ADJUSTING SCALED AGILE FOR SYSTEMS ENGINEERING

Drutchas, Jake Farlon;
Eppinger, Steven

Massachusetts Institute of Technology

ABSTRACT

Scaled agile development of large systems has primarily followed the approach used in traditional systems engineering – system decomposition and static teams assigned to subsystems. However, this arrangement may result in an inefficient allocation of resources and uneven progress. This paper presents an alternative approach in which problem-based decomposition replaces system-architecture-based decomposition, and resources are flexibly allocated to problems for each sprint using ad hoc teams. Using a field study approach, we examine a mechatronic system development project utilizing these agile adjustments and discuss situations in which these methods may be successfully utilized in other projects and organizations.

Keywords: New product development, Project management, Systems Engineering (SE)

Contact:
Drutchas, Jake Farlon
Massachusetts Institute of Technology
United States of America
drutchas@mit.edu

Cite this article: Drutchas, J. F., Eppinger, S. (2023) 'Adjusting Scaled Agile for Systems Engineering', in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.48

1 INTRODUCTION

Application of agile development methods to large systems has primarily followed the approach used in traditional systems engineering – system decomposition and assigning teams to subsystems for sprint execution. Scaled agile software projects are generally organized in this manner as well, with teams having responsibility for specific portions of the system for many sprints. However, this arrangement may result in an inefficient allocation of resources and uneven progress. This paper presents an alternative approach in which problem decomposition replaces system decomposition, and resources are flexibly allocated to problems for each sprint.

2 BACKGROUND

The agile transformation of technical project management began with the Agile Manifesto for software development in 2001 (Agile Manifesto, 2001). Since then, agile methods have evolved, and standard practices have emerged, with a process based upon time-boxed sprints consisting of sprint planning, daily standup meetings, sprint review, and retrospective (Schwaber and Sutherland, 2020; Atlassian, n.d.). Planning techniques for the sequence of sprints center on a product backlog and a sprint backlog (Schwaber and Sutherland, 2020; Atlassian, n.d.), which prioritize and assign the work to the various team members. The organization's overall structure depends on the needs of the project but generally consists of a few key roles: Development Teams are tasked with the execution of the development process. These teams are generally interdisciplinary and include all specialties needed to complete assigned work. Product Owners are responsible for maximizing the value of the product being created and are accountable for ensuring the backlog is properly prioritized. Scrum Masters are responsible for maintaining the agile process and ensuring that all teams and team members are enabled to complete their work (Schwaber and Sutherland, 2020).

Agile practices have matured and evolved as organizations adjust agile to fit their needs. Several frameworks have become popular for utilizing agile processes in product development. These include Scrum, eXtreme Programming (XP), Lean software development, and many others (Dingsøyr et al., 2012). The selection of a framework depends on many factors, including the size and scale of the organization and the maturity of its process (Conboy and Carroll, 2019).

When applied to larger system development projects, with many agile teams involved, there is an increased need for governance and planning across teams. The tension that can occur when scaling agile practices is the need for defined methods that the broader team can use to all work fluidly together against the need for flexibility and empowering team members closest to the work (Thompson, 2019, p.180; Eklund and Berger, n.d.). There are many approaches to agile at large-system scale, including SAFe, LeSS, Scrum-at-Scale, DAD, and the Spotify model (Edison, Wang, and Conboy, 2021; Ebert and Paasivaara, 2017). According to an industry survey by Cprime, SAFe is the most commonly used scaled agile framework, with 37% of respondents noting its use (Cprime, 2019).

For the development of hardware systems using scaled agile methods, the challenges around team organization are perhaps even more complex and acute. Hardware teams have historically resisted the adoption of Agile processes due to various perceived limitations (Atzberger and Paetzold, 2019; Drutchas and Eppinger, 2022). Three such concerns have been well documented (Drutchas and Eppinger, 2022): 1) Physical products are harder to iterate within the limited time frame of a two- to four-week sprint. 2) Many products are difficult to decompose into modular subsystems with minimal interfaces. 3) System integration efforts may be difficult to break down into small backlog tasks.

