Applying Scrum Methods to ITS Projects

www.its.dot.gov/index.htm

Final Report — August 2017

Publication Number: FHWA-JPO-17-508

U.S. Department of Transportation

Produced by Noblis, Inc.
U.S. Department of Transportation
ITS Joint Program Office

Notice

Technical Report Documentation Page

1. Report No. FHWA-JPO-17-508	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle **Applying Scrum Methods to ITS Projects**		5. Report Date August 2017 6. Performing Organization Code
7. Author(s) Barbara Staples, Dawn Hardesty, Blake Christie, Taylor Deurbrouck, and Josh Seder (Noblis), and Manny Insignares and Patrick Chan (ConSysTec)		8. Performing Organization Report No.
9. Performing Organization Name And Address Noblis 600 Maryland Ave., SW, Suite 755 Washington, DC 20024 Consensus Systems Technologies Corporation (ConSysTec) 301 East 87th Street, Suite 7A New York, NY, USA 10128-4807 Under contract to Cambridge Systematics, Inc.		10. Work Unit No. (TRAIS) 11. Contract or Grant No. DTFH61-12-D-00042 T-5010
12. Sponsoring Agency Name and Address ITS-Joint Program Office 1200 New Jersey Avenue, S.E. Washington, DC 20590		13. Type of Report and Period Covered Final Report 14. Sponsoring Agency Code HOIT-1
15. Supplementary Notes Kingsley Azubike (FHWA, Office of Operations), Ed Fok (FHWA Resource Center), Jesse Glazer (FHWA, California Division)		

16. Abstract

The introduction of new technology generally brings new challenges and new methods to help with deployments. Agile methodologies have been introduced in the information technology industry to potentially speed up development. The Federal Highway Administration (FHWA) developed this document to help those interested in learning about Scrum Methods, one of the Agile Methodologies, and how to incorporate Scrum into ITS project development. The FHWA wants stakeholders that choose to develop ITS projects using Scrum methods, to have appropriate information and assistance for successful implementation.

The two primary audiences for this document are State and local transportation agencies, and FHWA Division Office staff. For state and local agencies, this document will provide descriptions of Agile development and how it might be used on ITS projects. For FHWA staff, this document offers information on providing assistance for ITS projects that use Agile development. Within these two primary audiences there are decision makers that need to be aware of Agile development and combining it with systems engineering only at a high level. In addition, there is also useful information for contractors who are interested in using Agile development for federally funded ITS projects. The more detailed information in this document is geared toward the technical reader. Readers unfamiliar with systems engineering or agile development should first review the resources and tutorial videos identified in APPENDIX A. These are considered recommended reading and training before proceeding to read this document.

17. Key Words Agile, Scrum, Systems Engineering, Vee Model, 23 CFR 940.11	18. Distribution Statement No restrictions		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 87	22. Price

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

Table of Contents

List of Tables

List of Figures

Executive Summary

The USDOT regularly works with stakeholders to find innovative ways to better manage development and deployment of Intelligent Transportation Systems (ITS) and Connected Vehicle (CV) projects. One new approach being adopted by some in the ITS domain is the Agile software development philosophy for project execution. The USDOT has developed this document to introduce the concept to agencies interested in learning about the Agile philosophy and how it can be incorporated it into their own project development processes. The USDOT wants stakeholders considering the use of Agile for ITS projects to have the necessary information for successful application of Agile software development practices.

The Agile software philosophy was developed with the goal of making software development more efficient (i.e., faster and cheaper). Anyone adopting any new practice will need time to learn how to use it and when it is appropriate to use it. This document is designed to help those in the ITS community, that choose to adopt Agile practices, to have a better understanding of what Agile practice is and how it may be applied. Note that the concepts presented in this document will likely continue to evolve as more projects implement Agile software development, and experience develops at USDOT, State/local agencies, and their contractors.

This document explains the Agile development background and practice and how it can be used as a complement to the systems engineering process, along with some foundation information and initial suggestions for how to incorporate Agile software development practices into the management of ITS projects. Additionally, this document introduces Scrum (the most popular and widely used Agile methodology) and provides some suggestions for how to use Scrum to complement existing systems engineering processes to deliver ITS projects. For example, this document discusses how Scrum can be combined with the Vee lifecycle development model, typically used with system engineering for ITS projects. Examples are presented showing steps for combining Scrum with the Vee model to manage ITS projects. This document also discusses the type of projects that may be a good fit for using Scrum and types of projects where Scrum perhaps should be avoided.

Additionally, this document addresses how agency personnel (State and Local agencies, USDOT personnel, and contractor organizations) can use Scrum to work with project contractors to monitor and control project quality and progress. Information is also provided for choosing a project development team with appropriate Scrum and Systems Engineering training and experience. This document also provides some discussion of the potential impacts on federal oversight for projects applying Scrum methods, specifically oversight that relates to system development and systems engineering as defined in the Code of Federal Regulations (CFR), Title 23 (Highways), Part 940.11 (Intelligent Transportation System Architecture and Standards Project Implementation), hereafter referred to as 23 CFR 940.11 (https://www.fhwa.dot.gov/legsregs/directives/fapg/cfr0940.htm). [3] Additionally, the reader will gain a better understanding of procurement options surrounding the use of Scrum on ITS projects.

Finally, this document will cover some of the common benefits, risks and lessons learned using Scrum. Some discussion for mitigating project risks (e.g., failed project expectations, early obsolescence) will also be presented.

While Section 1.3 provides information on how to use this document and identifies recommended sections for particular audiences, presented below are two key take aways that apply to all readers.

Two Key Take-Aways for all Readers of this Document

1. **There are some projects where Agile methods (or Scrum) are appropriate, and others where traditional systems engineering should be used.**

Characteristics for projects that may be better suited to using Agile (or Scrum) include:

- The client does not have a good vision of specific product (or a single unit) functions and needs to see something tangible to help them decide on said functions
- System upgrades to existing systems where the new/needed functionality is well understood by the stakeholders
- New human interfaces that require frequent user trials to perfect the interface
- Web sites that require frequent user trials to perfect functionality
- Functionality that can be delivered incrementally

Characteristics for projects that are not suited to using Agile (or Scrum) include:

- Systems or system components dealing with safety critical or safety of life features/functions
- Systems requiring long-term maintenance and/or thoroughly documented project design decisions
- Systems consisting of high levels of integrated disparate systems

2. **There will be several challenges that need to be addressed by those adopting Agile methods (or Scrum)**

- Don’t use Agile methods (or Scrum) when safety of life, long-term maintenance, and integrating large disparate systems are at risk
- Consider the skill set, staff knowledge, and resources required when using Agile methods (or Scrum)
- Remember Agile is new to ITS community and implementation is still evolving

Safety of life pertains to a system function or feature that could impact loss of life (or injury) to users of the surface transportation network. Examples of safety critical functions and features may include, but are not limited to: traffic signal control, safety eminent vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) (I2V is implied) applications (e.g., forward collision warning, red light violation warning). For more information on safety of life and examples of features and functions that are not considered safety of life see letter from Michael O’Rielly, FCC Commissioner. [1] The operating agency needs to ascertain what projects have safety of life risks and what is acceptable in terms of risk for its projects. Comprehensive system documentation is needed to prove a thorough system analysis for safety critical systems. There may be instances (e.g., projects under litigation) where decision makers will need to be able to prove that ample consideration was made for safety of life features incorporated into the system. This can be achieved through rigorous systems engineering analysis and documentation practices. Furthermore, system documentation is needed to maintain systems that are fielded for 10 plus years, which is the norm for ITS.

List of Acronyms

Acronym	Meaning
CFR	Code of Federal Regulation
ConOps	Concept of Operations
COTS	Commercial Off-the-Shelf
CV	Connected Vehicle
DMS	Dynamic Message Signs
DoD	Department of Defense
DOT	Department of Transportation
DSRC	Dedicated Short Range Communications
FAA	Federal Aviation Administration
FBI	Federal Bureau of Investigation
FHWA	Federal Highways Administration
FTA	Federal Transit Administration
GAO	Government Accountability Office
GHG	Greenhouse Gas
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
HMI	Human Machine Interface
I2V	Infrastructure-to-Vehicle
ICD	Interface Control Document
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering
ITIS	International Traveler Information Systems
ITS	Intelligent Transportation Systems
LOE	Level of Effort
MVP	Minimum Viable Product
NASA	National Aeronautics and Space Administration
NRTM	Needs to Requirements Traceability Matrix
OSMA	Office of Safety & Mission Assurance
PCB	Professional Capacity Building
RSU	Roadside Unit
RTCM	Radio Technical Commission for Maritime Services
RTM	Requirements Traceability Matrix
SAE	Society of Automotive Engineers
SDD	System Design Document
SEDC	System Engineering Conference in District of Columbia
SLS	Space Launch System
SyRS	System Requirements Specification
USDOT	U.S. Department of Transportation
V2I	Vehicle-to-Infrastructure

Acronym	Meaning
V2V	Vehicle-to-Vehicle

1 Introduction

The introduction of new technology can bring new challenges and new methods to deployments. Agile methods have been introduced in the information technology industry to better manage the software development process where technologies are rapidly evolving. The Federal Highway Administration (FHWA) is offering this document with information to help those interested in learning about Agile software development methods and how Agile development methods can be considered for delivering Intelligent Transportation Systems (ITS) projects.

Use of Agile methods appears to be gaining traction within government agencies (e.g., the Department of Defense (DoD), US Department of Transportation (USDOT)) and also within private sector companies (e.g., Boeing, IBM). Agile methods were developed with the goal of making software development more efficient (i.e., faster and cheaper). Like any new practice, it takes time to learn how to use it and when it is appropriate to use it. This document explains the Agile development background and methods, and explains how these development methods can be used to complement the systems engineering process. The concepts presented in this document will likely continue to evolve as more projects implement Agile development, and experience develops at USDOT, State/local agencies, and their contractors.

This document introduces several Agile methods including one of the most widely used, Scrum, and provides some suggestions for how to use Scrum to complement the use of existing Systems Engineering practices to achieve greater efficiencies in the management and control of ITS projects. As will be explained in Section 2, there are several Agile methods; however, the rigor and structure of the Scrum methods nicely complements systems engineering Vee Model processes.

This document also addresses how agency personnel (State and local agencies, FHWA Resource Center and Division Office personnel, and contractor organizations) can use Scrum to work with project contractors to monitor and control software development quality and progress. Information is also provided for choosing a project development team with appropriate Scrum and Systems Engineering training and experience. Additionally, this document discusses the types of projects best suited for using Agile and key procurement considerations.

Finally, this document will cover some of the common benefits, risks and lessons learned using Scrum. Some discussion for mitigating project risks (e.g., failed project expectations, early obsolescence) will also be discussed.

The goal of this document, Applying Agile Software Development Methods for ITS Projects, is to provide information and assistance to those interested in applying Agile development to ITS projects. Some Agile development methods are discussed and examples are given to show how Agile development can be used in a complementary way with traditional systems engineering processes.

Agencies are currently following the systems engineering process to deliver ITS projects. In addition, some interest is growing for applying Agile practices to the development of software solutions for ITS projects. A potential benefit of Agile software development is reducing overall risks to some types of ITS projects based on frequent releases of useful, smaller capabilities or products. This is consistent with the Agile concept of Minimum Viable Product (MVP) to build the smallest thing that is of value to your customer. Building a MVP can help get a working piece of code in front of the client or component to market quickly. [7] There is a belief that Agile development and in particular Scrum can improve the rigor of project execution for software development projects.

1.1 Intended Audience

The two primary audiences for this document are State and local transportation agencies, and FHWA Division Office staff. For state and local agencies, this document will provide descriptions of Agile development and how it might be used on ITS projects. For FHWA staff, this document offers information on providing assistance for ITS projects that use Agile development. Within these two primary audiences there are decision makers that need to be aware of Agile development and combining it with systems engineering only at a high level. In addition, there is also useful information for contractors who are interested in using Agile development for federally funded ITS projects. The more detailed information in this document is geared toward the technical reader. Recommended sections for each audience is provided in Section 1.3. Readers unfamiliar with systems engineering or agile development should first review the resources and tutorial videos identified in APPENDIX A. There are a number of resources provided in APPENDIX A. , which are considered prerequisite reading and training before proceeding to read the bulk of this document. **Readers are encouraged to take a brief look at the resources listed and determine based upon the reader's skill set, experience, knowledge and needs those that are best and most appropriate for the reader to pursue.**

1.2 Compatibility with Federal Regulations

This document demonstrates how Agile practices can be utilized and fit within the framework of 23 CFR 940.11. That Rule states: *"All ITS projects funded with highway trust funds shall be based on a systems engineering analysis"* [3]. The rule does not prescribe how software needs to be developed, so developers are free to use Agile when it is appropriate.

Additionally, there are two primary systems-engineering guidebooks specific to ITS projects: 1) Systems Engineering Guidebook for Intelligent Transportation Systems [5] and 2) Systems Engineering for Intelligent Transportation Systems [6]. This document **does not replace** those two guidebooks; they both remain highly relevant and essential. Rather, it complements them and other existing system engineering resources for ITS projects. Those two guidebooks generally recommend the Vee Model for ITS projects, and the Vee Model remains one of the best project delivery methods when long term maintenance is required, major physical components are involved and being integrated, and safety risks must be minimized. But Agile development is emerging as an effective method for software delivery. Both the Vee Model and Agile development are compatible with 23 CFR 940.11 – and with each other – when used properly. Sections 3 and 4 of this document provide information and examples for using Agile development combined with systems engineering and identify some of the types of systems that are well suited for agile development.

1.3 How to Use this Document

The remaining Sections of the document are briefly described below. We recommend that:

- **State/local-Agency SE practitioners** should read at least the Executive Summary and Sections 1, 2, 3, 4, 6 and 8. **State/local agency decision makers** should read the Executive Summary.
- **FHWA Division Staff with ITS oversight duties** should read at least the Executive Summary and Sections 1, 2, 3, 5, 6, 7, and 9. **FHWA decision makers** should read the Executive Summary.
- **Consultants/contractors** to State or Local agencies should read the entire document.

1. **Introduction –** Provides the purpose, scope, and background to establish the document context.

2. **Fundamentals of Systems Engineering, Scrum Development, and the Vee Model –** Summarizes fundamentals of systems engineering and the Vee Model; describes the Scrum methodology; and introduces the concept of combining each.

3. **Getting Started - Why and When to Use Agile –** Project managers can use this section to guide the decision making process for when and where to use Agile (Scrum Method).

4. **How Agile (Scrum method) fits into the Vee Model –** This section explains how Scrum and the Vee model relate. Project managers can use this section to consider how to fit the Scrum method into the overall Vee Model.

5. **Cross-Cutting Activities** – This section explains activities that will cut across all systems engineering and Agile methods that should be considered to successfully manage the development of a system.

6. **Roles and Responsibilities when using the Scrum Method** – Understanding who does what in a system environment is one of the key concepts to success. This section explains the roles and responsibilities in a combined Scrum and Vee model project.

7. **Considerations for Federal Assistance When Using Agile** – This section identifies when and how using Agile methods is not in conflict with the requirements of 23 CFR 940.11.

8. **Procurement Options/Contracting** –This section provides information to help project managers develop procurement and contracting specifications for projects where Agile is a possible option.

9. **Summary and Next Steps** – This section summarizes the application of Agile on ITS projects. It also provides potential next steps for improving this document.

References – This section provides a list of all references used in the document and links to those references where available.

Appendix A – This appendix provides links to systems engineering, Agile, Scrum and popular Agile management tool resources.

Appendix B – This appendix provides information for the truck parking examples presented in Section 4.2.2.

2 Fundamentals of Agile and Scrum

The origins of applying “agile” concepts to system development began long ago and in other domains. To set the context for the reader of this document, a brief summary of the background and history of how agility and Agile originated is provided in this section. Also provided in this section is an overview of Scrum, an Agile methodology. A brief summary of systems engineering for ITS is presented in Section 2.1. For practitioners, who are very familiar with the Vee model, please skip Section 2.2.

In the 1990s, the Agility Forum (previously called the Agile Manufacturing Enterprise Forum) was formed. Its goal was for the United States to regain its competitive lead to become once again a global force in manufacturing. The Forum, whose members were representatives from industry, government and academia, focused on agile manufacturing enterprises. Agile enterprise concepts were formulated in several commercial domains (e.g., automotive, semiconductor, telecommunications) and the military. Agile system frameworks and agile enterprise reference models were being developed in the mid to late 1990s, and eventually migrated over to the software development community [8].

In 2001, seventeen people met to find common ground on an alternative to documentation driven, heavyweight software development processes [8]. Naming themselves “The Agile Alliance,” this group of freethinking software developers produced “The Manifesto for Agile Software Development” and the “12 Principles Behind the Agile Manifesto.” [20] The Agile Manifesto and the 12 supporting principles were written specifically as a philosophy for software development teams to easily adjust to stakeholder and user needs by focusing on people and interactions, not processes and tools. Using agile development allows project teams to incrementally deliver planned functionality earlier in the development cycle.

In March 2014, the Department of Defense published a guide, *Defense Agile Acquisition Guide: Tailoring DOD IT Acquisition Program Structures and Processes to Rapidly Deliver Capabilities* [10], to help them capitalize on the benefits of Agile development practices. “This guide provides DoD acquisition professionals with details on how to adopt Agile practices within each element of their programs.” Systems Engineering is one of the acquisition disciplines included in the guide.

The National Aeronautics and Space Administration (NASA) is applying Agile practices and traditional approaches to develop critical applications and functions. While NASA recognizes the value and flexibility of Agile practices that come from limited documentation and incremental product development, they also strive to ensure safety and software assurance needs are met [11]. NASA created their own customized Agile approach where portions of the development lifecycle are performed using Agile methods while others are performed using more traditional methods [11].

Much of the background research, so far, has concluded that combining Agile with systems engineering is still very much a developing concept. A number of frameworks and approaches have emerged and have been applied, yet there is still limited guidance and best practices. Currently, there are no industry accepted <u>standards that can currently be recommended for the application of the Agile philosophy for software development to systems engineering</u>. However, this document attempts to provide background information on Agile methodologies, and recommend some emerging practices for combining Agile with systems engineering for ITS projects.

2.1 Systems Engineering Process in ITS

For safety of life and system maintenance reasons, it is important to maintain a strong systems outlook for disparate systems (both large and small) in order to build complete and correct systems. For this reason, this document advises readers to be cautious in applying processes that are too flexible or not well vetted. The Vee model has obtained a certain amount of peer review and industry acceptance for system development, including safety critical systems.

Safety of life pertains to a system function or feature that could impact loss of life to users of the surface transportation network. Examples of safety critical functions and features may include, but are not limited to: traffic signal control, safety eminent vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) (I2V is implied) applications (e.g., forward collision warning, red light violation warning). For more information on safety of life and examples of features and functions that are not considered safety of life see letter from Michael O'Rielly, FCC Commissioner. [1] The operating agency needs to ascertain what projects have safety of life risks and what is acceptable in terms of risk for its projects. Comprehensive system documentation is needed to prove a thorough system analysis for safety critical systems. There may be instances (e.g., projects under litigation) where decision makers will need to be able to prove that ample consideration was made for safety of life features incorporated into the system. This can be achieved through rigorous systems engineering analysis and documentation practices. Furthermore, system documentation is needed to maintain systems that are fielded for 10 plus years, which is the norm for ITS.

As noted in Section 1.2, there are two primary systems-engineering guidebooks specific to ITS projects. FHWA's systems engineering guidance [5] recommends the Vee life cycle model as the development model for ITS projects. The Vee Model focuses on defining the problem to be solved and delivering the system. The process identifies and evaluates risks to deploying <u>the system</u> at each stage of the process (Concept definition, system requirements definition, etc.). The focus is on controlling <u>system risks</u> early so the number of changes to the system and elements of the system are minimized. The Vee process requires more definition work up front including prototyping (if needed) to define the requirements and then the design, implementation, and testing stages are all based on those requirements. The process is less flexible due to the need to define requirements early in the process.

However, flexibility is provided in the Vee via feedback provided between each stage. This feedback causes changes to the controlling documents that focuses the development and testing efforts, and guides maintenance of the system. The Vee supports the reduction of system risks early in the life-cycle. Major changes late in the life-cycle are much more expensive to correct (up to 1000 times more than when caught and corrected in the ConOps and Requirements stages) [12]. To its credit, the Vee also promotes the integration of multiple engineering disciplines and diverse systems. Large transportation systems (e.g. regional traffic management centers) use a wide variety of sensors, controls, and information dissemination devices to accomplish their missions. The Vee is a direct result of the need to consider and integrate the wide variety of devices, sensors, and mobile assets into a working whole. Its greatest advantage is that it looks at and constantly assesses the system. The Vee lifecycle model supports an iterative and incremental development and delivery approach of a system. Extensions to the basic Vee model, such as evolutionary deployment containing multiple build cycles, deal with incomplete requirements and incremental delivery approaches. In this case, the Vee model breaks development down into multiple concurrent sub-projects, and moves the traditional Vee approach towards a more agile approach while maintaining a strong systems outlook.

Figure 2-1 is another way to show the Vee Model. This figure illustrates the Vee Model with multiple, concurrent product developments and where each departs and reenters the main system Vee.

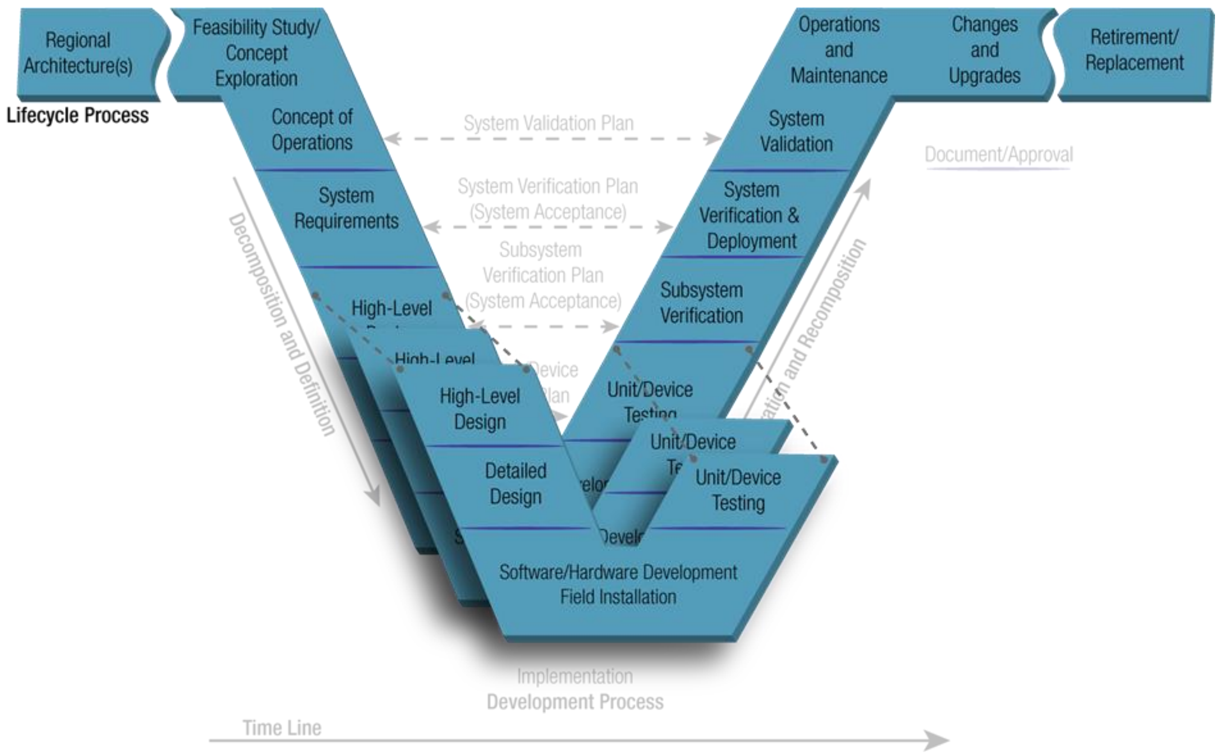


Figure 2-1: FHWA Systems Engineering "Vee" with Multiple Product Developments (Source: FHWA 2007 and modified by Noblis 2017)

The Vee uses systems engineering documentation to control system development. The type of documents used to control the development, manage, and maintain the system can be found in the two ITS systems-engineering guidebooks [5] [6] and the International Council on Systems Engineering (INCOSE) Handbook [12].

