Applying Scrum to Stabilize Systems Engineering Execution

Richard Carlson, Boeing
Philip J. Matuzic, Boeing
Robert L. Simons, Boeing

Abstract. Years before the Agile Manifesto [1] was created in 2001, effective practices for managing projects were being applied to reduce project cycle times and reduce costs while increasing productivity and quality. Agile [2] principles that emerged from the manifesto promulgated rapidly throughout industry on software development projects at first, and eventually into projects that were not software centric. This article focuses on the application of Scrum [3], an agile project management method that uses simple practices to enable stability for the execution of systems engineering (SE) activities and the development of requisite systems engineering work products throughout the product development lifecycle.

Introduction

The background and history of using agile on software development projects are well documented by the IEEE, the National Defense Industry Association conferences, and through workshops, papers and technical articles published by the SEI. The successes of agile software development projects have attracted tremendous interest throughout the SE community. There has been an upsurge within academia and industry advocating the use of these practices and principles to implement agile SE and reports of successes are emerging. This article identifies key drivers that are paramount in stabilizing agile-based SE product development efforts, lessons learned from empirical experience, specific areas of practice that were difficult to implement, and requisite infrastructures that must be in place before considering agile at the SE level.

Agile Systems Engineering Stability Drivers

There are some interesting contrasts between traditional (or waterfall [4]) project management and agile project management:

- High overhead vs. focused work: Traditional projects suffer excessive overhead caused by lengthy meetings that waste valuable team time. Agile projects concentrate on guiding committed work in progress through short, time-boxed ceremonies.
- Mature designs vs. incremental development: Traditional projects place too much emphasis on first completing mature architecture and design including features and functions that may rarely, if ever, be used. Agile thinking demands that the product and its associated artifacts be developed incrementally through a series of short iterations guided by frequent customer feedback throughout product development.
- Command-and-control vs. empowered teams: Project managers are trained (and certified through formal project management training) to exercise authority and direction of project members in order to accomplish predefined and measured goals. Agile projects, with participation of the existing project management authority, allow teams to self-manage and self-organize so they can become increasingly synergistic and efficient.

Agile Project Planning

The goals of agile project planning are defined in four critical artifacts: product vision, initial backlog of requirements, roadmap, and release plan.

The product (or project) vision must be created by the project visionary, typically the product owner. When implementing a Scrum approach, the product owner coordinates with key stakeholders to describe a desired state of the product in a series of release increments. The vision should be a clear and succinct statement describing the product's capabilities and features that set it apart from anything built in the past. The vision statement must convey the expected value expressed by the customer or as specified in the contract. It should also include the company's or project's motivation of using Scrum to manage its development, the product's key features, potential, and most significant risks with corresponding mitigation strategies.

The backlog, or product backlog as it is coined in Scrum, is a prioritized list of tasks encompassing everything essential to the success of the product. The priority of items listed in the backlog is typically driven by the customer.

The product roadmap is a translation of the product vision into product feature development terms. It describes "what" needs to be completed or implemented, and it should map development timelines of the product that reflect major features, a set number of releases, both internal and external, and the number of iterations planned for each release cycle. The roadmap should be updated prior to the start of each release cycle to ensure changes directed from the customer or as a result of product backlog re-prioritization are communicated to the core project team and its key stakeholders.

Release planning involves key stakeholders in determining incremental releases for the product development. The release plan describes the "how" and includes the details of the product roadmap. It also includes details of product development that extend throughout the lifecycle of the project.

About Scrum

Scrum is an agile framework for managing complex projects. Originally Scrum was formalized for software development projects, but works well for any complex, innovative scope of work. The possibilities are endless.

The Scrum framework is deceptively simple:

- The product owner creates a prioritized feature wish list called a product backlog.
- The team has a defined amount of time, called a sprint, to complete its work. A sprint is usually two to four weeks and the team meets each day to assess its progress (called the daily Scrum).
- During a sprint [5] or iteration planning, the team pulls a small quantity of tasks from the top of the wish list—the sprint backlog—and decides how to implement those features during the current sprint.
- During the sprint, the Scrum master keeps the team focused on its goal.
- Task results at the end of the sprint are an executable increment of the product which is potentially deployable to a customer, put on a storage shelf, or demonstrable to a stakeholder.
- The sprint formally ends with a sprint review and team retrospective.
- The next sprint begins with sprint planning where the team chooses another chunk of the product backlog and begins working on the new features.

