ALTEN TECHNOLOGY

AGILE HARDWARE DEVELOPMENT APPROACHES APPLIED TO SPACE HARDWARE

by
Todd Mosher, Ph.D.: Vice President of Engineering, AIAA Associate Fellow
James Kolozs: Staff Systems Engineer
Carissa Colegrove: Senior Systems Engineer
Erica Wilder: Program Manager

Updated 26 March 2024

ALTEN

EXECUTIVE SUMMARY

The Agile software methodology, commonly implemented using the Scrum framework, is a proven and popular approach to software development both inside and outside the aerospace industry. But is Agile an effective approach for developing space hardware? At first glance, Agile-based approaches seem incompatible with traditional space-hardware development methods, which have relied on rigorous processes and a clear development plan to reduce risk. Agile, however, is less structured and more nimble in responding to deviations from initial plans as the design matures. Because of the commercialization of space and the speed at which technology is evolving, traditional approaches can limit innovation. By contrast, Agile's advocates often rally against strict, top-down, and overly structured traditional development, which doesn't take into account the realities of changing product requirements and an evolving market. ALTEN Technology has used both methods and identified ways to get past the traditional-vs-Agile debate by blending the approaches for space and nonspace projects. Integrating these two approaches to multidisciplinary problems that have hardware and software elements, such as satellites, combines the benefits of both. ALTEN Technology implementation uses the overall structure provided by traditional approaches with the flexibility and accountability of Agile methods to perform daily and weekly execution. This paper describes the steps required to transform the traditional space-hardware development process into one that uses Agile as part of its daily execution. This approach has been validated through development projects with several clients in the space community, including MMA Design and Spaceflight Industries, as well as other large, traditional aerospace companies.

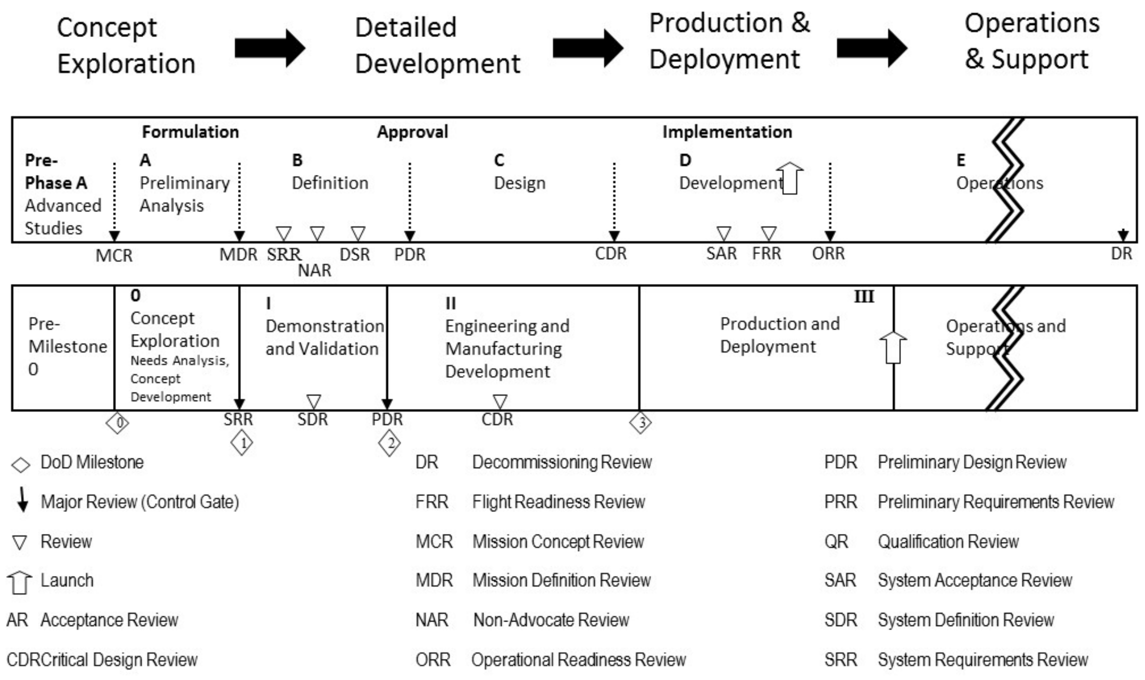
INTRODUCTION

Space projects are completed in phases, as shown in Figure 1. Over the years the government has used letters to designate its various phases (A, B, C, D, E) or numbers (0, I, II, III). However, as shown in Figure 1, the phases have similar definitions and usually end in a milestone review (SRR, PDR, CDR, etc.). More detailed descriptions of the traditional process and the phase descriptions can be found in [1–3].

This phased approach and the Gantt charts that accompany them are sometimes referred to as waterfall processes. The detailed schedules attempt to lay out every step in a project, including every milestone and every delivery date. Critics claim that these schedules are always wrong [4]. The assertion is that the charts are inaccurate the instant they are completed because most projects are too dynamic to successfully capture the actual schedule in real time.

The Agile Manifesto [5] has changed how a great many organizations develop and deliver software. Its success has inspired the product development industry to experiment with extending its principles to hardware design projects. However, there is often an incorrect assumption that applying Agile is synonymous with merely using its specific methodologies, like Scrum. As many have found [6–10], applying Scrum to integrated hardware and software systems is not straightforward. However, we have found that the underlying values and principles of Agile (see Appendix) can not only be successfully applied to integrated systems but also are an improvement on existing hardware development processes such as traditional systems engineering and the classic waterfall process.

FIGURE 1. TRADITIONAL GOVERNMENT PROGRAM LIFE CYCLE


Concept Exploration
Detailed Development
Production & Deployment
Operations & Support
Formulation
Approval
Implementation
Pre-Phase A Advanced Studies
A Preliminary Analysis
B Definition
C Design
D Development
E Operations
MCR
MDR
SRR
NAR
DSR
PDR
CDR
SAR
FRR
ORR
DR
Pre-Milestone 0
0 Concept Exploration Needs Analysis, Concept Development
I Demonstration and Validation
II Engineering and Manufacturing Development
III
Production and Deployment
Operations and Support
SRR
SDR
PDR
CDR
DoD Milestone
Major Review (Control Gate)
Review
Launch
AR Acceptance Review
CDR Critical Design Review
DR Decommissioning Review
FRR Flight Readiness Review
MCR Mission Concept Review
MDR Mission Definition Review
NAR Non-Advocate Review
ORR Operational Readiness Review
PDR Preliminary Design Review
PRR Preliminary Requirements Review
QR Qualification Review
SAR System Acceptance Review
SDR System Definition Review
SRR System Requirements Review

APPLYING AGILE TO HARDWARE PROJECTS

From our perspective, hardware design involves mechanical and electrical engineering. Although they require different disciplines and skills, mechanical and electrical design have similar product development processes, manufacturing costs, and lead times. In this sense, they are more similar to each other than to software (see Table 1). These differences with software are the reasons why attempts to apply Scrum to hardware development have met with resistance and difficulties [8], [9].