Fundamental to agile development at any scale is the process of backlog decomposition and the assignment of development work to individuals or teams. The product backlog is the "emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team."(Schwaber and Sutherland, 2020). It is from this product backlog list that the sprint backlog is selected. In scaled agile for hardware development, the backlog is generally organized by

https://doi.org/10.1017/pds.2023.48 Published online by Cambridge University Press

subsystems or major components (Thompson, 2019 p.211). Sprint goals therefore relate to the ongoing development of each subsystem. In other words, the system decomposition into subsystems becomes the structure of the sprints in scaled agile.

The Agile Manifesto refers to "self-organizing teams"(Agile Manifesto, 2001), and in practice, this generally means that interdisciplinary development teams have flexibility in accomplishing their assigned backlog items. It is generally recommended that development teams maintain a degree of cohesion over an extended period so that team rapport can be built and metrics for work completion (sprint velocity) can be meaningfully tracked over time (Thompson, 2019, p.211; Scaled Agile Framework, n.d.).

Our study aims to consider adjustments to scaled agile for systems engineering, focusing on two aspects of agile organizations: development team organization and backlog decomposition. Based on field research at Ocado Technology, we explored two specific agile adjustments made by the organization to support their systems engineering process. We describe how these changes are used and how other organizations might consider making similar changes. Specifically, we aim to answer two research questions:

- RQ1. Sprint Decomposition: How does Ocado's problem-based sprint decomposition differ from the standard system-architecture-based decomposition used in scaled agile?
- RQ2. Sprint Team Structure: How does Ocado's structure of self-organized teams differ from the standard fixed-teams structure of scaled agile?

After discussing these two agile adjustments, we then consider under what circumstances they may be appropriate alternatives to standard agile practices.

3 RESEARCH METHODOLOGY

We explore the agile development process being used at Ocado Technology (hereafter referred to as Ocado), a leader in online grocery ordering and delivery headquartered in Hatfield, United Kingdom. Ocado's technology division designs and manufactures mobile robots for their fulfillment centers (Figure 1). As an organization, they push to continually improve the robots to enhance their performance and capability. Part of the engineering organization focuses on the next iteration of their fulfillment center robot with a team distributed over four European countries. While Ocado utilizes scaled agile for both hardware and software development, what appears to be unique is their highly flexible approach to development which allows project leaders to define short-duration sub-projects, and development teams to self-assemble on a sprint-by-sprint basis.

Figure 1. Mobile robots on a grid operating a fulfillment center (Ocado technology)

The robotic device (bot), shown in Figure 2, comprises a number of subsystems, like any other complex hardware system. Their engineering organization is grouped into several specialties called "crafts," which include mechanical engineering, manufacturing and assembly, mechatronics, testing and insights, electronics, and industrial DevOps. They utilize three-week sprints and many of the standard scaled agile processes, including sprint planning, sprint review, and regular check-ins.

https://doi.org/10.1017/pds.2023.48 Published online by Cambridge University Press

This research explores the seemingly unique way Ocado implements agile product development on a complex robotic device while maintaining a fast-paced development process. Using a field study approach, we observed a sprint cycle, from planning to review, with their development team of over 70 people, partially embedded with teams in Stockholm and partially observed through online meetings and discussions. This included ten sprint ceremonies: the leadership review of the project backlog, project proposal, project selection, sprint kickoff, weekly check-ins, and sprint review. We met technical leaders and development team members of core specialties (crafts), including the leaders of the mechanical teams, manufacturing and assembly, mechatronics, and others. This allowed us to understand how team members experienced the planning and execution of agile sprints. Further interviews included in-depth conversations ranging from 30 to 90 minutes with the director of engineering and the project manager lead for agile process.

Detailed notes were taken for all meetings and interviews to provide the background for understanding how the Ocado agile process unfolds. Interviews were unstructured and conversational to provide flexibility in exploring how this sprint process worked and how team members operate within this structure at various levels. This paper focuses on the agile adjustments around systems decomposition and development team structure.

