INTERNATIONAL DESIGN CONFERENCE – DESIGN 2022
https://doi.org/10.1017/pds.2022.16

Guidance on Application of Agile in Combined Hardware and Software Development Projects

J. Drutchas ✉ and S. Eppinger

Massachusetts Institute of Technology, United States of America
✉ drutchas@mit.edu

Abstract

Agile has its roots in software, improving the pace and quality of development projects. The application of Agile to hardware is less mature, and anecdotal evidence suggests there are specific challenges in this space. Based upon qualitative interviews with organizations working simultaneously across hardware and software applications, this paper presents three key challenges we identified. We also document several useful adjustments to Agile practice which provide a toolset that teams can use to more successfully apply Agile to hardware projects.

Keywords: product development, agile development, agile management

1. Introduction

The Agile project management approach, hereafter referred to as "Agile", has its roots in the software industry and is known for improving the pace and quality of development projects. The application of Agile methods to hardware design is less mature. Anecdotal evidence suggests specific challenges with Agile for hardware, resulting in its slow adoption, despite its strength in application to development projects with inherent uncertainties. This paper reviews several successful applications of Agile to hardware development and the characteristics of these projects. The purpose of this study is to provide a toolset and components of a framework that can be used by design and development teams wishing to adopt or improve Agile. Through interviews and a review of relevant literature, we have identified three key challenges that limit the adoption of Agile to combined hardware and software projects: constraints of physicality, sprint cadence, and backlog creation. Constraints of physicality are well documented in the literature. We will review this and further explain the latter two limitations before exploring adjustments to standard Agile development practices that can be helpful to overcome these challenges.

This research is based on qualitative interviews with organizations that have applied Agile methods in projects involving both hardware and software development. This has allowed us to identify strategies that have enabled these teams to successfully navigate their efforts to ideate, design, build, and iterate on new and existing products using Agile. The nature and complexity of these combined projects has forced teams to formulate solutions to the perceived limitations of Agile in hardware and to implement various adaptations to execute in their particular context.

2. Background

2.1. Hardware Development

"Developing hardware is hard" is an industry mantra. The steps of hardware development can contain uncertainty which can cause significant variance in time, resource needs, and overall success. The waterfall methodology is a traditional process with sequential stages and gates in which teams must complete each step before moving on. While organizations have specific approaches to the process, a generic sequence of such phases is: planning, concept development, system-level design, detail design, testing, and production ramp-up (Ulrich, et al. 2020). For this paper, hardware development refers to the development of specifications for devices that are intended to be manufactured (Thomson, 2015). The juxtaposition of hardware and software emphasizes both the differences and the adjustments that are possible with Agile.

2.2. Software Development Methodologies

Agile software development is a term for a set of frameworks and approaches based on the values and 12 principles defined in the "Manifesto for Agile Development" (Kent Beck et al., 2001). Frameworks and processes emerged to apply essential tenets of the manifesto, including Scrum, eXtreme Programming (XP), Lean software development, and many others (Dingsøyr, et al., 2012). After Agile's emergence in the early 2000s, it quickly grew in prominence primarily in software development due to what many described as its potential benefits, including increased collaboration and flexibility, decreased planning burden reduction of unnecessary work, increased focus, and inclusion of the customer and user (Dingsøyr, et al., 2012). More recently, Agile has been applied in other industries such as pharmaceuticals, aerospace, construction, finance, and more (Which Industries Can, n.d.; Narayanamurthi, 2017).
Agile's benefits arise from incremental delivery of value to the user, a structure of cross-functional teams that improve information flow, and feedback from users to enhance product iteration (Hoda et al., 2011; Hannola et al., 2013). This process is executed in a time-boxed period called a 'sprint.' Development process steps are performed iteratively to complete the planned work within each sprint. The framework of sprints allows teams to break work efforts into smaller pieces, thereby reducing the burden of planning, allowing faster feedback, bounding errors, and more (Rubin, 2013).
Various roles are fundamental to executing in an Agile environment. For this research, we will focus on two: the product owner and the development team. The product owner is responsible for deciding what features and functionality should be built and in what order of priority (Rubin, 2013). The development team is a diverse, cross-functional team responsible for executing all aspects of the process (Rubin, 2013). In applying Agile and its various frameworks, there can also be the role of a product manager. We consider the roles of product owner and product manager synonymous.

