ALTEN TECHNOLOGY

APPLYING AGILE TO UNCREWED VEHICLE HARDWARE DEVELOPMENT CHALLENGES

by
Stefan Elsener: Program Manager
James Kolozs: Staff Systems Engineer
Erica Wilder: Program Manager

UPDATE 26 March 2024

EXECUTIVE SUMMARY

Robotics and uncrewed systems have reached a convergence between legacy development strategies in the aerospace and defense industries and rapid commercial product development cycles. How will developers meet the need for shorter schedules and frequent product refreshes while preserving the reliability of highly complex systems, all within demanding budget constraints?

ALTEN Technology has demonstrated success combining rigorous waterfall development planning with the flexibility and speed of Agile Scrum daily execution. This paper discusses applying these integrated practices to recent robotics hardware development projects by overcoming key challenges: organizing people, enhancing collaboration, managing complexity, handling change, and reducing risk.

By integrating Agile Scrum into hardware product development, a team can successfully overcome these common challenges to achieve the best version of their product while minimizing cost and development time.

INTRODUCTION

Agile has been actively used by software developers for more than 20 years and is now being embraced by hardware developers as well. Agile embodies an underlying philosophy for software development without regard as to how it is implemented. However, the philosophy provides a set of core values and principles to guide development.

AGILE VALUES

When it comes to values, the Agile Manifesto[1] states the following (emphasis added):

"We are uncovering better ways of developing software (products) by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software (product) over comprehensive documentation

Client collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

The Agile values derived from the Agile Manifesto tell us what values to fall back on when we are faced with a difficult situation, and they guide our high-level thinking about a project.

AGILE PRINCIPLES

The Agile Manifesto also provides a set of 12 principles that describe in more detail the incorporation of Agile values into a software development organization. We have taken these principles, thought deeply about how they apply to both hardware and software development, and devised the following product development principles based on Agile:

- **Demonstrate Value Often**—The principle that something of value is delivered and demonstrated at every sprint review. This reduces risk and shows tangible progress is being made. Useful feedback is provided when there is still time to incorporate it. The stakeholders and team members are aligned about how the product works and the direction it is heading. The team is accountable for producing value in every sprint.
- **Embrace Change**—The principle that it is impossible to know all the requirements for a product up front. There is a necessary amount of learning that will occur during a project, both internally (project team uncovering things about the system) and externally (better understanding the market needs), but this does not mean blindly embracing changing requirements. We have to understand that changes cost money relative to the budget and time relative to the schedule. Any change must be justified. Not only can requirements change, but the way we do product development can change, even within a project. Project change derives from the daily stand-ups, sprint retrospectives, stakeholder feedback, and lessons learned along the way.
- **Collaborative Empowered Teams**—The principle that we strive to include the entire engineering team in how a project unfolds. We also strive to include stakeholder input regularly throughout the project. This keeps everybody focused on a common goal and minimizes divergence of thinking. People work best when they have the time, authority, and will to make things happen. When you treat your people well, respect their input, and give them the time and tools

to be successful, they will happily work hard for you. Teams encourage a variety of ideas, serve as a check and balance on individuals, facilitate self-review, and create shared ownership of the project.

- **Technical Excellence**—The principle to incorporate technical excellence in all of your design and development activities. Address bugs, issues, and problems as soon as you find them; don't leave them for someone else to deal with later. Capture your best practices and lessons learned, and distribute the knowledge. Avoid non-value-added work. Create templates, and preserve good examples so that future work is done more efficiently. Regularly refine your processes, tools, and templates. Tailor the activities and deliverables of each project on a case-by-case basis. There is no one-size-fits-all process in product development.

INCREMENTAL DEVELOPMENT

Scrum is one of the most popular frameworks for implementing Agile.[2] However, it was created with software in mind. Modified Scrum is our preferred product development methodology based on Agile. Why modified? After experimenting with "pure" Scrum for hardware, we found it wasn't 100% compatible.[3] Other authors have found this issue as well.[4-8] While the changes are minor, they are important. In effect, they boil down to incorporating sprints into an overarching project plan and loosening the definition of "working product" at the end of a sprint for demonstration purposes.

Incremental development implements a key characteristic of Scrum project execution by breaking up the traditional product development phase into smaller, more manageable efforts. The difficulty of a long-duration phase is that the work details are lost and the actual status is obsolete almost immediately once the phase starts. With incremental development (Figure 1), we take a large and ungainly phase and break it up into sprints.

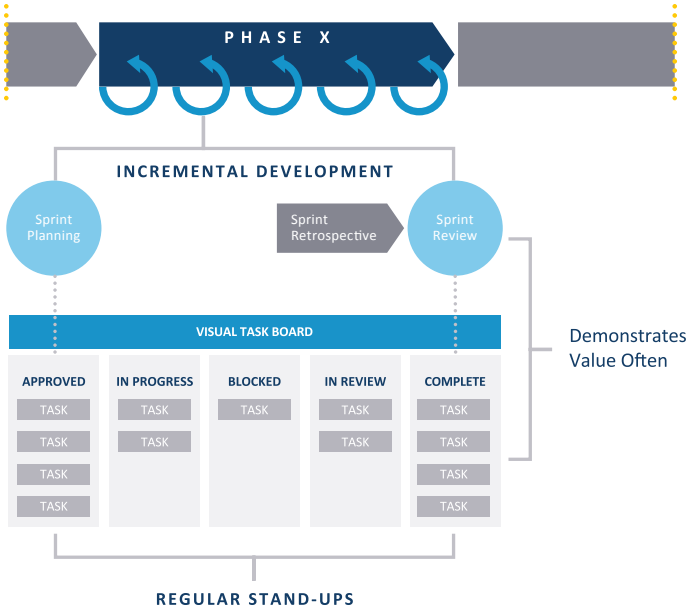


FIGURE 1. INCREMENTAL DEVELOPMENT PROCESS[3]

