Intertwining Lean and Design Thinking: Software Product Development from Empathy to Shipment

Tobias Hildenbrand and Johannes Meyer

Abstract

A few years back, everybody in the industry seemed to be talking about how "Lean Thinking" can improve software development. Best practices emerged, books were written and Lean Thinking, associated with agile process frameworks became somewhat of a standard work culture in software development. Now that many people are actually practicing lean and agile development, they have started to wonder about something called "Design Thinking". When we coach development teams in a large software company, we're frequently being asked whether Design Thinking is the next big thing substituting lean software development. After having guided several teams through successful projects, our verdict is: Design Thinking is not Lean's heir; in fact the two schools can be intertwined in many ways and complement each other very well. As we will elaborate in this case study, they share some integral core values and goals, and can therefore be applied in the same project without corrupting each other. As a proof of concept, we combined and utilized the underlying set of methods in order to explore a yet relatively unknown and unusual domain for SAP business applications: Software for professional sailors and their coaches that helps them to optimize their training experience and competitive performance.

1 Introduction: Related Work and Research Objective

Before we get into the actual sailing case study, it is important to note that SAP, world market leader in business software for large enterprises, has started a broad initiative to educate teams in Design Thinking, not only in development but across

T. Hildenbrand (✉)
SAP AG, Walldorf, Germany
e-mail: tobias.hildenbrand@sap.com

J. Meyer
Hasso-Plattner-Institut Academy GmbH, Potsdam, Germany
e-mail: johannes.meyer@hpi-academy.de

all business areas. The work culture we describe is not restricted to this project, but is currently being broadly adopted in the company. Why would the teams that already practice lean and agile development for several years (Schnitter and Mackert 2011) need additional values, practices and another set of tools to do their job?

Let us be frank here from the beginning: developing business software is becoming more and more challenging. Together with the transforming requirements of business customers in different industries, products have to be in a constant cycle of innovation and adapt to ever new environments (Smith and Reinertsen 1992). Operating in such an environment, a steady flow of good ideas is the only justification for a business software company to flourish (Reinertsen 1997, 2009). Such a company therefore needs a structured framework not only on *how* it turns ideas into sellable products, but also on *how to come up with* those ideas in the first place. Design Thinking is such a framework, intended to increase the likelihood and reliability of innovations developed in teams (Brown 2009; Martin 2009).

Lean Thinking, on the other hand, has proven to make teams and organizations more efficient and transparent for almost 20 years, if you take *Scrum* as an exemplary process framework (Schwaber 1995; Sutherland and Schwaber 2011). However, Scrum assumes that teams already start with a “product vision” and a “product backlog” without a clear picture as to where that vision will come from (Highsmith 2009; Pichler 2010). On the enterprise level, Lean Thinking tells us to “focus on customer value” (Womack and Jones 1990, 2003), but besides a basic definition of “value”, i.e. “what the customer is willing to pay for”, Lean Thinking does not provide according guiding principles on how to find out what is actually valuable to the customer.

Besides pressure for innovation (Martin 2009), there are other good reasons why both Lean and Design Thinking make particular sense together and have their respective niche in the business software domain: *First of all*, business software projects for large enterprises can get rather bulky, delivering complex products with the help of many different teams (Larman and Vodde 2008). Without frameworks like Scrum and the ability to scale beyond single teams, it’s almost certain that resources will be wasted, especially because software is not as transparent as other products (Leffingwell 2011). *Moreover*, developers in business software companies are often not actual users of their own products. Instead, they are expected to deliver something that their – often very IT-skilled – customers would not be able to build at the same price; however, most often without being experts for their customers’ respective business domain and business processes. Hence, *empathy* is needed to take the famous walk in the customers’ shoes and discover potentials for innovative applications. Once these opportunities have been discovered, it’s not enough if they are just desirable to the customer (Pichler 2010): To be reasonable as a product, they also have to be viable in terms of business value, i.e. generate revenue for the software company, and feasible to be developed in the first place. Hence, an innovative and successful software product has to be desirable, viable, and feasible at the same time (Meinel and Leifer 2011). Design Thinking has successfully proven to help teams and organizations balance these "three spaces of innovation" for products, services, and customer experiences (Brown 2009).

It is therefore no surprise that both thinking schools share a fundamental set of core values and commonalities: *First*, both recommend forming and empowering interdisciplinary or so-called "cross-functional" teams (Schwaber 1995; Kelley 2008; Brown 2009; Blau et al. 2011b). This means that a team contains all skills required to address a certain market or customer need and control is decentralized as far as possible (Reinertsen 2009; Sutherland and Schwaber 2011). *Second*, both are about taking an economic perspective on product development, i.e. taking business value, viability, and revenue streams into account when managing the overall product portfolio and prioritizing requirements for particular products (Brown 2009; Reinertsen 2009). *Third*, the development process leverages on fast feedback cycles and gaining additional insights for further iterations. The principle of inspecting and adapting both product and process is inherent to lean and agile development (Reinertsen 2009; Sutherland and Schwaber 2011). In the same vein, Design Thinking suggests early, regular, and cheap prototyping to "deliver fast results and generate useful feedback" (Brown 2009, p. 87; Ries 2011).