2.2.1. *Key Terminology*

The following is a brief overview of key terms associated with Agile development.

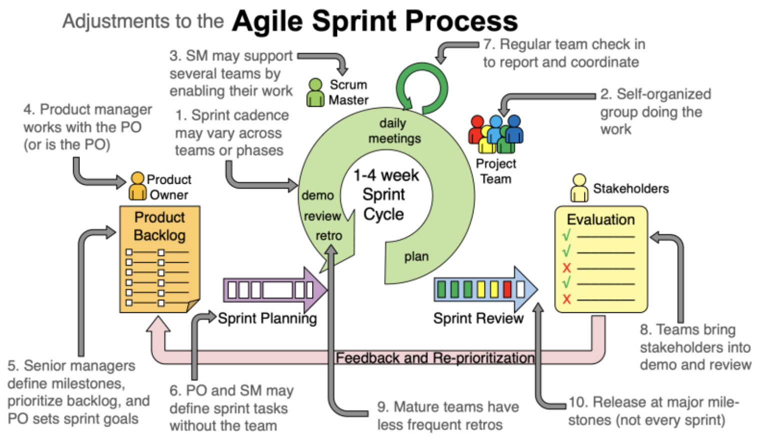
Adjustments to the Agile Sprint Process
3. SM may support several teams by enabling their work
Scrum Master
7. Regular team check in to report and coordinate
2. Self-organized group doing the work
4. Product manager works with the PO (or is the PO)
1. Sprint cadence may vary across teams or phases
daily meetings
Project Team
Product Owner
1-4 week Sprint Cycle
demo
review
retro
plan
Stakeholders
Product Backlog
Evaluation
Sprint Planning
Sprint Review
8. Teams bring stakeholders into demo and review
Feedback and Re-prioritization
5. Senior managers define milestones, prioritize backlog, and PO sets sprint goals
6. PO and SM may define sprint tasks without the team
9. Mature teams have less frequent retros
10. Release at major mile-stones (not every sprint)

Figure 1. Agile Sprint Development Process

- Product Backlog: a prioritized list of work for the project team to do
- Sprint Goal: the objective for the sprint, leading toward achievement of project milestones and deliverables

https://doi.org/10.1017/pds.2022.16 Published online by Cambridge University Press

- Sprint Backlog: the selected subset of the product backlog items which align to the sprint goal
- Sprint: a short development period (usually one to four weeks) which aims to complete the sprint backlog and includes the Agile events: daily meetings, sprint planning, sprint demo and review, and sprint retrospective
- Daily Meeting: a regular check-in to keep the team aligned, informed, and on track to complete the sprint
- Sprint Planning: the first event of each sprint, in which the team determines the sprint backlog, which is a subset of the product backlog
- Sprint Demo and Review: a sprint event in which the outcome of the sprint is reviewed first with stakeholders (demo) and against the sprint goal to help inform future work efforts
- Sprint Retrospective: the final event of each sprint in which the team identifies ways to improve their Agile practice

2.3. Agile limiting factors in hardware development

Application of Agile to hardware is the subject of a growing body of literature. For this paper, we chose to study both challenges that have been identified and explored in other research, as well as some challenges that have not been as well studied or documented in the literature. To begin, it is important to identify a few of the compounding factors and limitations for Agile applications in hardware.

Volatility, uncertainty, complexity, and ambiguity (VUCA) is a term used to describe highly challenging product development environments (Schmidt et al., 2018). Many of the challenges faced in product development can be attributed to the VUCA nature of this process (Schmidt et al., 2018; Atzberger and Paetzold, 2019). Much has been researched related to VUCA and product development, which will not be the focus of this paper.