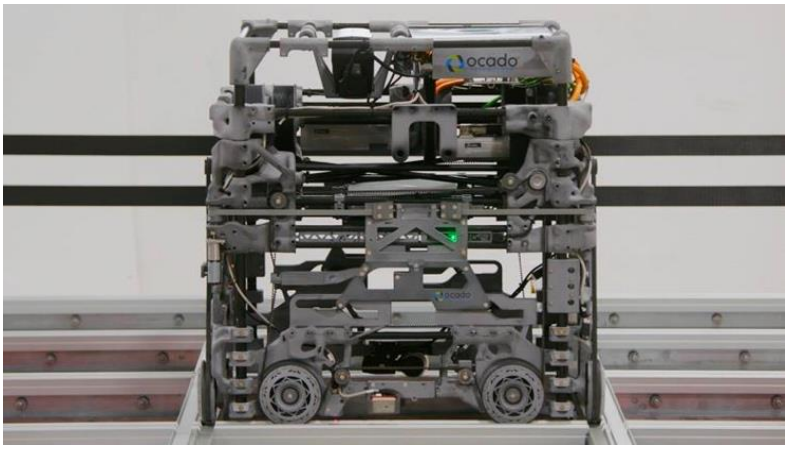

Figure 2. Ocado's mobile robot for grocery warehouse fulfillment (Ocado technology)

4 RESULTS

Based on our detailed observations of Ocado's scaled agile process, we discuss each research question in turn.

4.1 RQ1. Sprint decomposition

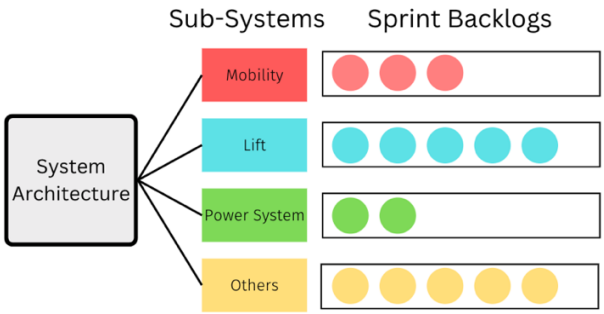

Figure 3. Conventional scaled agile sprint backlog decomposition

The conventional approach to develop sprint backlogs in scaled agile is through a system-architecture-based decomposition, as depicted in Figure 3. The program process, known as program increment planning, involves teams, each aligned to a subsystem or component, defining their sprint goals, estimating their sprint capacity, and defining the backlog items to develop and integrate their subsystem for the planned milestone(s) (Leffingwell, 2018).

https://doi.org/10.1017/pds.2023.48 Published online by Cambridge University Press

Hypothetically, Ocado's fulfillment center mobile robot being developed by a conventional scaled agile organization would have teams assigned to subsystems. For example, the mobility team would develop all parts related to the movement of the bot on the track. The lift team would focus on the mechanism that enables the bot to lift and lower totes. The power system team might handle work related to power management, including energy storage and power controls. Other teams would address the remaining subsystems. Backlog items in this subsystem-based approach would be slotted into the relevant subsystem development teams and prioritized within their lists.

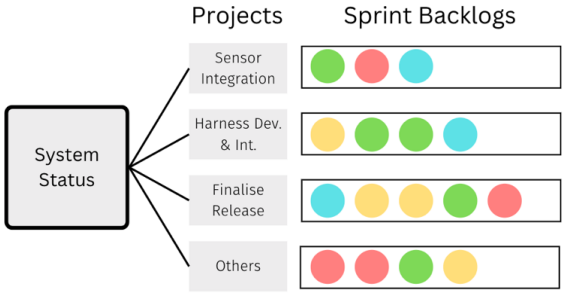

Figure 4. Ocado's project-based sprint backlog decomposition

Where the Ocado process differs is that sprint planning starts with an assessment of the system status and considering what "problems" are most important to address in the next sprint (Figure 4). A problem may involve multiple components across several subsystems; in this way, robustness and integration issues are naturally handled earlier in the process. The product owners, scrum masters, and other leaders also consider the upcoming milestones to ensure that the work done in the upcoming sprints aligns with the milestones. These problems are then prioritized and organized into a sprint structure in which a single problem or group of related problems forms what the Ocado team calls a "project".

