Introducing 'Human-Centered Agile Workflow' (HCAW) – An Agile Conception and Development Process Model

Leonhard Glomann(✉)

LINC Interactionarchitects GmbH, Munich, Germany
leo.glomann@linc-interaction.de

Abstract. Amongst today's successful digital companies, customer-centricity is, in one way or another, at the core of their business strategies. Successful roll-outs without previous analysis of context, needs and tasks of the actual people who are supposed to use the digital product, are becoming less and less. At the same time, highly competitive industries, changing requirements and the demand for efficient and constant delivery of new software products, has led to an astonishing success of agile development methodologies. It is often claimed that they work well together with customer-centered approaches such as Design Thinking. In today's software development reality, however, this is still far from being an established routine. In this article, the *Human-Centered Agile Workflow (HCAW)* is introduced as process model for true integration of customer-centered conception and agile development.

Keywords: Software development process · Agile · Design Thinking · Human-Centered Design · User experience · Agile UX · Lean UX · Interdisciplinary

1 Introduction

Software development projects today usually involve managing distributed teams of various sizes in changing environments [1] whether it is outsourcing development tasks, near- and offshoring teams or enabling home office or other remote working relationships. The established *agile* processes used for software development are fit to meet changes in requirements, resources or other factors. Most *agile* software development methodologies are well-equipped to support changing and distributed development teams, however business, design or concept activities are mostly not seen as integral parts of these methodologies. However, "the companies that succeed will be those that can develop high-quality software faster, more reliably and more cost-effectively than their competitors" [2].

Customer-centered approaches for concept and design promise to achieve exactly that: High customer acceptance rates with the customer as the ultimate judge of the perceived quality of products [3]. The best-established approaches describe necessary activities, methods, deliverables and decision points in high detail. Unfortunately, these approaches hardly provide information about an integration of themselves into actual product or software creation and development models. In recent studies, practitioners rated the influence of such approaches on their *agile* projects just 4 out of 7 (with 1 meaning 'low influence' and 7 'great influence') [4].

A combination that brings together the best of two worlds into a single workflow and works in practice is hard to find.

2 Objective

The objective is to define an *agile* collaborative conception and development process model that fulfills the following requirements:

- Follow the principles of *Human-Centered Design* [5] to full extent
- Complete integration of business, concept and development activities
- Practical applicability to interdisciplinary remote teams

Follow the principles of Human-Centered Design to full extent. A key part of the objective is to fulfill customer-centricity. The well-established standard of *Human-Centered Design* defines a set of principles that are essential when following this approach. These principles are defined as follows: "a) The design is based upon an explicit understanding of users, tasks and environments. b) Users are involved throughout design and development. c) The design is driven and refined by user-centered evaluation. d) The process is iterative. e) The design addresses the whole user experience. f) The design team includes multidisciplinary skills and perspectives." [5]

Complete integration of business, concept and development activities. Another core part is to integrate all processes into one single workflow that is shared and known to each stakeholder involved. Specifically, the following activities need to be considered for combination: Research, ideation, prototyping, evaluation, specification, business, development and quality assurance.

Practical applicability to interdisciplinary remote teams. The collaborative process model is not supposed to be abstract in nature but needs to prove itself in practice. In addition, teams consisting of members from various disciplines and backgrounds need to be addressed. Regarding a remote set-up, co-location is often suggested as the teams' ideal working structure [6]. In contrast, according to the 2016 *State of Agile Report*, more than 82% of projects "had at least some distributed [*agile*] teams, […] up from 35% just three years earlier" [7], making remote working more necessary than ever.

In the following, existing customer-centered conception approaches and *agile* software development methodologies are described.

3 Process and Collaboration Models

3.1 Customer-Centered Conception Approaches

Amongst various other customer-, usage- or user-centered conception approaches, two well-established approaches exist: *Human-Centered Design* (HCD) and *Design Thinking* (DT).

Human-Centered Design. *Human-Centered Design* is “an approach to interactive systems development that aims to make systems usable and useful by focusing on the users, their needs and requirements, and by applying human factors/ergonomics, and usability knowledge and techniques” [5].

The approach of *Human-Centered Design* is split into various activities: Plan the *Human-Centered Design* process, understand and specify the context of use, specify the user requirements, produce design solutions to meet user requirements and evaluate the designs against requirements. After passing through several iterations, the design solution aims to meet the user requirements.