Along with VUCA, several other factors of Agile for hardware development have been defined and researched: constraints of physicality, paradigm perplexity, designer's dissent, team distribution dilemma, and education and maturation (Atzberger and Paetzold, 2019). Of these factors, we will only discuss constraints of physicality in depth in our findings, as this is an often-cited limiting factor for Agile use and applicability for hardware. Constraints of physicality are the challenges unique to hardware products and systems due to the physical nature of hardware (Atzberger and Paetzold, 2019). The other items above, which are not explored further in this paper, while important, did not emerge during our research as relevant blockers for the industries and teams we interviewed.

We have selected three challenges for further explanation in section four due to their relevance to the interviewed organizations and projects: physical constraints, sprint cadence, and backlog creation. We also discuss ways that standard Agile approaches can be adjusted to address these challenges.

3. Research Methodology

We focused our research on organizations with applications of Agile in combined hardware and software product development. Within this space, we interviewed organizations across industries and team sizes and selected study participants based on satisfying three general criteria:

1. Their work had to focus on development or improvement of new or existing hardware products.
2. They had to operate using some or all of the following Agile practices: regular team meetings, retrospectives, teams consisting of personnel with multiple specializations, time-boxed sprint cadence with regular planning and review events.
3. Their project structure has more than one development team but less than ten teams working on a single effort due to our focus on Agile (not Scaled Agile, which is often made up of ten or more teams).

Once we identified potential target organizations, we organized a two-part series of qualitative interviews. We began with an introductory conversation to review the product development project, confirm their alignment to the study, and collect background information. We also focused our interviews on persons in the product owner role or similar responsibility (although job titles varied) to

https://doi.org/10.1017/pds.2022.16 Published online by Cambridge University Press

ensure critical insights into the product development process. If study criteria were met, we had a second interview where we explored in more depth a set of topics around the participants' personal and organizational process of development and application of Agile methodologies. Interviews were conducted with structured and unstructured interrogation of the topic through conversational exploration. Questions were structured by themes around the steps of the Agile methodology such as planning, work units, how sprints or equivalent are structured, and more. These were followed by a walk-through of their processes to understand these component parts in context.
Interviews were conducted with detailed notes which were later compared to the transcript to ensure thorough coverage. Question and answer themes were then compared across organizations and an affinity map was created around patterns within the conversation to isolate challenges. These challenges were then compared against the strategies used by the organization itself or the other organizations to determine if any of the interviewees had strategies to overcome said challenge. All interviews were recorded and transcribed, along with notes to review and categorize key learnings.

Table 1. Interview Participants

Company	Industry	Interviewee	HW/SW Dominance	HW and SW Competencies Combined vs. Siloed
Bose	Wearable Electronics	Creative Lead	Hardware	Integrated
Bresslergroup	Product Innovation Consulting	Senior Program Manager	Mix	N.A.
Cubii	Health and Wellness	VP Software product management	Hardware	Siloed
Ford	Automotive	Enterprise Connectivity	Hardware	Siloed
Nozomi Networks	IoT security	Co-Founder / CTO	Software	Siloed
Whirlpool Corporation	Home Appliances	IoT Digital Product Manager	Hardware	Siloed
Alpha - Unnamed IoT	Mobile IoT Platform	VP Product	Hardware	Integrated
Beta - Unnamed Consumer Electronics	Consumer Electronics	Product Manager	Hardware	Integrated

4. Results

This section examines the three types of challenges that our research identified as limiting the adoption of Agile to combined hardware and software projects: constraints of physicality, limitations of sprint cadence, and challenges of backlog creation. For each challenge, we explain strategies utilized to improve or alleviate the limitations and discuss the context in which they have been shown to benefit with regard to the subject organizations. These strategies are limited due to the context within which each strategy is applied. Agile in hardware, even more so than for software, is a practice which should be customized to the organization applying it. The strategies offered therefore provide useful examples and anecdotes for how these organizations overcame the challenges and may provide useful to other organizations facing similar challenges. It is not the assumption of this paper that these strategies are universally applicable to these challenges, again due to the individuality of context which causes these challenges. It may therefore be beneficial for an organization navigating similar challenges to investigate a solution more tailored to said organization.

