Agile: From Software to Mission System

Jay Trimble
Mark H. Shirley
Sarah Groves Hobart
NASA Ames Research Center, Mountain View, CA, 94035, USA

The Resource Prospector (RP) is an in-situ resource utilization (ISRU) technology demonstration mission, designed to search for volatiles at the Lunar South Pole. This is NASA's first near real time tele-operated rover on the Moon. The primary objective is to search for volatiles at one of the Lunar Poles. The combination of short mission duration, a solar powered rover, and the requirement to explore shadowed regions makes for an operationally challenging mission. To maximize efficiency and flexibility in Mission System design and thus to improve the performance and reliability of the resulting Mission System, we are tailoring Agile principles that we have used effectively in ground data system software development and applying those principles to the design of elements of the mission operations system.

I. Introduction

We define the Mission Operations System (MOS) to be the people, team(s), mission products and processes needed to operate the mission. The Ground Data System (GDS) consists of the hardware, software and facilities used by the MOS, and the Mission System (MS) is the combination of the MOS and the GDS. In this paper, we focus on methods for designing the many parts of the MOS, such as the positions, tasks, interfaces and staffing plans that make up the team definitions; the flight rules, procedures, shift handovers and other processes followed by people; and the activity timelines, Deep Space Network scheduling requests and other products passed between people or teams. We touch on methods for designing the GDS, to the extent needed to support the methods for designing the MOS. However, the design of the GDS is not the focus of this paper.

A previous paper[1] laid out the idea of applying Agile principles to MOS design and described a series of paper simulations. Since then, the Resource Prospector team has designed, built and integrated a prototype rover, and the MOS team has driven it remotely, thereby testing much more detailed processes, procedures and tools. We have also improved our understanding of how to integrate Agile methods into NASA's project systems engineering process.

II. Background: From Waterfall to Agile Software Development

In our previous work with software development for missions[2], we moved from using a traditional waterfall development methodology, to adopting parts of an Agile methodology, then to a user-centered Agile methodology. This evolution was driven by the need to solve problems we encountered in the development process, rather than by a desire to fit a particular development paradigm.

A. Problems with Waterfall in Software

We started with a six month cycle of requirements definition, development, testing and delivery. The fundamental problem was the length of the cycle, which created a large number of requirements to implement over each long cycle. This in turn complicated design, development and testing. Difficult and complex development tasks were sometimes deferred over the long cycle. We used traditional measures such as lines of code as progress metrics. While they were easy to measure internally, we found that these metrics did not translate to meaningful demonstrations of progress either internally or externally. Testing six months of new code at the end of a cycle meant that any bugs found were expensive to fix, and there was always a strong pressure to ship since the customer had been waiting for months. The result was that minor issues were resolved with workarounds, and the customer had to wait another six months for resolution of issues found in acceptance testing.

Six months is a long time from specification to delivery. Over each six month period we fell out of touch with our customers. Our expectations diverged, creating a mismatch that resulted in disappointment and frustration for both the developers and our NASA colleagues to whom we delivered.

B. Steps Toward Agile Software Development

Our first step towards Agile was to shorten our delivery cycle, initially to six weeks, then to three. Each three week develop/test/deliver cycle was called a sprint. Four sprints constituted a release (figure 1). Although this allowed us to find and fix bugs found during unit (development) testing much more efficiently, we found that just shortening the cycle was not enough. Each sprint needed a clear set of objectives, tied to a strategic road map. We set up a three-tiered structure for interaction with our customer. The first was a nightly or continuous build. Every day the customer could download the latest software (figure 2). The measure of progress became what we demonstrated with working code, not what we described on a presentation chart.

The nightly build allowed the customer, and our internal quality assurance team, to try new features as they rolled out, giving us feedback immediately. Each three-week sprint was a deliverable, which the customer put through testing, and then accepted or rejected features. While sprints were intended for testing and feature acceptance/rejection, a release was intended for deployment into a mission operations environment.