2.2 Agile Methodologies

There are a variety of Agile methodologies which include Scrum [7], Kanban [14], Extreme Programming (XP) [15], and others, each with their own unique processes, timelines, terminology, and practices. **This document uses Scrum methods because Scrum is a well-established and a broadly adopted Agile methodology. It is also well suited to the type of system development often encountered with ITS projects that are not impacted by safety of life or long-term maintenance.**

Scrum is an iterative agile methodology for managing product development “within which people can address complex adaptive problems, while productively and creatively delivering products of high value” [7]. Scrum focuses the team’s efforts on quick and incremental delivery of the product with regular feedback from stakeholders. This framework allows product development to respond quickly to changing requirements and adapt to evolving technologies. The usage and success of Scrum depends on the correct application of the roles, events, artifacts, and rules defined within the Scrum framework [7]. The Scrum Guide written by Ken Schwaber and Jeff Sutherland is an excellent starting point for guidance on Scrum basics which are summarized in the subsections below.

2.2.1 Scrum Roles

The three primary roles defined in the Scrum framework are the Product Owner, Scrum Master, and Development Team. These roles collectively make up the Scrum Team. Scrum Teams are self-organizing and cross-functional, that is, they manage their own work rather than being directed by others outside the team [7]. Scrum Teams work together to incrementally deliver a working version of the product that meets the requirements listed in the Scrum artifact called the Product Backlog. The roles and responsibilities of each Scrum role are described below.

- **Product Owner**: The Product Owner understands the needs and requirements of the product and is responsible for defining and prioritizing the work that needs to be completed. The Product Owner must be actively involved throughout the project to ensure the work being done aligns with the product requirements. The Product Owner is accountable for populating and prioritizing the items in the Product Backlog and communicating any change in requirements to the Scrum Team.
- **Scrum Master**: The Scrum Master provides support to the Product Owner and Development Team as the Scrum expert and ensures that the Scrum Team follows Scrum practices and rules.
- **Development Team**: The Development Team consists of typically five to nine people who do the work on each increment of the product.

2.2.2 Scrum Ceremonies

Scrum prescribes a process of events and artifacts (deliverables/ by-products of Scrum development) that center around a time-boxed (time-constrained) iteration called a Sprint. Once a Sprint begins, its duration is fixed and cannot be shortened or lengthened. Typically a Sprint will last from two to four weeks with two weeks being the most common. Specific strategies for implementing and using the Scrum framework vary and are described in many publicly available resources. A good resource to start with is the Scrum Alliance website [13]. The basic Scrum Method is described below and illustrated in Figure 2-2.

2.2.2.1 Sprint Planning

As illustrated in Figure 2-2, each Sprint starts with a Sprint Planning Meeting where the Scrum Team plans the work to be completed during the Sprint. In collaboration with the Product Owner, the Development Team selects a subset of the prioritized items in the Product Backlog (see 2.2.3.1) that they can reasonably complete in the fixed Sprint duration (usually 2 weeks). This subset of items and/or features that the Development Team commits to completing is called the Sprint Backlog (see 2.2.3.2).

2.2.2.2 Daily Scrum Meeting

During the Sprint, the Development Team meets daily for a 15-minute Daily Scrum Meeting. Each Development Team member discusses daily progress, planned work, and any roadblocks they have encountered. Roadblocks or impediments to the team are captured by the Scrum Master and assigned to a team member to find a resolution but no detailed discussion should happen within the Daily Scrum Meeting.

2.2.2.3 Sprint Review and Retrospective

Each Sprint ends with a Sprint Review Meeting where the Scrum Team reviews the work completed and the Development Team conducts a demo of the working product to the Product Owner. The Scrum Master also gathers the team for a Sprint Retrospective to solicit feedback on improvements to the team processes for the next Sprint cycle. This Sprint cycle continues until the product is released.

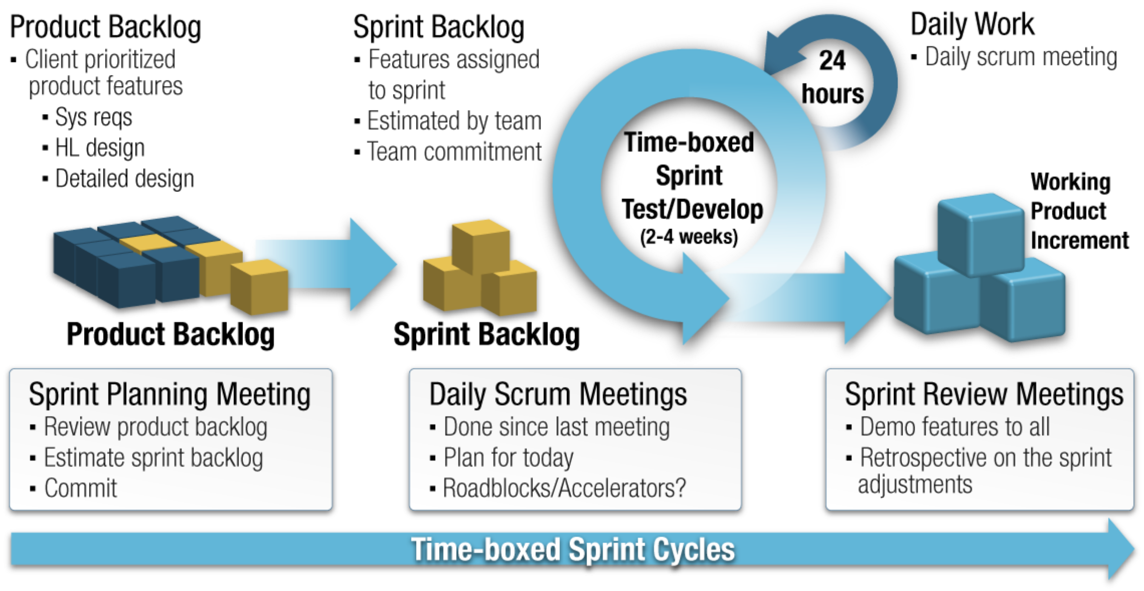


Figure 2-2: Illustration of the Scrum Method (Source: Noblis 2016)

2.2.3 Scrum Artifacts

Scrum artifacts (by-products of Scrum development) are designed to be transparent so that they can be inspected and adapted by the Scrum Team [7].

2.2.3.1 Product Backlog

The Product Backlog is a prioritized list of any requirements, features, or other items that might be needed in the product and is managed by the Product Owner. The initial Product Backlog may only include the preliminary requirements for the product and then evolve as user needs are identified, requirements change, or technology progresses. The Product Backlog is never complete [7]. The Product Backlog can be considered a “living” document that evolves as requirements change and exists as long as the product exists. Although not required by Scrum, Product Backlog items are often linked together in hierarchical fashion to show themes and unifying concepts. Figure 2-3 illustrates an example hierarchy structure for the Product Backlog.

As shown in Figure 2-3, the initial high level requirement, called an Epic Story, is a significantly large body of work that might need to be delivered over a set of Sprints. After conversations within the Scrum Team, this Epic Story could then be broken down into more manageable requirements called User Stories that are reasonable to complete within a single Sprint. Each User Story is then further broken down into detailed sub-tasks for the Development Team to complete. Each user story also includes Acceptance Criteria which are individual to each story. The Acceptance Criteria provide additional detail on how the Product Owner expects the feature to work and how they will assess the ability of the feature to perform its intended function.

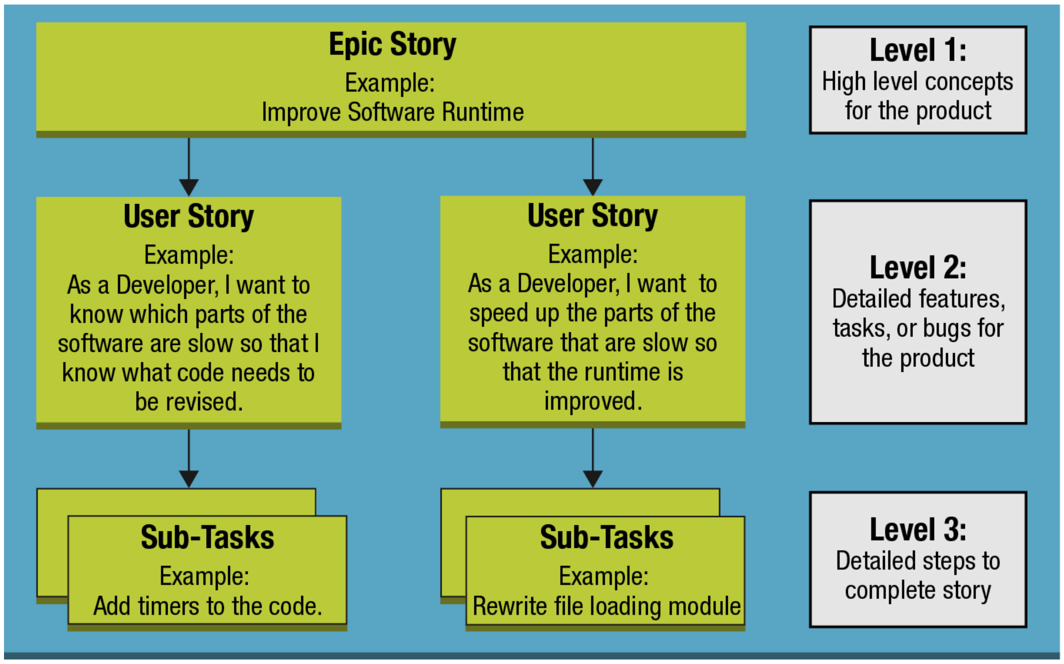

Epic Story
Example:
Improve Software Runtime
Level 1:
High level concepts for the product
User Story
Example:
As a Developer, I want to know which parts of the software are slow so that I know what code needs to be revised.
User Story
Example:
As a Developer, I want to speed up the parts of the software that are slow so that the runtime is improved.
Level 2:
Detailed features, tasks, or bugs for the product
Sub-Tasks
Example:
Add timers to the code.
Sub-Tasks
Example:
Rewrite file loading module
Level 3:
Detailed steps to complete story

Figure 2-3: Sample framework for Product Backlog items (Source: Noblis 2017)

2.2.3.2 Sprint Backlog

The Sprint Backlog is the set of Product Backlog items selected by the Scrum Team to be completed during the current Sprint. If using the structure in Figure 2-3, these would be a selection of user stories and their sub-tasks. After the Sprint Backlog is populated in the Sprint Planning Meeting, the Development Team uses the Sprint Backlog to track and estimate work progress. Therefore, the Sprint Backlog is a highly visible, real-time picture of the work that the Development Team plans to accomplish during the Sprint [7].

2.2.3.3 Working Product Increment

The Working Product Increment is the result of the work completed by the Development Team at the end of each Sprint. Each Increment represents the sum of all the Product Backlog items completed during the Sprint and each previous Sprint [7].

2.2.3.4 Definition of "Done"

The Scrum Team must decide as a whole on a definition of "Done" that must be met for each Product Backlog item or Product Increment to be considered complete [7]. Unlike acceptance criteria which are individual to each user story, the definition of "Done" applies to every user story and Product Increment. The definition of "Done" is a checklist of requisites or activities that determine if a story or increment meets project requirements. For example, "Done" might require that the code meets specific standards and includes unit testing.

2.2.4 Scrum with Large Teams

Due to the amount of interaction and collaboration required for Scrum, the recommended size of a Scrum Team is five to nine people [7]. Having more than nine members requires too much coordination and generates too much complexity for Scrum to be managed effectively [7]. When the team is too large, Scrum projects scale by having teams of teams. Although there are many techniques for scaling Scrum in this way, a popular method is the use of a “Scrum of Scrums” meeting [16]. With this approach, Scrum Teams proceed as normal but each team identifies one person who attends the Scrum of Scrums meeting to coordinate the work of multiple Scrum Teams [16]. Specific guidance on these techniques is beyond the scope of this guidance but is available in many publically available resources. Additional guidance on best practices for applying this concept in an Agile Systems Engineering approach is provided in Section 6.3.

2.3 Combining Scrum with Systems Engineering

Some organizations have adapted Scrum methods with system engineering processes to capitalize on its flexible structure while still maintaining the best practices of systems engineering. For example, NASA’s Marshall Space Flight Center’s Space Launch System (SLS) software team uses tailored Agile methodologies combined with traditional system engineering to assure that appropriate safety measures are taken during safety critical application development. NASA Software Assurance Technical Fellow Martha Wetherholt stated, “[agile is] an ideal approach in private industry, where you want to be first to market. You get your product out there and then you release updates. But when you’re heading for one launch of one vehicle, where safety needs to be provided and documented, a straight agile approach may not be the best option. Some Agile development processes like sticky notes and task boards aren’t appropriately documented, which means you can’t see problem areas, issues and how they are resolved” [11]. NASA’s process allows for portions of the software development lifecycle to be performed using Agile (within sprints) and others to include overall planning, black-box requirements development, and final product integration is performed outside the Agile process.

Combining Agile with traditional systems engineering, described in this document, consists of combining Scrum with the Vee Model for a more holistic approach to developing complex transportation systems. The following example is provided to show one way of using Agile for ITS projects. There could be multiple ways, however we present this example for consideration to show how Scrum can be integrated into the traditional systems engineering process.

The Agile approach, used in this example, is based on the Agile Systems Engineering Framework defined by Kennedy [9], and reinforced by NASA’s combining tailored Agile methodologies with more traditional methods. This approach applies to software development only, identifies issues, adds system level rigor, and adds documentation needed to maintain systems for the long term. For ITS projects, it is suggested that development of hardware, mechanical, physical plant, firmware, and safety critical software components follow the traditional systems engineering approach.

This example defines five main system phases: Release, Increment, Sprint, Integration, and Retrospective. These phases are described in the following subsections and illustrated in Figure 2-4. Where Scrum fits into the Vee Model is illustrated in Figure 2-5. Scrum is the middle portion of the figure. Additional phases are added prior to starting Scrum to help provide system planning and after Scrum to help provide for system integration. The additional before and after activities are not typically part of the Scrum process.

The planning shown in Figure 2-4 in the top bright blue boxes correspond to the left bright blue dashed box in Figure 2-5. This is the "before." The Sprints in Figure 2-4 in the middle green boxes correspond to the middle green dashed box in Figure 2-5. Lastly, the Integration and Retrospective phases shown in Figure 2-4 in the bottom muted blue boxes correspond to the right muted blue dashed box in Figure 2-5. This is the "after."

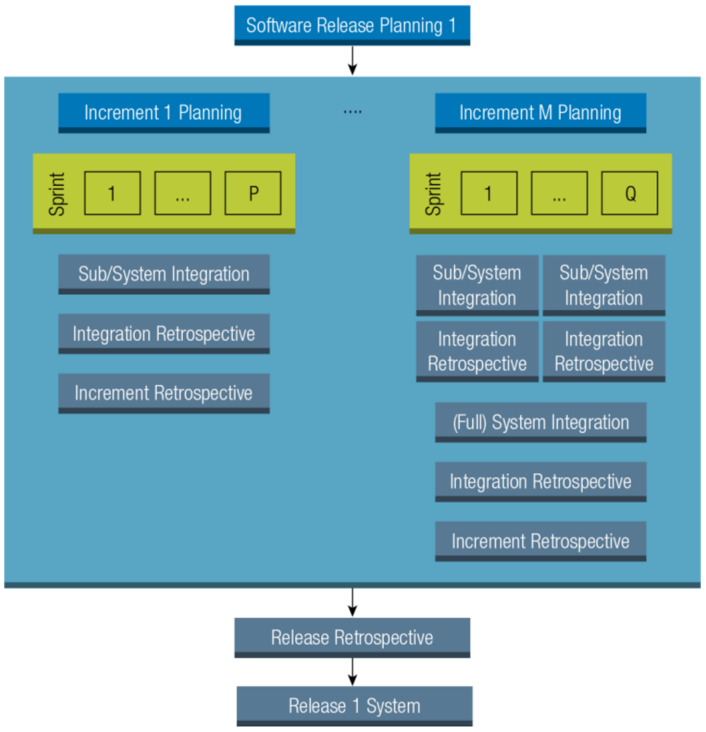


Figure 2-4: Example Agile Systems Engineering Framework (Source: Noblis 2016 adapted from Kennedy 2013)

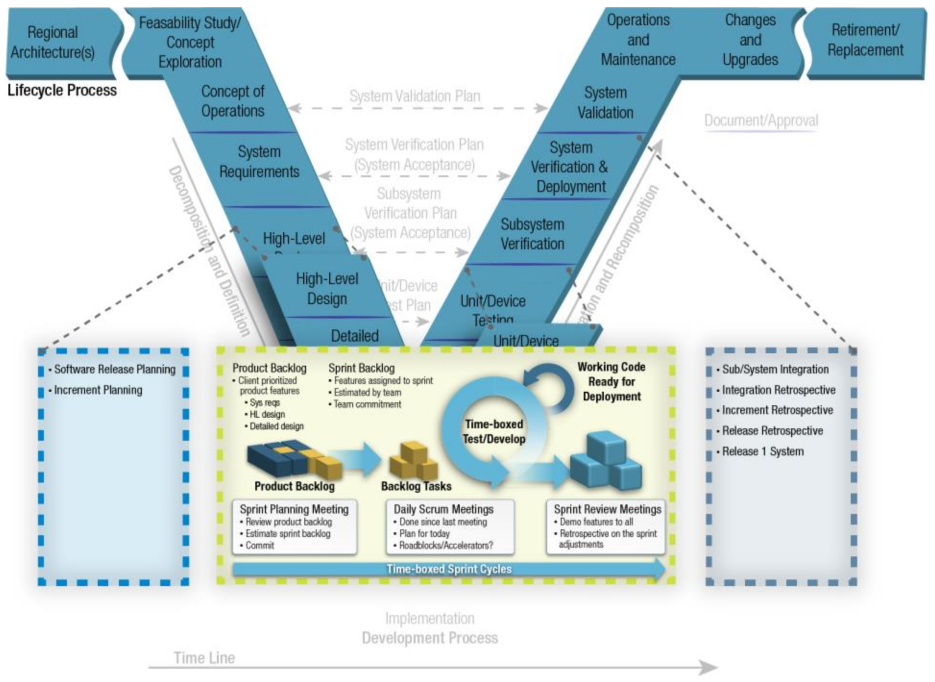


Figure 2-5: Combining Scrum with the Vee Model (Source: Noblis 2017)

2.3.1 The Release Phase

A system can be broken down into one or more software Releases. The Release phase starts with planning the Releases for the system. The system requirements are the basis for determining and prioritizing the Releases. It is important to realize the planning of the Releases depends on a solid understanding of what the system needs to do and relies on a common understanding of how the development of system requirements should be sequenced. The Release Product Backlog consists of the prioritized approved user needs or systems requirements. A release typically is an 8 to 9 month time block and results in an operational system. The system may or may not be deployed at the end of a Release. That is it could be an initial operating capability, or limited operation (prototype with limited deployment).

2.3.2 The Increment Phase

A Release can consist of one or more Increments. An Increment is a logical decomposition of the Release into smaller components or shorter timeframes. The combination of Increments make up the Release 1 system. An Increment may consist of one or more Sprints. The Sprints are usually logically associated and determined through the Release planning and Increment planning processes. Increment planning is based on what is necessary to organize each Increment and to get the Sprint process started. Each Increment has its own prioritized backlog, which is a subsection of the user needs or system requirements from the Product Backlog. An Increment typically is a 3 to 4 month time block. An Increment results in an Increment product, which is typically a functional piece of the overall system.

Increments can be structured to meet the needs of your project and resources. Figure 2-4 illustrates a Release that consists of multiple Increments (i.e., 1 to M). Increment 1 represents an effort that has limited resources, but is time flexible (i.e., a single Integration and a single Retrospective). Conversely, Increment M represents an effort with flexible resources, but is time-restricted (i.e., multiple Integration and Retrospective phases occurring in the same timeframe). Increments may occur serially or in parallel depending on the resources available.

2.3.3 The Sprint Phase

Sprint planning includes identifying and defining features and associated tasks to be completed during each time-blocked Sprint. Epics are identified from the Product Backlog as part of the Sprint planning meeting. The Epics are broken down further into User Stories and Tasks and become the Sprint Backlog for a specific Sprint cycle. Stories are either a requirement, a sub-function of a requirement, or a small set of requirements. Each Sprint produces a working product increment (a smaller component of the system). An Increment typically has 6 to 8 Sprints that are approximately 2 weeks each. Figure 2-4 illustrates Increment 1 having 1 to P Sprints, and Increment M having 1 to Q Sprints.

2.3.4 The Integration Phase

Using Scrum, code/module integration is continuously happening during the Sprint, resulting in the working product increment at the end of the Sprint. System integration can start once the first Sprint Review is completed and the Product Owner (Product Owner or representative) approves the product. System integration will help address and identify system issues and risks. Keeping the system integration separate from the Sprint unit testing is important to allow the systems engineering team to focus on the system level issues.

2.3.5 The Retrospective Phase

The Retrospective meeting is held to improve the work and team’s process at the Integration, Incremental, and Release levels. Each of the previously described four phases end with a Retrospective that is used to identify lessons learned – what did/did not work and improvements for the next phase.

2.3.6 Example Documentation

The Agile Manifesto values “working software over comprehensive documentation”. Agile attempts to strike a balance between discussion and documentation. Documentation should be:

Essential: Document with just barely good enough detail. Each system and environment has its own unique documentation needs, and certainly one size does not fit all. Keep the documentation as concise as possible.

Valuable: Document only when needed, not wanted. Agile projects prioritize work in the order of value, so the stakeholders must understand the value and total cost of ownership for a document, along with other requirements, and must explicitly choose to make that investment.