Each sprint is run like a miniature project with a defined scope and goals for a set of tasks that need to be accomplished. Not only does the team get a regular sense of accomplishment, but the stakeholders receive tangible, actionable value more often. Having an incremental cadence and focus on collaboration across all stakeholders addresses change in real time and avoids a buildup of technical debt that might not be reconciled until the next major design review. This overall approach provides multiple benefits:

1. It facilitates team and stakeholder alignment.
2. It addresses issues quickly.
3. It provides transparency about project progress to the stakeholders.
4. It facilitates continuous improvement of the product and product development process.
5. It reduces risk.
6. It delivers value to stakeholders on a regular basis.

APPLYING AGILE

Our experience with applying modified Scrum to uncrewed systems programs enables a high degree of collaboration, delivering value continuously while anticipating change and adjusting plans gracefully, at a high standard of quality and capability. Every product development effort encounters a variety of challenges as it progresses. This paper will describe how application of Agile principles via modified Scrum tools can help tackle the key challenges faced by complex uncrewed systems programs:

- **Organizing people** is the challenge of estimating, defining, and ramping up a project team.
- **Enhancing collaboration** is the challenge of coordinating multiple project teams, often from different organizations, and making sure everyone has a common understanding of the product direction.
- **Handling change** is the challenge of maintaining respect for the project scope while being open to changes that will improve the product or product development process.
- **Managing complexity** is the challenge of dealing with multiple subsystems and disciplines, hundreds of requirements and interfaces, and thousands of tasks.
- **Reducing risk** is the challenge of understanding the risks to the project and the product and having a plan to deal with them.

CHALLENGES

Complex programs like those in the uncrewed systems industry will face stumbling points during the product development effort. The strategies described below recommend pragmatic approaches based on Agile principles to effectively assess, resolve, and move on from these common challenges.

ORGANIZING PEOPLE

Most product development organizations have multiple projects in play at any given time. Additionally, these organizations are often beholden to a variety of stakeholders with competing priorities (e.g., investors, managers, clients, and vendors) and are limited by a fixed pool of resources (e.g., staff, funding, budget, space, and equipment). Planning for and organizing your team is the first step to ensuring the project has the right foundations to succeed.

To effectively plan for and ramp up team members, we need to understand the scope, schedule, and skill sets required for a project. Yes, Agile promotes self-organizing teams, but the reality for smaller or highly mixed programs is there is often competition for resources. Providing a quantifiable plan up front helps decision-makers prioritize work and resources most efficiently.

Some projects have the benefit of long durations where existing team structures are well exercised and processes are proven to be effective. However, most projects are smaller and can benefit from Agile planning to start nimbly and scale efficiently. When planning to leverage Agile techniques in a hardware development effort, determining the appropriate skill sets and personalities required on the development team is crucial. Considerations for technical expertise must be weighed against personality traits and leadership styles to optimize team member deployment at the right time with enough backlog and preparation completed to enable team members to be effective.

One way ALTEN Technology approaches project management differently than traditional Scrum is our definition of “self-organizing teams.” In an ideal project

world, you would have every required expert, at the precise moment needed, to get all the work done efficiently with no downtime. Reality often forces us into compromises to demonstrate progress, establish confidence, and deliver value to stakeholders as soon as possible, within the budget and resource constraints placed on us. In this situation, Agile principles guide us to ramp up quickly (Demonstrate Value Often) to build a backlog of well-defined work, which reduces the need for more experienced personnel on every task, as long as the team structure (Collaborative Empowered Teams) and support resources are available to ensure technical excellence.

We find the most efficient approach to team organization is to establish comfortable leadership ratios and drive individual development tasks to the lowest resource level possible given the task's requirements for success. ALTEN Technology's Agile implementation promotes this method of delegation to keep the doers on the team unblocked and engaged at all times, while maintaining leadership ratios at the minimum level required to keep project risk manageable and resolve challenges quickly and effectively. The key is to start with a framework that allows for change and scalability, fully anticipating that as the project ramps up, roles and available resources can and will change to be most effective.

In our experience, small project teams of one or two resources per discipline, or area of expertise, can generally get by with a sole leader/doer or even a doer with the appropriate project management or technical oversight (Figure 2).

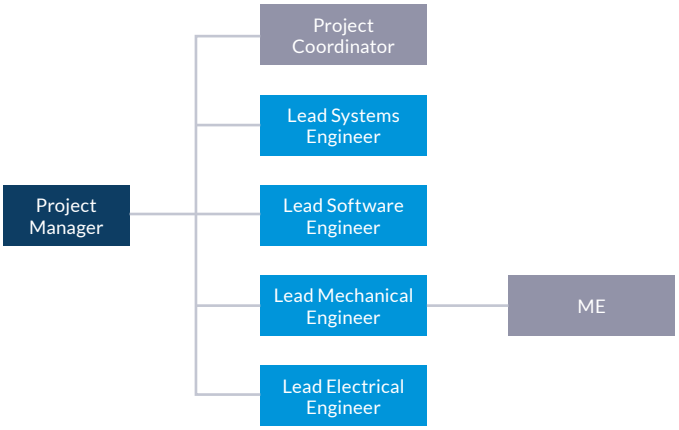


FIGURE 2. EXAMPLE OF CORE TEAM STRUCTURE

However, once a project team reaches three to five resources per discipline, strong technical leads must be identified and, more importantly, empowered to support task planning and prioritization and serve as the subject matter expert (SME) for that discipline. Additionally, once a given discipline grows above five resources, an additional "lead" is advantageous so as not to dilute the team's ability to fully define tasks, resolve issues in a timely manner, and react to the needs of the project as it evolves (Figure 3).