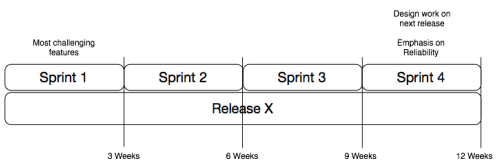

Figure 1 - Four Sprint Release Cycle

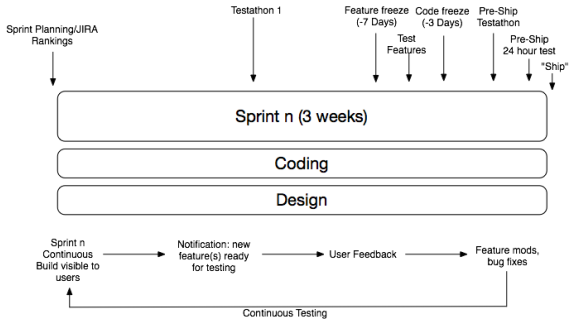

Figure 2 - Agile Sprint Detail

A key principle in this approach is that the schedule of deliveries is predictable and rapid. In effect, the train (the delivery) always goes on time. Features that are ready for delivery make it onto the train. Features that are not ready must wait for the next train. For this to work, the trains must leave regularly and frequently. In other words, sprints must be short and consistent, they must be delivered on time, and features that are not ready to ship may ship in a subsequent sprint. The only reason for delaying a shipment is if the software does not work.

The move to Agile solved our most pressing development and delivery problems. The short delivery cycle created a manageable set of features to develop and test in a bounded period of time. We were able to show demonstrable progress to ourselves and our customer on a frequent basis. We were able to solicit nearly instant feedback on the development of new features. Customer and team engagement and satisfaction went up.

III. From Waterfall to Agile for Mission Operations Systems

We know from experience that we can gain large benefits from using Agile to develop ground software. We are now exploring the potential benefits of using Agile for MOS design and development. We start by looking for those features of the RP Mission where the benefits are likely to be largest. Our first step is to look at the typical processes at NASA used for designing MOS.

It is plausible that space missions are the best case for using a waterfall process. They are extremely expensive and have a high, inherent risk of failure. Decades of experience have shown it to be cost effective to carefully consider up front all aspects of the mission before committing to its construction, because the cost of the team to design a mission is much less than the cost of the much larger team to build the hardware to fly it. Therefore it makes sense to finish designing, as much as is possible, before committing to construction. As a result, NASA's Space Flight Program and Project Management Handbook, which all missions must follow, mandates a waterfall process.

This is NASA's formal process, but the reality is more subtle. First, although it is waterfall, the recommended process is also concurrent. That is, all elements of the mission, including operations, are involved from very early in the process. However, in the early phases, MOS is typically staffed at a level that permits discussion and creation of requirements and planning documents but not at a level that could design or evaluate new processes, should they be

needed. Second, it's hard to overestimate the importance of the high flight rate early in the development of rocketry to the evolution and maturation of systems engineering and all aspects of mission design, including operations. In the case of robotic spacecraft, a sustained flight rate over decades has allowed NASA to mature its processes to the point where many aspects of most missions are based on what has been done before. Finally, NASA supports technology development and maturation efforts outside of missions. By the time a technology is selected for use on a mission, it has usually been tested in environments as close to the space environment as is possible.

Missions resist using any new technology unless it has been proven by previous missions, or unless the mission can't succeed without it. This risk aversion is rational and largely driven by the low flight rate. It exists in all mission elements, including operations. Mission operations practices in NASA are well established; hence MOS is not typically a focus area for innovation. While one could debate the merits and potential benefits of innovation in MOS, most projects do not focus their efforts there. That said, there are examples of new design and innovation in operations. The next Mars rover mission, Mars 2020, is redesigning the planning cycle to be able to conduct planning on a shorter timeframe. Given the lack of emphasis on mission operations for design and innovation, project managers normally expect to invest only a small amount on ops and ground systems during mission design, ramping up expenditures later during construction. This is why the MOS is typically designed and implemented using the same waterfall process that is used by the hardware mission elements. If the requirements are well understood, with only minor changes from the last mission, then a waterfall process can work well.

We believe that achieving the RP mission goals requires new MOS processes and that, absent those processes being designed and tested by technology development efforts outside of the RP project, it is both possible and desirable to do it within the project. The next section gives an overview of the mission goals and approach, and then it focuses on what is novel from the ops standpoint.

IV. Mission Overview

RP is designed to address gaps needed to plan human missions to the Moon before going to Mars. Returning to the Moon first involves shorter, less risky missions and the possibility of using lunar resources to reduce costs. We know there are some resources, but we don't know the amount or the accessibility of those resources.