Timely: Documentation should be done in a just-in-time (JIT) manner, when it’s needed. Ideally, documentation should be created throughout the entire software development life cycle when it makes the most sense. Take an evolutionary approach to its development so that you gain feedback as to its actual value. [21]

The documentation provided when combining Agile with systems engineering is flexible. A tenet in Agile is that working software is more valuable than detailed documentation. The documents are determined by the team and what the contract specifies. Table 2-1 lists potential documents generated when combining Agile with systems engineering and may be in either electronic or printed form.

Table 2-1: Example Model Documentation

Document	Purpose	Phase(s)
Backlog Views (Product and Sprint)	Consists of Feature descriptions (requirements), acceptance criteria, LOE or hours estimates, priority, Sprint assignment, who is assigned and what Sprint the feature is assigned to, tasks, and any other details the Scrum Team tracks in their Product or Sprint Backlogs	Software Release Planning and Increment Planning
Source Code	Working source code developed according to organization software practices	Sprint
Test Description	How the tests will be conducted	Sprint
Test Results	Focuses on automatically generated test logs and reports. Also indicates product stakeholder acceptance.	Sprint
Detailed Design	May be optional, needed for maintenance, safety of life purposes, and contractual purposes.	After Release
Test Documentation	Focuses on planning and verification/testing during system and subsystem integration. This documentation would follow the normal system engineering Vee Model for these phases.	System Integration

Section 4 of this document provides a more detailed example showing the combination of Scrum with the Vee Model. However, in Section 3, there is first a brief discussion with suggestions on why and when to use Scrum for ITS projects.

3 Getting Started - Why and When to Use Scrum

Now that the concepts of the Agile philosophy and using Scrum methods for project execution have been introduced, an understanding of when and why to apply these methods within the context of ITS projects will be offered. First, it is highly suggested that Federal, State/local, and contracted project teams have training and experience with both Systems Engineering and Scrum before applying them to ITS projects. These are not project development practices that should be used in an ad-hoc fashion. Please use caution before proceeding with the information in this document. If your project team is not qualified in Systems Engineering and Scrum, stop and get assistance (e.g., get training, hire experienced team members). Some training resources for Systems Engineering and Scrum can be found in APPENDIX A.

Those unfamiliar with SE or Agile methods should consult appropriate resources before proceeding

Another caution for readers to consider is that when starting an ITS project it is important to consider potential options and constraints for system development. When developing and deploying ITS projects that will use federal funding it is important to remember that, 23 CFR 940.11 requires that a systems engineering analysis be performed. Using Agile methodologies does not mean that ITS projects do not have to follow applicable Federal, State, and local regulations and policies. Further discussion on using Agile methodologies with 23 CFR 940.11 will follow in Section 7 of this document. In the rest of this section (Section 3) the discussion will focus on when and why it makes sense to apply Scrum to your projects. Figure 3-1 below shows a graphic that depicts how Scrum is used in combination with the Vee Model. A detailed example, showing how to apply a combination of Scrum with the Vee Model, will be provided in Section 4 of this document. This formalized example is provided to help readers better understand how they can apply Scrum on their own projects.

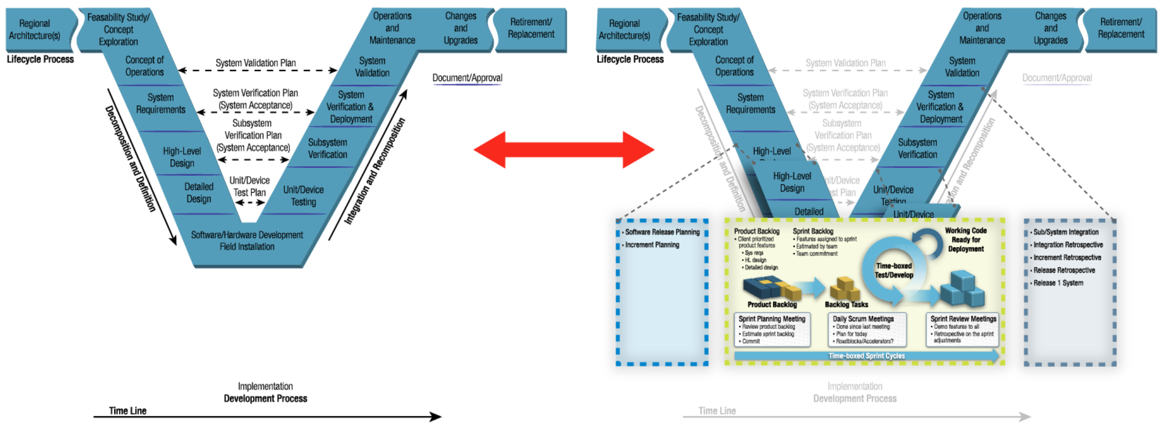


Figure 3-1: SE Only or SE & Scrum (Source: FHWA 2007 and modified by Noblis 2017)

3.1 Why choose Systems Engineering only?

When weighing factors for which types of projects Scrum may be best suited for, consider safety first. **Scrum may not be the best method to use if you need to thoroughly document project design decisions**. [11] Comprehensive documentation will still be needed to prove a thorough system analysis for safety critical systems. There may be instances (e.g., projects under litigation) where decision makers will need to be able to prove that ample consideration was made for safety of life features incorporated into the system. This can be achieved through rigorous systems engineering analysis and documentation practices. Note: This is not to say that a modified process using traditional systems engineering and Scrum could not be used, and a combined approach will be discussed in the next section (Section 3.2) and in Section 4.

Additionally, for systems where functionality and potential solutions are already well established, it may be unnecessary to use Scrum because the existing software may only need to be minimally tailored to meet specific stakeholder needs. For example, if you are deploying Dynamic Message Signs (DMS) and it is estimated to take less than a week to alter some software for the system, you may not need to use Scrum. This determination can be made by the project team assigned to the project, or may already be determined under any existing systems engineering analyses or change management processes previously outlined for the system.

3.2 Why choose the combination of Systems Engineering with Scrum?

Systems engineering and Scrum can be used in a complementary way to manage development of new systems with potentially greater success. Systems engineering provides defined project development processes to bring rigor to the project and Scrum can be a great way to accelerate and control project execution. As shown earlier, Scrum formalizes project execution so all participants have a grasp of project expectations and it keeps those expectations in front of the participants for the short Scrum timeframe. As discussed earlier, Scrum can improve teamwork and communication and it strives to reduce the timeframe for task completion.

One reason that Scrum could work well with the systems engineering processes is due to the fact that, systems engineering does not prescribe a method for how a chosen solution (software or hardware) is developed. Once a system is defined through the high-level requirements and architecture phases and a solution is selected to fulfill the requirements, solution developers are then free to select a development methodology that is best suited to their solution needs. Typically ITS solutions are going to involve software and hardware, which is why a software development method, like Scrum, compliments traditional systems engineering processes.

Analogies can be drawn between some of the stages of the systems engineering lifecycle and some of the steps that are followed in the Scrum method. Just like with the system analysis and definition lifecycle stages of systems engineering (e.g., concept of operations, high-level requirements and architecture) Scrum uses a functional decomposition process to populate backlogs with user stories that begin with high level descriptions of system needs and decompose into more detailed descriptions that can lead to an understanding of smaller developable pieces of a given system. These detailed user stories are somewhat analogous to system requirements. They help build the subsystems that ultimately lead to the integration and development of a complete system. The skill and experience with which teams are able to define a system will ultimately lead to its success or failure.

3.3 When to use and not use Scrum

It may be best to use Scrum when new systems with new functionality are being developed and the stakeholders are not able to fully define the system requirements due to incomplete details at the onset of the project. These would include projects that initially have a higher number of development unknowns or uncertainties. This does not mean there are unknown project needs or uncertainty about the objective or vision of the result. Project needs and objectives should still be defined clearly before considering Scrum.

Characteristics for projects that may be better suited to using Scrum include:

- The client does not have a good vision of specific product (or a single unit) functions and needs to see something tangible to help them decide on said functions
- System upgrades to existing systems where the new/needed functionality is well understood by the stakeholders
- New human interfaces that require frequent user trials to perfect the interface
- Web sites that require frequent user trials to perfect functionality
- Functionality that can be delivered incrementally

Characteristics for projects that are not suited to using Agile (or Scrum) include:

- Systems or system components dealing with safety critical or safety of life features/functions
- Systems requiring long-term maintenance and/or thoroughly documented project design decisions
- Systems consisting of high levels of integrated disparate systems

It is suggested that project managers not start their Scrum experience on a large complex project. Start small if you have never used Scrum. Begin with a project that requires a reduced level of effort. This will enable you to become familiar and comfortable with the application of Scrum in an environment with potentially fewer risks.

3.4 Initial Scrum Considerations

If you are thinking about using Scrum for project execution, some of the first considerations should include:

- Make sure the project team includes qualified systems engineering expertise
- Make sure the project team includes qualified Scrum expertise (e.g., Certified Scrum Master and trained Scrum Team)
- Train the entire team on how to implement Scrum
- Understand the roles and responsibilities of Scrum participants (this will be covered more in Section 6)
- Be aware of those that claim to be experienced in Scrum, but are not. (e.g., if you are currently managing a project and your contractor says they are using Scrum, but you have not been invited to a Scrum meeting, there is a problem.)
- Choose a good Scrum project management tool to adequately capture project details
- Understand that in many cases Scrum will be used in a complementary way to execute the already defined systems engineering project development processes
- Understand when it may be appropriate to include Scrum in the project development process (a detailed example will be presented in Section 4)
- Maintain traceability whether using systems engineering or systems engineering combined with Scrum
- Make sure that in the backlog the team is not deviating from addressing real system requirements to just addressing a list of items that need to be fixed (the backlog is designed to address requirements)
- Ensure that all public-agency participants will be able to dedicate the required person-hours, and will be able to meet short deadlines.
- Understand that the need for increased project development resources may increase project costs.
- Understand how to include Agile or Scrum in the acquisition process (this will be covered in Section 9).
- Use Scrum to potentially reduce the risk of software rework to accommodate changing specifications because software is released in smaller and more frequent releases. Note however that agile methods may not be able to reduce the risk of hardware obsolescence.
- Consider that the standard systems engineering process may be the best choice to avoid system obsolescence. The agile approach encourages hardware and software investments earlier in the acquisition process, and once those investments are made the system enters into the realm of obsolescence. The standard systems engineering process delays investments until the system is fully defined and those delayed investments may help to delay system obsolescence.

Additionally, it is suggested to **<u>not</u>** use Scrum unless the following criteria can be met for the project Scrum Team [17]:

- Team structure should be between 5 and 9 team members [7]
- Work should be conducted with collocated team members
- Team members should be completely dedicated to a single product or project
- Team structure should be static – minimal to zero variations in team
- Team members should be self-managed

- Team members should have cross-functional competencies

3.5 Suggestions for getting started with Agile Methodologies

In July 2012, the Government Accounting Office (GAO) publicly released a report entitled Effective Practices and Federal Challenges in Applying Agile Methodologies (GAO-12-681) [2]. This GAO report identified 10 practices that were found effective by officials from five agencies used in their study. The ten practices include:

1. **Start with Agile guidance and an Agile adoption strategy.** This practice advocates having these elements in place at the start, even if they must be copied from external sources.

2. **Enhance migration to Agile concepts using Agile terms and examples.** For example, use terms like user stories instead of requirements, and Agile Center of Excellence instead of Project Management Office. Provide examples, such as one illustrating the small scope of a user story to teams writing these stories.

3. **Continuously improve Agile adoption at both the project level and organization level.** This practice invokes the discipline of continuous improvement, meaning always looking for ways to improve. For example, improvements can be made by adding automated test and version control tools, and enhancing team rooms. These issues can be tracked in project and organizational-level backlogs.

4. **Seek to identify and address impediments at the organization and project levels.** This practice encourages organizations to be frank about identifying impediments so that they can be addressed.

5. **Obtain stakeholder/customer feedback frequently and closely.** For example, feedback is obtained during the iteration and at its completion at an iteration retrospective. This practice was linked to reducing risk, improving customer commitment, and improving technical staff motivation.

6. **Empower small, cross-functional teams.** Empowered teams of 5 to 9 people decide what to deliver and how to produce it. The teams should not over-rely on one member's skills.

7. **Include requirements [stories] related to security and progress monitoring in your queue of unfinished work (backlog).** Including activities such as security reviews and status briefings in the backlog ensures their time and cost are reflected and that they are addressed concurrent with, and not after, iteration delivery.

8. **Gain trust by demonstrating value at the end of each iteration.** This practice includes demonstrating key requirements in early iterations, and showing customers that requirements in the backlog are delivered and not forgotten.

9. **Track progress using tools and metrics.** Progress can be tracked using tools and metrics, which can be automated, and by success indicators such as reduced staff stress and overtime.

10. **Track progress daily and visibly.** This practice stresses that status is checked daily and publicly. For example, a progress chart is posted openly in the team's workspace, with timely revisions to reflect ongoing feedback.

3.6 Challenges with Applying Agile Practices

The GAO report also identified 14 challenges (see pages 15-20 in the report for more details) with adapting and applying Agile in the federal environment [2]:

1. Teams had difficulty collaborating closely
2. Procurement practices may not support Agile projects
3. Teams had difficulty transitioning to self-directed work
4. Customers did not trust iterative solutions
5. Staff had difficulty committing to more timely and frequent input
6. Teams had difficulty managing iterative requirements
7. Agencies had trouble committing staff
8. Compliance reviews were difficult to execute within an iteration time frame
9. Timely adoption of new tools was difficult
10. Federal reporting practices do not align with Agile
11. Technical environments were difficult to establish and maintain
12. Traditional artifact reviews do not align with Agile
13. Agile guidance was not clear
14. Traditional status tracking does not align with Agile

Presented above are some high level considerations and suggestions for why and when to use Agile to develop ITS software systems. The next section (Section 4) will provide more details for how to apply Scrum to ITS project software development by combining it with the Vee model.

4 Combining Scrum and Systems Engineering

This section provides an example of how Scrum can be combined with systems engineering and in particular the Vee Model for system development. This section is broken into three parts. The first part provides an overview of how Scrum and the Vee Model can work together. The last two parts provide a detailed description and an example of how Scrum works with the Vee Model.

4.1 Overview of how Scrum fits into the Vee Model

Deployers of ITS may choose to use Agile methodologies that have the potential to speed up software development. However, for large disparate systems, development of a product must retain the ability to relate to the total integrated system. The suggested ITS systems engineering/agile approach described in this section consists of combining Scrum and the Vee Model for a more holistic approach to developing ITS (see Figure 4-1).

It is important to understand that the Vee Model and Scrum are not equivalent to each other. Each has a different focus with different advantages and disadvantages. The Vee Model focuses on looking at the whole system, decomposing components into more manageable chunks (a top down approach), and manages risk at the system level for the whole development lifecycle. The Agile development methods focus on short iterations of working software and manages risk as part of frequent customer deliveries. While Scrum has demonstrated to be faster and lower in cost in developing software, the Vee manages risk in a holistic manner for the whole system and reduces lifecycle schedule and cost.

Figure 4-1 shows how Scrum can fit into the Vee Model at the design, implementation, and unit testing stages. However, there are also other Vee Model stages where Scrum can be entered from (one can begin Scrum as early as the Concept of Operations Stage). There are also a range of stages where Scrum can reenter the Vee Model (Scrum can reenter the Vee Model as late as the System Validation stage). The stage that the Scrum is entered and exited is dependent upon the size and complexity of the project, and the level of risk the project team is willing to take on. Note as indicated in Figure 4-1, the Vee Model is used for the total system and additional development/implementation activities, for separate units or subsystems (hardware, safety of life related components, etc.). The separate units or subsystems are added to the overall system Vee Model (notice the regular Vee path and Scrum path additions in the figure). Scrum is one of those possible additions.

In Figure 4-1 below, Scrum is shown as an addition to the Vee Model, see box with dashed green lines. Additional activities are added prior to starting Scrum and after Scrum to help provide for system planning and system integration that are not typically found in the Scrum process, see boxes with gray and blue dashes lines. These pre and post activities also address documentation needs required to provide for long term maintenance. Although Scrum can be faster, it may do so at the cost of sufficient documentation needed for maintenance and early systemic issues that may show up during integration. If the systemic issues are only addressed as one integrates modules and subsystems, then the costs saved using Scrum are quickly lost since the most expensive time to make changes is during integration. The Scrum process is described in Section 2.

As shown in Figure 4-1, the traditional stages of the Vee Model were followed until system requirements and high-level design were completed. This helps deployers define what they need and sets the stage for using Scrum. As noted above, Scrum has a range of potential starting points. This approach gives projects greater flexibility. If the start is earlier, greater risk could be introduced to the project since combining Scrum products into system units and subsystems also have systems issues that will not be verified until those components are integrated together. The Vee Model uses methods to analyze potential system level risks early and mitigates them when they are less costly to address. Scrum fixes problems as they are found in smaller sized modules. Most of the time these smaller modules keep the cost to fix problems very low. However, system level issues mostly show up at system level integration where it can be more costly to fix the problem. Projects may start using Scrum earlier if the risk is determined to be acceptable by the owner and developers. If the project starts after the low level design stages the project will have less flexibility but there is also considerably less risk.

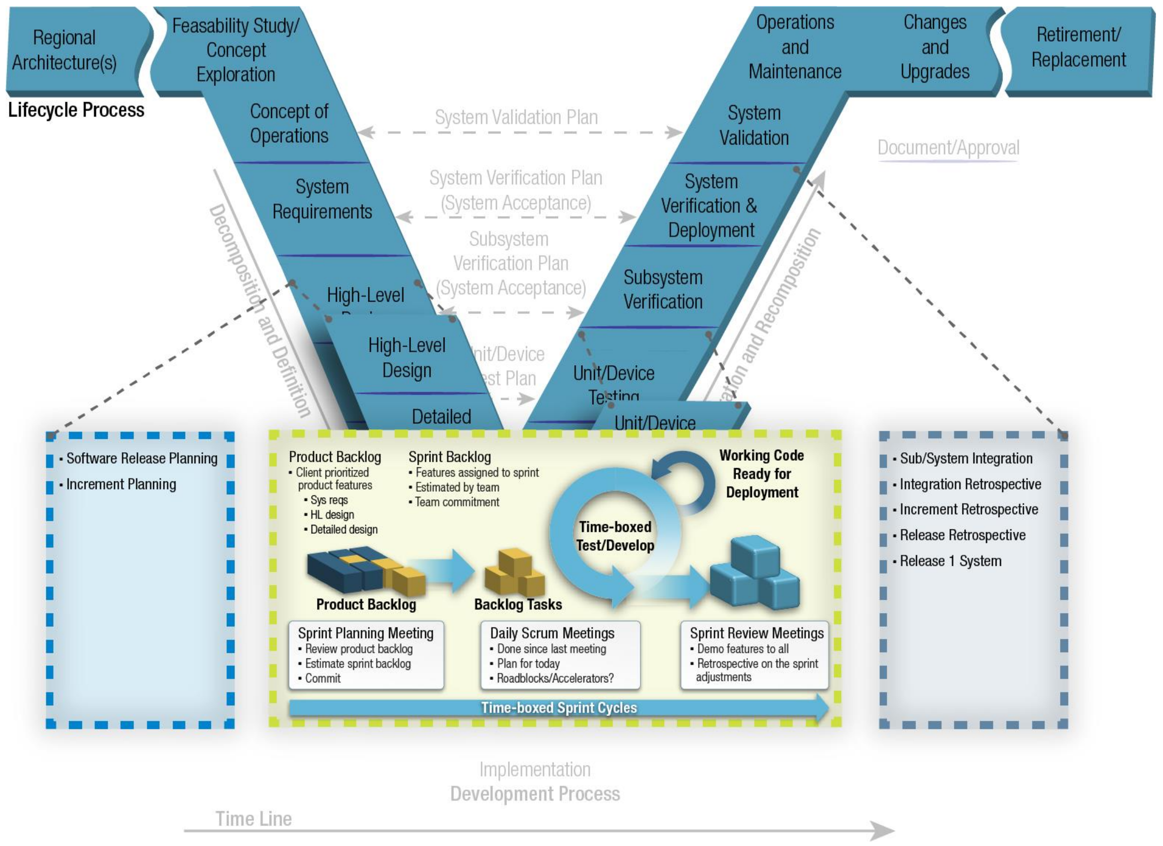


Figure 4-1: Systems Engineering Vee Model Adding the Scrum (Source: FHWA 2007 and modified by Noblis 2017)

As shown in Figure 4-1, traditional stages of the Vee Model are resumed at Subsystem and System Validation stages. It is suggested that projects follow the traditional systems engineering approach for hardware, mechanical, physical plant, firmware, and safety-of-life critical software components.

Scrum uses the requirements from the Vee Model and follows the process described in Section 2.3 with the system/subsystem or lower element requirements becoming the Product Backlog. When Scrum is complete, for the software product, it then reenters the Vee Model via the system integration activities (in this case after unit testing). The systems engineers that will take the Scrum product into integration will collaborate with Product Owners during the Scrum process. It is important to note that a released Scrum software product may have gone through multiple Sprint cycles with each Sprint cycle delivering a portion of the product until the complete product (or planned increment release) is ready for integration with other parts of the system. In the Vee Model all units, lower elements, or subsystems will eventually be integrated into the complete system independent of the method of development (Scrum or Vee Model).

Some projects may be implemented using only Scrum. For example, implementing a set of traveler information pages on an existing website where the pages are being modified, removed, or created often. This type of project is well suited for using Scrum to maintain the websites since the system the websites exist on, is already in place. Scrum can be considered where long-term maintenance, safety of life issues, and high levels of disparate systems are not being integrated.

Another key concept used in developing systems is the concept of traceability. If the origin of each of the project user needs, requirements, design, and verification is clear and if it facilitates the referencing of each user need, requirement, design, and verification in future development or verification documentation then the project has traceability. Traceability in the Vee Model is used to help verify that the system, subsystem, elements (and products) are complete and correct. It is used to guide the development of the system and individual products. It is also used to show that the verification activities (Unit testing, Sub-system Verification, System Verification, and System Validation) are complete and correct. The requirements are the focus of all traceability and are the key to verifying the deliverables at various stages as shown in Table 4-1.

Table 4-1: Traceability in the Vee Model

Tracing	Purpose	Comments
From: User Needs and Constraints To: System Requirements	Verify that the requirements satisfy the user needs and constraints	Usually found in a needs to requirements matrix (NRTM) and provided in the System Requirements Specification (SyRS)
From: System Requirements To: Sub-system requirements	Verify that the sub-system requirements fulfill the system requirements	Usually found in a needs to requirements matrix (NRTM) and provided in the System Requirements Specification (SyRS)
From: Sub-System Requirements To: Element Requirements	Verify that the element requirements fulfill the sub-system requirements	Elements breakdown into lower levels until the requirements are defined. Usually found in a needs to requirements matrix (NRTM) and provided in the System Requirements Specification (SyRS)
From: Requirements To: Functional Architecture	Verify that all requirements are satisfied by one or more function.	The functional architecture is decomposed at the same time as the sub-system and element levels. Found in a matrix developed to show the functional architecture element and in the description of the functional architecture element.