4.1. Challenges of Physical Constraints

The driving concept behind the constraints of physicality is that Agile does not apply well to hardware engineering due to the timetable and demands of engineering physical products, i.e., prototyping,

https://doi.org/10.1017/pds.2022.16 Published online by Cambridge University Press

sourcing, manufacturing, etc. This physicality constraint limits the ability of hardware teams to work in an Agile sprint cadence. This notion relates directly to the Agile manifesto's call for "working software" (Beck et al., 2001), which in the software development industry is the need for so-called shippable increments (Schmidt et al., 2017). The value of consistently delivering increments of working software is getting the product into users' hands. Users can give more impactful feedback when working with a tangible product or a prototype instead of documentation (Schmidt et al., 2017). Constraints of physicality limit teams working with hardware to readily complete these iterations of shippable increments in the one to four-week sprints typical in software.

As the name implies, it is not a single constraint. The constraints of physicality are a myriad of limitations that exist when working with hardware that limits the capacity for frequent iterations. Previous research has identified 153 challenges and 160 interdependencies that are contributing factors. These constraints point to issues around separating efforts into iterations, flexibility, and scaling (Schmidt et al., 2017). Our research supported these findings, as multiple interviewees reflected on frustrations due to the limited time in Agile sprints being a key point of frustration with implementing Agile.

4.1.1. Adjustment: Reviewable Increments

A common expectation is to end a sprint with "usable" output. However, hardware teams push back around the ability to provide new usable hardware or even a noticeable difference in hardware due to lead times and the constraints of physicality. An approach to mitigating the challenge of expectations around achieving acceptable work output within a sprint is to redefine the purpose of a sprint. Each sprint begins with a sprint planning event where the team comes together to decide the goal or endpoint of a sprint. This determines the backlog of work that is to be completed within that sprint.

The essence of each sprint should evolve around the team’s progress toward understanding the problem at hand to improve future efforts to move closer to the end product/goal. For a combined hardware and software project, this means that the purpose of sprint planning is setting goals that focus the team on uncovering these key learnings, which frees the hardware team from the expectation around providing "new" or "usable" hardware. The sprint cycle should therefore focus not on usable hardware but on achieving valuable learnings and progress which can be reviewed. This helps to better define the product backlog and its prioritization.

The Bose wearables team process focuses on improving the pace of learning to de-risk and improve the next steps. When a new potential product moves out of ideation and into product exploration, one of the Bose team's first steps is to create a plan for de-risking and shortening the feedback loop. The method of divergence and convergence is viewed as a "double helix" in which the team comes together weekly to share learnings, then separates to address questions. The goal of each phase is to build the maturity of the idea. "So it's constantly this double helix coming back and forth on a weekly cadence, looking at the work until we feel confident that we can move forward". Hardware works off a conventional waterfall methodology, and software runs their planning to fit into that process. But even in this phase, the notion of a double-helix continues as they run multiple process steps where all hardware and software progress is brought together to create a new shared understanding, thus enabling the re-prioritization of tasks and needs.

4.1.2. Adjustment: Risk Reduction

Many actions in Agile development center around risk and its impact on the development process. It is important to have a clear understanding of the relevant risks present in any phase of development in order to maintain an appropriate risk level given the goals and objectives of the organization (Moran, 2014). One way to structure the sprints in Agile product development is to address the most critical questions and most significant risks at any given time to create more clarity around the next tasks. Iterative planning drives this effort, and regular checkpoints are the key to learning and moving forward as a team (Cohn, 2005). A strategy for managing risk is to include it in the prioritization of the product backlog. Prioritizing for risks of failure can allow for fast learning and re-prioritization which can reduce risks for future iterations.

https://doi.org/10.1017/pds.2022.16 Published online by Cambridge University Press

Bresslergroup is a product design and development consulting firm that works with a wide range of clients. They use a flexible method deferring to the client if they have preferred methodologies to use in the product development process. Regardless of whether a project uses Agile, Waterfall, or other methods, the notion of a backlog exists. The Senior Program Manager that we interviewed sees the role of a backlog, especially early in the processes, as a "risk register." This risk register serves to document risks at various levels from tactical risks which may derail a sprint to project level risks which might change the trajectory or a major component of the product. Its purpose is to prioritize and address these risks and capture the key learnings to move forward. Even the organization of the team and the co-location of team members can then be used to more effectively and efficiently address this backlog.

