SIN-58-E
February 2019

Agile Methodologies and Organizational Agility

Claudia Röthke
Robert Wayne Gregory

Introduction

Organizations are experiencing a profound shift in the way they need to deliver value to customers. While classic project management approaches emphasize efficiency and meeting internal business demands in the delivery of projects, they tend to lack flexibility in their way of approaching change, dealing with uncertainty and reducing time to market. This leads to many projects not being delivered on time, within budget or with the required scope or quality. Classic project management, which typically follows the waterfall model in software development, uses a linear path of planning and executing that is modelled in Gantt charts. The planning phase focuses on analysis and requirements definition, while the execution phase consists of building and delivering the required offering. While this approach provides great benefits and is appropriate in predictable environments where requirements and user needs are clear, specific and stable, in more dynamic and complex environments the waterfall approach has significant limitations.

The contexts and conditions under which value offerings are developed and delivered today has changed dramatically over the last one to two decades, where digital technology has converged with traditional products and has become embedded into customers' everyday lives. As a result, customers' expectations and needs have become more diversified and individualized, their technical literacy has advanced significantly, and their demands for seamless customer experiences have been increasing from year to year. These customer- and product-related changes are reinforced through the fast-paced evolution of digital technologies in terms of their affordability, accessibility and ability to offer highly personalized and joyful experiences. In combination, these developments are irreversibly reshaping the environments of established firms across different sectors and business fields. An additional source of pressure for change is a continual stream of new entrants that leverage the windows of opportunities stemming from these technological and environmental changes and the inertia of incumbent firms and regulators.

This technical note was prepared by Claudia Röthke, Product manager and coach, and Professor Robert Wayne Gregory. February 2019.

Last edited: 2/18/19

In sum, organizations have to become more agile and adjust the way they plan and build offerings, by increasing the frequency with which they deliver and the way they learn from customer feedback during iterative product development and delivery. Nowadays, we find ample examples across industry sectors, including more traditional and regulated sectors such as banking—which strives to respond to the opportunities and threats from more agile fintech startups—but also manufacturing, where traditional automotive players strive to compete with platforms focusing on rentals and sharing.

Agile methodologies, mindsets and organizational models were introduced into organizations to help development teams adapt more quickly to change, build iterative feedback from customers into the development process and focus more on creating differential customer value. A core underlying belief is the need to better leverage the unique potential of each individual participating in this process, as well as the need to reduce the amount of waste produced (e.g., as a result of delivering products that really don't meet customer expectations). Agile methodologies are frameworks that promote iterative learning as well as early delivery and continuous improvement throughout the development of an offering.

The purpose of this technical note is to convey the essential characteristics of the agile mindset that underpins a variety of agile methodologies and frameworks used in practice. Secondly, different agile methodologies and their advantages are explained in detail. Finally, the implementation of agile methodologies in organizations is discussed. In addition to known operational changes that are necessary, the broader organizational transformation that this type of change entails when introducing agile thinking from the start-up world into large established firms is also discussed.

Agile Mindset

In 2001 representatives from several innovative and agility-oriented software development practices founded the Agile Alliance in order to move beyond the rigid waterfall approach and heavyweight documentation that has driven software development processes for decades. The vision was to reinvent the mindset and methodologies for software development in an interconnected, complex and hyper-competitive world. The group formulated the "Manifesto for Agile Software Development,"[1] which recommends conditions in which software can and should be built more effectively through an agile approach. The manifesto and its underlying principles promote the creation of trust within teams and self-organization. It also focuses on frequent delivery and collaboration with customers. Overall, the beliefs and principles underlying the manifesto extend far beyond the specific context of software development from which they emerged; they also offer general guidance for organizational change and leadership in the digital era. Each agile methodology and approach has its own set of unique characteristics and associated vocabulary. At the same time, adopting an overarching agile mindset is crucial for the successful adoption and use of any one of these different agile approaches.

The manifesto provides behavioral guidelines for the use of agile methodologies in day-to-day work. In the following guideline statements, what is mentioned first is valued more in relative terms over what is mentioned second.

- Individuals and interactions over processes and tools

[1] "Manifesto for Agile Software Development", 2001, https://agilemanifesto.org/.

- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

In addition, there are twelve principles[2] that enhance the manifesto and give more detailed practical guidance. These principles are:

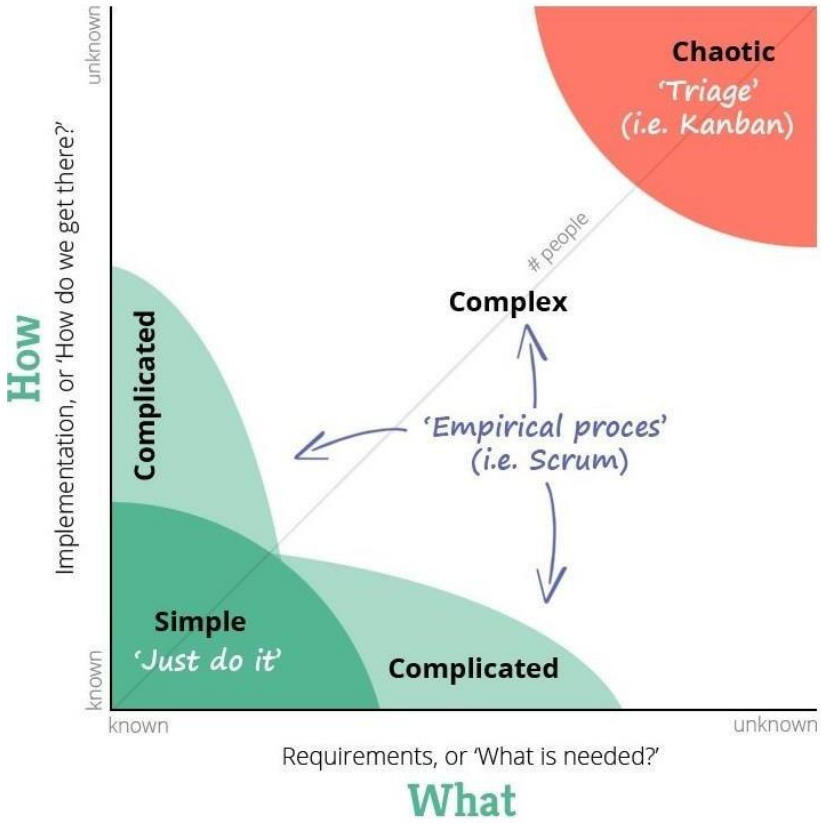
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, between every couple of weeks and every couple of months, with a preference for 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 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.

Agile Methodologies

The use of agile methodologies is not the best option for every project, considering the cost the agile transformation incurs. Prior to deciding for or against developing your next product with an agile approach, the context of the project and product-related work should be evaluated. The Stacey Matrix (see Figure 1) is a model that helps to characterize organizational complexity and implications for management. It is often adapted to explain the value of agile versus traditional approaches for the development of new solutions. The significant factors in the model are firstly, the level of clarity about the requirements for the new solution (What is needed?), secondly the extent to which requirements for implementation are clear (How do we get there?), and thirdly, the number of people involved in the development of a new solution.

[2] "Manifesto for Agile Software Development", 2001, https://agilemanifesto.org/principles.html.

Figure 1
Stacey Matrix



Source: "Stacey Matrix", https://freeagile.org/2017/10/01/stacey-matrix-adapted-to-software-development.

Projects are situated in the simple zone when there is no uncertainty or risk about the technologies and expertise needed to create, maintain and deliver the offering, and when requirements of customers and stakeholders are clear, agreed and communicable. This is the likely scenario in routine product work or fixed-time and fixed-scope projects. In this zone, the classic waterfall development approach makes sense and the efforts of introducing an agile methodology would most likely not pay off.

When the opposite is true, and both technological and requirement uncertainty is high, organizations find themselves in the chaos zone. A good example is the first attempts of automobile manufacturers in the 1990s to build connected cars through telematics, typically driven by research and development (R&D) units focused on blue ocean exploration. For these projects, development methodologies like Kanban can be applied, with the goal of increasing knowledge about requirements and implementation strategy by breaking the project down into spikes. Spike are stories with time-boxed estimations that promote learning.