Tracing	Purpose	Comments
From: Requirements To: Detailed Design	Verify that the detailed design fulfills the requirements	Found in a requirements traceability matrix (RTM) and in the detailed description of each design component.
From: Requirements To: Test Cases/Procedures	Verify that every requirement will be verified by some verification type	Verification types are inspection, analysis, demonstration, and test. Found in a requirements to test case traceability matrix (RTCTM). Also, knows as the Test Design Specification in IEEE 829.

Traceability in Scrum is used to document develop epics, stories, tasks, code, and verification (testing) and to link them together. Traceability in Scrum begins in the Product Backlog. The Product Backlog is created based on documented requirements taken from the systems engineering effort. A subset of the requirements (from the Product Backlog) are transferred to the Sprint Backlog and are usually captured in Scrum management tools (e.g., VersionOne, JIRA, etc. see APPENDIX A.). It is the responsibility of the Scrum Team to break down the work into epics, stories, tasks, and test cases to populate the Scrum backlog. At product testing, the Product Owner in collaboration with the systems engineer will use the original requirements associated with the product to approve or not approve the results of the testing.

4.2 **The Application of Scrum and the Vee Model**

The sections below provide three examples of how to apply Scrum and the Vee Model in combination. The first example shows a simplified project where Scrum is used as part of the Vee Model. The purpose of this first example is to help the reader become familiar with the concepts. The next two examples demonstrate a more complex project with two approaches, one using the Vee Model only and one integrating Scrum into a Vee Model. These last two demonstrate how to the use the Vee Model or Scrum while deploying an ITS standard. These last two help the reader understand the issues related to using or not using Scrum in a more complex project.

In all of the cases below, the regional architecture was used to identify the interfaces and appropriate standards based upon use of the National ITS Architecture. Therefore any additional architecture concepts found in the examples were derived as part of the design and are not available from the National ITS Architecture.

4.2.1 Bike Sharing Example

The first example shows how to integrate Scrum into the Vee Model in a simplified example. It also shows how to start using the Scrum process earlier in the Vee Model stages. Implementing a user application for smart phones and laptop computers via the web will be developed using Scrum starting after the ConOps stage. The Bike Sharing System (see Figure 4-2) is really a simplified system of systems (SoS) as shown in Figure 4-3. It consists of two systems an external component, and actors:

- Bike Location System
- Bike Tracking and Rental System
- Bike users (actor)
- Bike Inventor (external component)

- Management and Inventory Control (actor)

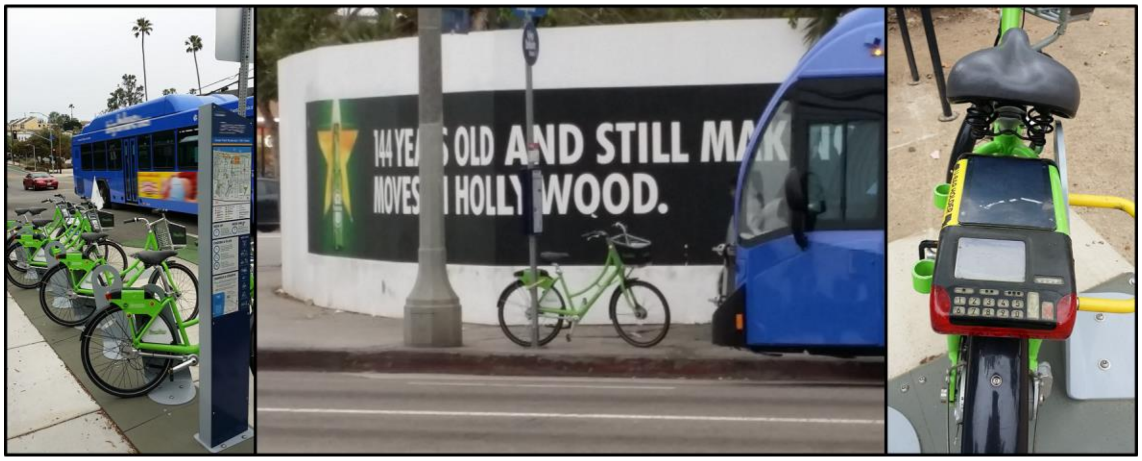


Figure 4-2: Photos of the Bike Sharing System (Source: FHWA Staff 2017)

The process starts with the development of a concept of operations (ConOps) as part of the Vee Model. During the development of the ConOps the following is identified:

1. Problem Statement:
    - To reduce traffic congestion/pollution/greenhouse gas (GHG) emissions, we need new alternatives for short trips in dense urban areas:
        - First-mile / last-mile access to transit
        - Shopping
        - Recreation
        - etc.
2. User Needs:
    - Pick up a bike anywhere and leave anywhere within the designated control area.
    - Bike user must be able to locate one or more bikes on their smart phone or via the internet on their home or office computer.
    - The application and internet bike locating system will not reserve or activate/deactivate a rental. There are no reservation of bikes in the bike rental capability and the activation/termination of a bike rental is the responsibility of the Bike Tracking and Rental System (see constraints below).
    - The bike rental fees will be part of the information provided to the user, including additional fees for leaving the bike out of the control area.
3. Constraints:
    - Tracking of bikes is to be done using a GPS and must have a resolution of at least 100 feet as part of the Bike Tracking and Rental System (existing system).
    - The bike provides a lock mechanism that works at the station racks or with a light or fence pole that the user can easily use. The Lock mechanism is fixed to the bike and not removable from the bike.
    - The bike provides a card reader that will read either a Metro rider card or a member card with sufficient funds to unlock the bike and to activate/terminate the bike rental (all user operated). Communications to activate/terminate the rental is part of the Bike Tracking and Rental System.

4. Context Diagram (see Figure 4-3)
   - The Bike Sharing system is a system of systems.
   - The project focus is to develop the capability to allow Bike Users to locate a bike near them (if available) using the Bike Location System. See user needs and constraints above.
   - The other parts include the Bike Tracking and Rental System, the bike inventory, and the Management and Inventory Control.

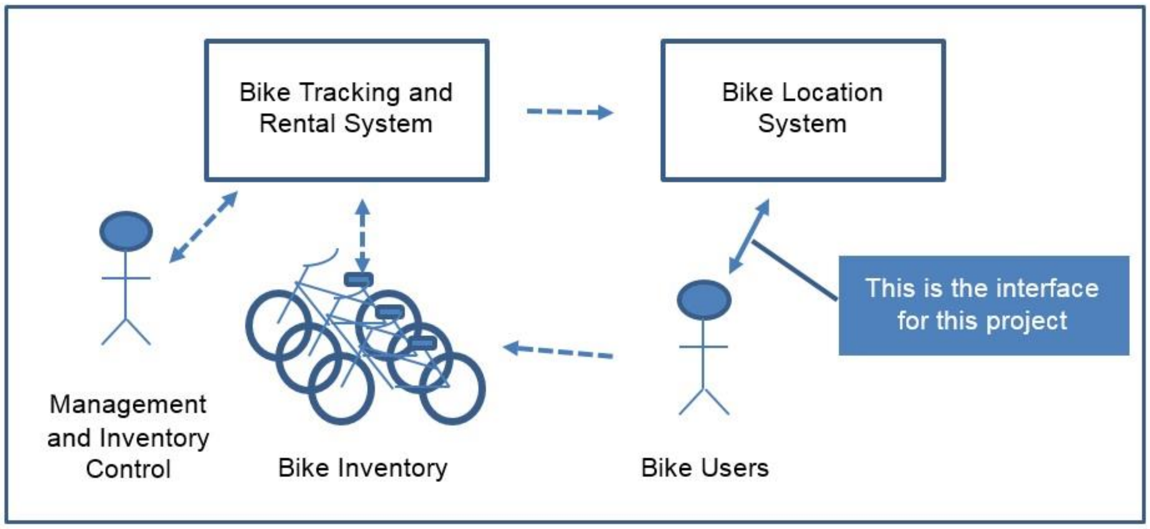

Bike Tracking and Rental System
Bike Location System
This is the interface for this project
Management and Inventory Control
Bike Inventory
Bike Users

Figure 4-3: Bike Sharing System Context Diagram (Source: Noblis 2017)

After the ConOps stage, the system requirements stage begins. These are the requirements for the system (the Bike Sharing System that consists of both the Bike Location System and the Bike Tracking and Rental System). Individual system requirements will be developed after the system of systems requirements are developed.

With the ConOps stage and system requirements completed and allocated using the Vee model to software and hardware solutions, the Bike Location System is broken off to be developed using Scrum. The Bike Tracking and Rental System will continue to employ the Vee model since it is mostly hardware and uses existing location systems (GPS) and commercially available software.

The Bike Location System will take the user needs and constraints into the Scrum process as product backlog items. The product vision is contained in the ConOps.

The following describes the process used to develop the Bike Location System:

- SCRUM Begins – Leave Vee Model
  - Software Release Planning (see Figure 2-4):
    - The approved Bike Location System user needs are entered into the Product Backlog by the Product Owner.
    - Software Release Planning includes breaking the Bike Location System user needs in increments. Each of these increments has its own Product Backlog (a subsection of the Bike Location System user needs). Each increment contains one or more Sprints where the increment Product backlog will be further broken down into logical

groups. An Increment Team will then develop the product associated with one or more Sprints using the Sprint method described below.

- Sprint Planning
  - Epics are identified from the Product backlog as part of the Sprint planning meeting. The Epics are broken down further into User Stories and Tasks and become the Sprint Backlog for a specific Sprint cycle. Stories are either a requirement, a sub-function of a requirement, or a small set of requirements. For example, one story can be a requirement to allow the Bike Location System to be able to identify available bikes within a set radius to the bike user. Another story can be a requirement to allow the bike user to register with the Bike Sharing Program as a user via the Bike Location System. For this example the smart phone and the internet apps are considered separate products.
- Sprint Process (see also Section 2.2)
  - With the Sprint Backlog established the Sprint can begin.
    - Daily Scrum meetings (report status on development of the stories, what is to be done today, issues and roadblocks)
    - Complete daily work (develop code, create test cases and procedures, test code)
    - Sprint Review meetings (demo features to all, Retrospective on Sprint adjustments if Product Owner and systems engineer accept product, then code documentation is finalized and the Scrum Team proceeds to Sprint Planning of the next Sprint cycle)
    - Sprint cycles occur until all Epics (and their Stories) for the product in the Software Release are completed. Note: More than one increment may have to be exercised to complete the Software Release and the increments may occur serially or in parallel depending on the resources available.
    - Specifications related to the interfaces with the Bike User and the Bike Tracking and Rental System will need to be looked at during the Sprint. In Scrum, Interface requirements are managed the same way as system requirements. The Bike Location System Product Owner will need to meet with the systems engineer for the Bike Tracking and Rental System. The Sprint team will address the Bike User interface as part of the Sprint Process.
    - Testing the Software Release of the smart phone app happens during the Sprint. In this example, testing of the Software Release is also unit testing in the Vee Model. This is where the Scrum begins to merge back into the Vee Model. Testing the web site application via a home or work computer is considered a separate product test (unit test) and is a separate product.

SCRUM Ends – Re-enter Vee Model

- Sub-System Verification Stage Begins:
  - Once unit testing is complete for each product (smart phone and internet via home or office computer), the unit (Software Release) is integrated with the next higher level element (Bike Tracking and Rental System) in the Vee Model. Note that each system in the Context Diagram (Figure 4-3) is considered a subsystem in the Vee Model. At this point integration and testing is based on a Verification and Validation Plan done at the high level design stage in the Vee Model. Details of the verification at this level are defined in the system test documentation. All verification will start at the product (or unit) level and end at the System requirement level (for acceptance testing). Documentation, required to maintain the product, is added in the Sub/System

Integration phase outside of the Sprint process. If the products developed using the Scrum do not work correctly during integration and a separate sprint may be organized to identify and resolve the interface issues and correct the product with the error.

- The integration and testing repeats until all units and systems are integrated (usually one at a time) and shown to work correctly.
- Sub-system test reports, test logs, and anomaly reports are created and anomalies resolved.
- The integration Retrospective will be conducted outside of the Sprint process based on the integration of the Scrum Software Release product into the sub-system.

- System Validation Stage:
  - Acceptance testing begins (based on the test documentation developed by the systems engineers and approved at detailed design review). All acceptance testing is done at the system level (includes external system interfaces to the Bike User, Bike Inventory, and Bike Management and Inventory Control) for the respective systems.
  - System test reports, test logs, and anomaly reports are created and anomalies resolved.
  - Design documents, operational documents, and maintenance documents are updated to as-built.
  - Training classes will be conducted either prior to final acceptance or after final acceptance depending on the clients scheduled needs.
- Operations and Maintenance Stage:
  - Once the system is fully accepted then operations and maintenance begins (based on operational and maintenance documents).
  - Updates based on design documentation.

4.2.2 Examples of Steps Taken when using the Vee Model Only vs. Scrum Combined with the Vee Model

The second and third examples provide the function of generating a message that provides truck parking information from a transportation agency or operating entity (see APPENDIX B. for details). This scenario depicts traveler information for a commercial vehicle operator of which parking availability is one of the message types. The scenario (see Table B-1), user needs and requirements for message contents were taken from the SAE J3067 Information Report. [19] In this example the system will use an ITS Standard to define the message format and message contents. The scenario describes the operation intent. The user need describes the specific need for commercial operators to be informed of parking availability upstream. Both the scenario and the user needs would show up in a ConOps developed during the Concept of Operations stage. The requirements represent both system level and interface level requirements for the operating entity to generate the message and its contents for this project. The requirements would show up in the System Requirements Specification (SyRS) developed in the System Requirements stage. The operational context is shown in Figure B-1. In this scenario, the Freight Vehicle system receives parking information from an operating entity (e.g. Transportation Agency). The parking availability information is provided by parking facilities to the operating entity who aggregates the information and formats it for transmission to freight vehicles.

4.2.2.1 Vee Model Only

In this example (see example details in APPENDIX B.), the user need, context diagram and detailed requirements have already been developed as part of the Concept of Operations and System

Requirements stages. The requirements and design have been decomposed (using systems engineering methods of requirements [12 (Sections 4.3, 4.4, and 4.5)], architecture, and design definition processes) to identify an interface that creates and sends a traveler information message about available parking to commercial vehicle operators. Traceability between the user need and requirements are shown in the Needs to Requirements Traceability Matrix (NRTM) in APPENDIX B. (Table B-2). These activities occurred in the Requirements, High Level Design, and Detailed Design stages of the Vee Model.

The Vee Model example is now near the end of the Detailed Design stage. The interface requirements reference message contents and message content design in a standard (SAE J2735-2016) and is ready to enter the detailed design review [12 (Section 4.9)]. Although the whole system is usually reviewed at this decision gate, this example will focus only on the Truck Parking Message Generator example. The detailed design review will evaluate the design architecture, how the interface requirements are being fulfilled, and how the requirements will be verified. Once approved, the following happens using the Vee Model:

Implementation Stage Begins:
- The design team takes the approved design for the Truck Parking Message Generator (captured in the Software Design Description (SDD)) and develops code [12 (Section 4.7)] (captured in the Source Code files).
- The design team conducts a code review to make sure the code is built right and fulfills all requirements shown in the NRTM in APPENDIX B.

- Unit/Device Testing Stage Begins:
  - The test team (may be the same team as the design team) develops the test cases and test procedures associated with unit testing for the Truck Parking Message Generator (captured in the test documentation). The test cases are based on the requirements (see NRTM in APPENDIX B.) and the test procedures will add the ITS standard details for the Parking Availability Message.
  - The test teams conduct a test readiness review (TRR) with design team and systems engineers. If all documentation and code is ready, unit testing will commence for the Truck Parking Message Generator.
  - The design and test teams conduct unit testing (based on test documentation and generates test reports, test logs, and test anomaly reports). Design documents are updated to as-built.
- Sub-System Verification Stage Begins:
  - Once unit testing is complete (passed all unit testing), the unit (Truck Parking Message Generator) is integrated [12 (Section 4.8)] with the next higher level element (up to and including subsystem) for integration and testing (based on Verification and Validation Plan done at high level design stage). For example, the Truck Parking Message Generator will be integrated with the Parking Facility Collection Module and eventually other functions (see Figure B-2 in APPENDIX B.).
  - The integration and testing repeats until all units and elements are integrated and shown to work correctly and ready for System Integration.
  - Sub-system test reports, test logs, and anomaly reports are created and anomalies are resolved.
- System Integration and Verification Stage:
  - Integration of all sub-systems (usually one at a time). Verification and integration processes to continue until all subsystems are working together.
  - System test reports, test logs, and anomaly reports are created and anomalies are resolved.

- System Validation Stage:
  - Acceptance testing begins [12 (Section 4.11)] (based on the test documentation developed by the systems engineers and approved at detailed design review).
  - System test reports, test logs, and anomaly reports are created and anomalies are resolved.
  - Design documents, operational documents, and maintenance documents are updated to as-built.
  - Training classes will be conducted either prior to final acceptance or after final acceptance depending on the clients scheduled needs.
- Operations and Maintenance Stage:
  - Once the system is fully accepted then operations and maintenance begins (based on operational and maintenance [12 (Section 4.12, 4.13)] documents).
  - Updates based on design documentation.

4.2.2.2 SCRUM Integrated into Vee Model Example

In this example, the user need, context diagram and detailed requirements have already been developed as part of the Concept of Operations and System Requirements Stages. The requirements and design have been decomposed (using systems engineering methods of requirements, architecture, and design definition processes) to identify an interface that creates and sends a traveler information message about available parking to commercial vehicle operators. These activities occurred in the Requirements, High Level Design, and Detailed Design Stages. Following the Vee Model, the example is now near the end of the detailed design stage. The interface requirements reference message contents and message content design in a standard (SAE J2735-2016) and are ready to enter the detailed design review. Although the whole system detailed design is usually reviewed at this decision gate, the focus will only be on the Truck Parking Message Generator example. The detailed design review will review the design architecture, how the interface requirements are being fulfilled, and how the requirements will be verified. Once approved, the following happens when including SCRUM as part of the Vee Model:

- SCRUM Begins – Leave Vee Model
  - Software Release Planning (see Figure 2-4):
    - The approved systems engineering Vee Model interface requirements are inserted into the Product Backlog by the Product Owner.
    - The Software Release Planning includes breaking the systems engineering Vee Model interface requirements in increments. Each of these increments has its own Product Backlog (a subsection of the Vee Model interface requirements). Each increment contains one or more Sprint where the increment Product backlog will be further broken down into logical groups. An Increment Team will then develop the product associated with one or more Sprints using the Sprint method described below.
  - Sprint Planning
    - Epics are identified from the Product backlog as part of the Sprint planning meeting. The Epics are broken down further into User Stories and Tasks and become the Sprint Backlog for a specific Sprint cycle. User Stories are either a requirement, a sub-function of a requirement, or a small set of requirements. For example, one story can be requirement 3.5.8.3.6.1 from the NRTM in APPENDIX B. (Default Anchor Point Position) and a second story can be requirements 3.5.8.3.6.3.1 through

3.5.8.3.6.3.2 dealing with describing a circular region presentation for a parking facility.

- Sprint Process (see also Section 2.2)
    - With the Sprint Backlog established the Sprint can begin.
        - Daily Scrum meetings (review what has been done, what is to be done today, issues, and roadblocks)
        - Complete daily work (develop code, create test cases and procedures, test code)
        - Sprint Review meetings (demo features to all, Retrospective on Sprint adjustments if Product Owner and systems engineer accept product, then code documentation is finalized and the Scrum Team proceeds to Sprint Planning or the next Sprint cycle.
        - Sprint cycles occur until all Epics (and their Stories) for the product in the Software Release are implemented. Note: More than one increment may have to be exercised to complete the Software Release and the increments may occur serially or in parallel depending on the resources available (see Figure 2-5).
        - Testing the Software Release happens during the Sprint. In this example it is also unit testing in the Vee Model.

SCRUM Ends – Re-enter Vee Model

- Sub-System Verification Stage Begins:
    - Once unit testing is complete (as part of the Scrum), the unit (Software Release) is integrated with the next higher level element (up to and including subsystem) in the Vee Model. At this point integration and testing is based on Verification and Validation Plans created at the high level design stage. Details of the verification at this level are defined in the system test documentation. All verification will be at the Sub-System requirement level. Documentation required to maintain the product is added in this stage outside of the Sprint process as part of the sub/system integration activity but is considered part of the Scrum.
    - The integration and testing repeats until all units and elements are integrated and shown to work correctly and ready for System Integration.
    - Sub-system test reports, test logs, and anomaly reports are created and anomalies resolved.
    - The integration Retrospective will be conducted outside of the Sprint process based on the integration of the Scrum Software Release product into the sub-system.
- System Integration and Verification Stage:
    - Integration is conducted on all sub-systems (usually one at a time). Testing and integration to continue until all subsystems are working together. All testing will be at System requirement level (captured in the SyRS and the system test documentation).
    - System test reports, test logs, and anomaly reports are created and anomalies resolved.
- System Validation Stage:
    - Acceptance testing begins (based on the test documentation developed by the systems engineers and approved at detailed design review).
    - System test reports, test logs, and anomaly reports are created and anomalies resolved.
    - Design documents, operational documents, and maintenance documents are updated to as-built.

    - Training classes will be conducted either prior to final acceptance or after final acceptance depending on the clients scheduled needs.
- Operations and Maintenance Stage:
    - Once the system is fully accepted then operations and maintenance begins (based on operational and maintenance documents).
    - Updates based on design documentation.

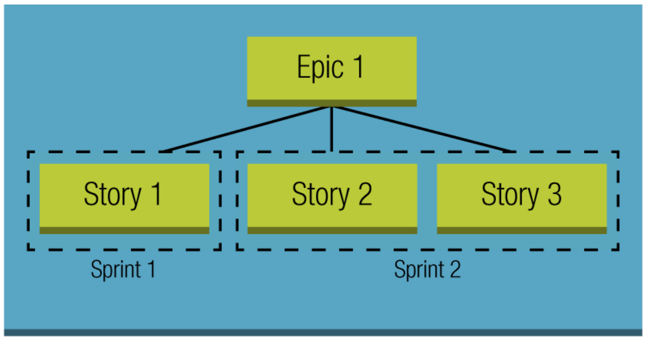


Figure 4-4: Relationship between Epics, Stories, and Sprints (Source: Noblis 2017)

5 Cross-Comparison Between Systems Engineering and Scrum

The purpose of this section is to reinforce the topics below that have been mentioned in other sections of this guide and to highlight the value that can be achieved when combining systems engineering and Scrum. The list of topics was drawn from the list of cross-cutting activities in the FHWA Systems Engineering Guidebook for Intelligent Transportation Systems [6]. These topics are viewed as key areas/activities that the reader should focus on when combining systems engineering and Scrum for ITS development.