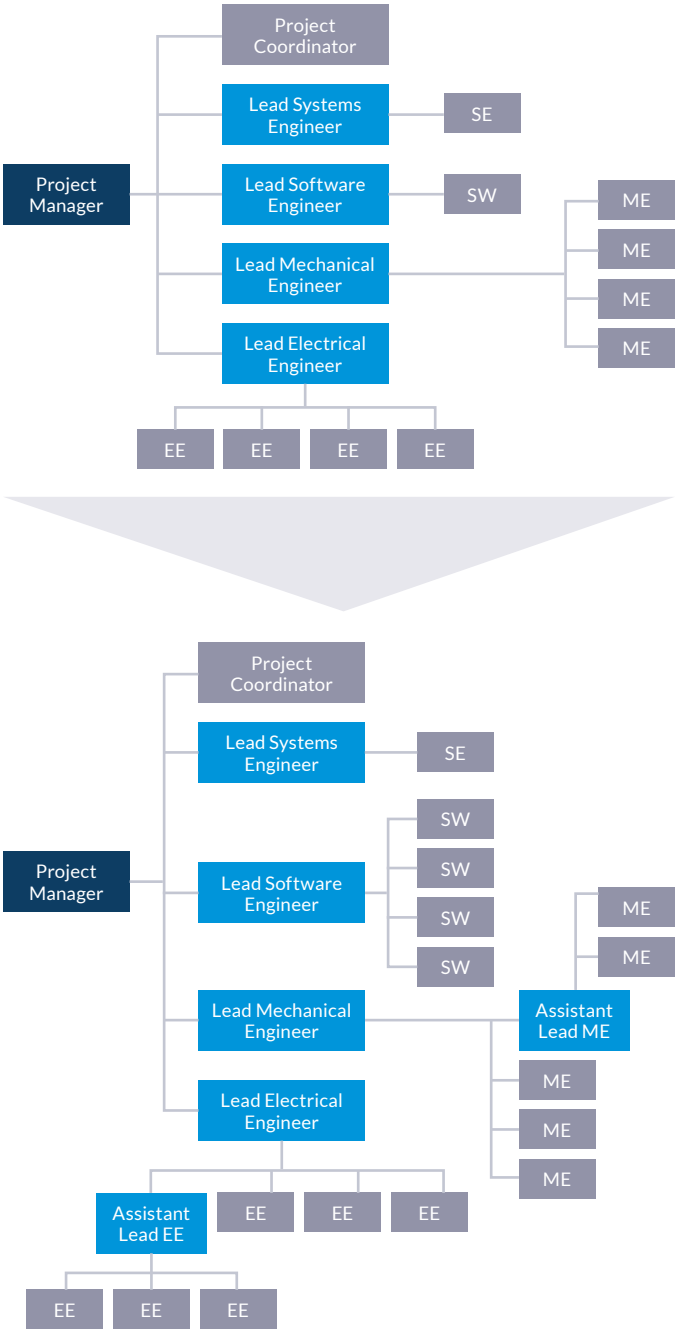


FIGURE 3. STRUCTURE
The core team provides a clearly defined structure as the team grows

This scalable team structure sets the stage for effective collaboration by defining roles and responsibilities. Additionally, we monitor for trigger points where the team may need to grow but always keep leadership and support structures in mind to ensure our team's effectiveness and efficiency are maximized.