Despite the differences, the enthusiasm and success of our Agile software teams encouraged us to reexamine our hardware development process to determine whether it could benefit from Agile methods. In particular, we were interested in enabling integrated product development teams, those with both hardware and software, to adopt Agile practices.

The complexity and integration of modern products demand a coordinated product development effort rather than one design methodology for hardware and another for software. We have found that it is possible to combine traditional methods of product development for project structure with Agile methods for project execution. This provides us with a number of benefits:

- Faster product development
- Higher quality deliverables
- Continuous risk reduction
- Enhanced collaboration both within the development team and with stakeholders
- More transparent and accurate project status
- Enhanced accountability at all levels
- Timelier issue identification and resolution

TABLE 1. KEY DIFFERENCES BETWEEN HARDWARE AND SOFTWARE DESIGN FROM AN AGILE PERSPECTIVE

Attribute	Hardware (Mechanical and Electrical)	Software
General Process	■ Design, Refine, Prototype, Test	■ Create Test, Implement Code, Refactor ■ Repeat
Productivity	■ Production requires drawing creation, quotes, building, and assembly. ■ Production may take days to months, depending on the complexity of the product. ■ Note: Rapid prototyping techniques decrease duration but are still time-consuming relative to software.	■ "Production" requires compiling code to create an executable or image. ■ "Production" takes on the order of minutes to hours.
Component and Testing Costs	■ Prototype costs can be high (especially good quality prototypes). ■ In addition to labor, testing costs typically include custom test and production equipment, plus assembly and packaging	■ There are few or no parts or distribution costs. ■ Testing costs are primarily labor.
Modularity	■ There is a high degree of integration on most products to minimize size, weight, and therefore cost. ■ Modularity is usually designed only for known changes/upgrades.	■ High modularity is encouraged as a best practice because size, weight, and computing cost are generally not a competing constraint.
Cost of Change	■ Most changes become exponentially more difficult to implement the later in a project they occur (especially when they affect production or test equipment).	■ If well-architected, the cost of change is mostly stable over a project. ■ Labor is the only cost.
Team Makeup	■ Team members are likely to be specialized. ■ Skills often not interchangeable.	■ Many skills are interchangeable.

IMPLEMENTATION OF AGILE

Surprisingly, the core of Agile—the Agile Manifesto (see Appendix)—does not describe how to implement it. It is a set of values and principles that can cut across all aspects of product development. There are various frameworks and methods used by software teams to implement Agile, the most popular of which appears to be Scrum [11].

In addition, lean principles, which have much in common with Agile, have made their way out of the manufacturing world and into product development [12]. In fact, lean has also morphed into a new approach to start-up product development called lean startup [13]. ALTEN Technology has embraced many lean startup principles, primarily encouraging clients to adopt the approach of a minimum viable product (MVP) as their initial design iteration and to minimize requirements to what is absolutely necessary.

What is currently lacking is practical guidance for how any company or project can start implementing Agile principles without upending their entire product development process. We have reviewed our own processes and identified those that have had the most impact without disrupting the overall structure of a traditionally phased product-development approach. The Agile methods, or tools, that we describe are primarily based on those advocated by Scrum and lean product development.

Using Agile tools and techniques during project execution, we keep projects and team members moving, communicating, and aligned. The tools also help us ensure that our stakeholders have complete visibility into our progress and facilitate continuous improvement.

In this way, we actively reduce risk and deliver value to stakeholders on a regular basis. The tools are an important step toward implementing Agile on a larger scale, both within projects and across the enterprise.

Although we use a variety of Agile tools and techniques (hereafter referred to as tools), those that have the most impact are the following:

- **Incremental Development.** Break up a phase into multiple increments (called "sprints" in Scrum), typically 2–4 weeks long. Plan and run each phase like a miniature project. Key activities include sprint planning, sprint review, and sprint retrospective.
- **Visual Task Boards.** Take all the tasks planned for a sprint, and visualize them on a shared board, either physical or virtual. Ensure that all team members and stakeholders have access. Break work up into small chunks, and have a clear definition of "done." Work toward completed features.
- **Daily Stand-ups.** Hold daily quick coordination meetings. Get everybody synced up, and determine dependencies and any necessary follow-up. Identify and resolve issues quickly.
- **Demonstrate Value Often.** Reduce risk and show that tangible progress is being made. Deliver something of value at every increment review, preferably a demonstration of a working product.