Through the refinement and sprint planning process, 10-15 projects of various sizes are selected based on an assessment of which ones contribute the most value toward the goal of a fully operational and robust system. The job of backlog refinement includes ensuring that projects are of an appropriate size and that the group of projects will utilize the development organization's capacity. Examples of such projects include: sensor integration – mechanical and electrical integration of a new proximity sensor; harness development and integration – design, build, and mechanical integration of a new electrical harness for the bot; and finalize release – preparation of all significant changes to be included in the next release.

As the engineering director at Ocado explains it, "Utilizing problem-based project prioritization provides great flexibility to unblock the critical development path – whether this be an unexpectedly urgent supply chain decision, providing data or an interface to another department, or ensuring one subsystem does not mature before another. It also allows for holistic pieces of work to be completed rather than one engineering craft being dependent on another's output."

This sprint planning process is a radical shift from a subsystem-based approach to sprint decomposition to what we call a problem-based approach to sprint decomposition. This also avoids the issue potentially faced by mature agile organizations in which there is too little work for a subsystem to do in a sprint, resulting in team underutilization.

4.2 RQ2. Sprint team structure

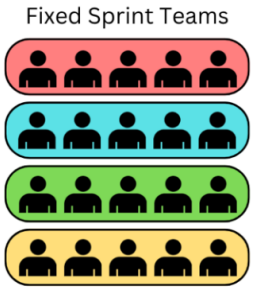

Figure 5. Conventional scaled agile fixed development teams

https://doi.org/10.1017/pds.2023.48 Published online by Cambridge University Press

In conventional agile and scaled agile, the sprint planning process is organized around fixed development teams which persist across several sprints, and ensures that each team has a sprint backlog sized to fully utilize the team for the sprint duration (Schwaber and Sutherland, 2020; Atlassian, n.d.; Leffingwell, 2018). When scaled agile follows this team-centered sprint planning approach, the product backlog is formally or informally decomposed by the development team or subsystem. Development teams can therefore maintain their backlogs based on their specialty or area of ownership. Unfortunately, during the project, a team's sprint backlog can drop below the effort available during a full sprint. They may then work ahead to prepare for future milestones or slow down efforts to fill the sprint duration.

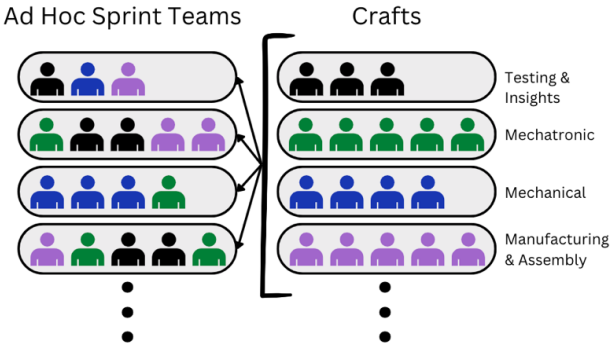

Figure 6. Ocado's Ad Hoc sprint teams comprising craft resources

Ocado uses a more flexible approach to team structure in which the development organization mirrors the dynamic structure of the projects. In the problem-based sprint backlog decomposition explained above, the most critical problems that can be reasonably addressed during the sprint are defined. Then, instead of being allocated to a fixed team, as in conventional agile, the Ocado team members individually assign themselves to the projects during sprint planning until all team members are allocated to projects that fully utilize their time.

This scaled agile organizational structure, therefore, employs "ad hoc sprint teams," with ad hoc meaning "formed or used for specific or immediate problems or needs"(Merriam-Webster.com, 2021). In this approach, the sprint teams do not have a subsystem specialty or any pre-existing composition. Instead, in defining the projects, the engineering leaders determine how many people of each "craft" are needed to accomplish each project. Crafts are groups of technical resources, such as mechanical engineers, testing engineers, manufacturing /assembly engineers, and others. They leave it up to the individuals in each craft to decide who will join each project and in what role, either as direct project team members or advisors.