ENHANCING COLLABORATION

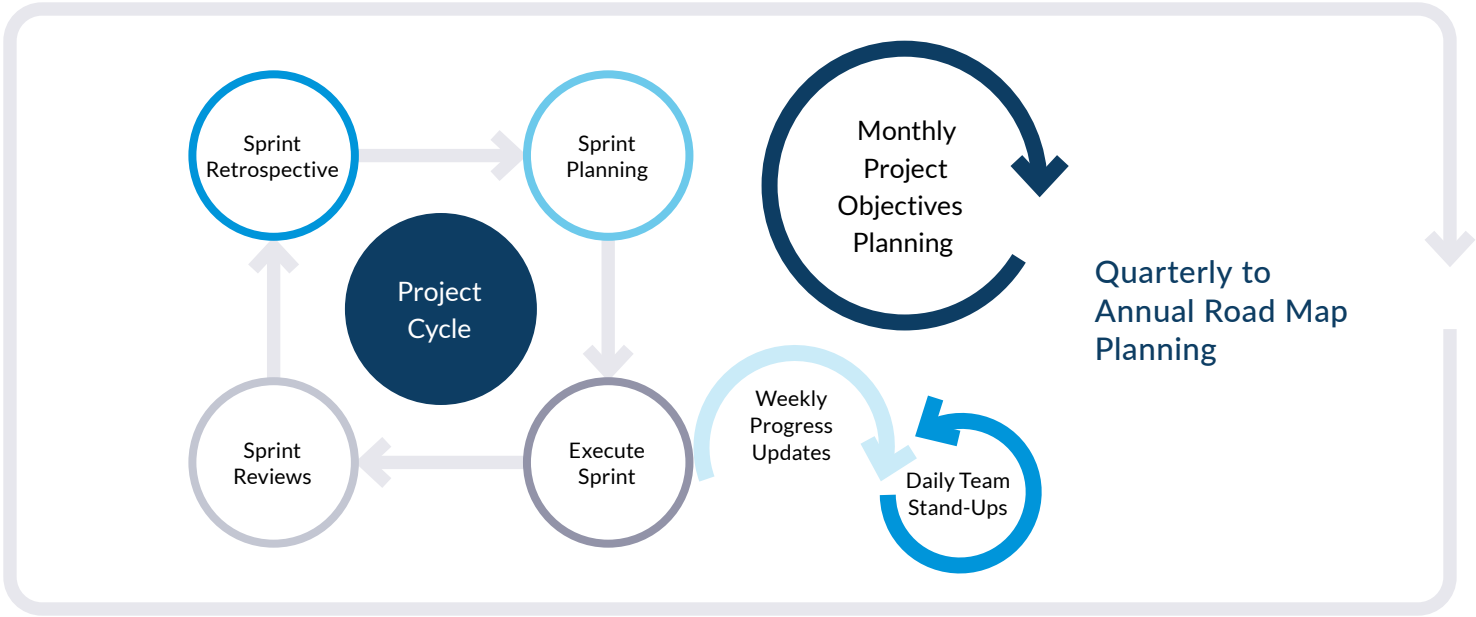
In today's global economy with talent distributed around the world, strategies for effective collaboration are more important than ever. Many organizations rely purely on reactive forms of communication to get by (i.e., emails or conference calls as needed). ALTEN Technology prefers a proactive communication approach to break down barriers to collaboration and aggressively encourage team members to communicate without delays. We use various tools and techniques to encourage collaborative teams from the start and ensure our project teams and partners are communicating effectively.

Most of our programs include collaboration with our clients (usually a product owner or prime contractor), other third-party engineering service providers (much like ALTEN Technology), and manufacturing partners and component suppliers. ALTEN Technology uses Agile methods during project execution to keep projects and team members moving, communicating, and aligned. Tools such as visual task boards help provide a central location to consistently plan, assign, and track progress on project tasks. More importantly, regularly scheduled gatherings promote collaboration and ensure team members are accountable to demonstrate value and deliver technical excellence to our partners. In general, we subscribe to the following cadence of important planning discussions, which Scrum calls "ceremonies" or "events," detailed below and shown in Figure 4.[2]

(**Organization leadership**) Quarterly or annual product road map planning

- (**Project leadership**) Monthly project planning for upcoming objectives (next two to three sprints)
  - (**Project leadership and SMEs**) Biweekly sprint planning of detailed tasks to meet upcoming sprint objectives
  - (**Project team and stakeholders**) Biweekly sprint reviews to demonstrate completed tasks
  - (**Project team**) Biweekly sprint retrospectives—may include key stakeholders if appropriate
    - (**Project leadership and SMEs**) Weekly status updates with project stakeholders—and key suppliers/vendors if needed to monitor progress
      - (**Project team**) Daily team standups to identify and resolve issues and ensure team members understand priorities and next steps

FIGURE 4. PROJECT LIFE CYCLE ELEMENTS



Without close collaboration, there is the risk of missed expectations between stakeholders and the project team. Everyone, stakeholders and team members alike, has a mental representation of what the product will be like at the end, and rarely do these representations actually align. The longer a project goes without resolving these representations, the further they will deviate, which means that when stakeholders see the first major prototype, they may disagree about how it was "supposed" to look, function, or interact. This often leads to a significant rework or grudging acceptance of a subpar solution.

Our Agile principle of Demonstrate Value Often directly addresses this issue. We do this via sprint reviews and demonstrating anything tangible to the stakeholders during these regular meetings (we prefer every two weeks). This allows everyone to recalibrate their internal model of the system so that any mismatches will come out in the sprint review. We can then refine the direction during sprint planning and the subsequent sprints. When the next sprint review comes around, we are a step closer to the final product, and everyone can see the progress based on the prior feedback. This moves the focus of integrated (and expensive) prototypes from seeing if they match the stakeholders' expectations to seeing how well the design conforms to the requirements. Thus, the integrated prototype should never be a surprise to the stakeholders (save the "big reveals" for marketing presentations).

We have also found regular integration meetings to be extremely helpful in finding and fixing design problems before they become expensive prototype issues. We completed an uncrewed vehicle project where we were in charge of mechanical packaging of components but another company was tasked with designing the printed circuit board assemblies. Given the extremely tight packaging constraints and the fact we had never met the other company, we had to ensure everything would fit. Integrating the other company's CAD models into our designs was obvious, but what about the details? Where is the primary coordinate system? Who is in charge of the profile and hole locations? What about component heights and routing of cables and connectors? How will we deal with changes? To coordinate all these design details within half a dozen subassemblies, we held focused integration meetings during the design phase. Each team was able to communicate (and show) what was happening at critical interfaces and decide what adjustments were needed. This enhanced collaboration and ensured the prototypes came together cleanly.