The cycle repeats until enough items in the product backlog have been completed to finalize the product, the budget is depleted, or a programmatic deadline arrives. Which milestone marks the end of work is entirely specific to each project. No matter which impetus stops work, Scrum ensures completion of the most valuable work before the project ends.

Scrum employs an iterative and incremental method for managing projects. Scrum has no ties to the Project Management Institute [6] and it is not part of the Project Management Body of Knowledge [7]. Although Scrum was originally recommended for software development projects, it is easily applied to just about any type of project. This means that Scrum can and has been used as a framework to manage a wide range of non-software-centric projects.

Scrum uses a set of simple practices that drive development stabilization:

- Scrum relies on active customer participation. Open communication with the customer provides visibility into issues and problems that may have adverse effects on project schedule, cost, and product release. When the customer is multitasking or otherwise not available to participate in technical interchanges, such as product reviews, they invite problems that will cause things to go awry.
- Daily stand-up meetings are short, time-boxed meetings that keep the team focused on communicating what they have accomplished since the last meeting with other team members, what they plan to do next, and any impediments that are keeping them from achieving planned work.
- Extensive planning is conducted prior to iteration activities to ensure development environments are available and that team members have everything they need to complete their tasks. Planning is conducted on the first day of each iteration so the team knows the overall goal and can determine how all work will be completed. Iteration planning assures the team commits to an amount of work it can complete during the iteration with a high probability of success.
- Estimating work in terms of requirements is conducted by all key stakeholders at the beginning of every project, and by the team at the start of each iteration throughout the project.
- Iterative development involves short, single passes through an entire development lifecycle. Short iterations are necessary to obtain early and frequent feedback from customers and other key stakeholders.
- Prioritized work contained in a product backlog encompasses all project requirements. Priorities are based on customer decisions or are business driven to ensure that only requirements bringing the most value to the customer are completed first.
- Reviews of work products completed during the iteration assure that adequate verification and validation take place.
- Self-organized agile teams must be empowered to make decisions, consult with domain and subject-matter experts, and select work tasks in an appropriate and logical sequence.

The Scrum Process

Scrum is a simple framework with three roles, three critical artifacts, and four low-overhead ceremonies (or meetings). The roles are:

Product Owner: The product owner represents the voice of the customer and is accountable to ensure that the team delivers value to the business. The product owner is responsible for taking all the inputs defining the product from the customer or end-user of the product, as well as from team members and stakeholders, and translating them into a product vision. In some cases, the product owner and the customer are one and the same; in other cases, the customer might actually be millions of different people with a variety of needs. The product owner writes customer-centric items, prioritizes them, and adds them to the product backlog. Scrum teams should have one product owner and, while they may also be a member of the development team, it is recommended that this role not be combined with that of the Scrum master. [Note: There are many instances of multiple individuals filling the role of the product owner to assure domain or technical knowledge is available to the team.] The product owner role maps to the product (line) manager position in many organizations.

Scrum Master: Scrum activities are facilitated by a Scrum master, also written as ScrumMaster, who is accountable for removing impediments enabling the team to deliver the iteration or sprint goals and deliverables. The Scrum master is not the team leader but acts as a buffer between the team and any distracting influences. The Scrum master ensures that the Scrum process is followed and is the enforcer of rules. A key part of the Scrum master's role is to protect the team from distraction and keep them focused on the tasks at hand. The role has also been referred to as servant-leader to reinforce these dual perspectives. The Scrum master is responsible for helping the team be successful. The Scrum master is not the manager of the team; he or she serves the team helping remove impediments, facilitating meetings, and supporting the practice of Scrum.

Team: The team is responsible for developing and delivering the product and is typically made up of five to nine people with cross-functional skills. The team should be self-organizing and self-managed. SE teams using Scrum should be staffed with functional and technical SEs, a domain-knowledgeable architect and software engineer, and testers.

Scrum Artifacts and Transparency Tools

The product backlog is a prioritized list maintained throughout the entire project. It aggregates backlog items that are broad descriptions of all potential features prioritized by business value as an absolute ordering. The product backlog is "what" will be built, sorted by importance. It is open and editable by anyone and typically contains rough estimates of business value and/or development complexity. Those estimates help the product owner gauge the timeline and, to a limited extent, prioritize tasks.