Design Thinking. For *Design Thinking*, there is not one single definition. According to Tim Brown, it is “a methodology that imbues the full spectrum of innovation activities with a human-centered design ethos” [8]. David Kelley puts it as “a method for how to come up with […] breakthrough ideas that are new to the world, especially with respect to complex projects, complex problems” [9]. It is seen as an alternative, more open way of getting to new solutions. The *Design Thinking* process consists of a set of stages which differ in definition from institution to institution. One of the most established definitions comes from the Hasso Plattner Institute: “Understand, Observe, Point-of-view, Ideate, Prototype, Test” [10]. A later version of the process is described by *Nielsen Norman Group* as “Understand, Define, Ideate, Prototype, Test [and] Implement” [11]. SAP attempts to summarize the various versions and comes up with the following common activities: “Understand the problem […], observe users ([…] visit them in their (work) environment, observe physical spaces and places), interpret the results ([…] empirical findings), generate ideas ([…] engage in brainstorming sessions to generate as many ideas as possible […]), prototype ([…] build prototypes and share them with other people […], test, implement [and] improve ([…] refine the design)” [12]. In all cases, revisions are planned for prototyping and evaluation/test activities and, if necessary, also for previous activities – in a similar way to that of *Human-Centered Design*.

Other conception approaches. Virtually all serious conception approaches are built upon and/or resemble one of the two above mentioned approaches to a high extent. For example, the *Design Sprint Process* as used by Google is based on *Design Thinking*. Its five phases are “Unpack, Sketch, Decide, Prototype and Test” [13]. The peculiarity of this approach is that there is a clear recommendation for the duration of phases and the iterations *(Sprints)*: A phase should last approximately one day, the whole *Sprint* respectively takes not more than one week.

Discussion. Essentially, *Human-Centered Design* and *Design Thinking* follow a similar activity procedure with the following structural deviations: HCD’s research activity combines the first two DT’s activities “Understand” and “Observe” into one, leaving it open to the project context to choose a certain type of method. At DT, this method is clearly set: Observation as a method of validating assumptions and gaining further insights. The “Point-of-view” activity is a less bureaucratic way of defining a solution’s requirements than the specification activity at HCD. This latter approach lacks “Ideation” as a separate activity, a fact that stresses the main difference between these two approaches: DT’s aspiration is to create concepts for innovation while the aim of HCD is to make systems more usable and useful. The remaining activities (“Prototype” and “Evaluate”/“Test”) are virtually identical procedures.

Neither of them, however, offers procedural standards. Of the above mentioned, the *Design Sprint Process* is the only one that makes an attempt in that direction. By putting this process into practice, its applicability for a human-centered *agile* development gets challenged: The advantage of the *Design Sprint Process* is the standardization with regard to timing of activities and the duration of *Sprints*, resulting in very fast design delivery. However, each design *Sprint* is supposed to include all *Design Thinking* stages, time-boxing potentially extensive research or evaluation activities in extremely short frames. Although this process might well be great for initial brainstorming and sacrificial prototype creation, it does not fit the requirements for an *agile* customer-centered conception and development process model.

3.2 Agile Software Development Methodologies

In general, the *agile* way of software development can be described as governed by “principles […] under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams.” [14]. The *Manifesto for Agile Software Development* consists of four core values: “a) Individuals and interactions over processes and tools. b) Working software over comprehensive documentation. c) Customer collaboration over contract negotiation. d) Responding to change over following a plan.” [6] “*Agile* refocuses software development on value. It seeks to deliver working software to customers quickly and to adjust regularly to new learning along the way.” [15] Amongst other *agile* methodologies such as *Extreme Programming* (XP), *XP Hybrid* approaches, *Scrumban* or *Kanban* [7], the most popular *agile* methodology *Scrum* is described in the following. This is followed by the more recent approaches *Agile UX* and *Lean UX*, which aim to combine *agile* development with a mindset familiar from the field of *user experience* (UX).

Scrum. The *agile* approach *Scrum* is about incremental software development. One of *Scrum*’s core ideas is the iterative nature of its development activities, which are supposed to take place in a defined time span: A single iteration (*Sprint*) is performed in one to four weeks. In accordance with the *Agile Manifesto*, *Scrum* relies on self-organizing teams, which are fully equipped to plan, execute and validate their deliverables. Each *Sprint* regularly concerns planning its content, reviewing the outcome and discussing and estimating requirements (*user stories*) [16]. Delivering business value in each *Sprint* is key. However, customer-centricity is not an explicit requirement when using *Scrum*.