The resources in question are volatiles, i.e., substances that are liquid at room temperature, such as water. The only place where lunar volatiles have been conclusively demonstrated is at the Lunar Crater Observation and Sensing Satellite (LCROSS) impact site[3]. Other permanently shadowed regions (PSRs) may contain volatiles, and temperature models and remote sensing data from Lunar Reconnaissance Orbiter (LRO) suggest volatiles may also exist just under the surface in areas lit at low angles for a short time each month. However, the spatial distribution of lunar volatiles is decidedly unclear. Therefore some form of prospecting is required, that is, the ability to move, to sense volatiles on and under the surface, and to access the volatiles.

RP is a rover containing prospecting and sample analysis instruments and a drill[4]. The rover will operate on the lunar surface for 7-10 days at a location where both sun exposure and direct to Earth (DTE) communication conditions overlap, and it will be tele-operated from Earth using the Deep Space Network (DSN). The rover will be capable of entering shadow and operating for up to 6 hours. At the end of 10 days, the sun will go down, and operating solely on solar and battery power, it is not expected to survive the cold of the lunar night.

We are focusing our use of Agile on areas that require new design. We have also tried Agile methods on more repetitive areas of the MOS, such as procedure walk throughs. Areas of new design that we have identified for RP MOS to date are:

- **Processes and planning cycles for real-time, science-driven, tele-operation**
  RP is similar to Mars rover missions in that the science team will determine where the rover will go, limited by the engineering team's assessment of capability and risk. However, the short mission necessitates 24/7 operations and much quicker decision-making. RP's short communications round-trip delay also makes tele-operation possible, which results in a very different command preparation and approval process from the daily command sequencing process used by many other missions. This results in a hybrid of rover-like and shuttle-like operations.

- **Fully Distributed Operations Team**
  The Mission Operations Team will be fully distributed. While distributed operations are not new, RP is taking distributed operations to a new level, having all operators at their home institutions. A standard distributed operations approach has clear boundaries, such as a payload team in one location and a vehicle

team in another. RP breaks this down further, for example, the rover team plans to have rover drivers in California and rover systems in Texas. Similar divisions exist within other parts of the operations team.

- **Harsh, lunar polar conditions**
  RP will be similar to the Lunokhod missions of the early 70's in that it will be a rover on the Moon operated in near real-time from the Earth. However, the environment at the lunar poles is more challenging than the locations visited by the Lunokhod program. With the Sun just above the horizon, uneven terrain will produce very long, very black shadows. In the permanently shadowed areas, the surface will be 40 Kelvin and the surface may contain material lower in density (almost 'fluffy') compared to the well-consolidated regolith near the equator. With the Earth also just above the horizon, line-of-sight radio communications may reflect off of the surface causing multi-path signal interference. These issues will complicate ops decision-making and risk management.

- **Class D mission with a (relatively) low budget**
  RP is similar to a Shuttle mission in its duration and potential for real-time, 24/7, shift-based operations. However, RP is class D, which means that mission operations, and the mission as a whole, must be done on a relatively small budget. For example, RP's budget doesn't allow construction of a special-purpose communications infrastructure like Apollo, so RP will use the Deep Space Network, which was not designed for tele-operations. The round-trip delay could vary between 10 and 30 seconds rather than the relatively steady several seconds experienced during Apollo. Another example is that we are considering reducing three driving shifts to two driving shifts plus an 8 hour charging period without ground contact, in order to reduce the staffing and space communications budgets. The impact of this option on science productivity will be a key issue evaluated this year.

V. Agile for Mission Systems

What does Agile mean in the context of a mission system? First, here's a brief overview of Agile principles, as originally elucidated for software.

Agile is a broad family of software development methods characterized by rapid iterations and continuous customer feedback. Agile methods began as a reaction against heavyweight software approaches that attempt to control the creation of software via formalized and pre-determined processes. Although formalized approaches allow good control overall of the software creation process, they often result in long lead-times and a product that is not what the customer needs. While there are many variations of Agile in practice, they all share common roots in the original Agile manifesto[5]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiations
- Responding to change over following a plan

By tailoring Agile to MOS, we hope to achieve, where applicable, benefits that are analogous to what we experienced with software. This includes efficiency and team engagement, focus of resources on the most important problems, and risk reduction.