The iteration backlog is the list of work the team must address during the next iteration. Selected product backlog items are broken down into tasks, which, as a best practice, should normally be between four and 16 hours of work. With this level of detail, the whole team understands exactly what to do. Tasks on the iteration backlog are never assigned. Instead, tasks are selected by the team members as needed, according to the set priority and team member skills. This promotes self-organization of the team. The iteration backlog is the property of the team, and all included estimates are provided by the team. However, the Scrum master may prefer to maintain this artifact to ensure it reflects iteration status in real-time. Often an accompanying taskboard or work-in-progress (WIP) board is used to view and change the state of the tasks during the current iteration.

The taskboard is a transparency tool that shows tasks in work during an iteration. The taskboard is updated by team members as they complete each task. As the tasks are accepted by the team members, they are checked out. Tasks are worked through completion and must be verified by the product owner before initiating another task. The team maintains the taskboard up to the "To Be Verified" column. Verification is done by the product owner and shown by the product owner moving the task to the completed column. If a team member completes a task, he/she cannot take credit until the product owner verifies it is complete. This is one of the most simple yet best transparency tools used on agile projects!

The iteration burndown chart displays a metric of remaining work in the iteration backlog. Updated every day, it provides a view of ongoing iteration progress.

Scrum Ceremonies

The daily Scrum or stand-up meeting is conducted each day and is facilitated by the Scrum master. During the meeting, each team member responds to three questions:

- What have you done since the last stand-up meeting?
- What are you planning to do today?
- Do you have any impediments that would prevent you from accomplishing your work?

The Scrum master facilitates resolution of these impediments, although all resolution occurs outside the daily Scrum itself to keep the meeting under 15 minutes. This meeting must always start on time, be conducted at the same location and time every day, and while anyone may attend, only team members are allowed to speak. Management is not allowed to speak and topics outside the strict agenda are scheduled for separate discussion.

Iteration planning for execution is conducted on the first day of the iteration. The core Scrum team (i.e. product owner, team, and Scrum master) meet to determine features that must be completed during the next iteration. At the opening of the meeting, the product owner explains the vision and product roadmap. From the product backlog of Prioritized Backlog Items (PBIs) he/she identifies "what" needs to be completed next. This meeting, facilitated by the Scrum master, consists of detailed planning activities between the product owner and the team. PBIs, identified by the product owner for the next iteration are estimated for effort and complexity and then selected by the team. The number of PBIs selected is dependent on the team's capability or their availability to complete selected PBIs during a single iteration. If the selected PBIs can be completed with a high level of confidence during the upcoming iteration, the team commits to those PBIs and the Scrum master enters the items into the iteration backlog. If they cannot be completed due to over-complexity or size, they are either decomposed or deferred unless they are the next highest priority items. During the second half of the iteration planning meeting, the team determines "how" to complete the selected PBIs by breaking down each item into quantifiable tasks and activities. This promotes "systems thinking" and ensures that all requisite and value-added steps are identified and implemented. As a standing rule for Scrum teams, all tasks must be completed and verified before a PBI can be declared "done" or available for deployment or delivery.

At the end of the iteration, team members conduct a review of all completed work with key stakeholders to validate the fact that requirements were interpreted accurately. The review sets up a conversation between the participants about what was done, a demonstration of prototypes completed, and a decision on what to complete next. Required iteration review attendees include all team members, the product owner, Scrum master (who facilitates the review), relevant key stakeholders, customer or customer representatives, users, and any interested engineers, domain experts, and managers. Attendance by all is vital to ensure that everything completed is presented, questions are properly responded to, and that feedback is given and received. During the review, the project is assessed against the iteration goal established during the iteration planning meeting. The Scrum master ensures the review does not exceed its time-boxed schedule (usually four hours or less). The team should be prepared to discuss what was done, how the iteration's goals were met, and to demonstrate the product in its current state. At the conclusion of the review, the product owner acknowledges either acceptance of work products presented or deferral of work until it is more mature or good enough for release into product development or production.