Ocado's engineering director explained, "A challenge with multidisciplinary teams is the fact that different types of engineers can talk different engineering languages. For example, 'sigma' can mean any of: mechanical stress, electrical conductivity, chemical bond type, etc. Putting two engineers from different backgrounds in a room and asking them to come to a decision is not always simple, and the conversation needs to start with deciding upon a common language. Our ad-hoc sprint teams mean the mixture of crafts is created at the start of the project, the engineers have the time to understand each other, work with each other, and value each other, such that at the end of the project and point of decision, the conclusions can be sympathetic to each other's perspective."

This process allows for a series of benefits to be realized immediately, including eliminating the effort of managing static team resources with various training needs, vacation schedules, and so forth. It empowers team members to pursue active areas of interest, a specific concern that the management of the Ocado organization considers when planning begins. It also allows them to focus greater effort on the most crucial work at any given time while the autonomy ensures that all team members are fully engaged.

5 APPLICATIONS OF AGILE ADJUSTMENTS

Having described the two main adjustments Ocado has made to conventional scaled agile, we now consider the broader applicability of these two practices. We will explore the following three questions: Why does this approach work at Ocado? Could other system development projects utilize this approach? Do these two adjustments only work together?

https://doi.org/10.1017/pds.2023.48 Published online by Cambridge University Press

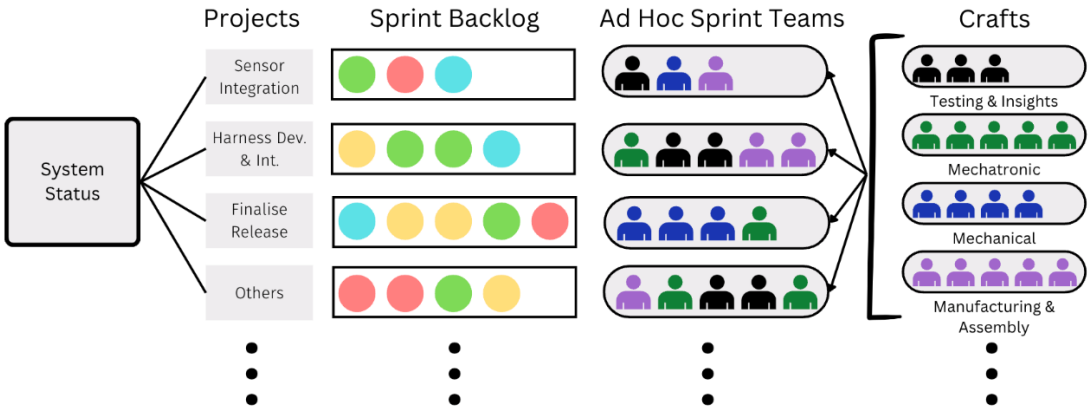

Figure 7. Ocado's approach to scaled agile development

5.1 Why does this approach work at Ocado?

Ocado's engineering leadership team developed this process over time with the launch of their new product group. The team had a starting point, an earlier version of their robot system already in operation, and was tasked with designing the next iteration. They knew the subsystems, components, and critical functions, and had to decide what and how they wanted to change. They decided to step away from the conventional systems engineering approach, mix up the subsystem teams, try new things, and re-integrate the parts. The team leading this effort was small and readily able to focus on mission-critical problems.

As the organization grew, they knew they had to develop a process that supported the expanding scale. They experimented with working across subsystems and focusing on problems. They immediately realized the benefit of defining sprints around critical problem areas rather than components. Pairing this approach with dynamically flexible teams allowed them to fluidly navigate the range of sprint topics and needed expertise. The model even supports significant team transformations, such as integrating new employees or an entirely new team into the expanding organization. This adaptability has been invaluable as the Ocado team has progressed through the system development lifecycle.