Table 5-1: Cross Cutting Topics and Cross-Comparison of Combining Systems Engineering with Scrum

Topic	Systems Engineering Vee Model	Scrum
Stakeholder Involvement	Most ITS deployments comprise representatives from many different agencies. Stakeholders include the planners, users, and agencies who may be the owners, operators and maintainers of the system. Sometimes stakeholders include the public. In systems engineering, maintaining stakeholder engagement and involvement throughout system development is important to ensure the needs of the stakeholders will be met by the delivered system.	When using Agile methods, the difference is focusing on getting real-time to near real-time stakeholder input and feedback on the current software release. High importance is placed on the Product Owner being the voice for all stakeholders. Two of the four values in the Agile Manifesto relate to stakeholders: **Individuals and interactions** over processes and tools; and **Customer collaboration** over contract negotiation. In combining systems engineering with Scrum, stakeholders are engaged at the appropriate level for the different stages of a project. This ensures a more holistic approach for engaging stakeholders in the system development.

Topic	Systems Engineering Vee Model	Scrum
Requirements Gathering	Requirements are the foundation for building a system. They determine WHAT the system must do and drive the system development. Requirements are used to verify if the project team built the system correctly. In systems engineering, requirements are based on the user needs gathered during system concept formulation and confirmed in writing in the Concept of Operations.	Scrum provides methods to support changing requirements based on stakeholder needs. The initial Product Backlog may only include the preliminary requirements for the product and then evolve as additional user needs are identified, requirements change, or technology progresses. Keep in mind that the requirements in the Sprint Backlog are fixed within a Sprint. Combining systems engineering with Scrum provides the project team with an initial set of requirements for the overall system and allows for flexibility to address changing requirements within the Scrum method.
Risk Management	Risk management seeks to understand and avoid the potential cost, schedule, and performance risks to a project. In systems engineering, risk management starts early in the project by identifying the full range of potential risks and continually monitors and manages risks throughout the project. System development in the traditional sense is a single large development that could span multiple years. So, the system owner may only have exposure to the product at major development milestones.	Scrum may reduce risk of developing the wrong product based on frequent releases of useful, smaller capabilities or products. It encourages frequent product reviews by the Product Owner, which can potentially reduce product risk. This translates into lower cost and minimal schedule risks. Systems engineering with Scrum adds defined control gates. That combined with a focus on system-level issues can help to reduce lifecycle schedule and cost risks.

Topic	Systems Engineering Vee Model	Scrum
Progress Metrics	Progress metrics are used to help monitor, recognize, and correct problems as early as possible. To be meaningful, metrics should represent the progress of the project, easy to collect and help make a decision. System engineering uses project management to track cost and schedule. Metrics are managed by the project manager/lead and possibly the systems engineering team. Typically, metrics are not visible to the entire team.	Agile metrics are in-process assessments that are used for insight on progress and to further enhance work on the product. In Scrum, key metrics include: velocity (measurement of the amount of work completed each Sprint), burn-down charts (measurement of completed work in a Sprint over time), and others. These metrics can be used to assess work load and productivity of the Development Team as well as to keep the Product Owner (and stakeholders) informed on the project status. Scrum formalizes tracking team performance and heightens awareness to the entire development team through Agile development-management tools that the team has access to. Systems engineering with Scrum combines the feature of tracking overall project cost and schedule with the detailed insight to the Agile software development metrics.
Configuration Management	Configuration management ensures that project documentation accurately describes and controls the functional and physical characteristics of the end product being developed. Processes are established for maintaining configuration management and control for software and documentation in systems engineering.	Agile methodologies do not prescribe specific configuration management strategies, but they can be easily applied within the Agile framework. Version Control Systems (e.g., like Git for software code) contain features that satisfy configuration management requirements. Content Sharing Tools (like SharePoint or Confluence) contain features to manage documentation (e.g., version number, last edited, historic versions). Systems engineering combined with Scrum uses as the foundation structure for overall system configuration management that incorporates information from Agile tools to provide a complete view into the configuration of the release system.

Topic	Systems Engineering Vee Model	Scrum
Documentation	The Vee Model uses systems engineering documentation to control system development. Some form of documentation is required for each stage of the project lifecycle.	Agile places more emphasis on working software and less emphasis on documentation. Scrum aims to provide essential, valuable, and timely documentation. Examples of Agile artifacts include the product backlog, sprint backlog and User Guides (created after the MVP is done). The documentation provided when combining systems engineering with Scrum is flexible. The documents are determined by the team and what the contract specifies (see Section 2.3). Systems engineering with Scrum combines the comprehensive documentation needed to prove a thorough systems analysis for safety critical systems with documents determined appropriate either by the contract or Agile Development Team.
Training	One of the underlying premises of this document is that team members must be trained and experienced in their respective areas of systems engineering and Agile. APPENDIX A. provides links to several training and subject matter resources for the reader requiring training or a refresher.	Currently, there are limited courses or training available on combining systems engineering with Scrum. Early adopters are encouraged to use this document and provide feedback on its utility and to share lessons learned on their specific application of combining systems engineering with Scrum for ITS development.

Topic	Systems Engineering Vee Model	Scrum
Traceability	The Vee Model uses traceability to ensure that user needs and concepts are addressed by the requirements and that the requirements are fulfilled by the high level and detailed design. Traceability also ensures that system and sub-system requirements are fully verified. Traceability supports impact analysis and configuration management for long-term maintenance, changes and upgrades, and replacement to the system.	Traceability in Scrum is used to document the development of epics, stories, tasks, code, and testing and link them together. The traceability in the combined systems engineering with Scrum approach begins in the Product Backlog, where a subset of the systems engineering requirements are transferred to the Sprint Backlog and are usually captured in the Agile project management tools. It is the responsibility of the Scrum Team to break down the work into epics, stories, tasks, and test cases to populate the Scrum backlog. Any work completed by the Development Team during each iteration of the product traces back to a documented requirement and user need. In the combined systems engineering with Scrum approach, at unit or product testing, the product owner in collaboration with the systems engineer will use the original requirements associated with the product to approve or not approve the results of the testing.

6 Roles and Responsibilities when using Scrum Combined with Systems Engineering

6.1 Introduction

This section of the document describes the roles (people and teams), responsibilities and activities when using a combined Scrum and systems engineering approach to develop ITS projects. Noted in Section 2, Scrum has become one of the most popular Agile methodologies. Other Agile methodologies exist and could be used by State/local agencies however, Scrum is the method being used in this document and is specifically called out in this section.

As part of 23 CFR 940.11, the systems engineering analysis includes identification of participating agency roles and responsibilities for ITS projects receiving Federal funding. The reader is encouraged to review descriptions of roles and responsibilities provided in the two FHWA systems engineering guidebooks, the Systems Engineering Guidebook for Intelligent Transportation Systems [5] (see Section 6) and Systems Engineering for Intelligent Transportation Systems [6] when the traditional systems engineering process is followed.

To develop a system using a combined Scrum and systems engineering approach, the State/local agency will be taking on additional roles, responsibilities, and activities to what is typical in a traditional systems engineering development process. The remainder of this section will provide some details about what additional roles and responsibilities may be needed when using Scrum for ITS projects and when combining Scrum with system engineering, in particular the Vee Model.

Traditional Systems Engineering Roles and Responsibilities

In the traditional systems engineering development for a system comprising software and/or hardware, there are three key roles: System's Owner, Systems Engineering, and Development Team:

- **System Owner**. The System's Owner and State/local transportation agency are responsible for the system and its operations and maintenance. The System Owner usually has the final authority to accept the new or updated system.
- **Systems Engineer**. The Systems Engineer provides system engineering support to the System's Owner. The Systems Engineer can be an in-house staff member or a Systems Engineering consultant with limited or "proxy" authority to act in the best interest of the System Owner. Contracted Systems Engineering services are useful for large complex projects and for smaller agencies that lack in-house Systems Engineer expertise. Note: the title 'Systems Engineer' is used in this document to describe either contracted or in house systems engineering staff that are providing systems engineering expertise on ITS projects.
- **Development Team**. The Development Team while usually a contracted service for system development could be composed of in-house staff. More often, the Development

Team is a system integrator that specializes in developing and integrating systems or components of systems. (these could be proprietary or commercial off the shelf (COTS) products or a combination of both) [5]. The development team may also include product vendors with product integration experience.

Combined Scrum and Systems Engineering Roles and Responsibilities

In a combined Scrum and systems engineering environment, the System Owner, Systems Engineer, and Development Team roles still exist; however, the titles may change as new roles, responsibilities, and activities are introduced. The Systems Engineering and Scrum teams work together in close coordination and collaboration throughout the project/system lifecycle. This is a vital aspect of the combined approach. As will be described in further detail in this section, the Agile systems engineering approach requires additional, frequent meetings, and requires the System Owner to have "on the spot" decision making authority, which is not typical in a traditional ITS systems engineering approach. As noted in Section 3, it is strongly suggested that all team members be trained, skilled and experienced in systems engineering and Scrum methods.

Potential Impacts on Agencies Using Scrum

Agencies need to plan for changes in activities when using Scrum. For example, agencies should anticipate increased participation among project team members. Scrum may also demand additional agency resources (e.g., staff, project management tools) that are not typically required for traditional project development practices. Agencies should be cognizant of the resource intensiveness when using Scrum alone and Scrum combined with systems engineering. Agencies are already familiar with using consultants to complement and extend their existing staff. These additional staff resources may be necessary when using Scrum or Scrum combined with systems engineering. For example, much like agencies contract for a Systems Engineer to support the system engineering processes, agencies may need to contract for Scrum expertise. Agencies implementing Scrum for the first time may want to consider this as an option.

Traditional Scrum software management tools may also be needed to manage project development activities. While you could certainly manage a Scrum process for a very small project with spreadsheet and project schedule programs these tools may not provide the efficiency needed. Tools specific to Scrum integrate all the Scrum activities into a single resource that can and should be made accessible to the entire team. These Scrum software management tools help the Scrum Team plan, track, and manage their progress. When contractors are hired for software development they typically provide and maintain the tools and also give the State/local agency Product Owner access to the tool throughout the development. These tools are helpful for communicating, in real-time, the current state of the development and backlog to the entire team and anyone else (e.g., Manager, Product Owner) that would like to be able to look-in on progress. These tools also provide reports that give the Scrum Team insight into their velocity/progress and areas where they could improve. Scrum Teams should use a tool that facilitates collaboration and communication between team members, but also meets the needs of the ITS Project Manager or stakeholders. Keep in mind that some tools can be complex and have a steep learning curve. See APPENDIX A. for a list and links to Agile and Scrum tools.

6.2 Combined Scrum and Systems Engineering Team Roles and Responsibilities

As noted in Section 3, for agencies just getting started with Agile, the best approach is to start small and keep the effort simple. The key roles and teams for this notion of a “core” combined Scrum and systems engineering effort are described below. Note that a person can be assigned more than one role depending on the project context and agency resources. The only two roles that should be combined are the Systems Engineer and Product Owner. All other roles should be separate.

For projects with a small level of effort, a single Scrum Team should be sufficient. A Release Team may be useful for planning long-term efforts that may start out with only one project. The initial project may be followed by one or more projects either in the short-term or long-term. The Release Team would ensure a comprehensive view of the overall system to be developed or the goal the agency is trying to achieve. The Scrum Team is focused on developing the product. Note that team size should be 5-9 with a maximum of no more than 9. [7] When team size goes beyond nine that is a signal to form multiple Scrum teams. Figure 6-1illustrates a possible core team when using a combined Scrum and systems engineering approach.

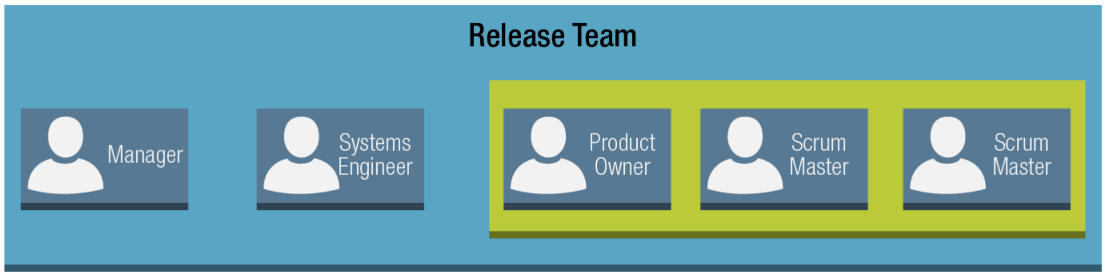


Figure 6-1: Example Core Combined Scrum and SE Team (Source: Noblis 2017)

Suggested core members of a combined Scrum and systems engineering team are described below:

Release Team - The Release Team may include the Manager, Product Owner, Systems Engineer, and Scrum Master(s). The purpose of having a Release Team is to define the product vision and to formulate a high-level plan for the project and releases to the systems engineering team, see Figure 4-1. The key responsibilities of the Release Team include:

- Conducting the Release Planning Meeting and ensuring that the minutes of the meeting are documented/captured and available to the team. During the Release Planning Meeting, team members review the user needs or system requirements (will depend on where project leaves the Vee model) and begin planning for the Sprints.
- Conducting meetings that may include Release Planning, periodic and ad hoc meetings, and Release Retrospectives.

Scrum Team - The Scrum Team consists of the Product Owner, Development Team, and Scrum Master. The maximum number of recommended team members is nine. If more than nine members are needed, then there may need to be multiple Scrum Teams. The key responsibilities of the Scrum Team include:

- Understanding and practicing the theory, values, and rules of Scrum

- Self-organizing within the team to complete the work defined for each Sprint
- Coordinating and collaborating with the Systems Engineer
- Articulating issues within the team and with the Systems Engineer

Manager - The Manager is the State/local agency lead for ITS Programs, ITS projects, or State/local agency District/ITS Operations. The title and position will vary for states and local agencies. The person filling this role has overall responsibility for ITS projects for the agency and is aware of the system to be developed. In addition to the standard responsibilities of a program/project manager, the Manager will also perform additional key responsibilities including:

- Delegating project and decision authority to the Product Owner. Agreements on project delegation and decision authority need to be made with the Product Owner prior to project initiation. Topics might include scope and bounds for the decision authority, understanding the need for clear lines of communication, and that these should be open and used when required to keep the project moving.
- Working with contracting officer to ensure that procurement offers flexibility needed for the combined Scrum and systems engineering development to be successful.
- Obtaining funding for project development, operations, and maintenance.

Systems Engineer - The Systems Engineer in the combined Scrum and systems engineering approach is concerned with the full lifecycle and development processes. In addition to the standard responsibilities of a Systems Engineer, the Systems Engineer will also perform key responsibilities including:

- Supporting the Product Owner while keeping focused on the overall system under development.
- Understanding the combined Scrum with systems engineering approach.
- Following the systems engineering process and collaborating with the Scrum Team to help address their needs. *Using the truck driver HMI example from Section 4, the* Systems Engineer *would provide the Scrum Team with documentation developed as part of the systems engineering process. Documents such as the ConOps, System and subsystem requirements would be provided to the Scrum Team. The Scrum Team uses the requirements to build the Product Backlog.*
- Establishing plans with Product Owner for the combined Scrum and systems engineering approach and process prior to project initiation.
- Participating in systems engineering meetings and Scrum ceremonies. However, the Systems Engineer does not participate in the Daily Scrum and may participate as needed in the Sprint Planning and Retrospectives.
- Communicating project status to the Product Owner.

Product Owner - The System Owner in Scrum terminology is called the Product Owner. Product Owner is the term that is used for Scrum and the combined Scrum and systems engineering approach in this document. In most instances, the Product Owner will be a State or local agency staff member. The key responsibilities of the Product Owner include:

- Representing stakeholders and users for the project or system.
- Understanding Scrum or the combined Scrum and systems engineering approach.
- Understanding the time commitment required of the scrum or combined Scrum and systems engineering approach.

- Defining business value and determining priorities of features (functionalities or capabilities), epics and stories on a regular basis.
- Conveying the vision of the end product to the Scrum Team.
- Coordinating meetings as required and responding to questions.
- Serving as the single decision making authority. Being the person empowered with decision making authority and able to make decisions as necessary at the daily Scrum, bi-weekly Sprint planning and Retrospective meetings, and other related meetings in order to maintain the momentum and tempo of the project. In some cases when resources are limited or constrained, the agency may need to assign a "proxy" Product Owner (see below).
- Maintain (e.g., create, prioritize) the Product Backlog.
- Approve implementation of work as it is completed via end of Sprint demonstrations.
- Establishing agreement with the Manager regarding how to address unsolicited requirements/user stories from superiors and high ranking stakeholders. *Based on feedback and observation in other federal agencies that have used Agile development methods, the Product Owner needs to be prepared to respond to unsolicited requests.* Having an established and agreed to protocol with the appropriate parties prior to the starting the project will go a long way in efficiently and effectively handling requests whether from superiors or out of sequence with the established procedures.
- Coordinating procurement and continuing open communication with Contracts/ Procurement Office.
- Supporting and continuously improving Scrum or the combined Scrum and systems engineering process.
- Facilitating the Sprint Review ceremony.

Proxy Product Owner - The responsibility of the Product Owner could be delegated to a "proxy" Product Owner that represents the agency at meetings with the Development Team. The proxy could be another agency staff member or contracted consultant. The proxy Product Owner is in close contact to inform the agency Product Owner of all decisions made on his/her behalf. The agency Product Owner has final decision authority, yet the proxy should be well versed in the goals, objectives and requirements for the system and be capable of being "the voice" for the agency Product Owner. See above for list of key responsibilities of the Product Owner that the proxy Product Owner may need to assume.

Development Team - The Development Team in Scrum terminology is the team developing the software. The key difference between Agile development and development in a traditional systems engineering approach is the development method is not specified in traditional systems engineering. In a combined approach, the development team uses an Agile methodology, specifically Scrum. The key responsibilities of the Development Team include:

- Understanding and experience in developing software using the Scrum method.
- Communicating and collaborating with the System Engineer and understanding the role the System Engineer has in the project.
- Participating in the Sprint Planning, Sprint Review, Daily Scrum, and Sprint Retrospective ceremonies.

Scrum Master - The Scrum Master is the champion or coach for the Scrum Team. The key responsibilities of the Scrum Master include:

- Understanding and practicing the theory, values, and rules of Scrum.

- Ensuring the Scrum Team follows the process and practice of Scrum.
- Coordinating and collaborating with the Systems Engineer and Product Owner.
- Facilitating the Sprint Retrospective and Daily Scrum ceremonies.

6.3 Expanded Combined Scrum and System Engineering Team Roles and Responsibilities

For complex projects, the agency may decide to develop the system as one single release, or multiple releases. In either case, the software development may involve multiple development teams. Depending on the specific project, these development teams could be conducting work either in serial, concurrently, or a combination of the two. For more complex projects, an expanded Scrum Team is envisioned. A typical scenario for ITS projects could consist of a single release with multiple increments, see Figure 2-4. Figure 6-2 illustrates a potential expanded Agile SE team composition—what is new are the Increment Teams.

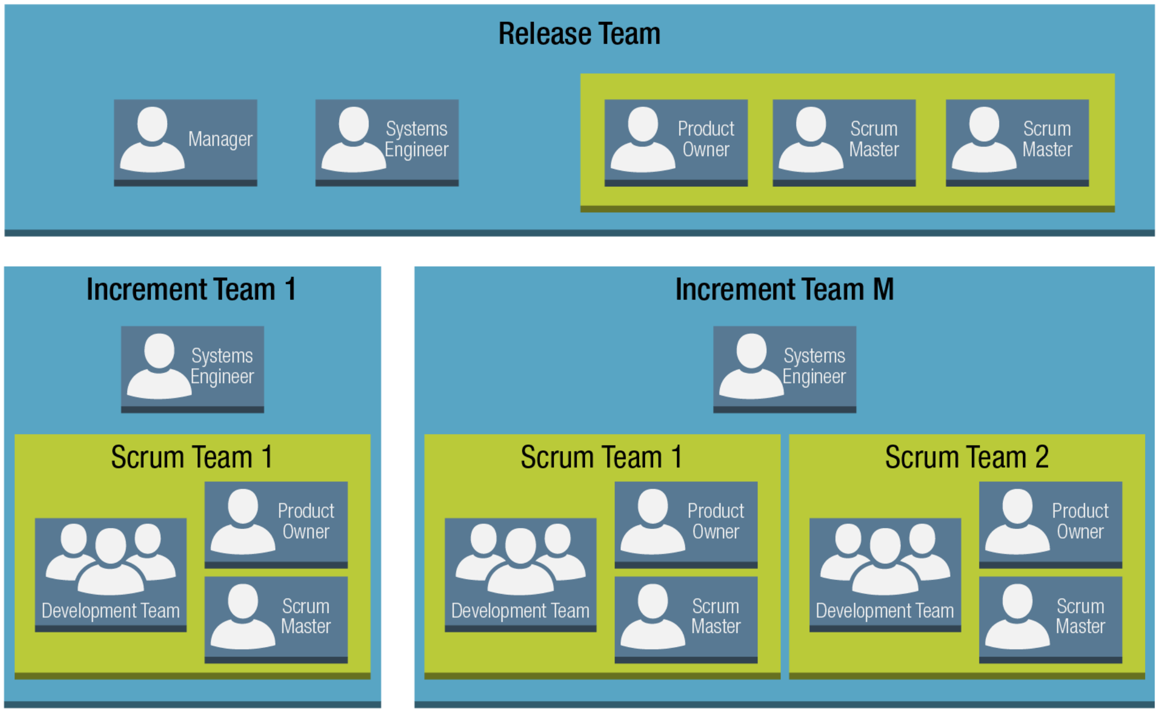


Figure 6-2: Example of an Expanded Combined Scrum with Systems Engineering Team (Source: Noblis 2017)

Release Team - In addition to the responsibilities in the core example above, team members break the user needs or system requirements (will depend on where project leaves the Vee model) into Epics (high-level logical requirements groups or themes for this example) and assigning Epics to Increments. The Epics are then assigned to increment teams. (See the Bike Sharing example in Section 4.2.1.)

Increment Team - The Increment Teams are sub teams within a complex Release Team. An Increment Team could consist of one or more Scrum Teams. The Scrum Teams are usually logically associated and determined through the Release Planning and Increment Planning processes. Far more complex projects could consist of more than one release, which could consist of multiple increments. Because this is *initial* information on Scrum combined with systems engineering for ITS, this document advocates for teams to start small and keep the team structure simple.

The Increment Team coordinates and collaborates with the Integration Team to ensure that the Minimum Viable Product (MVP) provided to the Integration Team has been tested, is ready for integration with other system components, and ready for sub/system integration verification. The **Integration Team** consists of the Systems Engineer, integrators/developers, and testers all of whom support the systems engineering process.

It is important to note that the title of individual staff, teams, and the core and expanded team compositions are not set in stone and can be adapted to meet the needs of the agency and project. However, the core tenets of Scrum still should be practiced.

Roles and responsibilities of the team members are essentially the same as those in the combined Scrum and systems engineering approach described in Section 6.2 with one exception. A member from each Scrum team represents the team at the Scrum of Scrums. A Scrum of Scrums is a stand-up, short meeting to keep people from the organization abreast of important issues. The main purpose is to provide insight into integration issues. The teams have flexibility in assigning their representative. It could be the Scrum Master, a technical member, or a manager. Ideally, the Scrum of Scrums is held daily, or it could be several times a week. The Scrum of Scrums Team develops and maintains their own backlog (e.g., completed items, next steps, and impediments). Resolution of the impediments is about coordination between the teams, understanding and addressing challenges.