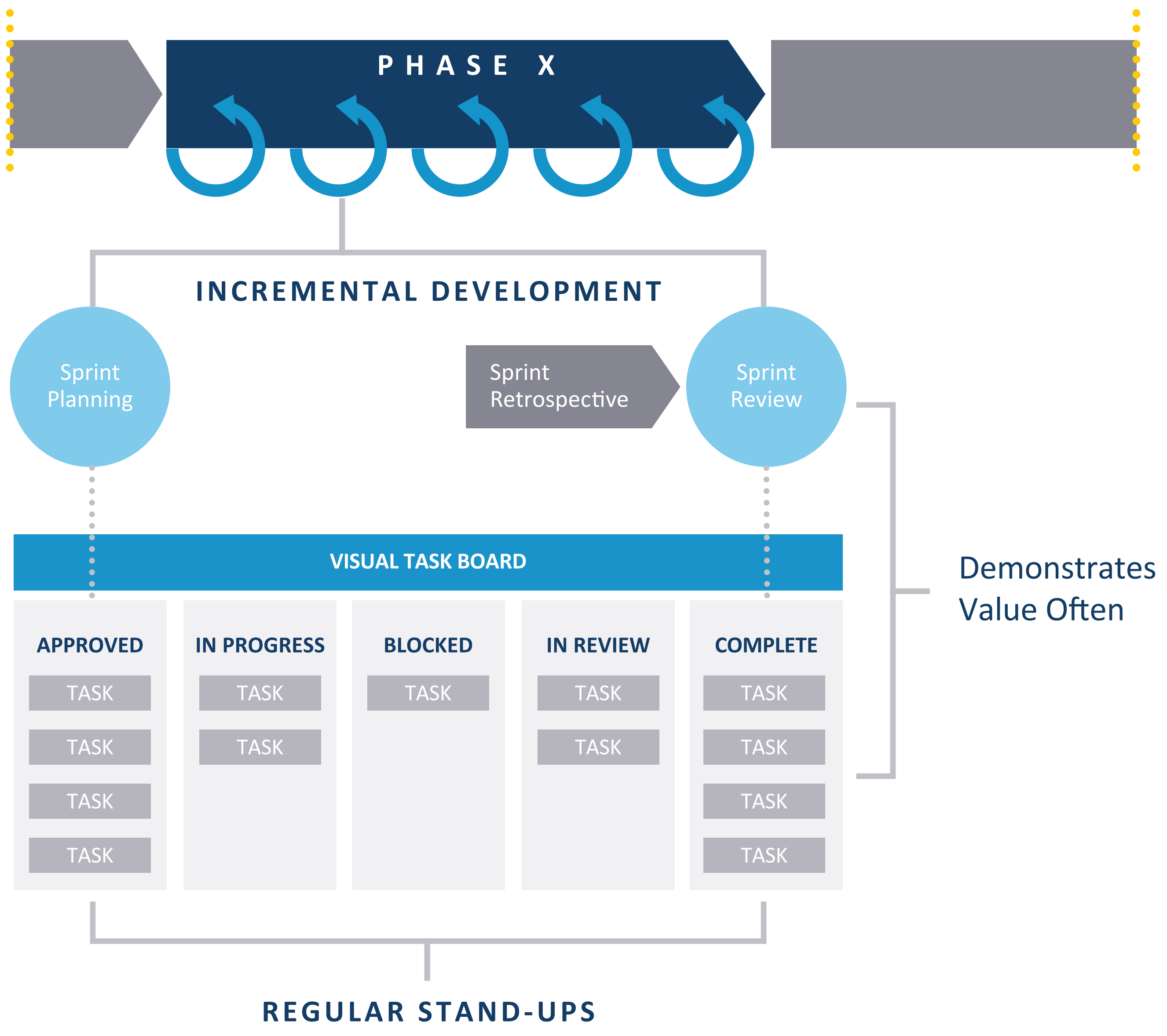


FIGURE 2. RELATIONSHIP OF THE RECOMMENDED AGILE TOOLS IN THE CONTEXT OF A PHASED PROJECT

INCREMENTAL DEVELOPMENT

A key characteristic of Agile project execution is breaking up the traditional phased approach into smaller, more manageable increments. The difficulty of a long- duration phase is that the work details are lost, and actual status is obsolete almost immediately once the phase starts. With incremental development, we take a large and ungainly phase and break it up into "sprints."

Each sprint is run like a miniature project with a defined scope and goals with a set of tasks that need to be accomplished. Not only does the team get a regular sense of accomplishment but the stakeholders also receive tangible value more often. By prioritizing features, risk is reduced earlier, and core features are emphasized, leading to a clearer understanding of project goals. Feedback is encouraged and can be incorporated before it becomes too expensive to implement.

There are three key meetings that bound and define a sprint: sprint planning, sprint review, and sprint retrospective.

Sprint Planning

The sprint planning meeting is used to start a sprint and, essentially, to create a miniature project plan for the sprint. All team members participate, and the plan is approved by the relevant stakeholders. Team members work together to determine the following:

- Sprint goal(s)
- Sprint duration
- Prioritized approved features and tasks

The goal of each sprint is to demonstrate something tangible during the sprint review. Preferably, the goal is to produce a set of features and related functionality. The initial focus is on high-risk aspects of the design and core functionality; this way, we are actively reducing risk and building a good framework for the rest of the product. Sprints iteratively build functionality to support an integrated prototype demonstration, as described in the Demonstrate Value Often section.

Goals are not tasks! A poor goal for a sprint is "Work on mechanical design." That is a high-level task. A good sprint goal for a mechanical project is "Demonstrate prototype design of the housing." There can be multiple sprint goals.

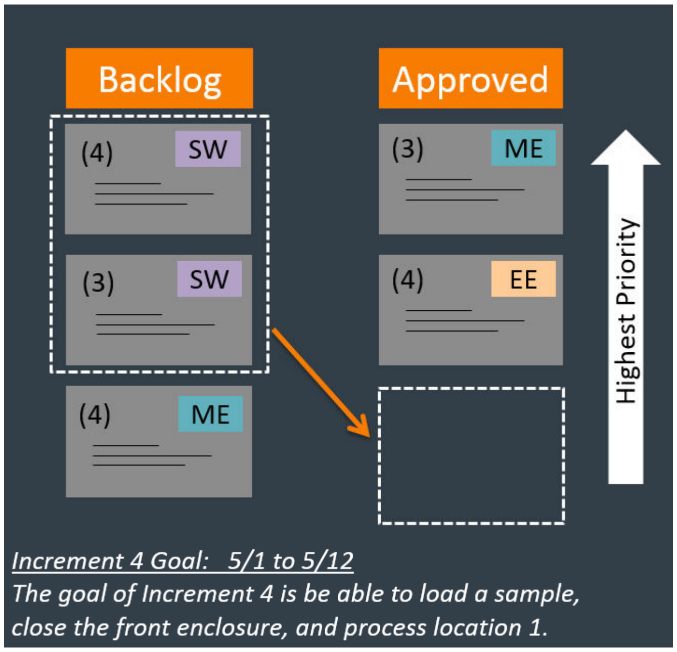
Many recommend a sprint duration of one to four weeks. We do not strictly advocate a specific sprint duration for hardware or software; this is left to the project team to decide. We find that two weeks works well for many projects. We also recommend maintaining a consistent sprint duration to help establish a project rhythm. If a project has multiple teams, sprints should be coordinated among the teams to all begin and end at the same time. However, if the sprint duration is not working well for the team, it can be discussed and adjusted during the sprint retrospective, which we will discuss later.

To achieve the sprint goal, a number of features or tasks are completed (software engineers may prefer "user stories"; we will use the more generic terms "features" and "tasks" for the rest of this paper). These features and tasks form a list of approved work that is assigned to a team member, implemented, reviewed, and completed.

Features and tasks are defined by the team. They are brainstormed at a high level as a set of features that must be implemented to achieve the phase goal. The complete feature set is called the backlog. At the sprint planning meeting, the team determines which features to include in the sprint to support the sprint goals. If necessary, the features are broken down into more detailed tasks that can be completed within a few days. In addition, any unfinished tasks from the prior sprint are automatically pulled into the current sprint.

It is important that this list of approved features and tasks be prioritized. Prioritization is based on dependencies, risk reduction, or anything else the team deems important. The visual task board, once set up with prioritized, assigned tasks, is used to track task progress throughout the sprint. Figure 3 represents this process.

FIGURE 3. SPRINT PLANNING
During sprint planning, a goal is determined, tasks are moved from the backlog to approved work, and the tasks are prioritized.


Backlog
(4) SW
(3) SW
(4) ME
Approved
(3) ME
(4) EE
Highest Priority
Increment 4 Goal: 5/1 to 5/12
The goal of Increment 4 is be able to load a sample, close the front enclosure, and process location 1.

Sprint Review

The sprint review meeting closes out a sprint. It highlights what happened in the sprint and demonstrates the state of the design to the stakeholders. Although it is called a review, it is not as formal as a gated review. Rather, it is a more casual meeting to wrap up the current sprint before continuing on to the next one.

Unfinished tasks will not necessarily be included in the next sprints because new tasks may have a higher priority.

That is part of the value of sprints—they allow you to stop, evaluate, and reprioritize as needed. A sprint review covers the following high-level items:

- Highlight progress and deliverables
- Demonstrate and receive feedback
- Discuss difficulties, new items, and next sprint plan

In some cases, the project stakeholders may have little or no contact with the development team during the sprint. The review provides an open and transparent setting to present the team's accomplishments as well as any completed deliverables.