Retrospectives are conducted to build team commitment, transfer knowledge to the next iteration, and share information with the customer, management, and other teams. The retrospective is the structured reflective practice that enables teams to learn and improve based on empirical experiences. It is a time-boxed meeting held with the team members, product owner, and Scrum master at the end of an iteration to:

- Discuss what was successful about the iteration or release.
- Realize what could be improved.
- Learn from experiences and plan for subsequent iterations.
- Plan to incorporate the successes and improvements in future projects.

- Share and pass along the learning experience.
- Make changes for the next iteration.

Retrospectives are conducted at the end of the iteration; however, a retrospective is very effective when conducted at the end of any event, that is, any time there is value for the team to pause for a few minutes to learn from its recent experiences. But be very careful; retrospectives are not conducted to identify mistakes and place personal blame or personal attacks. They should not involve the resolution of issues and problems, and should never become a planning meeting.

Agile Helps to Stabilize Systems Engineering Defects

Mapping the potential costs of defects found by various detection techniques to common development strategies versus the cost-of-change curve very clearly shows that using an agile approach is less costly than traditional approaches. It has long been realized that the earlier in the lifecycle a defect is corrected, the less costly. In the chart below, errors detected early and often by self-organizing teams who develop work products iteratively are much cheaper to fix than latent defects detected later in the project lifecycle. [Source: Numerous articles and papers published on the subject.]

Agile/Scrum Practices for SE

Agile SE for software development was first implemented in Boeing on two very large programs consisting of multiple engineering domains. Agile SE teams defined and developed requirements and functions in the form of user stories [8], developed and demonstrated prototypes and received real-time user feedback, and created a product backlog for software development.

Using agile to conduct SE activities and create SE work products is a viable approach to reduce overhead and lengthy schedules. Most of the agile principles and Scrum practices can be applied to SE teams, for example:

- Conducting four-week "requirements iterations": Considering the amount of coordination and collaboration with domain and subject-matter experts required, four-week iterations is a good point of departure for requirements iterations. During project execution, if the team feels they can improve productivity by changing the iteration duration, then it should be allowed. Test for a month or two to validate the increase in productivity is working.
- Staffing systems engineering teams with systems engineers, business and/or functional analysts, tester, and at least one domain-knowledgeable software engineer. Much of the verification work is accomplished using this mix of expertise.
- Empowering the team by providing them with the requisite authority and everything they need to be fully functional and productive.
- Developing SE work products incrementally using the agile principles and Scrum practices.
- Writing requirements in user story format including requisite acceptance tests.
- Conducting iteration planning meetings on the first day of every iteration.
- Conducting brief daily stand-up meetings (15 minutes or less).

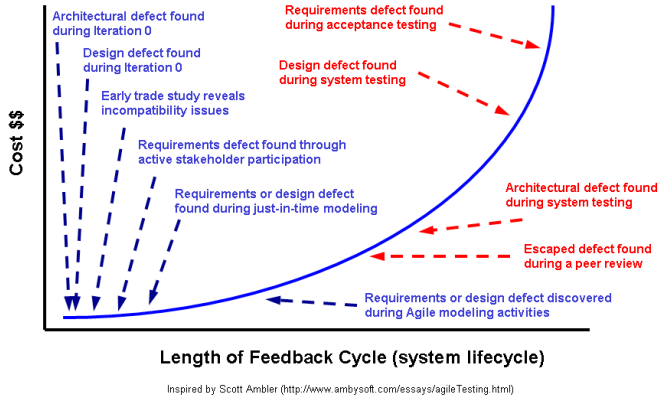

Figure 1 – Defect Cost Varies By Feedback Cycle

- Conducting peer reviews for all significant work products.
- Ensuring systems and sub-systems integrated verification and validation.
- Holding iteration reviews of SE work products including prototypes, documentation, trade studies, analyses, user stories, and groomed product backlogs.
- Conducting iteration retrospectives at the end of each iteration, and encouraging holding frequent mini-retrospectives.
- Holding Scrum of Scrums meetings with Scrum master and other stakeholders at least two or three times a week.
- Establishing an active product owner core team consisting of multiple product owners on multiple domain projects, and projects that require multiple product owners to provide sufficient domain and subject-matter expertise.
- Supporting geographically distributed teams with simple transparency tools to enable a highly collaborative and visible working environment.
- Implementing agile on large programs across multiple engineering domains.