Agile UX. Countless approaches exist that combine *agile* development processes with the user experience mindset and its activities. Jongerius’ “principles of *agile* UX & development” for example, are made up of the following: “End users first. Freedom vs. commitment. Eliminate waste. Self-propelled team. Timebox everything. Shippable product.” [17]. These principles, amongst various others based on the *Agile Manifesto*, do conform with the thought model of UX disciplines [18]. The *Agile UX* approach is an enhancement of *agile* software development methodology with principles and techniques from the field of UX design. Its aim is to bring developers and designers together in an *agile* product development process.

Lean UX. *Lean UX* is based on *Lean Startup* [19], *Design Thinking* and *Agile Development*. "*Lean UX* uses these foundations to break the stalemate between the speed of *agile* and the need for design in the product-development lifecycle" [15]. The approach describes methods, including their actual application, especially for unstable startup environments. It aims to unite product development and business through iterations and constant measurement. Development teams are "using these [lean] cycles as a competitive advantage – releasing early and often, gaining market feedback, and iterating based on what they learn – and (perhaps inadvertently) raising customer expectations in terms of quality and response times" [15].

Discussion. *Agile Development* addresses the activities which are necessary to iteratively create business value through cross-functional teams. In addition, for *Agile UX* and *Lean UX*, Gothelf names four core practices with which to work with: Firstly, "working in short cycles", which means validating each instance of success or failure and taking appropriate action. Secondly, "hold[ing] regular retrospectives" at the end of a *Sprint* and adjusting the working model accordingly. Thirdly, putting "the customer at the center of everything", mainly with regard to team discussions and priority setting. And lastly, to "go and see", addressing the team's constant need for, ideally face-to-face, communication [20]. The aim at the core of these methodologies is clear and the UX aspect of these lean practices is fully covered by the principles of *Human-Centered Design*.

3.3 Conclusion

The customer-centered conception approaches described above are useful for setting out activities and frame methods in order to create (innovative) design solutions. At the same time, they lack a comprehensive dependency- and time-based process model, integrating development, quality assurance and release cycles.

On the other hand, the younger *agile* software development methodologies described above list principles and values in accordance with *Human-Centered Design*. Recent studies state that UX activities become more and more integral parts in *agile* environments [4], however, there is no procedural standard existing. None of the above satisfactorily solves the problem of how to integrate the actual customer-centered tasks in practice.

4 Human-Centered Agile Workflow

In order to attain the objective delineated above, and to fulfill its requirements, an integrated process model needs to be defined that combines collaborative *agile* software development with customer-centric conception. This process model is introduced as *Human-Centered Agile Workflow* (HCAW).

Baseline. HCAW combines collaborative *agile* software development with customer-centered conception. Its software development methodology follows the established *Scrum* approach which in turn upholds the core values of the *Agile Manifesto*. The conception approach follows the principles of *Human-Centered Design*. With regard to its practical application, there are no restrictions about the team members' professional discipline, their location, the size of teams or the scope of projects.

Structure. In order to integrate conception activities into *agile* software development, the *Scrum* process is enhanced by iterations of two sizes and speeds: *Cycles* and *Sprints*. *Sprints* are set as the incremental unit as defined in *Scrum*. *Cycles* are introduced, framing multiple *Sprints* and including a prior conception phase called *Sprint 0*, referring to *agile* definitions. In this model, the *Sprint 0* is split into two phases: the conception of the current *Cycle* and the conception of the first *Sprint* in this *Cycle*. The *Cycle Conception* phase is specifically about research activities, the definition of the *Cycle*'s *Epics* [21] and the overall planning of the *Cycle*. This phase is followed by the *Sprint 1 & 2 Conception* phase, describing the concept for *Sprint 1* in detail and outlining a rough concept for *Sprint* 2. Furthermore, the development groundwork for the *Cycle* is getting done. See Fig. 1.

