Using Modern Methodologies with Maintenance Software

Barbara A. Streiffert and Laurie K. Francis
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109 USA

Benjamin D. Smith
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109 USA

Jet Propulsion Laboratory uses multi-mission software produced by the Mission Planning and Sequencing (MPS) team to process, simulate, translate, and package the commands that are sent to a spacecraft. MPS works under the auspices of the Multi-Mission Ground Systems and Services (MGSS). This software consists of nineteen applications that are in maintenance. The MPS software is classified as either class B (mission critical) or class C (mission important). The scheduling of tasks is difficult because mission needs must be addressed prior to performing any other tasks and those needs often spring up unexpectedly. Keeping track of the tasks that everyone is working on is also difficult because each person is working on a different software component. Recently the group adopted the Scrum methodology for planning and scheduling tasks. Scrum is one of the newer methodologies typically used in agile development. In the Scrum development environment, teams pick their tasks that are to be completed within a sprint based on priority. The team specifies the sprint length usually a month or less. Scrum is typically used for new development of one application. In the Scrum methodology there is a scrum master who is a facilitator who tries to make sure that everything moves smoothly, a product owner who represents the user(s) of the software and the team. MPS is not the traditional environment for the Scrum methodology. MPS has many software applications in maintenance, team members who are working on disparate applications, many users, and is interruptible based on mission needs, issues and requirements. In order to use scrum, the methodology needed adaptation to MPS. Scrum was chosen because it is adaptable. This paper is about the development of the process for using scrum, a new development methodology, with a team that works on disparate interruptible tasks on multiple software applications.

I. Introduction

The Mission Planning and Sequencing (MPS) group, an element of Multi-Mission Ground Systems and Services (MGSS) at Jet Propulsion Laboratory (JPL), has developed software to plan, simulate, translate and package spacecraft commands and activities for science and engineering. The activities include observations, experiments, calibrations, maneuvers, etc. This software is in maintenance even though it is regularly updated to meet the specifications of newer spacecraft capabilities. MPS maintains nineteen applications that are NASA rated Class B (mission critical) or Class C (mission important). Software components that simulate the state of the spacecraft based on the commands that are being scheduled or translate and package the commands are considered class B because if they produce incorrect results the spacecraft could be damaged. Software elements that edit the commands or plan activities are considered class C because they aid in development of the command generation, but are not the final constraint checking software that is used. Figure 1 provides a functional diagram of the MPS maintenance applications. The color-coding indicates the component and the inputs for the component and its outputs. Since the software is being used on spacecraft that are currently flying as well as spacecraft that are in development.
development with specific launch dates, it is imperative that any deficiencies found in the software be fixed as quickly as possible. The MPS team has six software engineers, one software engineer/system engineer, two test engineers and one system engineer. Also since the software is in maintenance, the software engineers have multiple software applications that they are responsible for. Due to the number of applications that are involved, it is difficult to schedule tasks and to track their status. The team has decided to look at modern methodologies to help schedule and track the work.

Figure 1. The MPS Applications in Maintenance (Each Software Engineer is responsible for multiple applications. Color coding represents the component and its associated inputs and outputs. Inputs in black represent inputs from other organizations).

II. Scrum Methodology

In the past the team has used Waterfall and Rapid Prototyping methodologies. For reference, in Waterfall the requirements are defined; then a design is developed. The next step is to implement and test and finally to deliver. Waterfall has not had good results for new development due to its rigidity and definitely has not been deemed suitable for maintenance. Rapid Prototyping is developing a capability, designing that capability and implementing it. The difficulty is that with many applications in maintenance it is difficult to track and there’s little input from users during its various stages. The team has decided to look at agile development methodologies. All of the Agile Development methodologies have several tenants in common. The first commonality is that all of them are incremental and iterative in development. Each one allows for the user to add or modify requirements at specified times during the process. Most of the Agile Development methodologies have processes where implementation is performed in bursts and typically pair programming is utilized; code features are not implemented until needed and code is reviewed regularly. Scrum is one of the Agile Development methodologies. It has significant flexibility and because of that flexibility it could be suitable for MPS.

In the Scrum development environment, teams pick their tasks that are to be completed within a short specific period of time called a Sprint. The tasks are based on priority and are time boxed. Sprints are usually two to four
weeks long, but the team specifies the Sprint length. Scrum is typically used for new development of one application. In the Scrum methodology there are three roles: Scrum Master, Product Owner and the Team. The Scrum Master is a facilitator who makes sure that everything moves smoothly. If a member of the team is blocked, then the Scrum Master attempts to remove the block. The Product Owner represents the user(s) of the software. The Product Owner sets the priorities. The team is responsible for selecting and completing the work that will be accomplished in the Sprint based on the priority of the items. At the beginning of each Sprint, a planning session is held where the team identifies the tasks for the Sprint. These tasks are small tasks from a group of tasks called a story. A story is a group of functionally related tasks. The priority of the small tasks is determined by the Product Owner. Each team member is responsible for ensuring that the tasks they’ve signed up for can be performed by the end of the Sprint. The unselected items are placed on the backlog. If one of the team members completes his/her tasks prior to the end of the Sprint and none of the other team members need help, an item from the backlog can be added to the Sprint. However, in Scrum the tasks should not be changed during the Sprint. The idea is that the Sprint is short enough that any changes can wait until the next Sprint. At the end of the Sprint a demonstration of the new capabilities is performed for the Product Owner and other interested stakeholders. Following the demonstration a retrospective is held. In the retrospective the team determines the elements of the Sprint to keep, the ones to change and the ones that need to be worked.