The zones in between both extremes are characterized by a certain level of uncertainty, on either one or on both axes. Projects in these zones are very suitable for the use of agile methodologies, because their semi-structured approaches help break down complexity while leaving room for flexible learning and adaptation through iterative delivery.

Developing software today is increasingly characterized by high interdependencies, complexity and risks. Digital offerings are interconnected with other offerings, some of them owned and controlled by the focal firm, others not. To ensure the reliability and performance of these offerings, these interconnections have to be managed well. Increasing technological complexities are reinforced by technological trends, like the Internet of Things (IoT) and artificial intelligence. In particular, technology mega trends continually reshape customer behaviors and expectations, making it increasingly difficult to predict user needs and market dynamics. As a result, reaching clarity and agreement over requirements has become a prime challenge. As discussed in the following sections, the most common agile methodologies are suitable for dealing with this challenge.

Scrum

Scrum founders Jeff Sutherland and Ken Schwaber define their methodology as: "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value."[3] Scrum is based on empiricism, which promotes the idea that knowledge comes from experience and decisions need to be made based on what is known. Empiricism affords transparency, inspection and adaptation in order to be implemented successfully. These are the three pillars of Scrum, and they are supported by the following core values: commitment, courage, focus, openness and respect.

Scrum teams live by and embody these values. The Scrum team comprises three roles. First, the product owner is responsible for maximizing the value of the product being built. This means being responsible for communicating the vision to the development team and prioritizing the product features by customer and business value. The product owner needs to align with customers and other stakeholders in the organization, but ultimately decides what should be built. Second, the development team is responsible for delivering product increments—based on the priorities by the product owner—at the end of the sprint. A product increment is an added piece or iteration of software that introduces a new feature or change in functionality, as envisioned by the product owner. The development team is self-organized and cross-functional and decides how the product increment is built. The development team size should be between three and nine members. Third, the Scrum master is responsible for promoting and supporting the Scrum methodology. The Scrum master helps the team members understand Scrum theory, practice, rules and values, and supports them in removing impediments. Scrum masters are servant leaders who help the product owners, the development team and the organization to maximize the value created by the Scrum team as a whole.

Scrum events are time-boxed meetings that create regularity and structure and support the pillars of transparency, inspection and adaptation. The heartbeat of Scrum is the sprint, which is the time frame in which the development team creates the product incrementally. The length of the sprint is decided at the beginning of the project and should not exceed one to four weeks. Short sprint cycles are preferred, since changes can be made only in between sprints, not during a sprint. The sprint itself consists of the sprint planning, daily scrums and development time, the sprint review and the sprint retrospective.

[3] Ken Schwaber and Jeff Sutherland , Scrum Guide (November 2017), https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf.

In the sprint planning, the entire team meets to plan the upcoming sprint collaboratively. The product owner explains the purpose and goal of the user stories, and the team estimates how the user stories will be achieved.

The daily scrum is the meeting in which the development team inspects the work done since the last daily scrum and plans work for the next 24 hours. The main goal for the daily scrum is to inspect the progress towards the sprint goal and discuss what is needed to accomplish their commitment. This is a fundamental inspect-and-adapt meeting that improves communication and level of knowledge in the team and identifies impediments. The typical questions answered in the daily scrum are:

1. What did I do yesterday that helped the development team meet the sprint goal?
2. What will I do today to help the development team meet the sprint goal?
3. Do I see any impediment that prevents me or the development team from meeting the sprint goal?