In practice we are implementing this as:

- Simulation for design
- Assessment of capability through demonstration
- Early and frequent builds and tests
- Risk reduction through targeted experiments
- Maturation of processes and tools through frequent use

A. Simulation for design

In operations, the team typically does not get a chance to test processes until mission simulations for training, which typically start a year before launch. We are advancing this flow by trying mission simulations earlier and by using simulations of variable fidelity as a design tool.

In 2015, the RP Project designed, built and integrated a prototype rover and payload. Along with the hardware team, MOS and GDS designed and built the necessary ground system and processes to operate the prototype. At the end of this cycle, the hardware was used by the MOS team for three days to conduct a Distributed Operations Test (DOT). This was a high fidelity exercise using a prototype GDS deployed at three NASA centers, a partial mission timeline, procedures, telemetry, commanding, and rover driving using early mission software builds.

Test objectives included:

- Validate distributed decision making
- Test distributed command and data flow
- Validate integrated distributed situational awareness tools
- Test integrated ground/flight system test procedures
- Test the allocation of tasks and responsibilities across the team
- Test waypoint driving

In addition to these direct test objectives, the DOT drove other aspects of Agile mission operations development:

- Integrated development of procedures for test and operations
- Maturation of GDS tools through use

Figure 3 - One of Three Control Nodes for the DOT

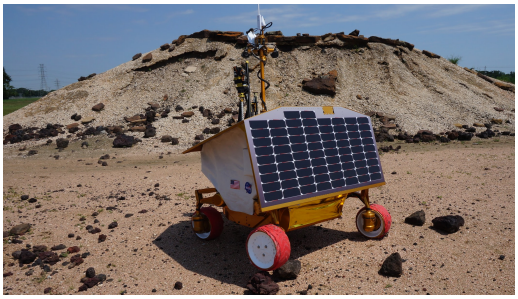

Figure 4 - Prototype RP-15 Rover and Payload

Based on results of the DOT, we altered the composition and task distribution of the planned operations team for the mission, updated procedures, and made changes to the distributed GDS architecture. The extended pre-phase A period of this project allows for substantial design changes based on simulation results. Another notable impact of conducting simulations has been the alteration of discussions in meetings. When an operational issues arises, we discuss it, but rather than having a series of ongoing discussions, we put it on the list of things to test in simulations.

B. Assessment of Capability through Demonstration

This is the mission operations analogy to our tailored Agile principle of "working software is the primary measure of progress." Before using Agile methods, we assessed the state of a software project with metrics and analysis but could not try the working code until testing (internal) and delivery (external). With modern software methods, we have nightly or continuous builds for code, and we can test capability every day.

However, there is no nightly build for mission operations systems processes similar to the nightly builds for the software. Our current method of assessment of capability through demonstration is to use the previously discussed simulations to assess our capability to perform the mission. We are investigating the right level of frequency and the optimal methods for these simulations.

Metrics include:

- Procedure execution errors or identified issues
- Time to perform a procedure
- Time to perform a portion of the mission timeline
- Rate of on-time uplink of correct commands (that have the desired effects)

- Forward movement rate of the rover while prospecting

C. Early and Frequent Builds and Tests

This philosophy has many variants and goes by many other names – "deploy early and often," "fail early so you can succeed sooner." The idea is simple - try things at the earliest opportunity. Simulate a procedure, test your command plans, and perform a segment of the mission timeline. At the early phases of testing, the goal is to find as many problems as you can. Here is an example involving procedures.

We put procedures to the test by "executing" them soon after they are first drafted. Rather than holding extensive and ongoing meetings, we draft a procedure, review it, try it, de-brief it, iterate, then re-try if needed. In our first run through of procedures, we used a telecon in place of voice loops, mockups in place of displays, and we used Google Hangouts to share display mockups across locations. Our early evidence indicates that the total time required by the team to iteratively develop, test and refine procedures is less than the traditional methods of meetings, reviews and late simulation. We will have more data on this assertion after we have been through the full mission life cycle.