The most important activity in the sprint review is the demonstration. In the Agile values and principles referenced in the Appendix, there are multiple mentions of delivering "working product." The demonstration shows the work produced. Projects with software compile what they have, and the stakeholders can actually use the software. Projects with hardware typically do not have the option of demonstrating stand-alone working hardware. However, given that sprints may be longer on projects that involve hardware and that the definition of "working product" can be fairly broad, there is plenty that hardware teams can demonstrate to stakeholders. Hardware working products include sketches, CAD models, mock-ups, or working hardware.

Unresolved issues that arose during the sprint are discussed in the sprint review and their resolutions planned into the next sprint. Emergent behaviors or properties of the system discovered as a result of the demonstration are discussed and addressed. Issues such as market changes and new or updated requirements are also reviewed, new backlog items are created, and goals are proposed for the next sprint.

The development team learns by putting a viable demonstration together, and the stakeholders have tangible evidence of progress. The stakeholders also get a real feel for how the system works and the state of the design. They also are able to provide feedback and direction directly to the design team to ensure that everyone is aligned with a common project vision at the end of each sprint. This contrasts sharply with a typical formal design review that covers months of work and tends to focus on slides rather than on tangible proof of progress.

Sprint Retrospective

The sprint retrospective typically happens after the sprint review, but the timing is flexible (it may occur after the next sprint has started). The retrospective is an opportunity for the project team to reflect on the last sprint and identify ways to improve the team's working process so that they are more effective and the work is more enjoyable. Not only can retrospectives help fine-tune the process for a particular project, but they also provide the chance to reflect on recent successes to improve team cohesion.

A retrospective is typically run as a time-boxed meeting with the team (usually one hour). It is helpful to start the retrospective meeting with a quick review of the improvements identified in the previous retrospective to determine whether the changes have been effectively implemented.

After reviewing the previous retrospective action items, the meeting focuses on the following questions:

- What went well?
- What didn't go well?
- What can we improve in the next sprint?

The first question is a chance to reinforce good practices and to celebrate the team's most recent victories. The second question is used to identify problems and look for solutions. The final question is treated in a much more free-form manner, befitting its creative nature. Anyone is free to call out suggestions. The end result is typically a list improvements for the following sprint.

To give the proceedings a little structure and to make sure everyone gets a chance to speak, it is typical to gather answers to "What went well?" and "What didn't go well?" in a round-robin. Anyone can "pass," but no one gets skipped. When everyone is done saying "pass," move on to the next question. For the final question, anyone can speak. A project is a team effort, and all input is valuable in ensuring the success of the project.

Sutherland [4] suggests a final question to gauge team happiness: "What is one thing that will make you happier in the next sprint?" This question will often reveal people's deep-seated concerns about the project. Often these are the same reactions associated with "What can we improve?" but this proposed question may provide a greater understanding of risk and priority.

VISUAL TASK BOARDS

Once a sprint has been kicked off with the sprint planning meeting, the team will have a list of approved and prioritized features or high-level tasks that need to be accomplished to attain the sprint goal. This is a good start, but the team needs a way to quickly communicate what they are currently working on (in case there are dependencies) as well as their status.

A visual task board is an effective way to track tasks and communicate status. Visual task boards come in a variety of formats and media. Like many, ALTEN Technology originally used sticky notes on a wall. Later we moved on to magnetic whiteboards. For some projects we used Excel. We currently use the online digital tools Jira, Trello, and Smartsheet. We prefer using a digital board because the information is accessible anywhere and anytime with a computer (stakeholders can review it offsite), digital task cards are easy to revise (no erasing or crossing out), and the cards retain their history.

Visual task boards help break features into smaller tasks. Each task in a sprint should have a duration of a half day to three days, and the board should explicitly define what is needed for a task to be considered completed. Tasks that are too big or inadequately defined may suffer from the "80 percent" or "almost-done" syndrome, which may inhibit quick progress and demonstration. Breaking up tasks at this level takes practice, and it is common for the team to complain or worry that they are being micromanaged. However, the clear objectives, frequent feelings of success, and easier status updates will soon demonstrate the value of this approach.

Examples of high-level hardware tasks include:

- Design the housing
- Design the circuit board

If we break down the "Design the housing" mechanical feature, it might produce the following list:

Design Housing (Feature)

- Task–Review housing requirements
- Task–Build skeleton model in CAD
- Task–Rough in lid
- Task–Rough in base
- Task–Decide on housing materials

- Task–Search for and select hinge
- Task–Search for and select latch
- Task–Finite element analysis for lid
- Task–Tolerance analysis
- Task–Lid refined model
- Task–Base refined model
- Task–Model interface
- Task–Design review
- Task–Update models
- Task–Create drawings
- Task–Review drawings
- Task–Release models and drawings

If we break down the “Design the circuit board” electrical feature, it might yield the following list:

Design Circuit Board (Feature)

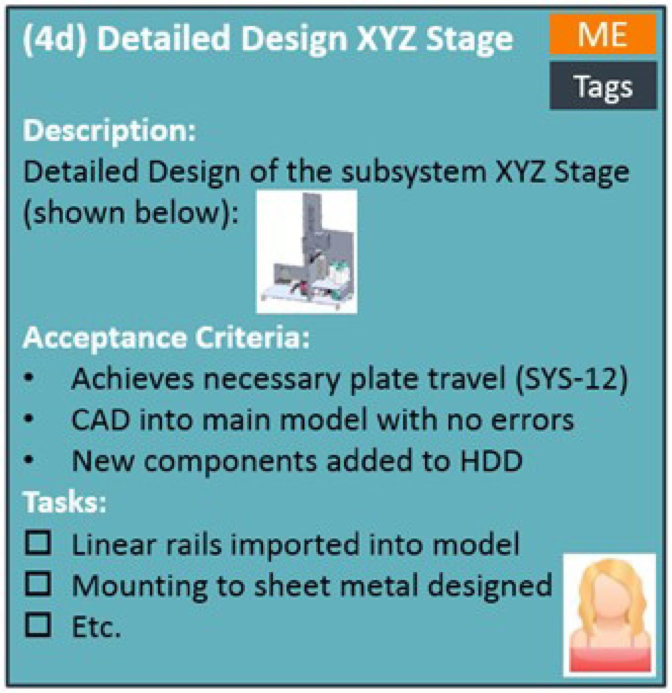
- Task–Review PCBA requirements
- Task–Component selection
- Task–Layout
- Task–Review layout
- Task–Design the power schematic
- Task–Design the communications schematic
- Task–Run a SPICE analysis
- Task–Review circuit
- Task–Schematic updates
- Task–Breadboard
- Task–Update layout
- Task–Bill of materials (BOM)
- Task–BOM review
- Task–Release files

The required tasks for each feature can be staggered in different sprints. There are many benefits to breaking up the work using this approach:

1. We have a way to visualize the amount of work required to complete a feature of interest;
2. Each task is easy to understand by the team members, project leads, and stakeholders;
3. Each task will take anywhere from half a day to a few days;
4. Everyone gets a sense that things are moving along when a task is completed.
5. It is easy to determine whether a task is completed.

When using a visual task board, we recommend creating a card for every task—cards are the workhorses of the visual task board. An example is shown in Figure 4. Each task is represented by a single card. Clients and projects may have different formats for their cards, so some content tailoring may be required.