Enhancing collaboration within Agile methodologies allows us to deliver tangible value to the stakeholders more often. Breaking up phases into smaller increments and measuring actual work completed as we go helps us understand what is remaining on a project. Perhaps more importantly, though, it helps us predict and communicate a more accurate completion date as the project unfolds. This provides our stakeholders insight into the product development process every step of the way and allows our teams to react early to issues identified by the stakeholders. But how can we predict and communicate when we will be done if we are following Agile?

Since uncrewed systems are following similar paths toward regulation that enabled safe product launches in the aerospace and medical device industries, pairing the flexibility of Agile techniques with proven strategies from traditional hardware development is the recipe for success. Traditional project management approaches are still leveraged to continuously maintain a high-level program schedule and resource plan, which is updated and communicated at each sprint review. Formal design reviews and other gating milestones ensure alignment is maintained among all stakeholders prior to moving on to the next phase of the project.

All of these collaborative tools give us and our stakeholders the rapport and confidence to communicate effectively, particularly when changes are brought up on a project.

HANDLING CHANGE

Change is a given within product development, and it is critical to balance technical improvements while maintaining project budget and schedule. Dealing with change on a project with hardware elements can be even more troublesome because of the high cost of change as the project unfolds. Leveraging an Agile systems engineering approach ensures critical artifacts such as requirements, interfaces, and risks are monitored and maintained to ensure project success. Embracing change is about assessing and understanding the impact any given change has on the project objectives and product capabilities in order to make informed decisions while working within the project constraints.

ALTEN Technology handles change by empowering our teams to identify and communicate issues as soon as they arise. The incremental development approach (discussed previously) is a mechanism for teams to address changes and avoid building up too much technical debt. Addressing change as early as possible leads to a lower development cost over the life cycle of the project (Figure 5).

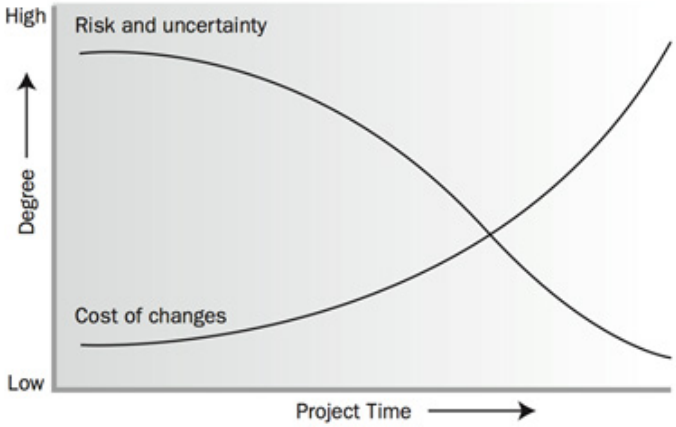


FIGURE 5. COST OF CHANGE OVER THE PROJECT LIFE CYCLE[9]

Monitoring scope change is a critical task that ALTEN Technology handles because it can affect all aspects of a project. The project manager closely monitors scope using a variety of Agile tools. Visual task boards are the main tool of choice, where individual tasks are planned and tracked. The high-level work breakdown structure (backlog) is roughly planned into future sprints. Every month, we plan the goals for the next two to four sprints (one to two months) in preparation for detailed biweekly sprint planning sessions.

The large backlog items are then broken down into well-defined tasks that engineers can complete. By using the visual task board in this way, any new scope requests are added to the board, where it's easier to see how the new scope fits in with the existing tasks. Stakeholders are encouraged to review the visual task board as well. We can then flexibly assess if the new scope can fit into the existing sprint plan or if an existing task should be deprioritized (delayed) or removed to make room for the new task within the existing project constraints.

However, one way Agile for hardware differs from software is we don't require tasks to be completed within a given sprint. This is the goal, of course, but if a task is not completed at the end of a sprint, we don't want to incentivize teams to artificially "complete" the task by shortcutting technical excellence or arbitrarily splitting the acceptance criteria just to "move a card across the board." Uncrewed systems programs are complex enough as they are, and we don't want to introduce artificial process requirements that could make them harder to keep track of.

As a team settles into a new project, they will find both new ways to work together and new ways to have conflicts. Sprint retrospectives are a wonderful tool to improve the process of product development because it will be slightly different for each group of people brought together to form a product development team. In a retrospective, the team has a roundtable discussion about what did and didn't go well on the project and possible ways to improve the process for the next sprint. A basic expectation exists that there will be friction along the way, and the retrospective is a way to expose it and improve upon it. The process itself is hardwired to embrace change, with benefits to improved team communication, efficiency, and cohesiveness.

MANAGING COMPLEXITY

As projects grow in size, they also grow in complexity. But why is it that a project that is twice as large (in staff, budget, or number of features) always feels much more than twice as complex to deal with?

Think of a project as a group of "elements," whether people, requirements, features, or components. Each of those elements has a relationship with a number of other elements. As the number of elements grows, the number of possible relationships grows even more. This

is a measure of the project's complexity (see Figure 6), and both the elements and the relationships have to be managed as well. This is one reason why projects can easily be underestimated by simply scaling smaller projects. Scaling a project does not incorporate the added complexity of the interactions within a larger project.

Uncrewed vehicles are a great example of a lot of complexity in a small package. There are a number of interacting subsystems, various technologies, and different engineering disciplines that all must work together to produce any uncrewed vehicle. In addition, there are related systems, such as ground control stations, support equipment, and operators, that add to the complexity. So how can Agile be used to manage the complexity of an uncrewed vehicle project?