4.2. Challenges of Sprint Cadence

Our second key finding relates to the issue of sprint length. Organizations implementing Agile product development often run into pushback over the limitations of time and expectations around the ability of a hardware team to provide output within that time frame. Challenges with sprint goals and acceptance criteria are complex when working across hardware and software because the model for software goes back to that shippable increment that users can interact with. But having an organization where part of the team (software) is regularly able to meet a quick turnaround time whereas the other part of the team (hardware) struggles to complete anything "shippable" or that a user can interact with can cause challenges around adoption and acceptance of Agile practices.

4.2.1. Adjustment: Decoupling Sprints

Decoupling sprints mean that different teams can run on differing sprint schedules, "decoupling" their sprint start and finish dates. For example, a software team may run on two-week sprints, and a hardware team may use four-week sprints. If teams use differing sprint lengths, we will say they are using "decoupled sprints." The purpose of sprints is to build a product in a series of iterations, allowing for the breakdown of big complex projects into manageable pieces (Rehkopf, n.d.). Working within this time-boxed cycle using iterative efforts helps ensure that any period's focus is on the items of the highest import and for learnings to be continually taken into account and used to improve future work (Rehkopf, n.d.). The value of sprints with hardware and software should be used to enhance knowledge transfer and ensure that effort is being focused on high-value actions to ensure future efforts are appropriately prioritized.

A hardware team concerned about not making impactful progress within a one- to four-week period should consider the key risks, concerns, or items of import and the fastest way to learn about these items to select their sprint length. Decoupling sprints empowers development teams to focus efforts on uncovering key learnings that can be used for improved current and future actions. This is how the work is organized by Beta, the large consumer electronics company. When product development is underway, the overall product is broken down into projects further comprised of epics. Each team has the flexibility to structure their efforts as needed. Whereas a software team may hold to a two-week sprint structure, groups working through hardware may choose a different sprint cadence. Instead, they have a process of divergence and convergence. "There is individual planning for each team. But selected individuals from every single functional area come together, weekly, and we say, okay, what is your team doing, where are you stuck. I want one person from my hardware team, my software team, my ops team because we then form this very tight-knit group where I know what my team knows."

The expectation is that all work is planned for, and for each effort, there is a responsible individual whose job is to report back to the larger group. Each week there is a meeting amongst these responsible individuals where project status, current efforts, dependencies and interdependencies, key learnings, and upcoming work are shared. The same person attends this meeting throughout a project. This mechanism ensures the progress of efforts while supporting the decoupling of sprint cadence across teams. This enables all teams to work off their cadence or structure with the reporting mechanism available to ensure that key learnings and progress are shared across teams. There may be

https://doi.org/10.1017/pds.2022.16 Published online by Cambridge University Press

no inter-connection of sprints within this model beyond the larger milestones or convergence events when all teams come together.

4.2.2. Adjustment: Grouping Sprints

Some work cannot break down into small pieces that fit within a sprint. When this is the case, a team has the option of grouping sprints together for the purpose of completing a larger piece of work. Utilizing a short sprint cadence but having work that goes beyond a single sprint allows the team to fully benefit from the communication, shared learning, and re-prioritization provided by the sprint structure.

Alpha is a company that creates hardware for cellular connectivity within the IoT industry. Their method is to group sprints into iterations that are six weeks in length. They break down their work using epics and user stories. If necessary, user stores can be further broken down into more granular tasks or efforts by the team. Epics do not have a standard timeframe of completion within their model; it is a vision for a product or component of the product. There is an understanding around more significant deliverables needing a longer "effort" than a single sprint. An iteration is then defined as a grouping of three or more sprints. Each sprint has a goal and still holds the idea of a shippable increment. Still, the notion doesn't have a strict software connotation because the shippable item aligns with the question: did you add value to or momentum to the project? Building momentum is not solely focused on new software but could be a research item, an infrastructure item, etc., so they may not "ship" anything into production in any given sprint, but work can go across sprints within the epic. "An iteration is like six weeks really to get the size and kind of features that our product suite needs. So, you might not actually ship something to production for that sprint. In terms of the epic, you might do a bit of infrastructure scaffolding work and have some engineering-facing user stories that really say 'I need to investigate a new database solution because the problem we're solving here is a performance and scalability problem'. So that team might really just write out the goals of the performance improvements they are looking for in the technology they evaluated, and that's like a shippable increment."