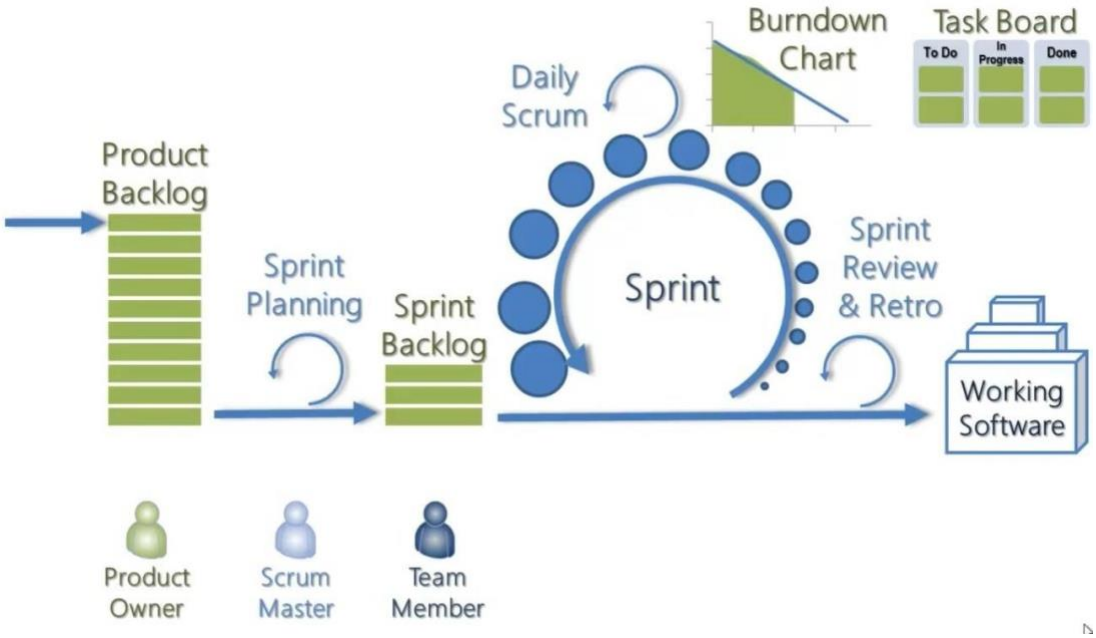
The sprint review can be regarded as one of the major success factors for Scrum. In the sprint review, the increment that was developed in the sprint is demonstrated by the development team and product owner and inspected by the stakeholders who have an interest in the product. The goal is to elicit feedback by the stakeholders and identify if the increment delivers value and if the timeline, budget, and ultimately the backlog, need to be adapted for the upcoming sprint.

The retrospective is the meeting that supports the scrum team in inspecting and adapting the way they work together. By discussing progress during the last sprint in terms of collaboration, communication, processes and tools, the team identifies things they want to improve and creates a plan for how to implement the improvements.

Figure 2 gives an overview of the Scrum roles, events, artifacts and workflow. While Scrum intends to be the opposite of a rigid, stagnant ruleset that has to be followed for project success, it is a framework that should be implemented in its entirety in order to be successful. For more information and details on the Scrum elements and implementation, check out the *Scrum Guide* by Jeff Sutherland and Ken Schwaber.[4]

[4] https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf.

Figure 2
Scrum Roles, Process and Artefacts



Source: Bryan Marshall, "Agile Scrum Overview", 2018, https://www.bryanmarshall.com/agile-scrum-overview/.

Kanban

Compared to Scrum, which is a rather extensively team-centric agile methodology, Kanban is a more lightweight methodology focused on value creation along the product development value chain.[5] Kanban was publicly introduced to the information technology (IT) sector by David Anderson but is originally based on Toyota's lean production principles and manufacturing. Kanban's fundamental goal is to avoid waste by creating an optimal and fast flow of tasks when building a product. This is accomplished by promoting several practical rules for Kanban application.