Each of our projects has a systems engineer (sometimes, more than one) who acts as the product owner for the team. Briefly, in Scrum, the product owner is responsible for defining the product backlog (requirements), managing and prioritizing the product backlog, making sure the team produces the most value at any given time, and keeping everyone aligned on the interpretation of the backlog items.[2]

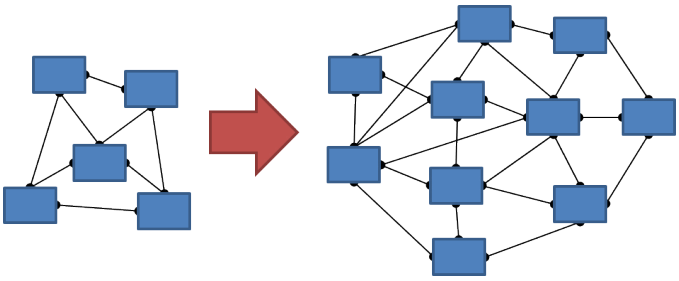


FIGURE 6: AS A PROJECT GROWS IN SIZE, IT GROWS IN COMPLEXITY

Our systems engineers also take on additional responsibilities in a project. They conduct or assign trade studies, guide the team to a common product architecture, determine and coordinate the definition of critical interfaces, and build a model of the system (via model-based systems engineering) so that it is easier to understand and measure the effect of making changes.[10]

Why are systems engineers so critical in managing project complexity? Because they do the jobs everyone assumes someone else is doing (Figure 7). Collaborative Empowered Teams is one of our key Agile principles. When engineers are focused on their particular disciplines, in a particular subsystem, they don't have the bandwidth to look at the whole system and ensure it makes sense. Product architectures often tend to grow organically, which may make sense at first, but later on it becomes clear the product doesn't make sense as a whole, even though many of the individual elements do. This is a symptom of project silos or a lack of coordination at the top levels of the project.

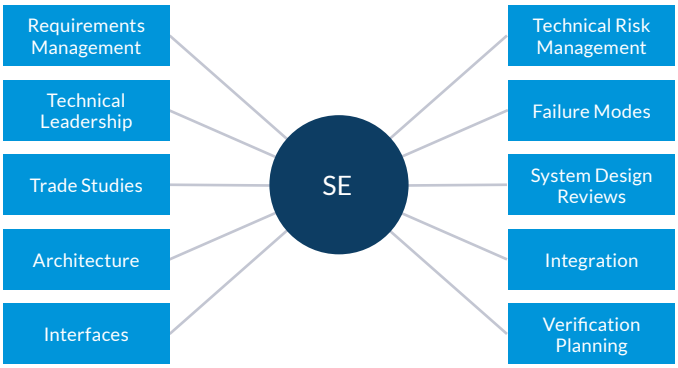


FIGURE 7: A SYSTEMS ENGINEER MANAGES THE SYSTEM[11]

Having an individual tasked with engineering the system (the systems engineer) frees up project engineers to focus on their disciplines and subsystems. The systems engineer arranges collaboration meetings with team leads with the goal of fleshing out the system-level concerns. Early on, these meetings will focus on decomposing, analyzing, and understanding the requirements. Trade studies will be coordinated to determine the best architecture for the system or various subsystems and major components. The focus will then shift to architecture and interface definition. Having a systems engineer onboard ensures that these activities occur at the appropriate times, that they are coordinated to follow an overall plan, and that there is an impartial arbitrator when difficulties or conflicts arise.

A visual task board is another way to manage project complexity. As a project ramps up, the number of tasks to be tracked increases tremendously. It is a common occurrence for individuals to end up assigned to multiple tasks simultaneously. When that happens, they become extremely inefficient at all of them.[12]

The visual task board shows what each team member is working on and allows us to set work in progress limits (see Figure 1 for a representation of a visual task board). We also have a set of acceptance criteria for each task so we (and the assignee) have a common definition of when it is "done." Not almost done, not mostly done, not 90 percent done, but DONE.

The measure of progress is not how many tasks you have started; it is how many tasks you have finished. Additionally, it is not uncommon for a task to become blocked so that the assignee cannot make useful progress anymore. This is also tracked and expedited so that blocks don't prevent other tasks from being completed or languish in obscurity until they have a potential impact on the project. Using visual task board software, we can also assign tasks to releases, subsystems, and disciplines. This allows filtering and prioritizing of the workload to occur across the project scope.

All of these elements of a visual task board make managing the project far less complex by limiting the amount of information that needs to be processed at any given time. Everyone has the same understanding of what is happening on the project, what happens next, and where the issues are. In addition, we empower our teams through the visual task board. While the project manager and systems engineer work together to set sprint goals (in the context of the overall project), the leads and team members are in charge of decomposing backlogs into tasks, defining them in detail, and providing estimates. The team members move their cards as they work on them, rather than being micromanaged by the project manager. This invests the team in the purposeful progression of the project because all team members can see their contribution on the board.

We have also been successful in managing project complexity through staggered subsystem development. It is often impractical to ramp up a large team and start working on all subsystems simultaneously. Hiring considerations, a massive burn rate, and project risk can hinder such a team. Instead, a core multidisciplinary team (as described in the Organizing People section) can be assembled to work on all subsystems. The trick is to start with a requirements and architecture phase, develop a really good understanding of the system needs, and address higher-risk or longer lead-time subsystems first. With the architecture and key interfaces defined, the core team can get to work on one subsystem at a time.