4.2.3. Adjustment: Kanban Sprints

Kanban can be beneficial when a sub-section of an organization requires flexibility to pair with the larger organization. In these situations, the sub-team may utilize Kanban, where sprint lengths are deprioritized. Kanban and the use of Kanban teams is an alternative Agile framework to Scrum. In Scrum, teams commit to completing work within set sprint intervals to create learning loops to improve. In Kanban, teams organize their work visually so the status and progress of all work can be tracked and then focus on shortening time to completion, improving workflow continuously, and not focusing on the use of iterative sprints (Radigan, n.d.). Two of the interviewed organizations utilize Kanban for all or part of their teams to help navigate the challenge of static sprint lengths.

Ford Motor Company has a large and complex system of hardware and software teams that bring new products to market and manage existing products. Their development teams tend to be siloed, with hardware and software operating independently while the work efforts have a high degree of interdependence. They operate in a controlled environment due to a low tolerance for risk in their product, which drives most of their development process to use a more conventional waterfall approach. The team that was interviewed has moved from a traditional Scrum approach to Kanban. In the past, they have tried sprints of various lengths but found that the frequency of Agile events and the expectations of metrics around sprints inaccurately represent teams and their progress. They shifted the focus of their Agile approach to backlog management and now focus on top priority items using Kanban. They found that weekly Agile ceremonies best fit for accomplishing this goal and improving learning, such as "Monday metrics," where they review the previous week and use this as a retrospective. This has allowed their team to work more fluidly and reduce knee-jerk reactions to changing metrics.

Another example of Kanban being utilized is Nozomi Networks, whose products provide industrial cybersecurity solutions for operations using IoT technology. Their development organization operates with hardware and software working independently. Their software organization is larger than the

https://doi.org/10.1017/pds.2022.16 Published online by Cambridge University Press

hardware team. Since their work is independent across competency, the software team uses traditional Scrum practices, and the hardware team uses Kanban. The decision to use Kanban was due to the need for a structure and system that more closely aligned with their efforts and conditions as a team that required flexibility. Kanban allows the hardware and management organization to manage priorities and focus on keeping the backlog updated actively.

4.3. Challenges of Backlog Decomposition

A product backlog is a list of potential work that the team may take on. It consists of new features, changes to features, tasks, jobs, tests, or larger, more systemic changes. The product backlog is the authoritative source for the potential efforts of the team. All work comes from the product backlog, but placing an item on the backlog is not a guarantee that it will be taken on. The product backlog represents optionality and priority, and a task only becomes a commitment when selected to achieve an outcome during a sprint. The challenge is creating a backlog the captures the work that needs to be completed and the order in which work needs to be executed. This challenge is compounded when managing across hardware and software domains in which there is a high degree of interdependent work.

4.3.1. Adjustment: Integration Points

Navigating how to go from an end-product vision to the tangible work that needs to be done and the order in which it needs to be accomplished to meet the organization's goals is challenging. Sprint iterations fit into a more extensive planning horizon within the product development lifecycle. The challenge becomes how to manage the long-term planning horizon to improve the pace and quality of efforts. This is a problem made more acute by the constraint of physicality. The long time horizon involving iteration through hardware creates an increased need for milestone planning.

The concept of a minimum viable product ("MVP") is one common in software development. It is not about creating a minimal product but learning from the first iteration (Ries, 2011). An MVP is a beneficial target for a product development team to continue the sprint cycle functionality of working to maximize learning to improve future efforts. The need is to organize planning horizons to get to MVP and then work beyond the MVP.

In moving from a conventional waterfall approach to Agile, the notion of stage gates and milestones can continue to provide value by forcing integration. As teams develop new products, having these connection points forces hardware and software to integrate and further provides insight into work completed and prioritization for the backlog. While these predetermined points of convergence can be valuable for all teams or organizations, it is especially valuable with siloed teams or sub-efforts within larger product development processes.