FIGURE 4. EXAMPLE OF A TASK CARD



(4d) Detailed Design XYZ Stage ME

Tags

Description:
Detailed Design of the subsystem XYZ Stage (shown below):

Acceptance Criteria:
- Achieves necessary plate travel (SYS-12)
- CAD into main model with no errors
- New components added to HDD

Tasks:
- ☐ Linear rails imported into model
- ☐ Mounting to sheet metal designed
- ☐ Etc.

As a starting point, we set up cards with the following fields:

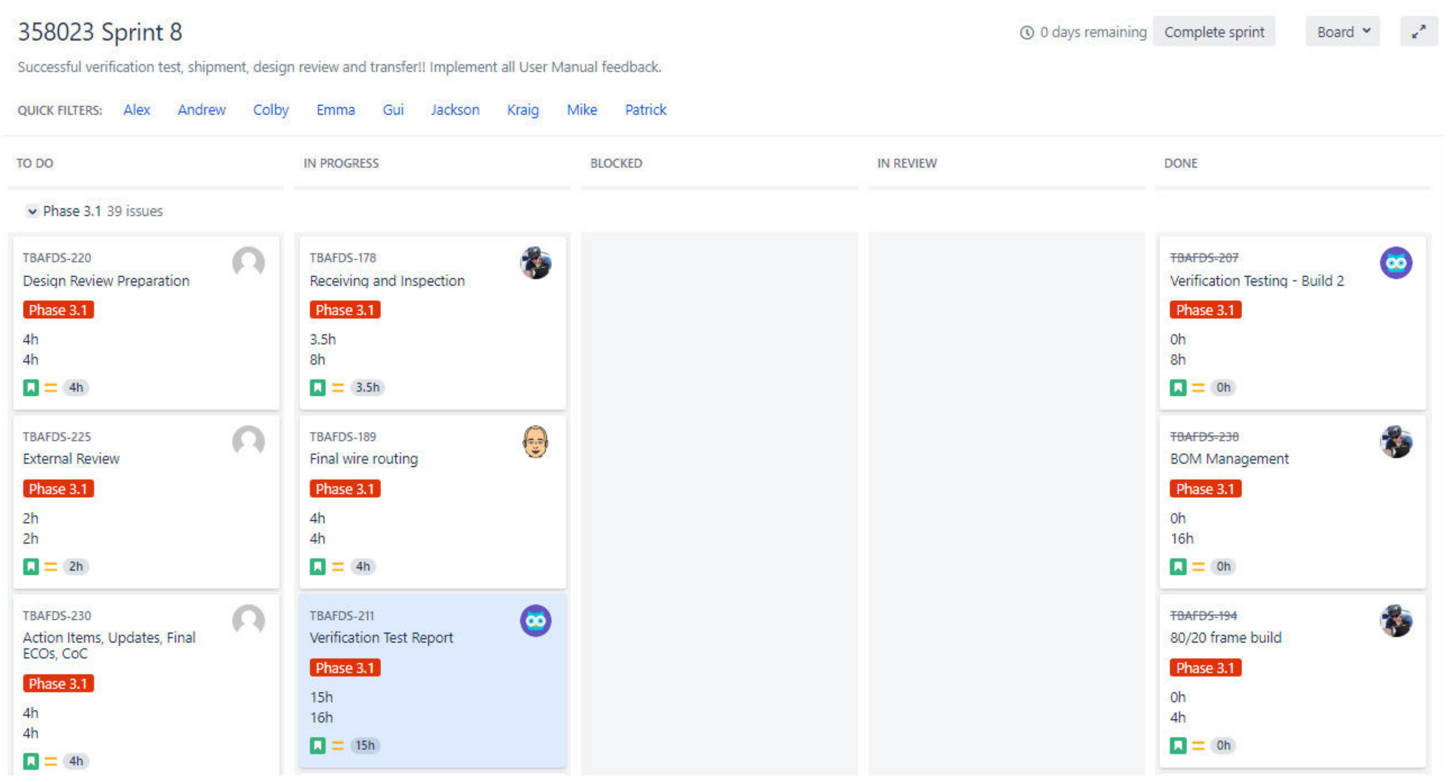
- **Title**—This is a short task name.
- **Description**—This is a more detailed description of the task.
- **Owner**—This is the singular person responsible for completing the task. Some tasks may require multiple people's involvement. However, only one team member is responsible for implementing it successfully and reporting on it. This is important to avoid ambiguity.
- **Subtasks**—Cards that are more involved may benefit from having subtasks to track incremental progress.
- **Acceptance Criteria**—This may be the most important field because it provides everyone with the same definition of "done." There may be multiple acceptance criteria for a card. The card's independent reviewer compares the work output against the acceptance criteria to determine whether the task is complete.

Once the content and utility of a card are defined, consider the following optional fields:

- **Estimate (optional)**—The owner's task duration estimate. This is useful to roll up on digital boards into a sprint-level work estimate.
- **Discipline (optional)**—The discipline(s) required to complete the task. This is particularly useful for large integrated projects.
- **Other Tags (optional)**—Project-specific fields. Note that each additional field increases the complexity of creating and tracking cards, so include only if value-added. Examples include tags for tracking to features, requirements, documents, or other general tasks.

Once cards are filled out, they are placed on the task board (if using a digital tool, the cards are created within the board itself). Visual task boards visualize and enforce workflows through columns. Each column represents a specific workflow step, and each card should move through every step in the workflow, except perhaps "Blocked" (see Figure 5).

FIGURE 5. EXAMPLE OF A VISUAL TASK BOARD


358023 Sprint 8

Successful verification test, shipment, design review and transfer!! Implement all User Manual feedback.

0 days remaining | Complete sprint | Board

QUICK FILTERS: Alex Andrew Colby Emma Gui Jackson Kraig Mike Patrick

TO DO | IN PROGRESS | BLOCKED | IN REVIEW | DONE

Phase 3.1 39 issues

TO DO:

TBAFDS-220
Design Review Preparation
Phase 3.1
4h
4h
4h

TBAFDS-225
External Review
Phase 3.1
2h
2h
2h

TBAFDS-230
Action Items, Updates, Final ECOs, CoC
Phase 3.1
4h
4h
4h

IN PROGRESS:

TBAFDS-178
Receiving and Inspection
Phase 3.1
3.5h
8h
3.5h

TBAFDS-189
Final wire routing
Phase 3.1
4h
4h
4h

TBAFDS-211
Verification Test Report
Phase 3.1
15h
16h
15h

DONE:

~~TBAFDS-207~~
Verification Testing - Build 2
Phase 3.1
0h
8h
0h

~~TBAFDS-238~~
BOM Management
Phase 3.1
0h
16h
0h

~~TBAFDS-194~~
80/20 frame build
Phase 3.1
0h
4h
0h

A typical starting point for columns are the following (ordered from left to right):

- **Approved**—These are the cards the team assigned to the current sprint during the sprint planning meeting. They are prioritized: the most urgent tasks and the unfinished cards from the previous sprint are on top.
- **In Progress**—These are the cards that the team is actively working on. Whenever a team member starts a new card, they transfer it from the "Approved" column to "In Progress." To limit inefficient multitasking, each team member ideally works on one card at a time, although this is not always possible.
- **Blocked**—Cards can sometimes be blocked by external factors. For example, a vendor's late delivery of a quote, or worse, of parts, could prevent the build of a prototype. As soon as a team member feels they are blocked on a task, they move the card to "Blocked." Tasks typically are not blocked for long because the team rallies to help remove the impediment or brainstorm alternatives.
- **In Review**—Once the owner of a card has completed the task, they move the card to "In Review" and assign an independent reviewer (usually someone who did not work on the card but is part of the project). The reviewer (often the project manager, systems engineer, or subsystem/discipline lead) reviews the completed work against the acceptance criteria on the card. If the work output meets the acceptance criteria, the reviewer moves the card to the "Complete" column. Otherwise, the card goes back to "Approved" for additional work.
- **Complete**—As stated above, cards in the "Complete" column have passed independent review against their acceptance criteria and are closed.