Key Things Learned Using Agile/Scrum

The following are examples of what was learned implementing agile on SE projects:

- Scrum is a natural approach to managing a project, and teams like the simplicity of the Scrum framework.
- Close collaborations improve significantly between team members and key stakeholders using Scrum.
- Impediments, issues, problems, and potential risks are identified early and often in real-time through short, daily stand-up meetings.
- Retrospectives can become boring and mundane if action plans are not implemented aggressively and followed through.
- Periodic Scrum of Scrums meetings on large programs optimize communications and information sharing across project teams and the greater organization.
- Product owners with sufficient domain knowledge and overall understanding of the product are hard to find and retain.
- Scrum masters are daunted when attempting to facilitate more than two teams.
- Adoption of agile on SE projects is slow due to an overall resistance to change.
- Simple inexpensive transparency tools are very effective (e.g. WIP board, backlogs, burndown charts).

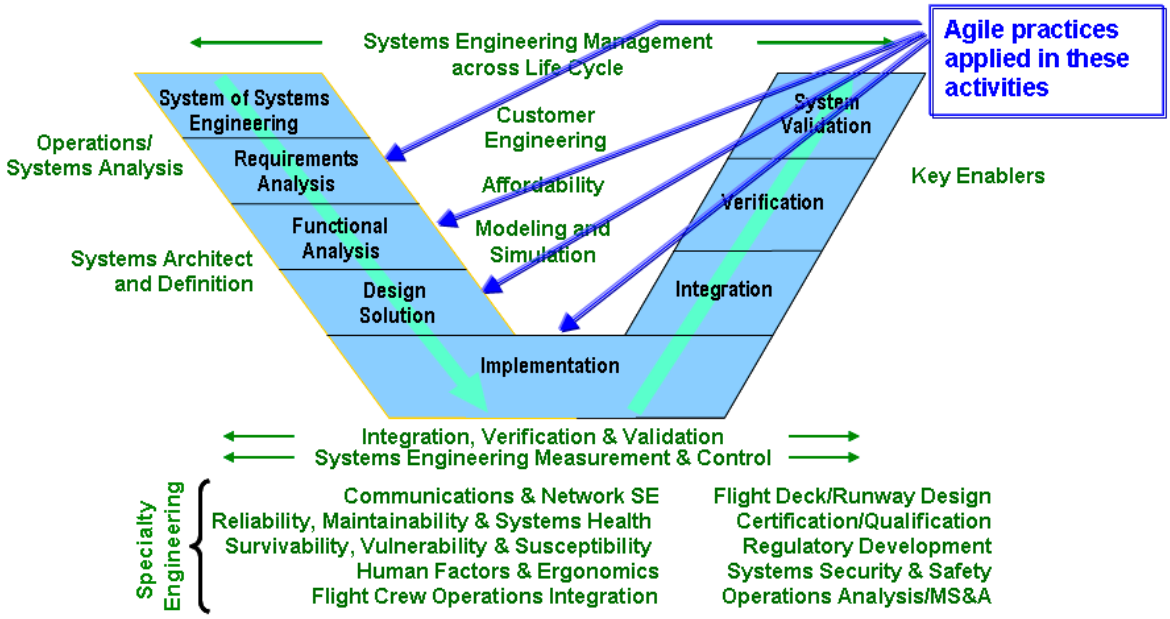

Figure 2 – SE V Diagram With Agile

The Systems Engineering "V" and Agile Enablers

An effective way to see where agile can be applied to SE is to revisit the SE V-model (see below). The fundamental SE processes include:

- Requirements analysis
- Functional analysis and allocation
- Synthesis and integration
- Verification and validation
- Systems analysis and control

The left side of the "V" involves identification and decomposition of requirements, functions, and design. The right side of the "V" includes the integration and characterization of product development. All of the attributes supporting the fundamental SE processes either have been or can be enabled using Scrum's simple practices.

During early system development an initial examination of customer needs, often referred to as a statement of need, is always necessary, but is especially critical if initial customer requirements are not available. The first product generated from examination of the statement of need is a feasibility analysis. The application of Scrum at this early stage introduces the framework for developing follow-on functional system requirements.

Alternative technology options and design considerations can quickly be identified and assessed during product backlog development activities. Because the customer's view is represented through the product owner, the team has this perspective available to them as they quickly identify, assess, and document their findings.