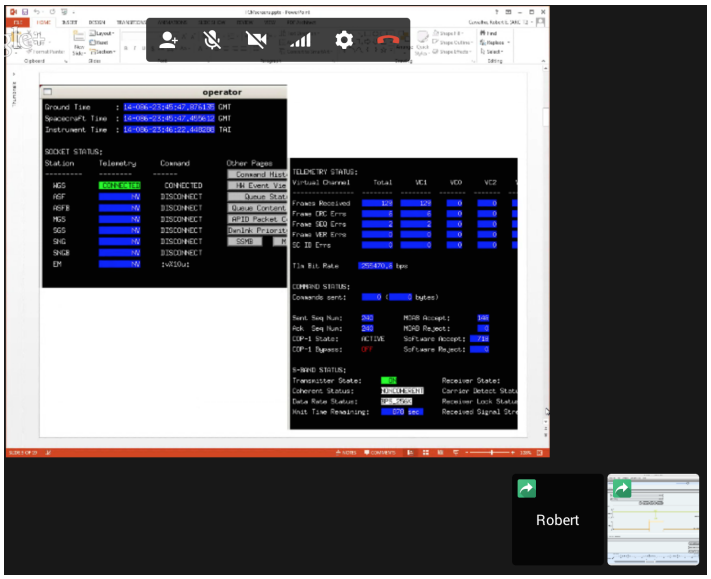

Figure 5 - Telemetry display mockup for paper sim, shared in a Google Hangout

In the same way we test procedures early, we try out portions of the mission timeline. This involves executing procedures in time order, while following a portion of the mission timeline. Our first mission timeline simulations used early prototype software to build the timelines, paper procedures, and a combination of Google Hangouts and a telecon as outlined above.

D. Risk Reduction Through Targeted Experiments

The purpose of risk reduction through targeted experiments is to gain operational experience with the highest risk mission elements early, to assess risk and to iteratively improve the design of a mission element. One example is developing the concept of operations for driving the rover and verifying the feasibility of our speed assumptions for the mission. Our ability to drive the rover is influenced by many factors and a lot of uncertainty. How long does it take (a) to downlink and process camera images, (b) to interpret the images and identify hazards to driving, (c) decide on a safe path forward and (d) uplink waypoint commands. How do changes in the camera fields-of-view or lighting or the downlink rate impact these times and affect average driving speed? Last year, the MOS and flight software teams built a discrete-event simulation to estimate these impacts on the driving speed, and we learned a few things about the proposed design. This year, we are developing a much higher-fidelity driving simulation and will greatly broaden the list of questions we can investigate with it.

E. Maturation of Processes and Tools through Frequent Use

This is distinct from simulations in that tools and processes are being used for real, not simulated, tasks. In practice this means using as much of the mission operations and ground data systems as is practical and possible during spacecraft development and test. This is also expressed as "test as you fly, fly as you test." In order to do this, we must ensure that our ability to integrate and deploy the operational software is as Agile as our ability to design and develop processes and tools.

To date, we have applied this with software and procedures. For the DOT, we used the same procedures for test and operations, with callouts for test specific sections. Procedures were developed jointly with the flight and ground teams, with help of the mission systems engineers to bridge the two systems. The project level emphasis is on system development and test in the early phases of the mission. The MOS team facilitates operational thinking and design by infusing operational processes early.

Where possible, we use operations software for flight system development and test. This means using the same software for telemetry and monitoring during flight system development and test as will be used for operations. We

also use flight-like GDS architecture as much as possible. For the DOT, the team who designed, integrated, and deployed the GDS was embedded in the operations team and worked closely with the flight team, enabling us to rapidly adjust and adapt as the processes and tools matured.

VI. Integrating Agile into the System Engineering and Requirements Cycle

NASA provides standard guidelines for project processes, including documentation and gate reviews. The RP project has tailored those guidelines to be appropriate for a class D mission. Individual project elements, such as MOS, GDS and Mission System Engineering (MSE) further tailor those guidelines. The MS, whether Agile or not, works within the project processes. The MS delivers project documentation, including requirements, interface control documents and system specifications.

Agile and standard project processes work together, with a back and forth flow of information between them. Agile processes, such as a simulation, may result in new or updated requirements. Requirements may impact or create a need for an Agile process.