5.2 Could other system development projects utilize this approach?

An important consideration around this system of changes and the value brought to an organization by operationalizing problem-based decomposition and ad hoc teams is in the circumstances of applicability. Consider four typical phases of the product development lifecycle: product planning, product design, integration, and maintenance. Generally, each phase requires different skill sets and specialties in different quantities. Perhaps the planning and integration phases require more holistic thinking and system engineering skills, whereas the design and maintenance phases may require expertise around particular components, subsystems, and operations. While this more flexible approach to scaled agile may apply across the development lifecycle, we believe its benefits are especially relevant in the integration phase, when the organization has to deal with the uncertainty and breadth of potential failure modes. Therefore, having teams that can flexibly adapt to meet the emergent and unpredictable challenges of system integration is particularly valuable. For example, subsystem and system-level testing, analysis, redesign, and integration within a single sprint require a highly focused, problem-based effort.

While we can readily envision these agile adjustments being applied to other medium-scale system engineering efforts, they may not be appropriate in very large system development efforts where a large number of team members would have great difficulty with ad-hoc sprint planning, and this may negate the value of the flexible approach. Furthermore, these adjustments may not be as helpful in highly modular or stable systems engineering projects like smartphones or automobiles. In these types of projects, the various subsystems may require specific types of expertise throughout the product development lifecycle; indeed, expertise tied to each subsystem and its unique integration challenges is built over time in the organization.

https://doi.org/10.1017/pds.2023.48 Published online by Cambridge University Press

5.3 Do these two adjustments only work together?

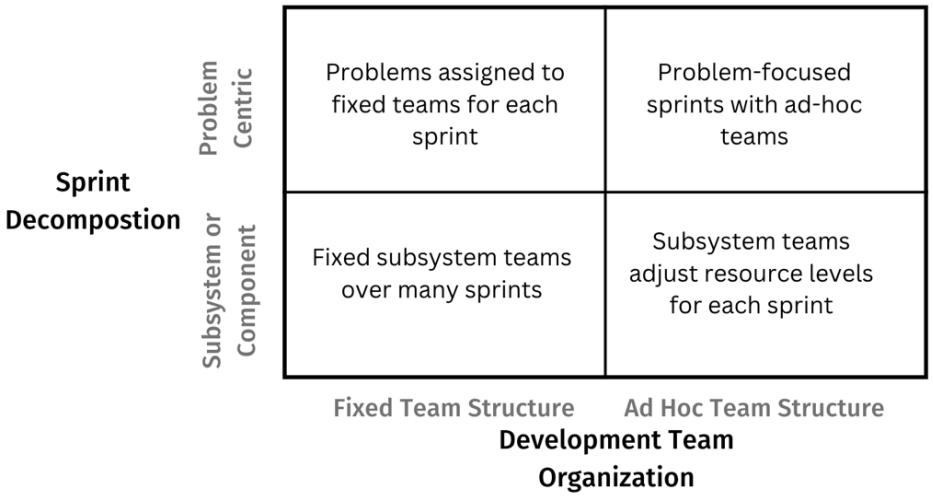

Figure 8. Four possible ways to organize scaled agile sprints for systems engineering

Notwithstanding Ocado's success in creating an agile systems engineering process and structure involving these two specific adjustments, it is clear that each adjustment provides a specific solution to a particular challenge. Thus, it is not difficult to imagine a development organization utilizing one of these adjustments or the other as it fits their unique situation. Figure 8 depicts four combinations of these two choices: fixed versus ad hoc team structure and subsystem versus problem-based sprint structure.

In the upper left quadrant, we envision an engineering organization with fixed teams but defining sprint projects based on prioritized problems rather than subsystems. In the lower right quadrant, we envision sprints defined around subsystems, but resources are flexibly allocated in an ad hoc fashion according to the needs of each sprint backlog. Whereas we did not study an organization separately utilizing either of these two agile adjustments, we believe these structures are possible and valuable options for engineering managers to consider.

6 CONCLUSION

We have observed and documented two key practices in the successful application of scaled agile at Ocado, which, importantly, differ from traditional practices. Ocado utilizes a problem-based approach to sprint decomposition instead of a subsystem decomposition. They use an ad hoc approach to sprint team structure instead of the conventional fixed team approach. While Ocado has implemented these practices for its mobile robot development, other organizations facing similar system development and integration challenges could also find the approach useful.