Scrum of Scrums is one of many techniques for scaling Agile. It can and should be tailored to meet the organization's needs. Agencies and implementers should not adopt scaling Agile methods until the implementation of Agile or Scrum combined with systems engineering at the team levels has matured.

7 Considerations for Federal Oversight When Applying Scrum Methods

This section of the document provides an assessment of the potential impacts on federal oversight for projects applying Scrum methods. This section only considers those elements of federal oversight that relate to system development and systems engineering as defined in 23 CFR 940.11 Project Implementation. This guidance does not intend to provide a comprehensive review of 23 CFR 940.11.

This section contains three subsections:

- **Overview of 23 CFR 940.11** - Identifies the seven items that comprise a Systems Engineering Analysis.
- **Approach to Assessing the Potential Impacts of Applying Scrum Methods on Federal Oversight** - Describes relation of 23 CFR 940.11 items to the Systems Engineering Vee Model. The Vee model identifies control gates (shown as "documents/approvals"). This section identifies systems engineering documents (e.g. ConOps, System Requirements Specifications, and Test Documentation) typically submitted for review and approval. We assume the federal oversight consists partially on review of systems engineering documents.
- **Summary and Discussion** - Provides a summary of potential impacts for Federal oversight of projects applying Scrum methods.

7.1 Overview of 23 CFR 940.11 Project Implementation

23 CFR 940.11 identifies the conditions that warrant the development of a systems engineering analysis for an ITS Project, and the seven items that together define a systems engineering analysis.

23 CFR 940.11 Project implementation requires:

> "(a) All ITS projects funded with highway trust funds shall be based on a systems engineering analysis.
> (b) The analysis should be on a scale commensurate with the project scope.
> (c) The systems engineering analysis shall include, at a minimum:
>
> > (1) Identification of portions of the regional ITS architecture being implemented (or if a regional ITS architecture does not exist, the applicable portions of the National ITS Architecture);
> > (2) Identification of participating agencies roles and responsibilities;
> > (3) Requirements definitions;
> > (4) Analysis of alternative system configurations and technology options to meet requirements;
> > (5) Procurement options;
> > (6) Identification of applicable ITS standards and testing procedures; and
> > (7) Procedures and resources necessary for operations and management of the system."

7.2 Approach to Assessing the Potential Impacts of Scrum Methods on Federal Oversight

The approach used to assess the impacts of Scrum methods on federal oversight was to:

1. Map the seven 23 CFR 940.11 items to the Vee model;
2. Identify the typical documents and content submitted for federal oversight; and,
3. Determine the potential impacts of applying Scrum methods.

The mapping of the seven 23 CFR 940.11 items to the Vee model is shown in Figure 7-1 below.

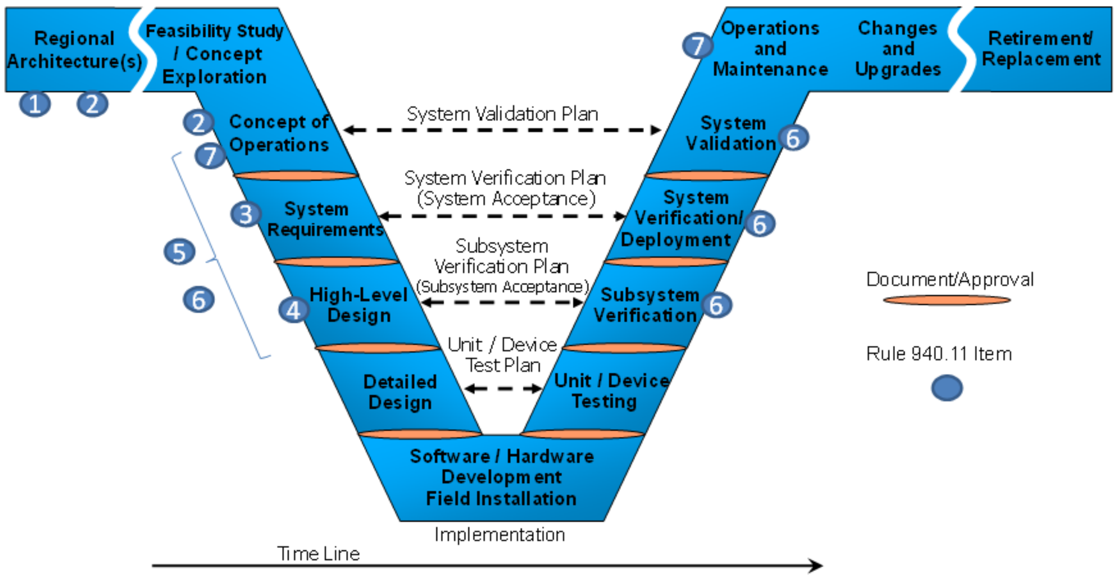


Figure 7-1: Mapping of 23 CFR 940.11 Items to the Vee Model (Source: FHWA 2007 and modified by ConSysTec 2017)

Table 7-1 below identifies systems engineering content that may be requested for federal oversight and compliance with 23 CFR 940.11. An additional column is provided to show traceability of requirements to user needs and test cases. Performing the requirements traceability helps identify gaps, errors, and omissions when advancing from one phase of the system life cycle to the next.

Table 7-1: Potential Systems Engineering Document or Content requested for Federal Oversight

Item	23 CFR 940.11 Title	Potential SE Content Requested for Federal Oversight	Requirements Traceability
1	Portion of Regional ITS Architecture being Implemented	Reference to portion of Regional ITS Architecture	Not applicable

Item	23 CFR 940.11 Title	Potential SE Content Requested for Federal Oversight	Requirements Traceability
2	Participating Agencies Roles and Responsibilities	Concept of Operations – ConOps (Including User Needs)	Requirements are traced to user needs in the NRTM. [See row below.]
3	Requirements Definition	System Requirements Specification (SyRS)	Requirements are traced to User Needs in the Needs to Requirements Traceability Matrix (NRTM)
4	Alternative System Configuration Options (Design Alternatives)	System Architecture Document (SAD)	Requirements to Functional Architecture.
5	Procurement Options	Request for Proposal, Proposal, Contracted Scope of Work	Not applicable
6	Applicable ITS Standards and Testing Procedures	Test Documentation: Test Plan, Test Design Specification, Test Case Specification, Test Procedure Specification, Test Reports.	Requirements are traced to Test Cases in the Requirements to Test Case Traceability Matrix (RTCTM) found in the Test Design Specification. Individual Test Cases are traces to one or more test procedures.
7	Resources Necessary for Operations and Management	Operations Manual, Maintenance Manual	Not applicable

Table 7-2 below shows systems engineering content that may be requested for federal oversight plus an assessment of the potential impacts of Scrum methods on federal oversight.

Table 7-2: Assessment of Potential Impacts of Scrum Methods on Federal Oversight

Item	23 CFR 940.11 Title	Potential SE Content Requested for Federal Oversight	Potential Impact of Scrum Methods on Federal Oversight Review
1	Portion of Regional ITS Architecture being Implemented	Reference to portion of Regional ITS Architecture	No impact
2	Participating Agencies Roles and Responsibilities	Concept of Operations – ConOps (Including User Needs)	ConOps needs to identify which agencies and representatives will have a role as Product Owner.

Item	23 CFR 940.11 Title	Potential SE Content Requested for Federal Oversight	Potential Impact of Scrum Methods on Federal Oversight Review
3	Requirements Definition	System Requirements Specification (SyRS)	More frequent review of smaller set of requirements. Requirements should trace to User Needs to identify portion of system implemented (released). The SyRS includes the system level, subsystem level, and lower levels of requirements.
4	Alternative System Configuration Options (Design Alternatives)	System Architecture Document (SAD)	No impact.
5	Procurements Options	Request for Proposal, Proposal, Contracted Scope of Work	No impact. But, may need a procurement option that supports Scrum methods.
6	Applicable ITS Standards and Testing Procedures	Test Documentation: Test Plan, Test Design Specification, Test Case Specification , Test Procedure Specification , Test Reports.	More frequent review of test results. Test results should reflect verification of a smaller set of requirements contained in product releases. No impact on applicable ITS standards.
7	Resources Necessary for Operations and Management	Operations Manual, Maintenance Manual	No impact.

Note: applicable to Item 3 and Item 4 above, 23 CFR 940.11 does not prescribe software development, detailed design, and implementation methods.

7.3 Summary of Potential Impacts of Scrum Methods on Federal Oversight

Scrum methods introduce the concept of piece-meal system delivery, in portions called releases. In the Scrum methods, for example, requirements are finalized within each Sprint. Impacts on federal oversight include:

- **Requirements Definition** (item 3) - The release-based approach only addresses the requirements associated with a given release. As such, federal oversight may include more frequent reviews of requirements, though the number of requirements reviewed will be a subset of the total requirements necessary to define a system. For each release, the requirements must be evaluated respective to how they relate to the rest of the system as part of the review.

- **Applicable ITS Standards and Testing Procedures** (item 6) - The release-based approach only addresses testing (requirements verification) of those requirements implemented in a release. As such, federal oversight of testing may include more frequent reviews of test reports (test results). There is no impact on identification of Applicable ITS Standards. Testing of how the release relates to the rest of the system will occur as the release is integrated into the rest of the system.
- **Participating Agencies Roles and Responsibilities** (item 2) - This item is impacted due to the additional role of Product Owner that one or more agencies will perform during the system development process. See Section 6 for a more detailed explanation of roles and responsibilities.

8 Procurement Options/Contracting

8.1 Overview

This section provides information to assist agencies developing procurement and contract specifications for projects where use of Agile software development methodologies, including Scrum methods, is an option.

Traditional ITS procurement contracting does not easily support an Agile development process. With traditional ITS procurements, the agency (or its consultant) generally states the requirements for the entire project, with fixed tasks, timelines, and deliverables. Contractors are paid a fixed amount, based on time and materials up to an agreed amount, or based on the satisfactory completion of a deliverable or milestone.

However, Agile development practices require contracts that support short development and delivery timelines, based on the requirements that may evolve as the agency gains experience with system software releases.

Table 8-1 provides a comparison between traditional contracting for ITS, and contracting needs for Agile development. Table 8-1 has been adapted from a guide developed for military contracting of Agile, but has been tailored for the needs of the ITS industry [10].

Table 8-1: Comparison of Contracting Needs

Area	Traditional Contracting Environment	Agile Contracting Needs
Timeline	• Based on best estimates of the completion of the overall project.	• Needs to support short development and delivery timelines. • Needs to identify where partial contract deliverables will be accepted.
Scope	• Functional requirements are locked-in at contract award or at the end of the requirements stage; changes may require contract modifications that may increase cost.	• Contract needs to allow the program to refine requirements throughout the Agile development process • Estimated costs are high when the uncertainty is high, so the contract should allow for prioritizing scope functions within a budget limit. • Contract should request traceability of requirements to contract deliverables to identify portion of project requirements completed by the deliverables completed.
Agency-Contractor Relationship	• Contractor executes the technical solution and reports the progress to the agency.	• The agency and contractor are working together on the development with daily interaction and collaboration. • Agency should consider how to maintain engagement during the course of the project (See Section 6).

Area	Traditional Contracting Environment	Agile Contracting Needs
Contracting Support	• Often centralized and unable to provide rapid turnaround on contract actions.	• Embedded contracting support that can quickly and efficiently execute contract actions.
Technical Evaluation	• Agency proposes the development methodology and contract is awarded based on the technical solution.	• Agency may choose to identify the development process, or be silent on a process. In either case, the contract is awarded based on the strength of the development team's Agile approach and experience with Agile, including Scrum.

Note: it is important to point out that each agency has its own regulations, and as a result procurement methods vary from agency to agency.

The contracting needs identified in Table 8-1 should be considered prior to making a decision to use Agile software methodologies including Scrum. Specifically, if contracting of Scrum methods cannot be supported by the agency's procurement regulations, then perhaps Scrum methods should not be used.

Key issues when considering the selection of Scrum methods include:

- Whether the scope of authority of the project manager, agency contracting manager, and contractor includes revising the scope of work, without requiring a modification to the procurement contract? Modifications to the procurement contract generally require approvals and signatures by authorized individuals from two or more parties and takes time to execute. If for example the product development process (or Scrum) is four weeks, but the time to make the necessary contract modifications is 6 weeks, then the time saving benefits of Scrum may be negated.

- Whether the procurement contract supports short development timelines and undefined deliverables? Issuing a procurement contract without clearly defined deliverables may be new to an agency. Moreover, can procurement support situations where the goals and objectives may be well-defined but the requirements to achieve those goals and objectives are not well-defined? If the requirements are not well-defined, traditional types of procurements results in constant change orders as the project manager and the agency make revisions to requirements that may impact contractors' cost. Change orders (for the purpose of refining requirements during the project contract) may lead to higher cost to the procuring agencies as the contractor makes adjustments and restarts work due to the changing requirements, in addition to any adverse project schedule impacts. If an agency is unable to support a procurement contract with undefined deliverables, then the flexibility and potential benefits of Scrum methods may be negated.

The remainder of this section focuses on understanding the constraints of current traditional ITS procurement processes, and identifies considerations for scrum methods in procurement contracts.

8.2 Procurement Contract Types

This subsection reviews the different types of procurement contracts that are commonly used for the acquisition of Intelligent Transportation Systems (ITS). Most contract vehicles for ITS acquisition can be categorized into two types: product-based and services-based.

With a products-based contract, the procurement involves an end product that is usually well-defined, and well-understood by the procuring agency. The contractor is accountable for the delivery of that product or capability where all the requirements for the product or capability are defined up front. Payment to the contractor is based on the satisfactory delivery of defined products or capabilities. Thus, the product-based contract is practical when the procuring agency can clearly define the final product or capability that is desired.

This is in contrast to an Agile development or Scrum methods, where all of the requirements may not be well-known up front, and where requirements (and requirement priority) may change as the product or capabilities are incrementally delivered. Agencies should state critical constraints and requirements, understanding that supporting requirements may not be well-known up-front. Also, requirements assigned a low level of priority may never be implemented. These characteristics may make a product-based contract impractical for the use of Agile methodologies including Scrum.

In a services-based contract vehicle, the focus is on the delivery of the contractor's expertise as opposed to a defined product. With a services-based contract, the procurement involves acquiring the expertise of the contractor to provide their development expertise AND a software development process, such as Scrum, to support the needs of the agency and/or their stakeholders. Thus, the services-based contract is practical when the procuring agency has a vision for a capability or product that can be clearly expressed, but for which the detailed requirements are not clear. A services-based contract provides the procuring agency flexibility to change or re-prioritize the requirements continuously as needed.

A contrast of product-based and services-based contracting is captured in the table below:

Table 8-2: Comparison of Product-based versus Services-based Procurement Contracting

Aspect for Comparison	Product-based Contracting	Services-based Contracting
Evaluation and selection criteria	Based on the end-product (can the contractor fulfill all the requirements).	Time (number of hours) and expertise of the contractor.
Agency involvement during the project	For product-based, the procuring agency effort is mostly upfront, to develop a procurement specification.	For services-based, the procuring agency effort is continuous throughout the project - managing the development process, participating in reviews and updating requirements and requirements priority.
Emphasis of responsibility	For a product-based, the contractor is responsible for delivering the end-product and capability.	For a services-based, the agency is responsible for managing the development process so that the end-product and capability is attained. The contractor is only responsible for the deliverables for each development cycle (e.g. Sprint) based on the priorities defined by the agency.

The table below summarizes characteristics of the different payment contract vehicles: Fixed-Price, Cost Plus Fixed Fee, and Time & Materials.

Table 8-3: General Characteristics of Various Contract Payment Options/Types

Contract Type	Characteristics
Fixed-Price Services	• Easiest contract type to manage. • Requires a deliverable for payment (e.g., monthly report) unless progress payments are authorized. • Cannot easily change labor mix and number of hours without contract modification.
Cost Plus Fixed Fee (Level of Effort) Contract	• Provides flexibility to change labor mix and hours as long as it does not exceed contract ceiling. • Does not require a deliverable for payment. • Contract ceiling may be difficult to establish based on Agile or Scrum requirements, which can affect upfront fee determination. • The contractor's cost accounting system must comply with acceptability standards. • Agency must monitor the contract for cost control assurance. • Less incentive for contractor to control costs thus risk of a cost growth that could exceed budget or stakeholder commitments.
Time and Material (T&M)	• Provides flexibility to change labor mix and hours as long as it does not exceed contract ceiling. • Does not require a deliverable for payment. • Profit is built into the hourly labor rate so it does not require extensive upfront fee negotiation. • Requires close client monitoring. • Contractor is not incentivized to control costs increasing the risk of a cost growth that could exceed budget or stakeholder commitments.

8.3 Tips and Guidelines for Procurement when Using Scrum or Scrum Combined with Systems Engineering

General

- It is important for the agency to clearly indicate the overall project goals and objectives. What is the vision, the roadmap, and possibly the budget for this project? What are the short term goals/objectives that the contractor should focus on during the initial development process, such as an initial Product Backlog with the priority indicated for each item? What are the constraints for the project? These details about the requirements will help the contractor develop their bid costs as the contractor understands what is important to the procuring agency.
- If Scrum will be used, the procurement specification should identify the expectations. Examples of expectations include:
  - Each development cycle (e.g., Sprint) should incrementally add a capability, and the capabilities will be jointly determined by the agency, the contractor, and the stakeholders.

- Adjustments by the contractor to each development cycle are permitted, but the agency and stakeholders are not allowed to make changes to the planned work during the Sprint. However, there should be some mechanism that allows them to make suggested changes that can then be considered by the contractor.
- A demonstration of the new capabilities is expected after each iteration, where the contractor can present what was finished and the agency and stakeholders provide feedback.
- Allow flexibility on the duration of the development cycle. Recognize that the duration of the development cycle will be decided by the project team (procuring agency, contractor, and other stakeholders). Cost-wise, the effect of the development cycle duration on the cost is probably negligible, but providing the expectation allows the bidding contractor to better understand what the agency has in mind. Ask the bidding contractor for their assumptions for the agile development process as part of the proposal.

Procurement Specification

- The procurement specification should briefly explain why Agile development or Scrum is being considered or required for the procurement. What are the potential issues that the agency is attempting to address by using Scrum as opposed to a traditional development process? Is it because rapid prototyping is desired; or because the client does not have a good vision of specific product (or a single unit) functions and needs to see something tangible to help them decide on said functions?
- If an agency determines it is willing AND able to support Scrum, then the specification should ALLOW for Scrum and mention it as such. Able means that the agency will **COMMIT to actively participate in the scrum methods agreed to between the agency and contractor**. This includes the selection of a knowledgeable agency representative who will **ACTIVELY participate in the meetings, including the Scrum, and will provide feedback in a timely manner**. Such participation may not be necessary for a commercially available system (with or without some customization), but it is more likely and critical for the development of a custom system.
- Cost is always a consideration, for both the procuring agency and contractor. Procurement specifications that are clear and specific as much as possible will help reduce the bid costs. The more unknowns, the more the risk the contractor perceives it must take, and that risk will be reflected in the contractor's bid cost. The total contract cost should be defined by the agency - it can be a Not To Exceed amount that is driven by the level of effort needed to fulfill the requirements for a development cycle; or it can be that the agency and contractor selects what requirements (based on priority) are to be fulfilled by the contractor based on a fixed amount (level of effort) available for the development cycle.
- The procurement specification must provide flexibility to allow the agency to update their requirements as system development progresses, and allow the contractor to propose different processes that they feel will best fulfill those requirements. Thus, the procurement specification should provide specificity when known. This can be accomplished by clearly indicating whether a requirement in the procurement specification is mandatory, desirable, or optional; and whether each requirement is of high, medium or low priority.
- The procurement specification should provide flexibility to allow the agency and contractor to adjust the project milestones and delivery dates. The delivery dates should be coordinated with the development cycle, which may vary from two weeks to a year,

mutually agreed upon by the agency and contractor. Traditional procurement specifications generally identify fixed deliverables from the contractor at specific project milestones, usually based on a pre-determined number of weeks after a Notice To Proceed.

- The procurement specification should authorize the procuring agency's project manager to adjust the needs and requirements of the project within a clearly defined scope. Allowing the project manager to approve technical changes for the procurement within specified limits lessens the need for contracting support to make changes to the procurement specification.

Roles and Responsibilities

The procurement specification should also clearly define the roles and responsibilities for all parties expected to be involved in the development process - or at least provide a baseline. Who is expected to participate in the development process and how often? The following are examples of roles and responsibilities that may be included, however, the procurement specification should also allow the proposer to offer alternatives.

The procuring agency will:

- provide contractual support and mechanisms for Scrum methods

That the qualified and appropriate stakeholders will:

- specify high-level requirements to the contractor;
- work with the contractor to develop user stories and establish acceptance criteria; and
- provide feedback in a timely manner.

The contractor will:

- provide the appropriate expertise to perform the development tasks using Scrum methods;
- work with the stakeholders to develop user stories and establish acceptance criteria;
- demonstrate the new capabilities to the stakeholders and use stakeholder feedback.

Qualifications

- If the procurement specifications REQUIRE scrum, the contractor must be familiar with and experienced with Scrum - that agency does not want the Contractor learning about Scrum methods while working on the project.
- A contractor who has Scrum experience, with a minimum of one project; and uses documented processes.

Project Documentation

- An agency considering Scrum should request project management documentation (e.g., Project Management Plan, and Systems Engineering Management Plan) to identify how the Scrum process will be implemented to fulfill contractual requirements.
- To support a collaborative, transparent development environment, the procuring agency and stakeholders shall have access to the Scrum management and software development tools used by the contractor to develop and maintain the Product Backlog, Sprint (iteration), and backlog, working increment of a capability (software, hardware, and support documentation).
- If Federal money is being used, indicate in the procurement specification that some of the outputs of Scrum will be used to fulfill some of the requirements of 23 CFR 940.11. Indicate that the use of the Agile development process is not a proxy for ignoring the

systems engineering process, such as requirements management, performing testing (verification and validation), and systems engineering documentation development.

9 Recommendations and Next Steps

The ITS community is in the early stages of using Agile methodologies for ITS project development. This document provides some initial suggestions for applying specifically Scrum combined with a systems engineering approach. This concluding section provides a consolidation of recommendations on combining Scrum with systems engineering for a holistic and cost effective approach to developing intelligent transportation systems. The section also provides potential next steps for applying Agile methodologies.