Besides these and other inherent commonalities, another reason why Lean and Design Thinking don't collide is that they focus on different challenges and aspects in a development project lifecycle: While Lean Thinking and agile practices help organizations to *build and ship products right*, meaning e.g. in time and in quality, Design Thinking focuses on *building the right product* in the first place. Hence, it can help teams to understand the full context of a problem space from the perspectives of potential users and relevant stakeholders. Building on this understanding, teams can develop a product vision and derive requirements for what the product could actually do for the users within their respective context. Lean Thinking and agile methods, such as Scrum, Extreme Programming (XP, Beck 1999; Beck et al. 2001; Hildenbrand et al. 2008), and Clean Code (Martin 2008), for instance, provide the process framework for development organizations as well as concrete engineering principles to efficiently bring the product vision to life as shippable software (Chow and Cao 2008; cp. Fig. 1).

It is thus understandable that business software companies are particularly interested in introducing practices that ensure continuous and reliable innovation through empathy and streamline development processes with minimal waste. Nevertheless we are at the start of this journey and Design Thinking is just now spreading from consultancies into in-house product development teams, just like Lean Thinking spread from manufacturing into the software industry via agile practices a few years ago (Poppendieck 2002; Poppendieck and Poppendieck 2003; Larman and Vodde 2008).

Research Objective: Based on the commonalities and possibly conflicting areas inherent to software development as outlined above, our case study intends to shed light on how to leverage Lean and Design Thinking in order to *build the right software right* in one practical project setting. In particular, we investigate the underlying research question of *how to come up with an innovative product vision and derive requirements in a yet unknown domain.*

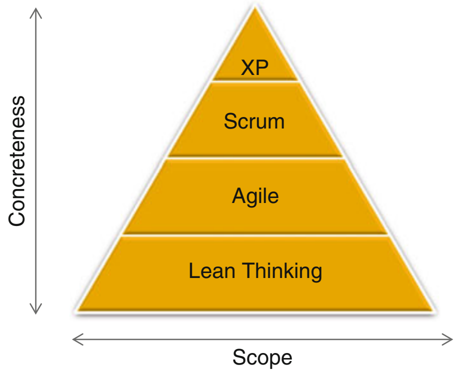


Fig. 1 From lean thinking to concrete software engineering techniques

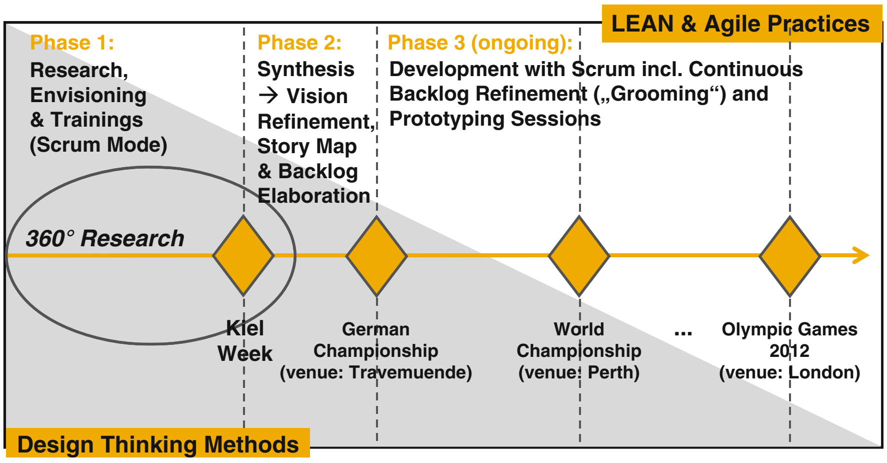

LEAN & Agile Practices
Phase 1:
Research, Envisioning & Trainings (Scrum Mode)
Phase 2:
Synthesis → Vision Refinement, Story Map & Backlog Elaboration
Phase 3 (ongoing):
Development with Scrum incl. Continuous Backlog Refinement („Grooming") and Prototyping Sessions
360° Research
Kiel Week
German Championship (venue: Travemuende)
World Championship (venue: Perth)
...
Olympic Games 2012 (venue: London)
Design Thinking Methods

Fig. 2 Integrated approach for sailing team Germany project at SAP

Methodologically, this work follows a design science approach (Hevner et al. 2004). In doing so, our main "artifact" is the team-based process from the initial challenge or "project brief" (cf. Brown 2009) to a first working and potentially shippable software increment that users can apply and assess in their environment (Schwaber 1995; see also approach in Fig. 2). Hence, we later evaluate the *usefulness* of our process by customer adoption, development team satisfaction, and other types of feedback. Our case study therefore serves as observational design evaluation for the integrated approach described in Sect. 3.1 and Fig. 2 in particular (Hevner et al. 2004, p. 86). The constituents of this process are based on a combination of Lean and Design Thinking principles and practices. Moreover, we also evaluate the process from an internal team perspective, i.e. team learning and satisfaction as well as the "fun factor" within our development team. Our evaluation approach is based on a *single case study* setting within SAP as the organizational context (Eisenhardt 1989; Yin 2008, 2011) and we derive an initial set of conclusions under the particular contingencies of this case as a first step towards an integrated theory of efficient and effective software product development with a combination of Lean and Design Thinking principles and practices (Van de Ven and Drazin 1985).