These subsystems must fit within the original architecture and respect the interfaces. And once subsystem development starts to ramp down, another subsystem begins to ramp up. The core team moves from subsystem to subsystem in this manner. On more complex subsystems, we can bring in additional team members. On simpler subsystems, we may run more than one subsystem in parallel. All the while, we are demonstrating the value of each subsystem and making tangible progress along the way without having to have a fully operational system. The systems engineer is there to make sure that subsystem designs are coordinated with the system architecture and interfaces.

The drawback to staggered subsystem development using a core team is that it may seem to take longer than running a full project team from the start. However, the core team approach will likely produce a higher-quality product with fewer integration issues because the team members will be the experts in the system as a whole, having worked on every aspect of it. The staggered approach allows for a more reasonable burn rate and a lower risk profile for the project as a whole.

REDUCING RISK

In terms of complexity, risks move in tandem with project growth. Some of these risks will be obvious at the outset, and others may be discovered along the way.

Some risks can affect the project's success, while others affect the product itself. Either way, the project team will need to deal with risks as the project progresses. The question is, Do you anticipate and deal with the risks preemptively or let things unfold and handle risks as you go?

We are big advocates of risk-based product development and believe that Agile points us in this direction as well. But why? Why not just let the project unfold and deal with things as they come up? Isn't that Agile?

Letting risks impose themselves upon the project is a recipe for chaos, confusion, and expensive and time-consuming rework. It makes for a dissatisfied project team because they go from fighting one fire to the next and for unhappy stakeholders who do not understand why the project can never stay on track. Just-in-time risk management is decidedly un-Agile for all these reasons.

We use Agile principles to reduce risks by identifying and tracking them right from the beginning of the project. Risks are collaboratively brainstormed by the team, prioritized, and shared with the stakeholders. Each risk has a responsible team member, and tasks are assigned to them to complete to mitigate the risk. Risks are regularly reviewed and their progress tracked. For significant risks, the project may even be restructured to deal with "project killers"—those risks that, if realized, could potentially derail the project.

For example, imagine a new technology that may not yet have become a product but still needs to be incorporated into a relatively low-risk system design. Perhaps it's a new type of sensor that needs to be integrated into an uncrewed platform. While the project may seem low risk as a whole, if the sensor subsystem fails, the whole project fails. In this case, it is probably worthwhile to restructure the project to get a working prototype of the sensor subsystem. The team will learn a lot along the way about the performance and limitations of the sensor subsystem, and stakeholders will have information in hand to determine if it is still worthwhile to pursue the project as a whole. Yes, this takes an investment up front, but imagine if the project were turned on with a full team from the outset. A lot of time and effort would be wasted if the sensor subsystem were built in and ultimately found not to work as needed six months into a full-team effort.

As mentioned earlier, project risks are brainstormed right from the beginning. These risks are added to the risk tracker, a spreadsheet that lists and prioritizes the risks in a single place. Without it, we find risk concerns are spread throughout a project, often in people's heads. The team (and stakeholders) have little understanding of the risk profile of the project and where the responsibility lies for dealing with the various risks. The risk tracker makes all this obvious to everyone, including the stakeholders.

Having a risk tracker is nice, but it is useless if it does not positively influence the project. Each risk has an assignee who is responsible for following through with the chosen risk responses and reporting regularly to the team on progress. Often, this is a full-time task, so it is assigned as a task(s) on our visual task board. That way, it is clear that the risks are being addressed and time has been allocated to do so.

A summary of the risk tracker is presented at every sprint review. We present the following at sprint reviews regarding risks: review any changes to risks (new risks, closed risks, or risks that change priority level), review any risks that are coming due, and review the risk charts (Figures 8 and 9).

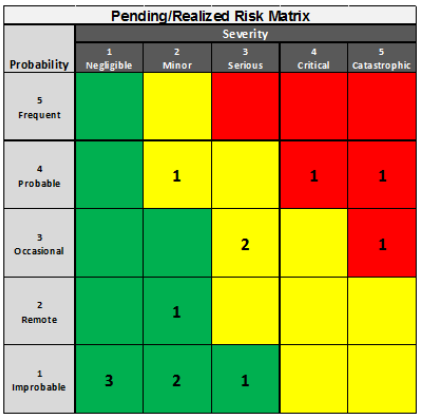


FIGURE 8:
RISK MATRIX CHART

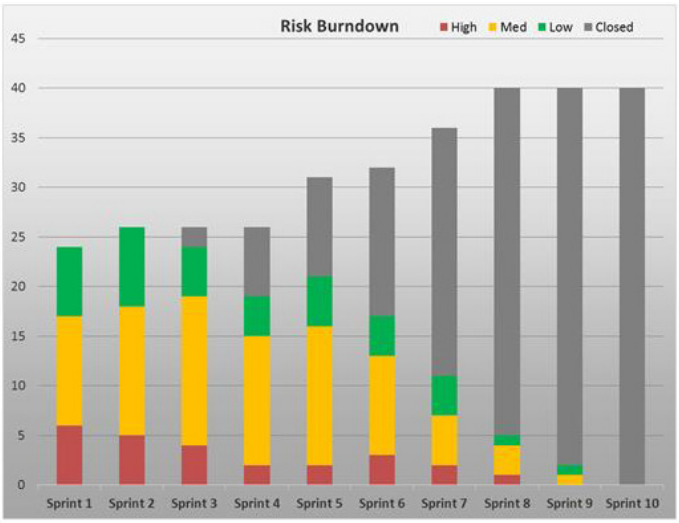


FIGURE 9:
RISK BURNDOWN CHART

The risk matrix gives a visual snapshot of the current risks that potentially could affect the project. In the example above, although most risks are in the lower left, they are not the ones to be overly concerned about. One should be very concerned, however, by the three risks in the upper right. The red risks have the potential to derail the project and resources should be applied to close them as quickly as possible.