The team's goal is to move all cards through the workflow from "Approved" to "Complete" within a single sprint. There are numerous direct and indirect benefits to using a visual task board:

- Team members like knowing what is expected of them. The cards in the "Approved" column show the anticipated workload and the acceptance criteria for their tasks, so they know what needs to be accomplished.
- Team members do not need to pause to plan their next task. They identified and prioritized their cards in sprint planning, so they merely grab the next card, pulling work instead of its being pushed to them.
- Team members know when someone else is working on something that might affect them.
- The project manager can see at a glance the state of the sprint. One can see what has been completed as well as what is left to do without interrupting the team. (Digital boards have a variety of tools that can extract and visualize other information from the board as well.)
- Stakeholders see immense value in the transparency that a visual task board provides. Digital boards can be shared with stakeholders, and the information on a board can easily be summarized for a status review.
- If a team member is overloaded or multitasking between different cards, the facilitators can step in and manage their workload by reprioritizing cards, reassigning cards to another team member, or sending some cards back to "Approved."
- Blocked tasks are directly visible to ensure they are not dropped. They tend to be an eyesore on the board, which encourages escalation and resolution.
- Each completed task demonstrates visible and incremental value for the stakeholders.

There is no ambiguity around whether a task is complete ("mostly done," "80 percent done," "done, except," or "really done"). A task has either been independently reviewed against its acceptance criteria or it has not. If it has, it is in the "Complete" column and is closed. At sprint close, cards that are still in the "Approved" column are moved to the top of the backlog. When planning the next sprint, the team may move them back to the "Approved" column or keep them in backlog for future implementation.

DAILY STAND-UPS

Daily stand-up meetings are relatively popular throughout product development and not just for those using Agile techniques. However, they may also be the most abused Agile tool in practice. Nominally, the point of a daily stand-up is to make sure the team is in sync, accountable, and able to communicate with one another. The team comes together at a preset time every day and quickly shares their progress, plans, and pains. Unfortunately, this is easier said than done. There are a number of best practices for keeping stand-ups useful and running smoothly.

- **Set the proper cadence.** Although this section is introduced as a "daily" stand-up, some projects may not have the urgency that requires a daily meeting. Have the stand-up in the morning so that the previous day's accomplishments can be described and the plan for the current day has meaning.
- **Limit the meeting to 15 minutes.** Keep the stand-up brief. A team of five to ten can perform a stand-up in 15 minutes, but it requires discipline. Larger teams may require multiple stand-ups, grouped by discipline or subsystem. In this case, the facilitators need to attend all stand-ups to ensure continuity and communicate cross-team considerations.
- **Use facilitators.** Although the team members are reporting to one another, facilitators promote continuity and bring a higher-level project perspective, such as ensuring that task priorities line up with the bigger project picture.
  - For us, the project manager acts as "scrum master" [11], facilitating meetings and keeping things on track.
  - We also have a systems engineer on every project. The systems engineer acts as the "product owner" [11]; they are the stakeholder interpreter and have the best in-house understanding of the product requirements. They also go back to the stakeholders for clarification of requirements when necessary.
  - Those familiar with Scrum may recoil at the involvement of the project manager and systems engineer as described above, but we have had success with this implementation.
- **Have structured updates.** Fifteen minutes is not much time, so each team member must stay focused. To get to the point quickly, each team member answers the following questions:
  - "What have you done since the last stand-up?"
  - "What are you working on next?"
  - "Is there anything blocking your progress?" Or "Is there anything you need from the other team members?"
- **View the task board.** This is not the time to manipulate the task board, which should be done by team members as they go. Rather, the task board is a useful reference to understand how the "In Progress", "Blocked", and "In Review" tasks are related to the team's stated work.
- **Book a follow-up meeting.** Inevitably, things will come up that require further discussion by some of the team members. After the stand-up reporting is complete, the facilitator ends the meeting. If further discussion is required, having a prebooked follow-up meeting provides those involved with a space and time already scheduled to complete their thoughts and exchange information.

Some bad habits can easily creep into and impact the effectiveness of daily stand-up meetings.

- **Sitting.** This seems obvious—the meeting is called a "stand-up meeting." Sitting starts to encourage comfort and detailed discussions. One way we use to enforce standing is to have a designated stand-up area or room that contains a monitor for review of the electronic task board but no chairs. A whiteboard is also useful for taking notes or for the follow-up meeting.
- **Task board is not up to date.** The facilitators need to encourage the team to keep the task board up to date in real time as they begin and complete tasks.
- **Wandering off topic.** Each person answers the three stand-up questions—that is all. Detailed discussion, brainstorming, and problem-solving are reserved for the follow-up meeting or individual meetings later in the day.
- **Too many people.** ALTEN Technology limits attendance to the core team. If there are specialists who come and go in the project, designate one of the core team members to report on their progress. If team members are not getting value out of the meeting, it is okay for them to leave or to split the meeting into multiple stand-ups.
- **Overrunning time.** This is the outcome of the items listed above. If you find that your team is struggling to stay within the 15-minute time slot, take a hard look at how the stand-ups are running and check above for ways to get back on track.

DEMONSTRATE VALUE OFTEN

Our goal in product development is to deliver value to the stakeholders as quickly and inexpensively as possible while still creating a quality product. With traditional development approaches, by the time the stakeholders see a deliverable, many decisions may already have been made and locked into the design. If the stakeholders disagree with the design team's decisions, it could lead to arguments about changing requirements, scope creep, and what the design "should" be. To avoid these problems, we use the concept of "Demonstrate Value Often."

- **Demonstrate**—This is an actual demonstration of the latest features at the end of a sprint. It does not have to be a production-ready demonstration; any demonstration of the product deliverables adds value.
- **Value**—Lean proponents define project value this way: "All project value is embodied in its deliverables. A deliverable is any tangible and transferable item that contributes to the commercialization of a new product. A deliverable can be a document, a drawing, a decision, a report, a prototype, a piece of hardware or software, etc. Deliverables are outcomes of tasks and activities" [12].
- **Often**—A demonstration is not a one-time occurrence; it happens often. In fact, every sprint review is an opportunity to demonstrate the state of the product. This means that you are demonstrating value every one to four weeks, depending on the chosen sprint duration. Doing demos often provides regular opportunities for face time with stakeholders. Demos also provide alignment and allow for adjusting the plan if necessary. We limit the amount of technical debt (bugs and fragility) that can pile up before the next demo. This forces us to make our designs more robust and keep them that way throughout the project.