The Bose organization operates with software and hardware teams working independently of each other. The creative lead within the wearables organization explained that, for planning purposes, hardware still follows a more conventional waterfall model, and software plans its development to coincide with that effort. Their product development lifecycle is punctuated by two integration points where the software and hardware progress are brought together. The first integration happens toward the beginning of development and after their product exploration phase is complete. Its purpose is to bring teams together and make sure there is a line of sight into all parts. This allows all parties to understand the current status and critical elements to be brought forward, informing future efforts. "We also have what we call a swag and swag is an integration of software to the hardware. So, and even though a line of code may not have been written at this point, we start planning out all the different sprints to coincide with the hardware milestones, not doing a large waterfall-type of planning out. It's more making sure that we have all the feature sets that we plan on releasing and start breaking this down and getting them in priority order. And making sure that the software teams have the backlinks that they need to be moving forward." The teams then carry forward, continuing their work in their siloed manner. The second integration happens later in the development process, usually around the alpha prototype of the product, and brings all the work together, and the teams then work hand-in-hand until launch.

https://doi.org/10.1017/pds.2022.16 Published online by Cambridge University Press

Beta, the consumer electronics company, uses a related approach. As described previously, this organization follows a process of divergence and convergence throughout the project lifespan. A project is broken down into epics and provided to various teams to execute. A responsible individual from each team meets with the other teams' responsible individuals weekly to review progress, findings, dependencies, etc. As each epic progresses to completion through this process of divergence and convergence, there is a final convergence. Toward the end of each epic is a "final convergence" before the epic is completed.

5. Conclusion

Our research presents current practices of Agile in combined hardware and software projects and calls forth areas and potential strategies for improved adoption of Agile outside of software development where it originated. We learned that Agile is being successfully used outside of software and that its adoption seems to be growing. There is much evidence to suggest that Agile application will continue to grow given its continued success and new areas of penetration. It continues to prove to be a valuable resource across industries and practices.

We have provided perspectives across diverse industries working with Agile processes in product development to show how new and emerging applications of Agile are being used. Using empirical evidence, we identified three challenges these organizations face, one of which has been previously well documented and researched and two which offer potentially new frames through which to see difficulties with Agile implementation. The challenges are physical constraints, sprint cadence, and backlog decomposition. These discoveries were uncovered through semi-structured interviews with eight companies whose development organizations span industries and approaches.

The challenge of physical constraints relates to teams working with hardware who, when using Agile, feel they cannot make visible progress during a sprint. Mitigating this challenge for our interviewees took two forms. First, an adjustment around reviewable increments as the output of a sprint. This is a change of objective in a sprint. Whereas conventional Agile calls for usable software, which instructs a hardware team to have usable hardware, we propose the sprint should be focused on achieving valuable learnings and progress which can be reviewed. A valuable learning is an insight which reduces the risk to the product or its development process. This could include the outcome of research into a feature, specification, user interaction and expectation, an audit of technical systems and interaction points, and more. The second adjustment to the physical constraints is around reducing risk. In this adjustment, risk is a key consideration in the prioritization of the product backlog. By structuring the product backlog to prioritize for high risk, teams are empowered to improve learnings and reduce risk over time.

The second challenge we considered is sprint cadence and the restriction that teams feel around working within the conventional one-to-four-week time-boxed sprints. The interviewees addressing this challenge used three different strategies. Decoupled sprints allow different sprint cadence for the different teams. They also have a mechanism for sharing key insights across on a regular basis separate from the sprint events. A second adjustment, grouping sprints, is a strategy used by teams to stretch larger bodies of work beyond a single sprint by combining multiple sprints. Grouping multiple sprints allows teams to continue to use the regular sprint events, ensuring that progress is being made toward the larger effort goal. Lastly, Kanban sprints utilize an alternate method for sprint planning. The use of the Kanban methodology allows teams to work in a more continuous effort, with reduced emphasis on sprint planning based on product backlogs.