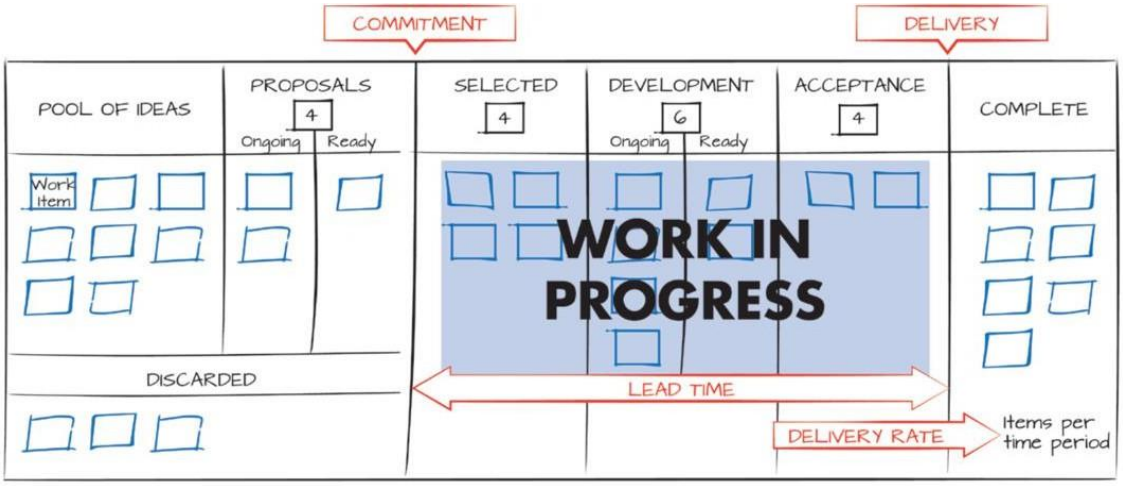
In order to visualize the flow and state of tasks when developing a product, a Kanban board is created. The board consists of at least three columns, which can be simplified as "To Do," "In Development" and "Done." More columns can be added if doing so helps the team to organize the work. The goal is that tasks from the left (in the backlog) move to the right (ready for release) in a continuous flow. While doing this, only a limited number of tasks are worked on at the same time. Therefore, the team defines work in progress (WIP) limits for each column, which helps with the visualization of bottlenecks. When the WIP limit is reached, the team knows it needs to complete tasks that have already been started before taking on new tasks. This keeps the work flowing from left to right along the chain of tasks to be completed for building the product. Another supporting element for keeping up the task flow along the chain is reducing the batch

[5] For an extensive comparison of Scrum versus Kanban, with a focus on team learnings and team maturity, watch this video: https://www.youtube.com/watch?v=HNd1_irOL5k.

size of tasks. This means that large tasks with a higher complexity of requirements need to be broken down and reduced in size.

An example of a more elaborate Kanban board is shown in Figure 3. The columns marked with "work in progress" consist of the tasks the teams are actually working on and are committed to delivering. The time it takes from commitment to delivery is called "lead time." This is the time that should be reduced to the minimum for any single task. The columns before the "commitment" line show tasks that might be built in the future. This gives the teams an outlook and perspective of what is coming up. The numbers symbolize the WIP limits the team agreed to.

Figure 3
Kanban Board



Source: http://dysfunctionalrequirements.com/wp-content/uploads/2016/10/Kanban.png.

Unlike in Scrum, there is no formally proposed development time frame like a sprint. Therefore, the development teams decide through collaboration with the product manager when to release a product increment. Moreover, there is no equivalent role like the Scrum master, so the members of the team or organization are in charge of organizing their own work. Nevertheless, there are a number of duties that using Kanban implies. They need to be taken up by somebody in the team or organization and can be guided by an internal or external agile coach. These duties include monitoring and controlling the flow of tasks and adapting measures, making the rules for the process explicit and improving leadership on all levels of the organization. The goal is to continuously and constantly improve the Kanban process in the teams and all over the organization.

To summarize, Kanban is less formalized than Scrum, but it builds on similar values and goals. Some agile enthusiasts promote Kanban as the logical evolution of working with agile methodologies.[6]

[6] For an exemplary case on the sequential development from waterfall to Scrum to Kanban, read the story of engineering director Grant Ammons. "Ditching Scrum for Kanban — The best decision we've made as a team." https://medium.com/cto-school/ditching-scrum-for-kanban-the-best-decision-we-ve-made-as-a-team-cd1167014a6f.

Extreme Programming (XP)

Ron Jeffries, Kent Beck and Ward Cunningham developed Extreme Programming as "a lightweight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly changing requirements."[7] They compare developing software to driving a car down a winding road. It is not enough to just point the car in one direction and go, but rather it must be steered in case of bends in the road and obstacles appearing suddenly. Using this metaphor, XP is a methodology that proactively embraces risk—and even plans for it. Moreover, XP takes on a mentality of sufficiency, meaning that software should be developed as if all resources were available. This includes taking time for recurrently testing and refactoring software, and discussing outcomes with fellow developers and customers; the way it should be done to maximize code quality.[8]

The main differentiator from other agile methodologies is that XP focuses specifically on producing higher-quality software. Therefore, XP prescribes operative engineering practices—specifically, coding and developer practices. Coding practices include simple coding and design, continuous refactoring of written code, developing shared code standards among the team and forming a common vocabulary. Developer practices consist of test-driven development, pair programming, collective code ownership and continuous integration.