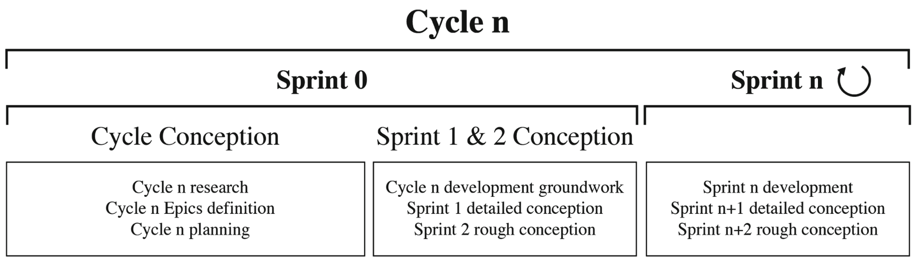
Cycle n
Sprint 0
Sprint n
Cycle Conception
Sprint 1 & 2 Conception
Cycle n research
Cycle n Epics definition
Cycle n planning
Cycle n development groundwork
Sprint 1 detailed conception
Sprint 2 rough conception
Sprint n development
Sprint n+1 detailed conception
Sprint n+2 rough conception

Fig. 1. The *Cycle* and *Sprint* hierarchy: each *Cycle* contains *Sprint 0*, and a number of *Sprints*.

Cycle-based concept groundwork. Following HCAW, each software project consists of at least one *Cycle*. If the project is of manageable scope, one *Cycle* may be sufficient. For larger projects, each *Cycle* may encompass a major release of the software. Each *Cycle* begins with a two-phased *Sprint 0*, followed by any number of *Sprints*.

The first phase of *Sprint 0* is about laying the concept groundwork for the complete *Cycle* and is therefore called *Cycle Conception*. The starting point is the business (usually represented through management resources) providing a problem scenario and business aims to the research team in a meeting called *Cycle Briefing*. Based on that briefing, the researchers start to plan their activities by choosing suitable methods (compare *Human-Centered Design*), conduct the research and complete their contribution by publishing an *Insight Report*. This report is presented as part of a *Co-Creation Workshop*, which is hosted by the ideation team, having invited colleagues from research and prototyping teams and potentially others. The prototyping team bases their work on the workshop's outcome and presents their result to the evaluation team.

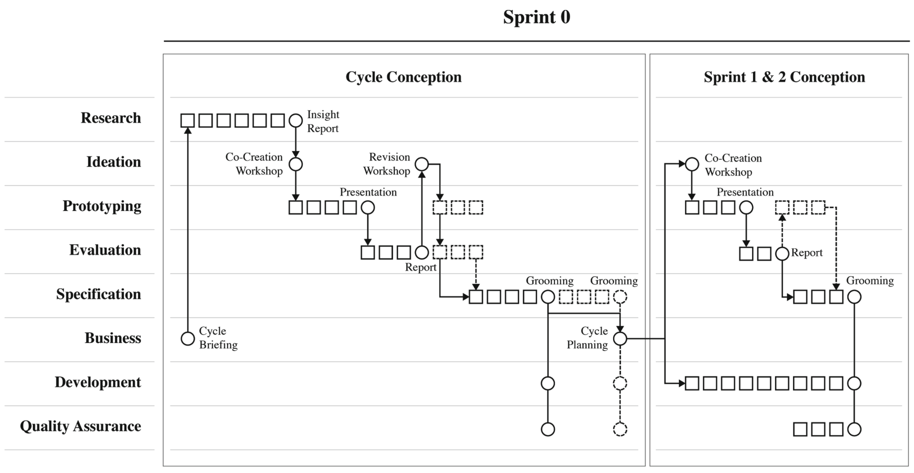

Fig. 2. The process model of *Sprint 0* including *Cycle Conception* and *Sprint 1 & 2 Conception*.

The evaluation team decides on a suitable method, conducts the evaluation and reports their findings at the *Revision Workshop*, hosted by the ideation team. Depending on the evaluation report and the workshop's outcome, the prototyping and evaluation teams might be involved in re-working and evaluating the prototype. Having done so, they hand over the validated prototype to the specification team. Once *Epics* have been created, these receive a rough assessment (estimation) by the development team in a *Grooming* meeting together with the quality assurance team. If need be, the specification team re-works the *Epics* and may conduct a second *Grooming*. The business team then holds a meeting called *Cycle Planning*, where the *Epics* are allocated to a series of *Sprints*. With that, the *Cycle Conception* phase is completed.