The third challenge is backlog decomposition. This concern reflects the complexity of developing a prioritized backlog of efforts to be completed when there may be complex interdependence from working in the hardware space and across competencies. Alleviating this challenge can be achieved by the use of integration points within the larger project planning horizons. As teams develop new products, having these connection points forces hardware and software to integrate and further provides insight into work completed and prioritization for the backlog. While these predetermined points of convergence can be valuable for all teams or organizations, it is especially valuable with siloed teams or sub-efforts within larger product development processes.

https://doi.org/10.1017/pds.2022.16 Published online by Cambridge University Press

While this paper takes a close look at Agile for hardware development through a limited set of case study interviews, the various adjustments to standard Agile practice may be useful for many types of Agile projects. Certainly there are further challenges to be discovered and documented, and many more refinements to Agile will be made, as this is an area of engineering design practice that is evolving rapidly at this time.

References

Atzberger, A. and Paetzold, K. (2019) ‘Current Challenges of Agile Hardware Development: What are Still the Pain Points Nowadays?’, Proceedings of the Design Society: International Conference on Engineering Design, 1(1), pp. 2209–2218. doi:10.1017/dsi.2019.227.

Beck, K. et al. (2001) ‘Manifesto for Agile Software Development’. Available at: https://agilemanifesto.org/.

Cohn, M. (2005) Agile Estimating and Planning. Upper Saddle River, N.J.; London: Prentice Hall PTR: Pearson Education.

Dingsøyr, T. et al. (2012) ‘A Decade of Agile Methodologies: Towards Explaining Agile Software Development’, Journal of Systems and Software, 85(6), pp. 1213–1221. doi:10.1016/j.jss.2012.02.033.

Hannola, L., Friman, J. and Niemimuukko, J. (2013) ‘Application of Agile Methods in the Innovation Process’, International Journal of Business Innovation and Research, 7(1), p. 84. doi:10.1504/IJBIR.2013.050557.

Hoda, R., Noble, J. and Marshall, S. (2011) ‘The Impact of Inadequate Customer Collaboration on Self-organizing Agile Teams’, Information and Software Technology, 53(5), pp. 521–534. doi:10.1016/j.infsof.2010.10.009.

Moran, A. (2014). Agile risk management. Cham ; Heidelberg: Springer.

Narayanamurthi (2017) ‘Top 5 Industries That Are Adopting Agile Other Than Software’. Agile Seeds. Available at: https://agileseeds.com/agile-in-other-industries/.

Radigan, D. (n.d.) ‘Kanban’. Atlassian. Available at: https://www.atlassian.com/agile/kanban.

Rehkopf, M. (n.d.) ‘Sprints’. Atlassian. Available at: https://www.atlassian.com/agile/scrum/sprints.

Ries, E. (2011). The Lean Startup. 1st ed. New York: Crown Business.

Rubin, K.S. (2013). Essential Scrum : A Practical Guide to the Most Popular Agile Process. Upper Saddle River, Nj: Addison-Wesley.

Schmidt, T.S., Weiss, S. and Paetzold, K. (2018) ‘Expected vs. Real Effects of Agile Development of Physical Products: Apportioning the Hype', in. 15th International Design Conference, pp. 2121–2132. doi:10.21278/idc.2018.0198.

Schmidt, T.S. et al. (2017) ‘Agile Development and the Constraints of Physicality: A Network Theory-based Cause-and-Effect Analysis’, in. 21st International Conference on Engineering Design (ICED17), Vancouver, Canada.

Schwaber K., and Sutherland J. (2020) ‘The Scrum Guide’. Available at: https://scrumguides.org.

Thomson k. (2015) ‘Agile Processes for Hardware Development’. cPrime, Inc. Available at: https://www.cprime.com

Ulrich, K.T., Eppinger, S.D., and Yang, M.C. (2020) Product Design and Development. Seventh edition. New York, NY: McGraw-Hill Education.

‘Which Industries Can Benefit from Applying Agile?’ (n.d.). kanbanize. Available at: https://kanbanize.com/agile/industries.

https://doi.org/10.1017/pds.2022.16 Published online by Cambridge University Press