The strength of the Scrum process of infusing customer insights through the product owner throughout the planned series of iterations performed is important. Quality Function Deployment (QFD) [9] is often used in systems engineering customer engagement activities. The QFD is designed to establish a communications dialog and elicit customer input on end-state goals and objectives. The findings of a QFD are used to frame the contents of the statement of need and reflect the customer's perspective in the feasibility analysis report. A fundamental weakness of QFD is that it is almost exclusively performed as a single activity, usually early in the feasibility analysis timeline. Because this is an instant-in-time view of the customer's perspective, emerging customer concerns and needs are lost. The Scrum framework of continual customer participation throughout an incremental lifecycle provides the most current customer view for the system development process. Applying a disciplined Scrum approach ensures the feasibility analysis has customer buy-in throughout the study. The resultant Scrum-based feasibility analysis establishes a customer-aligned baseline that drives the development of operational user requirements, conceptual designs, and follow-on system-level requirements and specifications.

Conclusion

The Scrum agile approach is by far the most widely used and implemented technique [10] adopted by software and systems development teams because of its simple practices and empirical process control [11]; but Scrum is a significant cultural shift from more traditional ways of project management. Scrum roles differ significantly from traditional roles and frequently cause confusion among general engineering and business communities alike. Scrum terminology is foreign to most and change from the status quo is not only difficult, but is often initially resisted. Scrum is most effective when used as a wrapper for the organization's existing engineering practices and improved as necessary during product delivery or deployment increments. Scrum provides an environment where everyone can feel good about their job, their contributions, and that they have done their very best.

ABOUT THE AUTHORS

Dick Carlson has worked for Boeing for more than 7 years. He retired from the U.S. Army where he spent the bulk of his career in communications-electronics and systems engineering. After the Army, Dick worked as a consultant and employee focused on the development of software systems and engineering technologies supporting DoD, commercial, and private industry.

He has been active for nearly 15 years in the implementation of agile practices on a variety of software development and non-software centric projects. Dick spends most of his work time coaching, mentoring, and training Boeing engineers on the application of agile implementation and deployment. Dick has a Bachelor of Science degree from the University of Maryland, and is recognized by the Scrum Alliance as a Certified Scrum Professional and Certified Scrum Master, and is certified in Lean-Agile Project Management.

E-mail: richard.carlson2@boeing.com
Phone: 714-350-9946

Philip J. Matuzic is an Associate Technical Fellow at the Boeing Company Satellite Development Center, in El Segundo, California, where he is Chief Software Technologist and lean-agile modeling proponent. Philip also consults and provides technical training for Boeing, IBM, Northrop Grumman, Raytheon, Lockheed, and the U.S. Government. Mr. Matuzic has a MS in Software Management from Carnegie Mellon University, and a Project Management Certificate from the California Institute of Technology.

E-mail: philip.j.matuzic@boeing.com
Phone: 310-364-7387

Robert (Rob) Simons is a Boeing Technical Fellow in Boeing's Defense System's System Engineering organization in St. Louis. Rob is responsible for systems engineering and analysis supporting C4I, Network Centric, and homeland security activities. He holds multiple roles in program, project and team initiatives, and leads several academic collaboration initiatives. Rob holds MS degrees in Telecommunications and Engineering Management and an Graduate Certificate in Project Management from Washington University in St. Louis, an MBA from Lindenwood University and an MA in International Relations from Webster University.

E-mail: robert.l.simons@boeing.com
Phone: 314-234-3107

REFERENCES

1. Agile Manifesto <www.agilealliance.org/the-alliance/the-agile-manifesto>
2. Agile Alliance <www.agilealliance.org>
3. Scrum Alliance <www.Scrumalliance.org/learn_about_Scrum>
4. Waterfall model <http://en.wikipedia/Waterfall_model>
5. Sprint <http://en.wikipedia.org/wiki/Sprint_(Scrum)>
6. Project Management Institute <www.pmi.org>
7. Project Management Body of Knowledge <www.pmi.org/PMBOK-Guide-and-Standards.aspx>
8. User stories <http://en.wikipedia.org/wiki/User_story>
9. What is QFD? <http://www.qfdi.org/what_is_qfd/what_is_qfd.htm>
10. Figure 1 - Agile Is Primary Approach <http://www.practiceagile.com/>
11. What's Unique About Scrum? <http://scrummethodology.com/>