The second phase of *Sprint 0* is the *Sprint 1 & 2 Conception*. So far, *Epics* have been described, estimated and planned for, a rough *Sprint* structure is given. With the help of that, the initial activity of setting the technical groundwork for the *Cycle* commences and lasts until the end of *Sprint 0*. In addition, the assigned *Epics* for *Sprint 1 & 2* are taken as input for another initial activity in this phase: the ideation team's *Co-Creation Workshop*. After having created a prototype for *Sprint 1* & optionally *Sprint 2*, the team presents them to the evaluation team, which performs a prototype evaluation and reports back on it. If necessary, the prototype is amended and the specification team writes *user stories* at least for *Sprint 1*. The *Sprint 1 user stories* are estimated in a *Grooming* session, together with development and quality assurance teams. The *Grooming* marks the end of the *Sprint 1 & 2 Conception*.

Iterative conception and implementation. After *Sprint 0*, the iterative conception and implementation starts, which is basically a concatenation of *Sprints*. Each *Sprint* follows the same pattern and has two major objectives: a) To deliver the *Sprint*'s scope to create business value (implementation). b) To deliver the validated concept for the upcoming *Sprint* (conception).

The implementation part of the *Sprint* commences with the *Sprint Planning* meeting, hosted by the business representatives together with the specification, development and quality assurance teams. The aim of the meeting is to plan previously estimated *user stories* to be part of this *Sprint*. The *Sprint* is officially set in motion once the *Sprint* scope has been agreed by all. The meeting is followed by a *Technical Planning* session of the development team and an activity by the quality assurance team to write *test cases*. At the end of the *Sprint*, *Checks* take place between the development and specification teams to review the delivered *user stories* from a conception point of view. The *Sprint* closes with a *Review* session, where the development team presents the delivery to the business, specification and quality assurance teams.

The conception part of the *Sprint* does not deal with the content of the current *Sprint* (n) but defines the content for the upcoming *Sprint* (n + 1) in detail (*user stories*) and prepares the content of the *Sprint* after that (n + 2) in a rough manner (initial prototype). The conception part starts again with a *Co-Creation Workshop* from the ideation team, focusing on the content of the upcoming two *Sprints* (n + 1 & n + 2). Based on the workshop’s outcome, the prototype team creates prototypes. The prototype for the upcoming *Sprint* (n + 1) is presented to the evaluation team, which, in turn, tests the prototype and reports back to the prototype and specification teams. If need be, the prototype is re-worked. The specification team creates the *user stories*, which are then estimated in a *Grooming* session amongst the development, specification and quality assurance teams at the end of the *Sprint* (after the *Sprint Planning* meeting of the implementation part).

At the very end of the *Sprint*, a *Retrospective* meeting is conducted amongst the same teams to reflect on the *Sprint’s* quality and performance (Fig. 3).

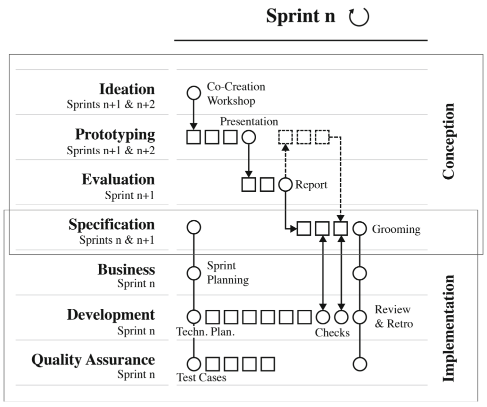

Fig. 3. The process model of each iterative *Sprint* with implementation and conception parts.

Scheduling. In order to deliver a high-quality human-centered development outcome, it is crucial to set the time span for the activities at the outset. To achieve a true integration between development and conception, relative time allocation and dependencies should be set as shown in Fig. 1 and 2: here, squares and circles stand for relative time units. For example, if a square or circle represents one workday, a *Sprint* would last ten days, allowing eight days for the actual development activities including two days with informal *Checks* and two additional days with formal meetings. Table 1 shows the time allocated for activities for one- to four-week *Sprints*. When more time is defined for a *Sprint*, the *Sprint 0* with its *Cycle Conception* and *Sprint 1 & 2 Conception* phases are prolonged proportionately. In order to achieve a decent balance between the effort invested and the quality of the result, the two-week *Sprint* rhythm is recommended. However, there are exceptions for projects which are follow-up projects of previous ones or which are generally very small. In such cases, a one-week *Sprint* rhythm may be considered. When, on the other hand, if a project's aim is extremely ambitious, broad or unclear, longer conception phases and more development time to deliver actual business value are acceptable. In such cases a three- or even four week *Sprint* rhythm may be chosen.