The risk burndown chart is an ongoing history of the project's risks. Initially, these charts aren't too interesting—they just show the number of risks at each level. However, as sprints go by, the charts should start to show a pattern of risks burning down. Ideally, the higher risk items are burned down first because they have the greatest potential impact to the project. If the trend does not look encouraging, this may be the first sign a project is out of control and serious discussions need to be had.

Thus, the status, number, and priority of risks are never far from anyone's mind. By evaluating the project's changing risk profile at each sprint review, informed decisions can be made in a proactive manner, often before risks morph into major project issues. This gives the team a crucial bit of breathing room to thoughtfully consider options and then redirect tasks in the sprint planning meeting. The team knows who is working on it, how long it is expected to take, and what the definition of "done" is. Risks are no longer lurking in a project waiting to derail it at inopportune times.

CHALLENGES

By applying the four Agile product development principles, we demonstrate value to stakeholders often, empower the team to work in a collaborative way with all partners, remain flexible to change when necessary, and maintain a high level of technical excellence. All of this allows us to effectively overcome the many challenges that product development throws at us.

We have explored five challenges faced by uncrewed systems programs and how we deal with them in an Agile manner. The challenge of organizing people is handled using a core team that is scalable and sets the stage for collaborative empowered teams. Enhancing collaboration is a challenge that is met with incremental processes, from the organizational quarterly road map down to the project daily stand-ups. By demonstrating value within each of these cycles, collaboration is enhanced within the team and with stakeholders. The challenge of handling change is met via visual task boards, sprint reviews, and sprint retrospectives. The team is empowered to bring up potential changes, and the flexibility to change is built into our product development philosophy and processes. Managing complexity is a challenge we tackle by embedding a product owner (systems engineer) within each program. The product owner handles the system-level tasks that others don't have time to consider. By using visual task boards and staggered subsystem development, value is demonstrated often, and the amount of information that has to be processed at any given time remains manageable. Finally, the challenge of reducing risk is dealt with by empowering teams to identify risks, having a centralized risk tracker, and aggressively burning down risks by priority.

The quick feedback loops of incremental development via modified Scrum are imperative as technology evolves at an increasing pace in the uncrewed systems market and organizations need to be able to react quickly to keep up. Adopting Agile techniques and a continuous improvement approach enables organizations to maintain focus on new technology innovation, while not losing sight of key goals to get a product through regulatory requirements and to market in an increasingly competitive environment. The shift to more commercial market demand and supporting emerging user needs that have not been considered before will be key to being competitive and first to market.

REFERENCES

[1] Kent Beck et al., "Manifesto for Agile Software Development," 2001, accessed November 9, 2017, https://agilemanifesto.org.

[2] K. Schwaber and J. Sutherland, "The Scrum Guide," 2017, in Scrum Alliance, November 2017, from https:/ www.scrumalliance.org/learn-about-scrum/the-scrum-guide.

[3] Todd Mosher, James Kolozs, Carissa Colegrove, and Erica Wilder, "Agile Hardware Development Approaches Applied to Space Hardware," 2018 AIAA Space and Astronautics Forum and Exposition, Orlando, FL. https:/arc.aiaa.org/doi/10.2514/6.2018-5233.

[4] Backblaze, Inc., "Application of Scrum Methods to Hardware Development," 2015, Backblaze.com, accessed November 9, 2017, https://www.backblaze.com/blog/wp-content/uploads/2015/08/Scrum-for-Hardware-Development-V3.pdf.

[5] E. Graves, "Applying Agile to Hardware Development (parts 1–7)," Playbook Blog, accessed November 9, 2017, https://www.playbookhq.co/blog/agile-hardware-development.

[6] N. Ovesen, "The Challenges of Becoming Agile: Implementing and Conducting Scrum in Integrated Product Development," (PhD diss., Aalborg University, 2012).

[7] P. Reynisdottir, "Scrum in Mechanical Product Development: Case Study of a Mechanical Product Development Team Using Scrum" (PhD diss., Chalmers University of Technology, 2013).

[8] K. Thompson, "Agile Processes for Hardware Development," 2015, cPrime.com, accessed November 9, 2017, https://www.cprime.com/resource/white-papers/agile-processes-for-hardware-development/.

[9] *A Guide to the Project Management Body of Knowledge*, 5th ed., Chapter 2.4.1 (Newtown Square, PA: Project Management Institute, 2013).

[10] N. Kass and J. Kolozs, 2016, "Getting Started with MBSE in Product Development," INCOSE International Symposium 26: 526–41, https:/onlinelibrary.wiley.com/doi/abs/10.1002/j.2334-5837.2016.00176.x.

[11] C. Black [now C. Colegrove], 2017, "The Impact a Systems Engineer Can Make in Medical Device Development," INCOSE International Symposium 26: 1584–96, https:/onlinelibrary.wiley.com/doi/abs/10.1002/j.2334-5837.2016.00247.x.

[12] M. Cohn, *Agile Estimating and Planning* (Upper Saddle River, NJ: Prentice Hall, 2005).