One other aspect of Scrum is the daily standup. The standup is not a status reporting meeting. It is typically 15 minutes or less. At the standup each team member states what they did the day before, what they are planning to do that day and if they are blocked in accomplishing their task(s). There is a short discussion of the task and/or any blockage. The Scrum Master runs the standup and the Product Owner attends the meetings. Typically the tasks are written on 3x5 cards. The selected tasks start in the “To Do” column, then move to the “In Progress” column; next each task moves to the “Verification” column and finally, to the “Done” column. It is important to define what “Done” means in terms of the project and the task. Figure 2 shows a sample standup board layout with the columns and the entries for the columns. In addition a burn down chart of the tasks is part of the standups. The burn down chart displays the rate of completion during the Sprint. It can be kept in terms of issues, hours or points. All of the values are determined by the team. Figure 3 displays a sample burn down chart.

Figure 2. Standup Board Layout.
There is one additional tenant that is part of Scrum basics and that is the concept that Scrum is adjustable and must meet the needs of the team. The use of Scrum and its various tenants must make sense in any situation.

III. MPS Scrum

MPS core software and the core software team doesn’t fit the normal Scrum development environment. Generally, a Scrum team works on a single software product and often those products are in new development. The MPS core team consists of nine people including software engineers, testers and systems engineers for all nineteen core software components. All nineteen are in maintenance. Each of the software engineers is responsible for multiple components. Each of the components is a separate product that is significantly different from the other components. The components work together in a mostly linear process and fall under the following categories: planning, simulation, packaging and various utilities that enable the other three categories to work more seamlessly. The bottom line is that each product is very different from the others. They perform separate tasks and are written in multiple languages including scripting languages and some utilize browser capabilities. These differences make it difficult to swap or add software engineers to tasks for other components than the ones that they are responsible for. Figure 4 shows the categories and the placement of the components into their respective categories. Another difference for MPS is that missions take priority. If a mission finds a problem with one of the components, that issue takes precedence over any other work the software engineer, system engineer or tester is working on. Having the software engineer able to work on the chosen task for an entire Sprint is one of the major tenants for Scrum because it increases productivity.

In order to figure out how MPS could tailor Scrum to fit MPS’s needs, the MPS manager, the system engineer and the test engineer attended a Scrum Master Certification course. The instructor specializes in training people to be Scrum Masters and Product Owners. The instructor made several suggestions for the MPS situation. He suggested that the team only be tasked at a percentage of their time to account for the fact that they could be interrupted by mission needs. In this way the Sprint would not have to be interrupted. In addition, he indicated that Scrum was meant to be flexible to meet the needs of those using it. He also suggested that the team tailor Scrum to meet the MPS needs, but to try to stick to the basic tenants of Scrum including the roles of a Scrum Master, a Product Owner and the team along with the prescribed meetings (the Planning Meeting, the Retrospective, and the Standup) and the concept of the Sprint. Everything else could be tailored.
MPS began to experiment with Scrum. The MPS Core System Engineer who meets with all the end users became the Product Owner and one of the Test Engineers became the Scrum Master. MPS uses Jira, a commercial issue tracking software, to track the MPS issues. Initially the Jira issues chosen for the Sprint were transcribed onto 3x5 cards and placed on a cork board. The cork board was divided into three columns ("In Progress", "Dev Complete" and "Done"). Jira contains the backlog. MPS started with two week Sprints. For the planning meeting the Jira issues were displayed and the cognizant software engineer for each product chose the issues with the highest priority that would be able to be completed in the two weeks. Figure 5 is the original cork board (minus the tasks). MPS also decided to use the burn down chart from Greenhopper, a commercial tool that is implemented by Atlassian, the company that produces Jira. Figure 6 is an early Greenhopper burn down chart.

Figure 4. The process categories and components placement in the categories.

Figure 5. The original cork board used for Standups.
At the end of the Sprint, a demonstration of the work completed in the Sprint is to be scheduled. Since all of the components are in maintenance the changes to the software are typically quite small. Often the tasks are bug fixes. The demonstrations are held after an accumulation of work that can be demonstrated. In addition, engineering/evaluation deliveries to mission customers occur frequently usually varying from one month to three months based on need and interest. Finally, larger modifications are demonstrated to the system engineer and the test engineers by the end of the Sprint.