Demonstrating value often is accomplished through sprint demonstrations and integrated prototype demonstrations. Both types of demonstrations occur as part of the sprint review. Regarding the definition of "working product," some authors claim that what you demonstrate has to be "potentially shippable" [11].

We find that any demonstration of deliverables to the stakeholders has value. We know from the earlier examination of the differences between hardware and software that hardware prototypes can have long lead times and cost a lot of money. Therefore, although software can deliver executable code at every sprint review, hardware has to plan far in advance and budget for every prototype. Therefore, hardware prototypes will not occur as often as software prototypes.

Progress in hardware development can be demonstrated with incremental deliverables in lieu of an anticipated forthcoming full prototype. Hardware deliverable demonstrations can come in many forms:

Mechanical

- Brainstorming sketches
- Renderings
- CAD mockups
- Detailed CAD
- Analysis
- Rapid prototypes

Electrical

- Schematics
- Analysis
- Layouts
- Breadboards
- Developer kits
- Prototype PCBAs

Other

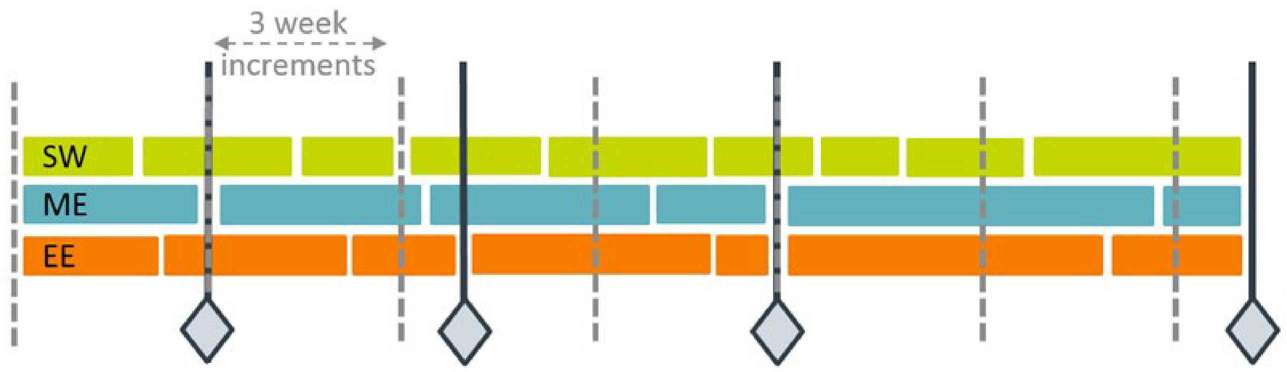
- Vendor selection
- Bill of material
- Cost estimates
- Lead time estimates

Each of the items above represents an incremental improvement in the evolution and understanding of the final product design without the time and expense of a production-equivalent prototype. Demonstrating any of the above to stakeholders as they are created has tremendous value in obtaining feedback and alignment without having to fully commit to the design. We also reduce uncertainty and risk as we progress through the various demonstrations of increasing capability.

While a demonstration of discipline-specific functionality is valuable, in a product that involves multiple disciplines, the most valuable demonstration is the integrated prototype demonstration, where working software is demonstrated on working hardware.

An integrated prototype requires planning at the phase and sprint levels. Completion of software and hardware features does not always coincide, so integrated prototypes may not line up with sprints. However, ideally, you can combine an integrated prototype demonstration with a sprint review. A bit of high-level planning is required to make sure that an integrated prototype is ready for demonstration to coincide with a sprint review (see Figure 6). There may also be a different stakeholder audience interested in attending and participating in the integrated prototype demonstrations. Colored bars in Figure 6 represent work on features for each discipline. Dashed lines represent sprint reviews and demonstration of those features. Diamonds represent integrated prototype demonstrations. This is an area of applying Agile where ALTEN Technology has fundamentally blended Agile with more traditional methods and their associated milestones and reviews.

FIGURE 6. SPRINT CADENCES ALIGNED WITH INTEGRATED PROTOTYPE DEMONSTRATIONS



We strive to have a physical mockup of the system in progress to help with development and write software on the target hardware (development kits are invaluable for embedded systems). These form the basis of early integrated prototypes. Integrated prototypes bring all of the disciplines together to provide a regular system-level demonstration. Projects are planned to ensure that the team is integrating important features into prototypes as they go. Each integrated prototype demonstration showcases more and better features and functionality (see Figure 7).

Integrated prototypes are important because they include the thing that most people are not thinking about as they work on their individual components and areas of expertise—the system. Only by building integrated prototypes do we gain an understanding of the following:

- As-built form, fit, and feel
- System behavior
- Interface compatibility
- Cross-discipline interactions
- Technical risk reduction
- Emergent system properties and behavior
- Common interpretation of the design direction

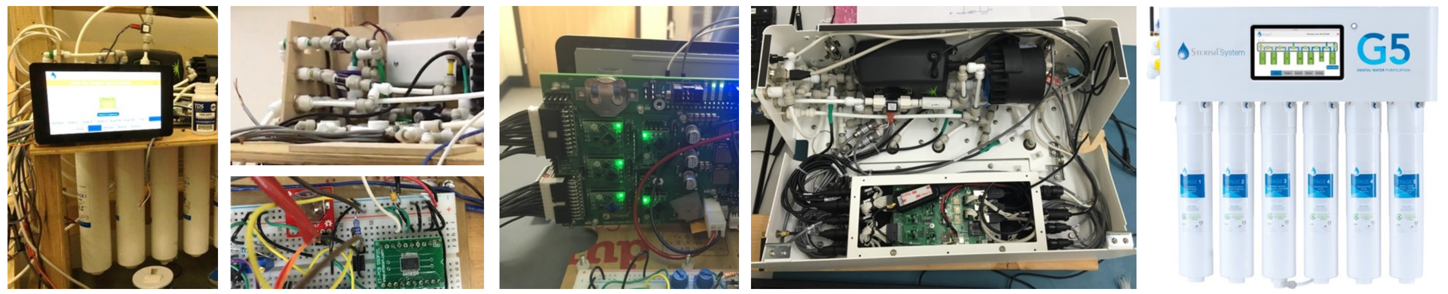
FIGURE 7. INTEGRATED PROTOTYPE EVOLUTION ON AN INTERNET-CONNECTED WATER FILTRATION PRODUCT

Plywood Prototype
Size/shape support frame with: install, power-on, filter replacement (calendar-based)

Hydraulic Prototype
Functional hydraulics with in-line sensors installed; get sensor readings and add flow-based expiration

"Alpha" Near-Final Prototype
Rev 1 hardware: PCBA, sheet metal, and cables; add internet conn., logging, secondary GUI screens

"Beta" Fully Functional Final
Rev A hardware, mfg./service screens, reliability and final GUI improvements, bug fixes



EXAMPLE AEROSPACE PROJECTS

ALTEN Technology has used this approach on several projects and is highlighting the work with two clients, MMA Design and Spaceflight Industries. Although these two projects were considered subsystem contributions, ALTEN Technology has demonstrated this methodology for large, complex projects with great success.