Table 1. Time allocation (in days) for activities for one- to four-week *Sprints*.

	Sprint n	Sprint 0	Cycle conception	Sprint 1 & 2 conception
One-week Sprint	5	17.5	12.5	5
Two-week Sprint	**10**	**35**	**25**	**10**
Three-week Sprint	15	52.5	37.5	15
Four-week Sprint	20	70	50	20

Practical application. The HCAW model described above is currently in use in three service design and implementation projects at LINC[1]. All three projects deal with similar challenges:

- All aim to deliver customer-centered high-quality business value.
- Environmental factors are changing constantly, which is why an *agile* development methodology needs to be followed thoroughly and consistently.
- Some parts of the international projects' teams are co-located, but the majorities of team members work from remote locations.

A field study of the model's application in three projects is currently ongoing. Although during the projects' current execution status the model's applicability in the respective environments and the acceptance of the workflow by the teams involved can be observed, measurable practical implications and results are not available at the present moment.

5 Further Research

The ongoing field study with application in initial projects is expected to prove the practical use of the *Human-Centered Agile Workflow*. On top of that, an extended field study including a higher number of projects is planned. Further research needs to be

[1] LINC Interactionarchitects GmbH is a service design company based in Munich, Germany.

conducted with the model in a structured fashion addressing various fields, such as completely remote projects or projects with large teams (> 100 executing members).

References

1. Chang, C., et al.: Time-line based model for software project scheduling with genetic algorithms. Inf. softw. Technol. **50**, 1142–1154 (2008). Elsevier, USA
2. Coverity: The Software Development Challenge. Coverity, San Francisco (2006)
3. Brown, T.: Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. HarperCollins, New York (2009)
4. Nielsen Norman Group: Effective Agile UX Product Development. (2017)
5. International Organization for Standardization: DIN EN ISO 9241-210 Human-centred design for interactive systems. (2010)
6. Beck, K., et al.: Manifesto for Agile Software Development. Addison-Wesley, Reading (2001)
7. VersionOne: The State of Agile Report. (2016)
8. Brown, T.: Design thinking. In: Harvard Business Review (June 2008), pp. 87–92. Harvard Business Publishing (2008)
9. Camacho, M.: David Kelley: From design to design thinking at stanford and IDEO. In: She Ji: The Journal of Design, Economics, and Innovation, vol. 2, issue 1, pp. 88–101. Tongji University Press, Shanghai (2016)
10. Plattner, H. et al.: , Design Thinking. In: mi-wirtschaftsbuch. Munich (2009)
11. Gibbons, S.: Design Thinking 101. https://www.nngroup.com/articles/design-thinking/ Retrieved 6 March 2017
12. Waloszek, G.: Introduction to Design Thinking. https://experience.sap.com/skillup/introduction-to-design-thinking/ Retrieved 6 March 2017
13. Knapp, J., et al.: Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days. Simon and Schuster, New York (2016)
14. Collier, K.: Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing. Addison-Wesley, USA (2011)
15. Gothelf, J., et al.: Lean UX: Designing Great Products with Agile Teams. O’Reilly Media, Sebastopol (2016)
16. Rubin, K.: Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley, USA (2012)
17. Jongerius, P., et al.: Get Agile!: Scrum for UX. Design and Development. BIS Publishers, The Netherlands (2013)
18. Brown, D.: Agile User Experience Design: A Practitioner’s Guide to Making It Work. Morgan Kaufmann, San Francisco (2012)
19. Ries, E.: The lean startup: How today’s entrepreneurs use continuous innovation to create radically successful businesses. Random House, New York (2011)
20. Gothelf, J.: Lean vs Agile vs Design Thinking: What you really need to know to build high-performing digital product teams. Gothelf Corp, Glen Rock (2017)
21. Jarrell, J.: Stories versus Themes versus Epics. https://www.scrumalliance.org/community/articles/2014/March/stories-versus-themes-versus-epics Retrieved 6 March 2017