At the retrospective at the end of each Sprint, the team goes over the Sprint activities to keep, the ones to change and the ones to work. The team has changed many aspects from how each sprint is named to how each sprint is documented. By the second retrospective, the decision to move to longer four week Sprints was made. Within the first two months the standup categories were changed to “To Do”, “In Progress” and “Done”. Later tasks became stories as long as there were multiple sub-tasks to be implemented. If the task is a “bug”, then the bug issue is the story. All coding tasks require a “verify” test sub-task. The definition of this task is to create the automatic test that is used to verify that the coding task is correct. The verification of the software changes occurs during various testing periods. JPL works a 9-80 work week so that every other Friday is considered a day off. It has been decided to eliminate standups on Fridays. Some teams don’t have sprints during the system test cycle before a delivery. MPS has decided to keep standups during the test cycle so that everyone on the team knows of any difficulties that have occurred during test and everyone knows what components are in the official test cycle. However, it was decided to only meet on Mondays and Wednesdays during the test cycle. Because of the volatile nature of needing to support missions, it was determined that tasks could be added to the Sprint as long as it is determined that the new task can be completed within the current sprint as well as the original tasks or a task of similar duration must be removed. This last item has caused some difficulties, but has worked out in many cases. The burn-down charts have been changed from tracking hours to tracking issues. Figure 7 shows a burn down chart where additions have created chaos during the sprint. The second chart (Figure 8) shows a much better outcome. Also within the first few months Greenhopper started to be used for standups instead of the 3x5 cards on the corkboard.
Figure 7. A Burn Down Chart for a Sprint that had too many additions to be able to complete the original set of tasks. The blue line at the bottom of the picture shows the tasks that were added to the Sprint.

Figure 8. A Burn Down Chart for Sprint that has had tasks added, but all the tasks were completed within the Sprint.
A set of wiki pages has been created to track each Sprint. On the wiki pages, the dates of the Sprint, the final burn down chart, the Jira issues being resolved in the Sprint and the Sprint retrospective results are listed. The documentation of each sprint has helped to keep track of the changes that have been made in the Sprints and the Stand ups as well as identifying how much work has been accomplished. The burn down chart lets MPS know how well (or not) the work went and enables the team to correct elements in the planning that go awry. Figure 9 shows the initial state of the wiki page for the first sprint of 2014. The task list identifies the task, the assignee and the status of the task. Even with the modifications Scrum has worked well for MPS.

![Image of wiki page](https://example.com/image.png)

**Figure 9. An initial wiki page for a Sprint.**

### IV. Current MPS Scrum Process

The current MPS Scrum Process contains the key aspects of the Scrum Methodology including the roles, the concept of the Sprints and set of meetings including Standups. However, other elements such as the team’s scheduling of tasks is different due to the nature of the application. The team members schedule themselves at 50% so that any mission needs can be addressed without aborting the Sprint. Because of the team member’s time allocation, software engineers are allowed to add tasks from the backlog if they finish their current set of tasks. Even with the reduced scheduling there are times that tasks need to be added to the Sprint (another adjustment to Scrum) because of mission needs. At the time the task is added it is determined if another task has to be removed. Another change is that the retrospective meeting occurs at the beginning of the planning meeting and is time-boxed to 20 minutes. Scrum often deals in terms of tasks being a given point value. MPS uses issues and time estimates per issue instead of the point system. The team is more accustomed to assessing tasks in that way. In addition, the process has moved to an electronic tracking of the tasks with Jira and Greenhopper. Every task is a Jira issue. Some of the tasks are stories because they have multiple sub-tasks. All coding tasks have at least one testing sub-task associated with it. The testing sub-task builds the automatic test that verifies the coding task has been implemented correctly. Each task is assigned to a Sprint or the backlog. Greenhopper is used to display the planning board (all tasks for a given Sprint) and the task board (the tasks for a given team member). Greenhopper also displays the chart board (see Figures 7 and 8). Figure 10 displays the Greenhopper planning board, the Greenhopper task board and the MPS Standup room.

![Greenhopper interface](https://example.com/greenhopper.png)

**American Institute of Aeronautics and Astronautics**
V. Conclusion

In many aspects MPS doesn’t fit the typical scenario of those using Scrum, however, the flexibility of Scrum has allowed MPS to tailor it to fit their needs. MPS utilizes the standard set of meetings and keeps the roles, but modifies the areas where there is built-in flexibility. In general using Scrum has been helpful in the following ways:

1. The estimations of the team on what can be done are better
2. The communication and understanding of the tasks among the team members is better
3. The communication and understanding of the tasks with the system engineer and the test engineers is better
4. Scheduling the work has been easier
5. Adding test tasks has allowed the component automatic tests to be built during development and not at the end of development.
6. Identifying problems earlier in the cycle so that they can be fixed earlier.
7. Allowing customers to know the status of the capabilities to be delivered and to take advantage of early releases in the form of engineering/evaluation versions. The MPS Core team has always been and continues to be incredibly professional and productive. Scrum has always been an experiment and continues to be an experiment – so far a successful one.

Acknowledgments

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work is funded by Multi-Mission Ground Systems and Services (MGSS).