The Ocado adjustments to scaled agile, individually or collectively, offer valuable outcomes to a team considering adoption. These adjustments offer an approach to building more flexibility into the scaled agile process and its teams. To implement it, team members need to be flexible to accomplish the highest value backlog items in a timely manner through focused team effort.

As agile project management becomes more widely adopted, it continues to evolve as organizations experiment with novel approaches. We believe there is a need and an excellent opportunity for future research in this space, particularly in the application of scaled agile to complex systems. While the focus of this study was the Ocado organization and its two specific adjustments, the potential of other organizations to follow this approach suggests the analysis of similar implementations to understand better where and when it is usefully applied. In addition, we have raised the possibility of selectively applying either the problem-based sprints or the ad hoc team structure. This suggests further analysis could also be undertaken to understand where these agile adjustments might best apply.

https://doi.org/10.1017/pds.2023.48 Published online by Cambridge University Press

REFERENCES

Agile Manifesto (2001). Manifesto for Agile Software Development. [online] Agilemanifesto.org. Available at: https://agilemanifesto.org/.

Atlassian (n.d.). How to create and use sprints in Jira Software. [online] Atlassian. Available at: https://www.atlassian.com/agile/tutorials/sprints.

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. https://dx.doi.org/10.1017/dsi.2019.227.

Conboy, K. and Carroll, N. (2019). Implementing Large-Scale Agile Frameworks: Challenges and Recommendations. IEEE Software, [online] 36(2), pp.44–50. https://dx.doi.org/10.1109/ms.2018.2884865.

Cprime. (2019). Agile at Scale Report. [online] Available at: https://www.cprime.com/wp-content/uploads/2020/10/Agile_at_Scale_Report_2019.pdf [Accessed 18 Feb. 2023].

Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N.B. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, [online] 85(6), pp.1213–1221. https://dx.doi.org/10.1016/j.jss.2012.02.033.

Drutchas, J. and Eppinger, S. (2022). Guidance on Application of Agile in Combined Hardware and Software Development Projects. Proceedings of the Design Society, 2, pp.151–160. https://dx.doi.org/10.1017/pds.2022.16.

Ebert, C. and Paasivaara, M. (2017). Scaling Agile. IEEE Software, 34(6), pp.98–103. https://dx.doi.org/10.1109/ms.2017.4121226.

Edison, H., Wang, X. and Conboy, K. (2021). Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review. IEEE Transactions on Software Engineering, pp.1–1. https://dx.doi.org/10.1109/tse.2021.3069039.

Eklund, U. and Berger, C. (n.d.). Scaling Agile Development in Mechatronic Organizations -A Comparative Case Study. [online] Available at: https://arxiv.org/pdf/1703.00206.pdf [Accessed 16 Nov. 2022].

Ocadogroup.com. (2017). Ocado Group. [online] Available at: https://www.ocadogroup.com/.

Leffingwell, D. (2018). SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises. [online] Google Books. Addison-Wesley Professional. Available at: https://www.google.com/books/edition/SAFe_4_5_Reference_Guide/zrdZDwAAQBAJ?hl=en&gbpv=0 [Accessed 16 Nov. 2022].

Merriam-webster.com. (2021). Merriam-Webster Dictionary. [online] Available at: https://www.merriam-webster.com/dictionary/ad%20hoc.

Scaled Agile Framework. (n.d.). PI Planning. [online] Available at: https://www.scaledagileframework.com/pi-planning/.

Schwaber, K. and Sutherland, J. (2020). The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game. [online] Available at: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdfa#zoom=100.

Scrum.org (2019). What is a Sprint Retrospective? [online] Scrum.org. Available at: https://www.scrum.org/resources/what-is-a-sprint-retrospective.

Thompson, K. (2019). Solutions for Agile Governance in the Enterprise (Sage).

https://doi.org/10.1017/pds.2023.48 Published online by Cambridge University Press

CAMBRIDGE
UNIVERSITY PRESS