9.1 Recommendations

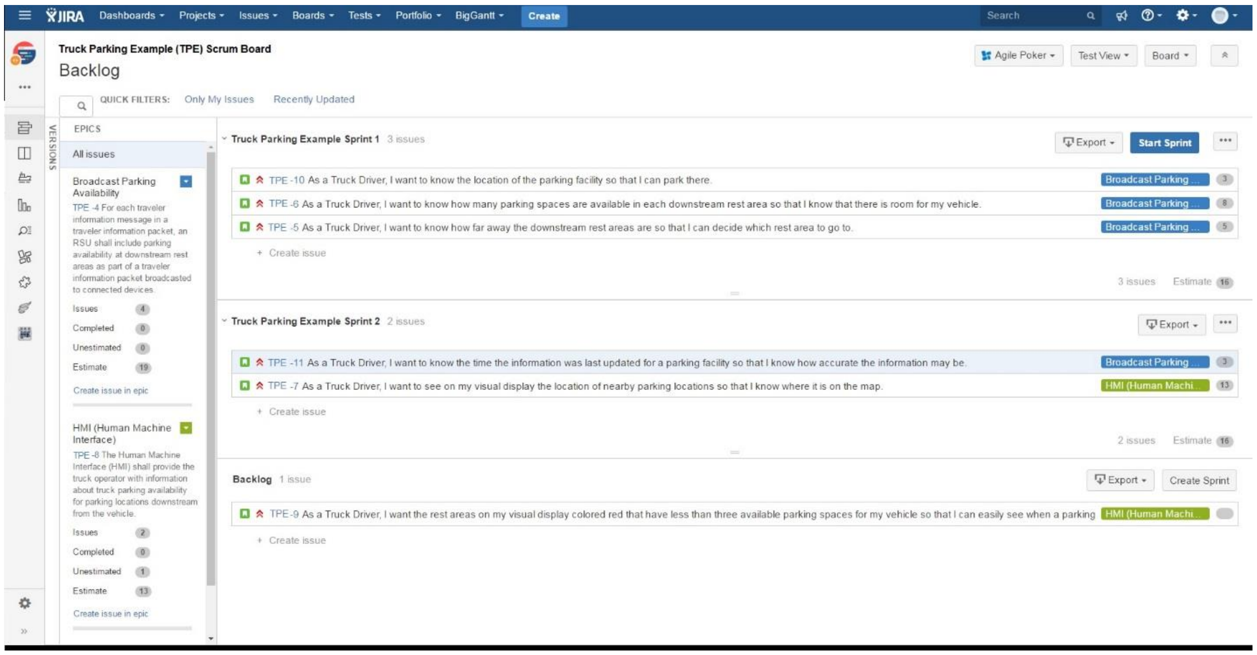
Given where the ITS community is in the progression towards using Agile methodologies, there are few to no examples of ITS deployments to extract key lessons learned. However, the Department of Defense (DoD), NASA, and FAA, as well as other federal agencies, have been combining Agile and system engineering and have documented successful observations, challenges and lessons learned. Below are 12 recommendations for using specifically Scrum with or without systems engineering to develop surface transportation systems. These suggestions are a combination of information extracted from several reports/sources within the previously mentioned agencies [4] [10] [18] [11]. This collection of suggestions was viewed as the most relevant to the ITS community as it explores and practices using Scrum and Scrum combined with systems engineering.

The top 12 suggestions are:

- Don't start project development unless all project staff are trained in systems engineering and Scrum. Project teams have also found that recurrent training on Scrum is useful.
- Don't use Scrum-only on safety-critical applications! Use combined Scrum with systems engineering for non-safety-critical system components. Software development is performed within the Scrum while overall planning, high-level requirements and architecture development, and final product integration are performed following the Systems Engineering process. This combination ensures requirements and design changes are documented, incorporated into the design, and verified and validated. Additionally, the combined methods can improve communication and coordination among requirements, design/implementation, and test teams.
- Begin using Scrum with systems engineering no sooner than when system requirements and high-level design are completed. This helps deployers define what they need and sets the stage for using Scrum methods. Use combined Scrum with systems engineering for software development only.
- ITS Contracts and Finance Specialists need to consider policies to address Scrum in contracts, and receive training on Scrum.
- Start small if you have never used Scrum. Begin with a project that requires a reduced level of effort (e.g., simple project, small Scrum team). This will enable you to become familiar and comfortable with the application of Scrum in an environment with hopefully fewer risks.
- Have a Scrum advocate. Adoption of Scrum on systems engineering projects can be slow due to an overall resistance to change. Having a Scrum advocate at the leadership/management level can foster collaboration and team-building between the Scrum

Team and systems engineers as well as support cultural change within the organization/agency. The absence of an advocate may diminish the potential for Scrum adoption and cross-team collaboration.

- Use Scrum to increase opportunities for communication among all project team members (stakeholders, agency, the system engineering, software development, and testing teams).
- Employ Scrum management and software development tools that facilitate real-time access to project status (see APPENDIX A.) for a list of the most popular Scrum project management tools and Figure 9-1 below for a screen shot of a tool sample using the truck parking example mentioned in Section 4).
- Use periodic “Scrum of Scrums” meetings on large programs to optimize communications and information sharing across project teams and the greater organization.
- Don’t overload Your Scrum masters by asking them to facilitate more than two teams.
- Avoid continuously deferring requirements or other items in the Product Backlog. When the development team determines it cannot complete all of the requirements in a Sprint, the requirements are moved to the next Sprint. To avoid an overload of incomplete requirements, use one of the later sprints in the increment as a “catch up” Sprint. No new requirements are added to the Sprint. Schedule and costs are kept in check.
- Frequent software demonstrations at the end of sprints leads to stakeholder confidence in product delivery commitments (quality, cost, and schedule) made by the software teams.



Truck Parking Example (TPE) Scrum Board

Backlog

Broadcast Parking Availability
TPE -4 For each traveler information message in a traveler information packet, an RSU shall include parking availability at downstream rest areas as part of a traveler information packet broadcasted to connected devices.
Issues 4
Completed 0
Unestimated 0
Estimate 19

HMI (Human Machine Interface)
TPE -8 The Human Machine Interface (HMI) shall provide the truck operator with information about truck parking availability for parking locations downstream from the vehicle.
Issues 2
Completed 0
Unestimated 1
Estimate 13

Truck Parking Example Sprint 1 3 issues

TPE -10 As a Truck Driver, I want to know the location of the parking facility so that I can park there.	Broadcast Parking ...	3
TPE -6 As a Truck Driver, I want to know how many parking spaces are available in each downstream rest area so that I know that there is room for my vehicle.	Broadcast Parking ...	8
TPE -5 As a Truck Driver, I want to know how far away the downstream rest areas are so that I can decide which rest area to go to.	Broadcast Parking ...	5

3 issues Estimate 16

Truck Parking Example Sprint 2 2 issues

TPE -11 As a Truck Driver, I want to know the time the information was last updated for a parking facility so that I know how accurate the information may be.	Broadcast Parking ...	3
TPE -7 As a Truck Driver, I want to see on my visual display the location of nearby parking locations so that I know where it is on the map.	HMI (Human Machi...	13

2 issues Estimate 16

Backlog 1 issue

TPE-9 As a Truck Driver, I want the rest areas on my visual display colored red that have less than three available parking spaces for my vehicle so that I can easily see when a parking	HMI (Human Machi...	

Figure 9-1: Scrum Project Management Tool Sample Screen Shot (Source: Noblis 2017)

Until sufficient experience and exposure to Scrum or other Agile methodologies is gained in ITS, FHWA should continue to review processes, observations and challenges from other domains and apply the lessons learned to specific project development needs.

9.2 Next Steps

This document was initiated by FHWA to provide a concerted response to the questions being asked about the use of Agile methodologies and specifically Scrum on ITS projects. This is in recognition that 1) Scrum methods are being used on ITS project developments and likely will continue, 2) there is a need for assistance on how to incorporated Scrum with traditional systems engineering for ITS project execution and oversight within the framework of 23 CFR 940.11, and 3) there is a need for this information to be shared with State and local transportation agencies.

Once the ITS community has more familiarity with, and gains deeper understanding of combining Scrum with the System Engineering, this document may be revised and updated to reflect the actual experiences and incorporate feedback from the State/local transportation agencies, FHWA Division Offices, FHWA Resource Centers, and contractors/consultants. After the above parties have had exposure and experience building and deploying ITS using the combined systems engineering and Scrum approach, the information can be refined. The revised document may include specific examples from the field and augmented processes to reflect actual stakeholder experience. Over the next few years, effort could be taken to collect feedback from agencies applying the combined Scrum and systems engineering (in particular the Vee Model) approach and document that feedback in an updated version of this document. FHWA may consider a specific outreach effort that includes tasks such as:

- Collecting feedback (what worked and what did not work).
- Documenting experiences and findings from ITS and advanced transportation deployments that followed the Scrum, and combined Scrum and systems engineering approach.
- Identifying specific gaps that exist between the current resource and field experiences with Scrum, and combined Scrum and systems engineering approach.
- Identifying quantifiable and qualitative metrics that show the benefits of Scrum, and combined Scrum and systems engineering approach. [18]
- Validating the Scrum, and combined Scrum and systems engineering approach through a use case study with non-safety-critical programs [18]. This could also be done in coordination with the JPO's Scrum-related initiatives.

The goal would be an updated resource that meets the needs of the deployers, and is specific to the ITS community.

References

1. Defining Auto Safety of Life in 5.9 GHz, June 8, 2016. https://www.fcc.gov/news-events/blog/2016/06/08/defining-auto-safety-life-59-ghz.
2. U.S. Government Accountability Office, "Effective Practices and Federal Challenges in Applying Agile Methods", July 27, 2012. http://www.gao.gov/products/GAO-12-681.
3. U.S. Department of Transportation. Federal Highway Administration. "Part 940 – Intelligent Transportation System Architecture and Standards," January 31, 2002. http://www.fhwa.dot.gov/legsregs/directives/fapg/cfr0940.htm.
4. Wrubel, E. et al. "Agile Software Teams: How They Engage with Systems Engineering on DoD Acquisition Systems." Software Engineering Institute, Carnegie Mellon University, Technical Note CMU/SEI-2014-TN-013
5. U.S. Department of Transportation. Federal Highway Administration. "Systems Engineering Guidebook for Intelligent Transportation Systems, Version 3.0," November 21, 2009. https://www.fhwa.dot.gov/cadiv/segb/files/segbversion3.pdf.
6. U.S. Department of Transportation. Federal Highway Administration, Federal Transit Administration. "Systems Engineering for Intelligent Transportation Systems: An Introduction for Transportation Professionals," January 2007. http://ops.fhwa.dot.gov/publications/seitsguide/seguide.pdf.
7. Schwaber, K., Sutherland, J. "The Scrum Guide." July 2013. http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf
8. *Agile Alliance*. Agile Alliance, 2015, https://www.agilealliance.org/.
9. Kennedy, M. "Case Study: Applying Agile Software Methods to Systems Engineering." The Journal of Cyber Security & Information Systems. Volume 1, No. 3, June 2013: pp.12-21. https://www.csiac.org/journal-article/case-study-applying-agile-software-methods-to-systems-engineering/
10. Modigliani P, Chang, S. "Defense Agile Acquisition Guide: Tailoring DoD IT Acquisition Program Structures and Processes to Rapidly Deliver Capabilities." The MITRE Corporation, March 2014.
11. Office of Safety & Mission Assurance (OSMA). "*Agile Development Brings New Challenges for Software Assurance at NASA*." Science and Technology, Software Assurance. September 4, 2014. https://sma.nasa.gov/news/articles/newsitem/2014/09/04/agile-development-brings-new-challenges-for-software-assurance-at-nasa
12. SE Handbook Working Group, INCOSE. "Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities." Version 3.2.2, October 2011.
13. Scrum Alliance. Scrum Alliance, Inc. 2016, https://www.scrumalliance.org/
14. Radigan, D. (n.d.). Kanban The Agile Coach. Retrieved February 10, 2017, from https://www.atlassian.com/agile/kanban

15. Wells, D. Extreme Programming: A gentle introduction. Retrieved February 10, 2017, from http://www.extremeprogramming.org/

16. Scrum of Scrums, Agile Alliance. Retrieved February 10, 2017, from https://www.agilealliance.org/glossary/scrum-of-scrums/

17. [PSInternalTraining]. (2013, June 8). *Scrum Training - Crash Course - 2013-06-18.* [Video file]. Retrieved from https://www.youtube.com/watch?v=wNwfFStmtw8.

18. Subowo, N. "Towards an Agile Systems Engineering Approach for the National Airspace System," System Engineering Conference in DC (SEDC), April 21, 2014. http://www.sedcconference.org/towards-an-agile-systems-engineering-approach-for-the-national-airspace-system/.

19. SAE J3067-201408 Candidate Improvements to Dedicated Short Range Communications (DSRC) Message Set Dictionary [SAE J2735] Using Systems Engineering Methods; issues 2014-08-26; http://standards.sae.org/j3067_201408/

20. Agile Alliance. Agile Manifesto, Agile Alliance, 2015, The Twelve Principles of Agile – https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

21. Sharma, A. Essential, Valuable, Timely Documentation. December 20, 2013. https://www.scrumalliance.org/community/articles/2013/december/essential-valuable-timely-documentation

APPENDIX A. Resources and Agile Project Management Tools

Systems Engineering Resources:

The FHWA has several resources available for persons interested in learning about systems engineering.

The Consortium for ITS Training and Education (CITE) through a partnership with the ITS Joint Program Office Professional Capacity Building (PCB) Program offers free web-based and blended courses including:

- For an **introduction** to get started on systems engineering concepts - Introduction to Systems Engineering (Blended)
- For more **advanced** and detailed understanding of systems engineering and system integration - Advanced Systems Engineering for Advanced Transportation Projects (Blended)

The PCB Program offers the ITS ePrimer, an online resource that provides transportation professionals with fundamental concepts and practices related to ITS technologies. Module 2 of the ePrimer, Systems Engineering, provides an overview of the systems engineering process.

FHWA developed the following two systems engineering guidance documents:

- Systems Engineering for Intelligent Transportation Systems - Provides an introduction to systems engineering and leads the reader step by step through the project life cycle ad describes the systems engineering approach at each step.
- Systems Engineering Guidebook for Intelligent Transportation Systems - provides a more in-depth reference for ITS practitioners applying systems engineering to plan, implement, manage, and operate ITS.

Agile Resources:

- The Agile Manifesto – https://www.agilealliance.org/agile101/the-agile-manifesto/
- The Twelve Principles of Agile – https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

Scrum Resources:

- The Scrum Guide by Ken Schwaber and Jeff Sutherland available online at http://www.scrumguides.org/scrum-guide.html or in PDF format at http://www.scrumguides.org/download.html
- Introduction to Agile – 7 minutes (video) https://www.youtube.com/watch?v=9TycLR0TqFA

Most Popular Agile Project Management Tools

A good resource for information on Agile Project Management Tools is the website at: https://project-management-software.financesonline.com/c/agile-project-management#products. This source provides information about these tools, features offered, and how to determine which tool is best for your project. Some popular Agile Project Management Tools are listed below.

- JIRA – https://www.atlassian.com/software/jira
- VersionOne – https://www.versionone.com/
- Wrike – https://www.wrike.com/
- dapulse – https://dapulse.com/
- Zoho Projects – https://www.zoho.com/projects/
- Nutcache – http://www.nutcache.com/
- Asana – https://asana.com/
- Projectplace – https://www.projectplace.com/
- Basecamp – http://basecmp.com/
- Planview – http://www.planview.com/

APPENDIX B. Truck Parking Availability Examples

Example and its Purpose

This Truck Parking example was selected for use in this document to help readers understand how to apply the concepts of Scrum with the Systems Engineering Process. This example was taken from the SAE J3067 Information Report and covers use of the ITS standard for building the message that provides truck parking information from a transportation management center. The predecessor to SAE J3067 was called the J2735 Systems Engineering Document. The J2735 Systems Engineering Document was developed by USDOT to help the connected vehicle industry understand how to develop standards using the systems engineering process and to capture the operational needs for the set of capabilities identified within the document. SAE later published the document as the J3067. The purpose of this example provides the framework to show how a Vee Model only approach would be conducted and then how a Scrum integrated with a Vee Model approach would be conducted to develop the message building module.

The example contains:

1. The Scenario that includes the parking information for heavy trucks (Commercial Vehicle Operations)
2. User need of parking information for heavy trucks
3. Requirements associated with building the parking information message sent to operators and fleet managers

Use the Example For:

The example comes as a product from using the systems engineering process to develop the scenario, user needs, and requirements. This example was used in this document to:

1. Show how to develop the example using strictly the Vee model; then
2. Show where using Scrum fits into the Systems Engineering Process (specifically the Vee Model);
3. Show how to develop using Scrum stories and other Scrum activities.
4. Demonstrate the concept of traceability and how it applies when using the Vee Model and when using Scrum within the Vee Model.

Other sections use this example to help explain concepts being introduced.

Scenario

The following scenario was taken from the J2735 Systems Engineering Document (predecessor of the J3067). The Scenario provides a high level description of the operation, who is involved, and what the involved parties are expecting or providing. The highlighted (and underlined) sections are specifically identifying the parking information needed in this scenario. Also provided is a context diagram showing the entities and paths of communication. Only the parking information provided to the operators is shown in this example.

4.9.2.2.8 Special Cases – Commercial Vehicle Operations

Table B-1: Special Cases – Commercial Vehicle Operations

Category Name:	Commercial Vehicle Operations
Description:	Describes the information that flows between a commercial vehicle and transportation regulatory agencies or transportation agencies. The types of information flows may support: • Credentialing and permitting; and • Safety inspection. These two applications are currently implemented with the vehicle sending identification to the roadside where the processing takes place in the infrastructure Vehicle clearance - I2V providing a message to the vehicle to proceed; and Infrastructure measured vehicle data (weight, size, and height) (I2V providing e.g. weight to the vehicle for later transmission to another roadside unit)
Alternative Operational Scenarios:	Parking space availability and service information (I2V)
Needs Addressed:	2.5.3.4.1 Commercial Vehicle Credentialing and Permitting 2.5.3.4.2 Safety Inspection Data 2.5.3.4.3 Vehicle Clearance 2.5.3.4.4 Receive Parking Space Availability and Service Information

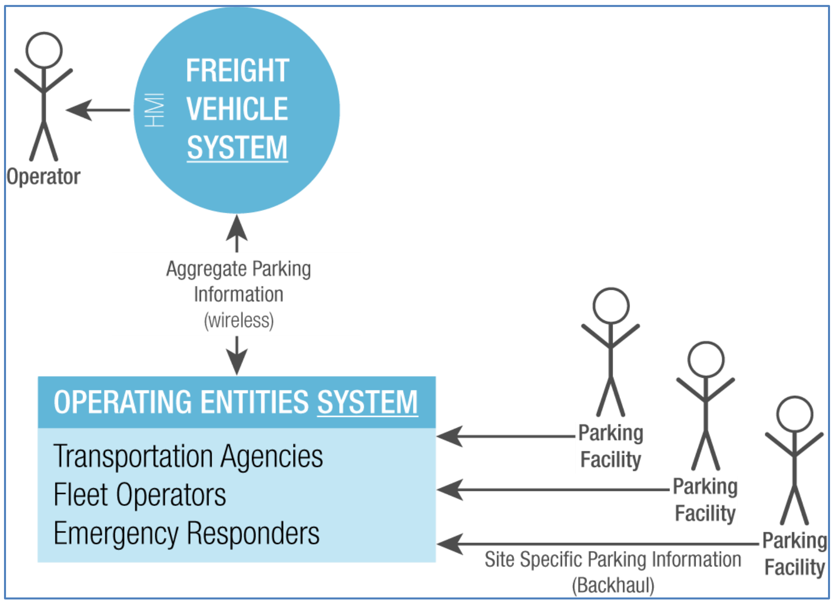


Figure B- 1: Truck Parking Availability Context Diagram (Source: Noblis 2017)

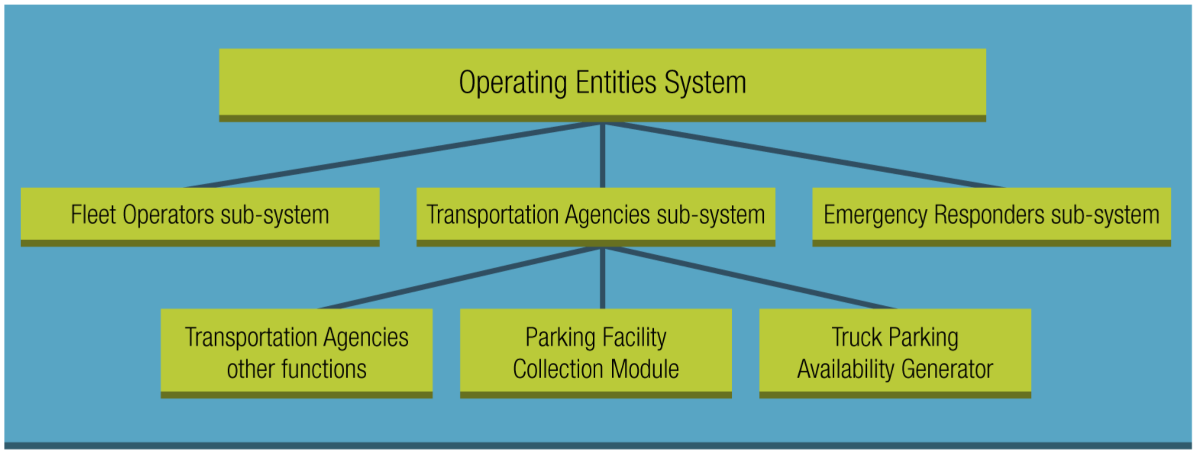

Operating Entities System
Fleet Operators sub-system
Transportation Agencies sub-system
Emergency Responders sub-system
Transportation Agencies other functions
Parking Facility Collection Module
Truck Parking Availability Generator

Figure B- 2: Operating Entities Functional Diagram (Source: Noblis 2017)

User Need

The user need, provided below, identifies the major capability and provides a justification for this major capability.

User Need 2.5.3.4.4: Receive Parking Space Availability and Service Information

Commercial vehicle drivers need the availability of parking spaces and services at upcoming parking lots or rest area facilities. This capability provides commercial vehicle drivers with information about the availability of parking and services information, including availability times and parking space dimensions. This capability has a safety aspect to it since commercial vehicle drivers that exceed hours of service requirements represent a safety hazard on the roads.

Requirements for Example 1 (NRTM for User Need 2.5.3.4.4)

The Needs to Requirements Traceability Matrix (NRTM) traces those requirements associated with satisfying the user need.