VII. Using Agile for GDS Development and Deployment (DevOps)

For most of this paper we have intentionally focused on the application of Agile techniques to the MOS part of the Mission System. The application of Agile techniques to ground software is already well understood and covered in previous papers, and it might be assumed that the application of the techniques would follow a similar pattern when adapting them to MOS. However, there is an important difference when starting to apply Agile techniques to MOS. While one can apply Agile techniques to the development of the GDS (the systems and software that support MOS) without applying it to the rest of the Mission System (the people and processes of MOS), the reverse approach of applying Agile techniques to MOS (people and processes) without applying it to the GDS limits testing to paper simulations. Once the MOS begins operating in a more Agile fashion, it demands Agile GDS integration and deployment methods so that the software and systems can be made available much more rapidly than traditional methods allow. The only effective way to mature the system and demonstrate capabilities while using Agile methods in design and development is to also use Agile methods in integration, deployment, and system administration of the systems being used.

For the DOT, our approach for dealing with this issue was very simple: the team members responsible for integration and deployment were embedded in the development and operations team. Since the DOT was a bounded field test, this approach worked well in enabling rapid collaboration and a holistic view of the system for the team. We used relatively light-weight, mostly manual processes to perform integration and testing. However, those processes will not scale well to the pace and scope of the upcoming RP mission phases, especially when factoring in distributed operations.

To keep up with Agile design and development, the GDS must be updated in a timely fashion across multiple operations locations, while also maintaining high quality with each iteration. We are in the process of prototyping a system based on industry tools (Kickstart, Ansible and Docker) that will enable continuous or near-continuous integration testing and deployment. The framework will allow us to integrate and deploy the software rapidly to a new environment, and also allow us to bundle the environment with the software in a self-contained module, or container. This approach provides flexibility and avoids the standard problem of finding out that a software application is missing a needed dependency when deploying to a new hardware environment. This will enable the GDS to rapidly and effectively support the MOS throughout the mission, from design through flight.

This approach is known as Agile System Administration, or DevOps. It takes the core principles of Agile and extends them beyond development to the integration and deployment phases. Similar to the Agile software approach, DevOps approaches emphasize collaboration across disciplines, a focus on successful outcomes over artifacts, an embrace of change, and holistic systems thinking. As Agile has transformed software design and development culture, so DevOps is transforming integration, deployment, and on-going, rapid update of operational tools and processes. Working together, these Agile methods can enhance the way we do mission systems from design to flight.

VIII. Conclusion

We have taken Agile principles originally evolved for software development and are applying them across the Mission System for one NASA mission. The principles apply to both the Ground Data System and to the Mission

Operations System (people and processes). We are tailoring the principles and focusing on where we believe they will provide the most value, specifically to the unique design challenges of the Resource Prospector mission. Our results from the DOT allowed us to make great progress early in the mission towards reducing risk and costs. Targeted experiments both reduce risk and support continuous learning. We have been able to assess and adjust required flight capabilities unusually early in the life cycle of the project. Further, by deploying and testing with an integrated operations and flight system, we are rapidly maturing the processes and tools. We plan to continue to apply and adapt these principles throughout the project phases, and we expect to continue to realize benefit.

References

[1]Trimble, J. "Lean Mission Operations Systems Design - Applying Lessons from Agile and Lean Software Development to Mission Operations Design", *SpaceOps 2014 Conference*, SpaceOps Conferences, (AIAA 2014-1816). http://dx.doi.org/10.2514/6.2014-1816

[2]Trimble, J. "Agile Development Methods for Space Operations", *SpaceOps 2012 Conference*, SpaceOps Conferences, (AIAA 2012). http://dx.doi.org/10.2514/6.2012-1264554

[3]Marshall, W. et al., 2011. Locating the LCROSS impact craters. *Space Sci. Rev*., 1–22, ISSN:0038-6308

[4]Quinn, J., J. Smith, J. Captain, A. Paz, A. Colaprete, R. Elphic, and K. Zacny. "Resource Prospector: The RESOLVE Payload." USRA Houston. Lunar Exploration Analysis Group (2015), 22 Oct. 2015. Web. 25 Mar. 2016. <http://www.hou.usra.edu/meetings/leag2015/pdf/2046.pdf>.

[5]Beck, Kent et al. "Manifesto For Agile Software Development". *Agilemanifesto.org*. N.p., 2001. Web. 29 Apr. 2015.