MMA DESIGN

ALTEN Technology was asked to produce a motor controller for one of MMA Design's hardware demonstrations. In less than a calendar year, ALTEN Technology was asked to design, fabricate, assemble, test, and deliver under a firm fixed-price contract an engineering demonstration unit (EDU) and a flight model (FM). Because of the aggressive schedule, a joint design review, EDU delivery, FM delivery, and a program final report were the primary deliverables with limited milestone reviews for each one.

Because this was a new client relationship, the ALTEN Technology approach to Agile helped build trust by demonstrating value often. Performing this development on a fixed price budget, ALTEN Technology learned many important lessons as captured via retrospectives throughout the project and at its overall completion. At times retrospectives were skipped because of an aggressive schedule, but the team quickly learned that this step in the process was extremely valuable, and the retrospectives were reinstated. Scope management and estimating were critical but difficult as the project changed. Frequent client interaction was important and was facilitated by the two companies being geographically close. The greatest validation of this process was the client feedback that "the controller continues to be a rock of reliability" throughout the testing and delivery of the overall product.

SPACEFLIGHT INDUSTRIES

Spaceflight Industries successfully launched a record 64 small satellites in a single launch for a variety of commercial, government, and education customers; and ALTEN Technology was fortunate enough to be a part of this mission. ALTEN Technology supported development of the launch sequencer that released the various satellites into a Sun-synchronous orbit, which was the largest ride-share mission to date on an American launch vehicle. ALTEN Technology helped to troubleshoot some challenges with this critical piece of mission hardware, and again the Agile approach was valuable in defining the scope of work and methodically completing it on a fast-track schedule. ALTEN Technology also helped Spaceflight Industries with the development of spacecraft avionics hardware. In both cases, Spaceflight Industries was happy with the ALTEN Technology Agile approach to completing the work successfully and with an aggressive schedule.

CONCLUSION

At the beginning of this paper, we stated that incorporating Agile tools would provide a number of benefits. Let us recap how these specific tools support these benefits:

- **Incremental Development.** Large phases are broken down into much more manageable one- to four-week sprints. Higher risk items and core functionality are worked on first, delivering value at a faster cadence. There are more opportunities for stakeholders and team members to get together and develop a common understanding of the project and product. There are a set of goals that are delivered and demonstrated at the end of the sprint.

- **Visual Task Boards.** The actual work is broken down into half-day to three-day tasks, which are easy to understand and have a clear definition of "done." Every task is independently reviewed for completeness and quality. The status of all tasks is transparent and visible to all team members and stakeholders. Each task has a single owner for accountability. Blocking issues are highlighted and addressed.

- **Daily Stand-ups.** Stand-ups provide "real time" status of what has happened on a project and where it is going. The team gets synced up daily, and team cohesion is enhanced. Blocking issues are brought up and plans are made to resolve them.

- **Demonstrate Value Often.** Something of value is delivered and demonstrated at every sprint review. This reduces risk and shows that tangible progress is being made. Useful feedback is provided when there is still time to incorporate it. The stakeholders and team members move into alignment regarding how the product works and the direction it is heading. The team is accountable for producing value in every sprint.

As with all changes to organizations, it is best to start with a small pilot project and a group of enthusiastic team members. Make them part of the decision-making process on how to incorporate these tools. Determine what works best for your projects. Feel free to customize. We have found these tools to be infectious. Once a team sees the speed and improvements brought about by these tools, they will spread the word for you.

These are just the first steps in incorporating Agile into your integrated product development process. In addition, Agile can also be applied at all levels of an organization (if you view internal initiatives as projects, all of these tools apply). Once you get a taste for Agile using these initial tools, there is a greater world of Agile to explore and incorporate into your business.

APPENDIX

The Agile values and principles from the Agile Manifesto are listed below [5]. The authors have added emphasis to generalize to all product development, not just software.

AGILE VALUES

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**

Customer 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.

AGILE PRINCIPLES

1 Our highest priority is to satisfy the customer through early and continuous delivery of valuable software (deliverables).

2 Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3 Deliver working software (product) frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4 Business people and developers must work together daily throughout the project.

5 Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6 The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7 Working software (product) is the primary measure of progress.

8 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9 Continuous attention to technical excellence and good design enhances agility.

10 Simplicity—the art of maximizing the amount of work not done—is essential.

11 The best architectures, requirements, and designs emerge from self-organizing teams.

12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

ACKNOWLEDGMENTS

The authors would like to recognize the ALTEN Technology team for being open to experimenting with these new approaches and our various clients in the aerospace, medical, and other markets we serve who are open to the benefits delivered by the ALTEN Technology Agile hardware development approach. There are too many to name all the contributors, both internal and external to ALTEN Technology, who have influenced this work.

REFERENCES

[1] Brown, C. D., Elements of Spacecraft Design, AIAA Education Series, AIAA, Reston, VA, 2002, pp. 13-17.

[2] NASA, *NASA Systems Engineering Handbook*, NASA SP-2016-6105, 2016, NASA, Washington, DC, Chapter 3.

[3] Department of Defense, *Systems Engineering Fundamentals*, Defense Acquisition University Press, Fort Belvoir, VA, Chapter 2.

[4] Sutherland, J. *Scrum*, 2014, Crown Business, New York.

[5] Beck, K., Beedle, M. Bennekum, A. V., Cockburn, A., Cunnighman, W., et al., Manifesto for Agile Software Development, 2001, viewed 9 November 2017, from http://agilemanifesto.org.

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

[7] Graves, E. "Applying Agile to Hardware Development (parts 1-7)", in Playbook Blog, viewed 9 November 2017, from https://www.playbookhq.co/blog/agileinhardwarenewproductdevelopment.

[8] Ovesen, N., 'The Challenges of Becoming Agile – Implementing and Conducting Scrum in Integrated Product Development', 2012, PhD thesis, Department of Architecture, Design, And Media Technology, Aalborg University.

[9] Reynisdottir, P., 'Scrum in Mechanical Product Development – Case Study of a Mechanical Product Development Team using Scrum', 2013 PhD thesis, Department of Product and Production Development, Chalmers University of Technology.

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

[11] Schwaber, K. and Sutherland J.,'The Scrum Guide', 2017, in Scrum Alliance, viewed 9 November 2017, from https://www.scrumalliance.org/why-scrum/scrum-guide.

[12] Mascitelli, R., *The Lean Product Development Guidebook*, 2007, Technology Perspectives, CA.

[13] Ries, E., *The Lean Startup*, 2011, Crown Business, New York

[14] Rigby, D.K., Sutherland, J. and Noble, A., "Agile at Scale," *Harvard Business Review*, May-June 2018, pp. 88-96.