Table B- 2: Requirements for User Need 2.5.3.4.4

User Need ID	User Need	FR ID	Requirements	Conformance	Support	Additional Specifications
2.5.3.4.4	Receive Parking Space Availability and Service Information			RSU:O	Yes / No / NA	
		3.5.6.4.1	Broadcast Location Corrections Detail - NMEA **(Location)**	O	Yes / No	
		3.5.6.4.2	Broadcast Location Corrections Detail - RTCM **(Location)**	O	Yes / No	
		3.5.8.1	Broadcast Traveler Information	M	Yes	
		3.5.8.2.1	Broadcast Traveler Information - Packet Identifier	M	Yes	
		3.5.8.2.2.1	Broadcast Traveler Advisories - Message Identifier	M	Yes	
		3.5.8.3.2	Broadcast Traveler Information - Start Time **(ValidTime)**	O	Yes / No	
		3.5.8.3.3	Broadcast Traveler Information - Start Year	ValidTime:O	Yes / No / NA	
		3.5.8.3.4	Broadcast Traveler Information - Validity Duration	ValidTime:M	Yes / NA	
		3.5.8.3.5	Broadcast Traveler Information - Importance	O	Yes / No	
		3.5.8.3.6	Broadcast Traveler Information - Presentation Requirements **(Presentation)**	O	Yes / No	
		3.5.8.3.6.1	Broadcast Traveler Information - Default Anchor Point Position	Presentation:M	Yes / NA	
		3.5.8.3.6.2	Broadcast Traveler Information - Heading Slice	Presentation:O	Yes / No / NA	
		3.5.8.3.6.3.1	Broadcast Traveler Information - Circular Region - Radius **(Circular)**	Presentation:O.4(1..*)	Yes / No / NA	
		3.5.8.3.6.3.2	Broadcast Traveler Information - Circular Region - Anchor Point	Circular:O	Yes / No / NA	
		3.5.8.3.6.4.1	Broadcast Traveler Information - Polygon Region - Offsets **(Polygon)**	Presentation:O.4(1..*)	Yes / No / NA	
		3.5.8.3.6.4.2	Broadcast Traveler Information - Polygon Region - Anchor Point	Polygon:O	Yes / No / NA	
		3.5.8.3.6.5.1	Broadcast Traveler Information - Shape Point Set - Default Direction **(Shape)**	Presentation:O.4(1..*)	Yes / No / NA	
		3.5.8.3.6.5.2	Broadcast Traveler Information - Shape Point Set - Default Width	Shape:M	Yes / NA	

User Need ID	User Need	FR ID	Requirements	Conformance	Support	Additional Specifications
		3.5.8.3.6.5.3	Broadcast Traveler Information - Shape Point Set - Offsets	Shape:M	Yes / NA	
		3.5.8.3.6.5.4	Broadcast Traveler Information - Shape Point Set - Direction	Shape:O	Yes / No / NA	
		3.5.8.3.6.5.5	Broadcast Traveler Information - Shape Point Set - Width	Shape:O	Yes / No / NA	
		3.5.8.3.6.5.6	Broadcast Traveler Information - Shape Point Set - Node Width	Shape:O	Yes / No / NA	
		3.5.8.3.6.5.7	Broadcast Traveler Information - Shape Point Set - Anchor Point	Shape:O	Yes / No / NA	
		3.5.8.3.7	Broadcast Traveler Advisories - Content	M	Yes	
		3.5.8.3.9	Broadcast Traveler Information - Uniform Resource Locator	O	Yes / No	
		3.5.8.3.10	Broadcast Traveler Information - Valid Vehicle Type	O	Yes / No	
		3.5.8.3.11.1	Broadcast Parking Availability - Mandatory Requirements	M	Yes	
		3.5.8.3.11.2.1	Broadcast Parking Availability - Location Description	O	Yes / No	
		3.5.8.3.11.2.2	Broadcast Parking Availability - Availability Time	O	Yes / No	
		3.5.8.3.11.2.3	Broadcast Parking Availability - Availability End Time	O	Yes / No	
		G.2.9.1	Minimum Transmission Rate - Location Correction Details Broadcasts	Location:M	Yes / NA	
		G.2.9.2	Default Transmission Rate - Location Correction Details Broadcasts	Location:M	Yes / NA	The default message transmission rate to broadcast location correction details information is once per ___ (default: 1000) milliseconds.
		G.2.11.1	Maximum Transmission Rate - Broadcast Traveler Information	M	Yes	

User Need ID	User Need	FR ID	Requirements	Conformance	Support	Additional Specifications
		G.2.11.2	Default Transmission Rate - Broadcast Traveler Information	M	Yes	The default message transmission rate to broadcast traveler information messages is once per ___ (default: 1000) milliseconds.

The following requirements are to be used to develop a message building handler to provide parking information to the commercial vehicles. Some of these requirements may be used to satisfy other user needs and as such were written to satisfy multiple user needs. The NRTM specifies what is needed to be conformant to the standard for this user need.

3.5.6.4 Broadcast Location Correction Details

Connected devices need information to improve its positional accuracy estimate. These differential GPS corrections allows a mobile GPS receiver, such as a GPS system in a connected vehicle, to achieve a greater absolute positional accuracy, compensating for errors that exist in satellite positioning. GPS receivers use the differential GPS corrections to improve the accuracy of their positioning, by comparing its position with the RSU's location, which is known.

The detailed requirements for transmitting differential GPS corrections are as follows.

3.5.6.4.1 Broadcast Location Corrections Detail - NMEA

The RSU shall broadcast National Marine Electronics Association (NMEA) 0183 differential GPS correction messages to connected devices. NMEA 0183 is a standard that defines the interface between two different devices, generally marine devices. NMEA 0183 includes support for a GPS receiver to provide its real time position information, such as position, velocity and time as computed by the GPS receiver.

3.5.6.4.2 Broadcast Location Corrections Detail - RTCM

The RSU shall broadcast Radio Technical Commission for Maritime Services (RTCM) messages, as defined by RTCM special committee number 104, to connected devices. These standards support very high accuracy navigation and positioning through a broadcast to mobile Global Navigation Satellite System (GNSS) receivers, which allows the receivers to compensate for errors that exist in satellite positioning without augmentation.

3.5.8 Traveler Information Requirements

Traveler information is used to provide connected devices with travel advisories and information. Traveler information can be customized for travelers in a specific location, traveling in a specific direction, or at specific times. Examples of traveler information include traffic information, traffic incidents, major events, evacuations, and road signs.

The detailed requirements for providing traveler information from the infrastructure to connected devices are as follows.

3.5.8.1 Broadcast Traveler Information

An RSU shall broadcast a packet containing traveler information to connected devices. Each packet may contain one or more individual traveler information messages. Multiple traveler information messages may be packaged into a single traveler information packet for transmission efficiency.

3.5.8.2 Broadcast Traveler Information - Mandatory Requirements

The following are the minimum requirements for an RSU to broadcast traveler information to connected devices.

3.5.8.2.1 Broadcast Traveler Information - Packet Identifier

An RSU shall include a packet identifier for the traveler information packet broadcasted to connected devices. A change in the packet identifier indicates a change in the packet content. Receivers of the packet can ignore subsequent packets with the same packet identifier, otherwise the receiver should parse the packet contents.

3.5.8.2.2 Broadcast Traveler Information - Message Identifier Requirements

For each traveler information message in a traveler information packet, an RSU needs to identify each message transmitted as part of a traveler information packet broadcasted to connected devices. This requirement allows a receiving connected device to ignore (not process) a traveler information message that has already previously received. The identifier for each message is dependent on the type of traveler information.

Two types of traveler information are supported - traveler advisories and road sign messages. Traveler advisories are temporary in nature, that is, the information being broadcasted are finite in duration. Examples of traveler advisories include traffic information, traffic incidents, major events, evacuations, etc… Road sign messages are static in nature and generally emulate the message on a physical roadside sign.

3.5.8.2.2.1 Broadcast Traveler Advisories - Message Identifier

For traveler advisories, an RSU shall include a message identifier for each traveler advisory message as part of a traveler information packet broadcasted to connected devices. This message identifier is expected to be assigned by a transportation agency. It is suggested that the message identifier should be the same as the identifier assigned to that event (e.g., icing conditions) or incident that results in the need for the traveler advisory message, creating a relationship between the traveler advisory message to the event or incident. This assumes that the transportation agency has an incident or event tracking system that assigns an identifier to each incident or event.

3.5.8.3.2 Broadcast Traveler Information - Start Time

For each traveler information message in a traveler information packet, an RSU shall include the start time that the message becomes valid as part of a traveler information packet broadcasted to connected devices. The start time is measured in minute of the year, in one minute units. If a start year is included, then the start time is the minutes of the included start year, otherwise the start time is the minute of the current year. This requirement allows an agency to preload traveler information messages into an RSU, then make that message valid at a certain time, such as warning travelers about upcoming roadway construction only after construction has begun. If no start time is broadcasted as part of the traveler information message, the traveler information message is immediately valid for the valid region.

3.5.8.3.3 Broadcast Traveler Information – Start Year

For each traveler information message in a traveler information packet, an RSU shall transmit the start year along with the start time that the message becomes valid as part of a traveler information packet sent to a connected device. If a start year is also transmitted as part of the traveler information message broadcasted, then the start time is the minute within this start year that the traveler information message becomes valid.

3.5.8.3.4 Broadcast Traveler Information - Validity DurationFor each traveler information message in a traveler information packet, an RSU shall include the duration from the start time that the traveler message is valid for as part of a traveler information packet broadcasted to connected devices. The duration is measured in one minute units.

3.5.8.3.5 Broadcast Traveler Information - Importance

For each traveler information message in a traveler information packet, an RSU shall include the importance of the message relative to other traveler information messages being broadcasted as part of a traveler information packet broadcasted to connected devices. Values shall be from 0 to 7, with 0 being least important and 7 being most important. The selection of importance will be made by the agency broadcasting the messages. How a connected device presents two messages with the same importance is outside the scope of this standard.

3.5.8.3.6. Broadcast Traveler Information - Presentation Requirements

Agencies may need to present traveler information messages only to specific travelers, such as travelers within specific geographic (spatial) regions or a direction of travel. Multiple valid regions and directions of travel may be defined for each traveler information message. The detailed requirements for defining the conditions when the traveler information content shall be presented to the driver are as follows.

3.5.8.3.6.1 Broadcast Traveler Information - Default Anchor Point Position

For each traveler information message in a traveler information packet, an RSU shall include the geographic location (latitude, longitude, elevation) of the default anchor point for which valid regions are determined as part of a traveler information packet broadcasted to connected devices. The latitude and longitude are measured in units of 1/10th microdegree. The elevation represents the height above or below the WGS-84 reference ellipsoid in units of 1 decimeter. The default anchor point is included so an anchor point does not have to be broadcasted for each valid region defined for a traveler information message.

3.5.8.3.6.2 Broadcast Traveler Information - Heading Slice

For each traveler information message in a traveler information packet, an RSU shall include the direction of motion (of the connected device) that the message is valid for as part of a traveler information packet broadcasted to connected devices. The connected device's direction is measured by one or more heading slices, with each heading slice 22.5 degrees wide. This requirement allows traveler information messages to be valid for only those connected devices traveling in a specific direction. For example, a weather message may apply to all directions, while an incident message may be applicable for travelers heading towards the location of an incident.

3.5.8.3.6.3 Broadcast Traveler Information - Circular Valid Region Requirements

A spatial region for which a traveler information message is valid for may be a circular region around an anchor point. The connected device should be located within the circular region for the traveler information message to be presented to the traveler. The detailed requirements for defining the circular region where the traveler information content should be presented to the traveler are as follows.

3.5.8.3.6.3.1 Broadcast Traveler Information - Circular Region - RadiusFor each traveler information message in a traveler information packet, an RSU shall include the radius for the circular region defining where the traveler information message is valid for as part of a traveler information packet broadcasted to connected devices. The valid circular region is defined by the radius, measured in one meter units, from the specified anchor point defined in Section 3.5.8.3.6.3.2. If that specified anchor point is not broadcasted, the default anchor point defined in Section 3.5.8.3.6.1 is used.

3.5.8.3.6.3.2 Broadcast Traveler Information - Circular Region - Anchor Point

For each traveler information message in a traveler information packet, an RSU shall include the geographic location (latitude, longitude, elevation) of the anchor point for the circular region of travel defining where the traveler information message is valid for as part of a traveler information packet broadcasted to connected devices. The latitude and longitude are measured in units of 1/10th microdegree. The elevation represents the height above or below the WGS-84 reference ellipsoid in units of 1 decimeter. If this anchor point is included, it takes precedence over the default anchor point, otherwise, the default anchor point defined in Section 3.5.8.3.6.1 should be used.

3.5.8.3.6.4 Broadcast Traveler Information - Polygon Valid Region Requirements:

A spatial region for which a traveler information message is valid for may be a polygon, which may represent the jurisdictional boundaries of a specific transportation agency or a work zone. The connected device should be located within this polygon region for the traveler information message to be presented to the traveler. The detailed requirements for defining the polygon region where the traveler information content should be presented to the driver are as follows.

3.5.8.3.6.4.1 Broadcast Traveler Information - Polygon Region - Offsets

For each traveler information message in a traveler information packet, an RSU shall include the area of travel defining where the traveler information message is valid for as part of a traveler information packet broadcasted to connected devices. The valid area of travel (polygon) is defined by a series of offset points (x-axis, y-axis based on the WGS-84 coordinate system), measured in one meter units. These offset points describes the perimeter of the polygon. Each offset point defines the offset from the anchor point defined in Section 3.5.8.3.6.4.2, or the default anchor point defined in Section 3.5.8.3.6.1. The anchor point in Section 3.5.8.3.6.4.2 takes precedence over the default anchor point.

Optionally, an elevation offset point (z-axis) and an elevation tolerance, both in one meter units, may also be transmitted for each set of offset points. If the elevation offset point is not transmitted, all connected devices within the x-y polygon area are part of the valid area of travel. For example, if a node has an elevation offset point of +5 meters with an elevation tolerance of ± 2 meters, any connected device located between +3 meters to +7 meters offset from the anchor point, while connected devices outside this range are not. This requirement is necessary in case the traveler information message is valid for specific overpasses or underpasses in a specific direction.

3.5.8.3.6.4.2 Broadcast Traveler Information - Polygon Region - Anchor Point

For each traveler information message in a traveler information packet, an RSU shall include the geographic location (latitude, longitude, elevation) of the anchor point for the area of travel defining where the traveler information message is valid for as part of a traveler information packet broadcasted to connected devices. The latitude and longitude are measured in units of 1/10th microdegree. The elevation represents the height above or below the WGS-84 reference ellipsoid in units of 1 decimeter. If this anchor point is included, it takes precedence over the default anchor point, otherwise the default anchor point defined in Section 3.5.8.3.6.1 should be used.

3.5.8.3.6.5 Broadcast Traveler Information - Valid Shape Point Set Region Requirements

A spatial region for which a traveler information message is valid for may be a shape point set, which allows a spline-like representation of a geographic area such as a road segment. A connected device should be located within the shape point set region for the traveler information message to be presented to the traveler. Unlike the valid circular region of travel or the area of travel (polygon), which both can be used to define a wide geographic area, the shape point set is used to represent a short segment of a specific roadway. The detailed requirements for defining the shape point set region where the traveler information content should be presented to the traveler are as follows.

3.5.8.3.6.5.1 Broadcast Traveler Information - Shape Point Set - Default Direction

For each traveler information message in a traveler information packet, an RSU shall include the default direction of travel along the shape point set as part of a traveler information packet broadcasted to connected devices. This requirement indicates the direction of travel along the series of offset points defined in Section 3.5.8.3.6.5.3. Valid values are forward (direction of travel follows node ordering), reverse (direction of travel is the reverse of node ordering), or both (direction of travel allowed in both directions). The default direction is included so the direction does not have to be broadcasted for each shape point set defined for a traveler information message.

3.5.8.3.6.5.2 Broadcast Traveler Information - Shape Point Set - Default Width

For each traveler information message in a traveler information packet, an RSU shall include the default width of the shape point set as part of a traveler information packet broadcasted to connected devices. The width, measured in one centimeter units, is used to define the width of the valid region of travel at each offset point defined in Section 3.5.8.3.6.5.3, with the offset point located on the centerline path of the geographic area (width). The default width is included so a width does not have to be broadcasted for each shape point set defined for a traveler information message.

3.5.8.3.6.5.3 Broadcast Traveler Information - Shape Point Set - OffsetsFor each traveler information message in a traveler information packet, an RSU shall include the shape point set defining where the traveler information message is valid for as part of a traveler information packet broadcasted to connected devices. The shape point set is a path defined by a series of offset points (x-axis, y-axis based on the WGS-84 coordinate system and its reference ellipsoid), measured in centimeters. Each offset point represents the center line of the shape point set.

The first offset point is from the default anchor point defined in Section3.5.8.3.6.1, or the specified anchor point in Section 3.5.8.3.6.5.7 (the specified anchor point takes precedence over the default anchor point). All successive offsets are from the previous offset point. For example, the first set of offset points (node) describes the offset from the anchor point, the second set of offset points describes the offset from the first node, the third set of offset points describes the offset from the second node, etc…

Optionally, an elevation offset point (z-axis) and an elevation tolerance, both in one meter units, may also be transmitted for each set of offset points. If the elevation offset point is not transmitted, all connected devices within the x-y shape point set are part of the valid area of travel. For example, a node has an elevation offset point of +5 meters with an elevation tolerance of ± 2 meters. Thus any connected device located between +3 meters to +7 meters offset from the previous node is within the area of travel, while connected devices outside this range are not. This requirement is necessary in case the traveler information item is only valid for specific overpasses or underpasses in a specific direction.

3.5.8.3.6.5.4 Broadcast Traveler Information - Shape Point Set – Direction:

For each shape point set in a traveler information message, an RSU shall include the allowed direction of travel along the shape point set as part of a traveler information packet broadcasted to connected devices. This requirement is used to indicate the direction of travel along the series of offset points defined in Section 3.5.8.3.6.5.3. Valid values are forward (direction of travel follows node ordering), reverse (direction of travel is the reverse of node ordering), or both (direction of travel allowed in both directions). This direction of travel takes precedence over the default direction defined in Section 3.5.8.3.6.5.1, otherwise the default direction is used for the shape point set.

3.5.8.3.6.5.5 Broadcast Traveler Information - Shape Point Set - WidthFor a shape point set in a traveler information message, an RSU shall include the width for the shape point set as part of a traveler information packet broadcasted to connected devices. The width, measured in one centimeter units, is used to define the width of the valid region of travel at each offset point defined in Section 3.5.8.3.6.5.3, with each offset point on the center line of the shape point set. This width takes precedence over the default width defined in Section 3.5.8.3.6.5.2, otherwise the default width is used for the shape point set.

3.5.8.3.6.5.6 Broadcast Traveler Information - Shape Point Set - Node WidthFor a shape point offset in a traveler information message, an RSU shall include the width of the geographic area at that node as part of a traveler information packet broadcasted to connected devices. Each shape point offset (node) defined in Section 3.5.8.3.6.5.3 represents a point along the center line of the shape point set. The width, measured in one centimeter units, represents the width of the geographic area at that node. This width takes precedence over the default width defined in Section 3.5.8.3.6.5.2 and the width of the shape point set defined in Section 3.5.8.3.6.5.5

3.5.8.3.6.5.7 Broadcast Traveler Information - Shape Point Set - Anchor Point

For each shape point set in a traveler information message, an RSU shall include the geographic location (latitude, longitude, elevation) of the anchor point for the shape point set defining where the traveler information message is valid for as part of a traveler information packet broadcasted to connected devices. The latitude and longitude are measured in units of 1/10th microdegree. The elevation represents the height above or below the WGS-84 reference ellipsoid in units of 1 decimeter. If this anchor point is included, it takes precedence over the default anchor point, otherwise the default anchor point defined in Section 3.5.8.3.6.1 should be used.

3.5.8.3.7 Broadcast Traveler Advisories - Content

For traveler advisory message in a traveler information packet, an RSU shall include the contents of the travel advisory information as part of a traveler information packet broadcasted to connected devices. Traveler advisory information consists of ITIS codes, as defined in SAE J2540/2, and free-form text. Traveler advisories include information on travel or route restrictions, work zone information such as work zone signs and directions, speed advisories including speed limits, and traveler services available at upcoming exits. For commercial vehicles, traveler advisories may be the availability of parking spaces and services information, including availability times and parking space dimensions.

3.5.8.3.8 Broadcast Road Sign - Content

For each road sign message in a traveler information packet, an RSU shall include the road sign information as part of a traveler information packet broadcasted to connected devices. Road sign information consists of ITIS codes, as defined in Section 9 of SAE J2540/2, and free-form text.

3.5.8.3.9 Broadcast Traveler Information - Uniform Resource Locator

For each traveler information message in a traveler information packet, an RSU shall include a uniform resource locator (URL) for the traveler information message as part of a traveler information packet broadcasted to connected devices. The URL is a network-retrievable locator where other resources can be found, which, in this context, provides additional information about the event or topic of the traveler information message. The URL, in the form of a text string, provides a link to the designated resources. For example, the URL may be the address of a link that points to a map image showing the location and extent of the event, or the address of a link containing a text file with additional information about the travel advisory such as start and end dates and times. These resources may be retrieved by the user via a different communications medium and device, such as a smart phone over a cellular or wi-fi network.

3.5.8.3.10 Broadcast Traveler Information - Valid Vehicle Type

For each traveler information message, an RSU shall include the vehicle types that the traveler advisory or road sign is valid for as part of a traveler information message broadcasted to connected vehicles. The vehicle type is defined by the FHWA 13-Category Classification System. The connected vehicle must be of the vehicle type for the traveler advisory or road sign to be presented to this traveler. All combinations of vehicle types are allowed. If the vehicle types are not included as part of the a traveler information message broadcasted to connected vehicle, it is assumed that the traveler information message is valid for all types of vehicles.

3.5.8.3.11 Broadcast Traveler Information - Parking Availability:

Information about parking availability downstream is important for commercial vehicle drivers who are nearing the end of their allowable hours of service. The parking availability information is treated as a traveler advisory message. The following are the detailed requirements for providing parking availability information that may be broadcasted from an RSU to connected devices.

3.5.8.3.11.1 Broadcast Parking Availability - Mandatory Requirements

For each traveler information message in a traveler information packet, an RSU shall include parking availability at downstream rest areas as part of a traveler information packet broadcasted to connected devices. Parking availability information broadcasted includes the following: the time the information was last updated, the location of the parking facility (latitude, longitude, elevation), and the number of parking spaces available at that facility by maximum vehicle size (length and width, in centimeters) that the space can fit.

3.5.8.3.11.2 Broadcast Parking Availability - Optional Requirements

The following are optional information that an RSU may include with the parking availability information broadcasted from the RSU to connected devices.

3.5.8.3.11.2.1 Broadcast Parking Availability - Location DescriptionAn RSU shall include a description of the parking facility as part of a traveler information packet broadcasted to connected devices. The textual description may include additional information about the parking facility, such as what exit to use to get the parking facility, or what parking amenities is available at the facility.

3.5.8.3.11.2.2 Broadcast Parking Availability - Availability TimeAn RSU shall include the time one or more parking spaces, by maximum vehicle size, will be available as part of a traveler information packet broadcasted to connected devices. This requirement is transmitted only if no parking spaces, by that maximum vehicle size, is currently available but expect to be. Parking spaces may become available when the parking facility opens, or if a reservation for a parking space expires.

3.5.8.3.11.2.3 Broadcast Parking Availability - Availability End TimeAn RSU shall include the time one or more parking spaces, by maximum vehicle size, will no longer be available as part of a traveler information packet broadcasted to connected devices. This requirement is transmitted only if parking spaces, by that maximum vehicle size, is expected to be no longer available after that time. Parking spaces may become no longer available because it is the parking facility's closing time, or if there is a prior reservation for that size parking space.

U.S. Department of Transportation
ITS Joint Program Office-HOIT
1200 New Jersey Avenue, SE
Washington, DC 20590

Toll-Free "Help Line" 866-367-7487
www.its.dot.gov

FHWA-JPO-17-508

U.S. Department of Transportation