Similar to Scrum and Kanban, XP is also based on the basic ideas and principles of the "Manifesto for Agile Software Development," including the goal to deliver software increments quickly to gather feedback from the market and customers. Similar to other agile methodologies, the core values of XP are simplicity, communication, feedback, courage and respect.

Implementation

There are many opinions about what the best way is to introduce and implement agile methodologies within an organization: Top-down or bottom-up? Starting in a staged process with one team, one project, one product or one department at a time, or in a big bang across the whole organization? Is there a blueprint of how agility matures over time in teams and organizations? Should we do Scrum first and then switch to Kanban?

Before deciding if an agile methodology should be introduced, analyze your project and assess whether requirements are clear—both from a customer and an organizational perspective. To create clarity, it is advisable to use Design Thinking, Customer Discovery and Lean Start-up methods.[9] Moreover, assess the technological risk that comes with starting the project or developing the product, and involve seniors who can work with these technologies.

When you decide to use an agile methodology, it is important to gain buy-in and trust from the management. Train them on agility, its values and how the change will affect *their* work. Managing expectations about release cycles, backlog prioritization and performance predictability is very important at this point, as well as giving management the opportunity to voice doubts. After all, you are igniting an organizational change, potentially an organizational

[7] Kent Beck and Cynthia Andres, Extreme Programming Explained: Embrace Change (Addison-Wesley, 2004) p. 3.

[8] More details on the principles and practices of extreme programming can be found here: https://www.agilealliance.org/glossary/xp.

[9] For more details, see technical notes by Claudia Röthke and Robert Wayne Gregory "Qualitative Customer Research", SIN-56-E, IESE, January 2019 and "Hypothesis Driven Experimentation.", SIN-57-E IESE, February 2019.

transformation, and it needs to be supported and understood by the management.[10] For example, managers often misinterpret the agile value "responding to change over following a plan" as not needing a plan and just dropping in feature ideas that will be ready after the next sprint.

Secondly, set up the team(s) and convey to them an agile mindset and set of principles, and create the roles, events, artifacts and development process that are needed to begin. Then, kick off the development and stick to the same methodology and processes for a couple of weeks, in order to reach a routine and learn. Apply the chosen methodology with all the elements it promotes, and collect data, measure outcomes and seek stakeholder feedback.

Thirdly, analyze the collected data from your experiences and inspect what works for your team and organization. Adapt what does not work well in the specific context of the team or organization. Revisit the analysis of software increment delivery, team happiness and stakeholder feedback at least quarterly and adjust continually. Potentially develop your own version of agile methodology over time.[11]

Conclusion

In summary, the value of agility lies in learning from experience, adapting to change and improving team performance, while working towards one important goal: producing tested, integrated, working, shippable product increments that create value for customers. Achieving this goal requires more than just introducing agile methodologies; it also entails a large organizational transformation. It is not a quick fix that can be prescribed by management to increase the number of releases. Yet, it comprises changes in culture, processes, organizational setups, communication and most importantly, individual mindsets.[12] Not applying agile methodologies and values might successively result in a competitive disadvantage for organizations developing digital products today.

There is a lot of dispute about the correct implementation of agile methodologies generally, and Scrum specifically. Before engaging in discussions with agile enthusiasts and promoters, it is more important to find a methodology that helps your organization and teams flourish. Agile methodologies underline a more human-, product- and customer-centric approach to developing and delivering (digital) products in a hyper-competitive world than classic practices ever did.

[10] John Cutler, "10 Ways the C-Suite Can Support Agility," December 8 2017, https://medium.com/@johnpcutler/10-ways-the-c-suite-can-support-agility-81a369bc549b.

[11] For insights into agile maturity, see the video on the engineering culture at Spotify: https://www.youtube.com/watch?v=R2o-Xm3UVjs.

[12] For a controversial post criticizing the hype around agile methodologies and the request to focus on the agile mindset and values in the time of transformation, see: https://freeagile.org/2018/12/26/the-year-they-kill-agile/.