The rest of this book chapter is structured as follows: the following Sect. 2 presents a brief overview of the case study setting and the respective customer, i.e. SAP's Sailing Program for Audi Sailing Team Germany. Section 3 then describes the process as set of practices that we designed, adapted, and intertwined for this particular team and setting. Section 4 will evaluate and discuss our methodological approach based on observations from the case study, while Sect. 5 will draw conclusions for software product development and provide an outlook on future development projects applying our approach and upcoming research programs in our pipeline.

2 SAP'S Sailing Program and Sailing Team Germany

As of 2011, SAP is sponsoring *Audi Sailing Team Germany* (STG), a recent initiative geared at promoting athletic sailing and building a common organizational structure between Germany's many diverse and clustered sailing clubs. As Marcus Baur, former Olympic sailor and co-head of STG, points out in an interview: "Sailing Team Germany has come together to achieve the goal of making Germany one of the best sailing nations in the world again" (STG 2011).

When analyzing the domain, many people underestimate that sailing is a highly strategic sport. In the majority of boat classes, it is not so much physical fitness, but experienced judgment of weather, water conditions and venues that make teams win. It is finding the adequate trim settings, being on the right side of the venue and anticipating events before they actually occur.

SAP as a software vendor chose to support STG's ambitious goal with software. Besides the classical support of a sponsor, SAP is particularly interested in showcasing its technology and ways of working by building software that provides true value to the sport. In an initial analysis of what SAP technology could do for the sailing domain, one of several emerging challenges was *How might we improve knowledge transfer among sailors and coaches so they can benefit from each other's experience and improve performance?* According to Marcus Baur, a fast knowledge transfer from experienced to younger sailors would offer a powerful and sustainable competitive advantage for the German sailing team towards the Olympic Games in London in 2012 and beyond (STG 2011).

You can call it lucky or not that the team that was selected to work on this challenge had not been in contact with the sailing domain before. The team included five developers, one Scrum Master, one Product Owner, as well as several part-time experts e.g. for user interface design, mobile applications, and Java platform services. From the beginning it was clear that there was a lot of learning to do before the team would get to a solid idea of what their product should do and what their impact on the users' daily experience should be.

On the other hand, starting a project without any assumptions gave us as coaches the chance to also intertwine our respective experience from both Lean and Design Thinking. This way, we developed a custom-made, integrated approach that was supposed to help the team get up to speed with sailing and build shippable software eventually. To be able to do so, they first needed to understand the sailors' problems, come up with an innovative product vision and derive high-level requirements for this yet unknown domain.

3 Intertwining Lean and Design Thinking

This section presents the integrated process and its application as part of the case study. In the overview, the different phases and objectives of the project are described along with the respective methods, practices, and techniques from Lean and Design Thinking.

3.1 Our Overall Approach

Lean software product development (Larman and Vodde 2008; Reinertsen 2009; Larman and Vodde 2010; Leffingwell 2011) has been adopted as a standard approach in SAP development for almost 3 years starting in 2009 (Schnitter and Mackert 2011; Blau et al. 2011a).

SAP's change template for this transition included (1) Lean Thinking principles as a basis (Womack and Jones 2003), (2) agile principles from the Agile Manifesto (Beck et al. 2001), (3) standard agile process frameworks such as Scrum (Sutherland and Schwaber 2011) and Kanban (Anderson 2010), as well as (4) well-established agile engineering techniques, e.g. from XP and Clean Code (Beck 1999; Martin 2009). The latter is seen as critical success factor in most Agile projects and missing link to the actual software "shop floor", i.e. the development teams, in large-scale Lean implementations (Chow and Cao 2008).

Figure 1 shows how Lean, agile principles, Scrum as a process framework, and agile engineering techniques such as XP can build upon and complement each other within SAP's overall "Lean Development Model" to enable successful software projects (Chow and Cao 2008):

Taken together, this new approach SAP is taking can be summarized for particular lines of business or bigger solutions as follows: instead of a large group spending a long time building a big piece of software, many smaller teams spend a short time (2–4 week iterations) building small pieces of software, while integrating regularly to see the whole (cp. also Kniberg 2007).

SAP's lean development approach, as described above, however, does not directly tell you how to come up with an innovative product vision and a good product backlog that includes the relevant requirements as "user stories"

Table 1 Overlapping terminology in lean and design thinking

Lean thinking and agile practices	Design thinking
Requirement	User need
Persona	Persona
Usage sequence	As-is scenario
Product vision	Solution
Epic (i.e. a coarse-grained user story)	To-be scenario
Product backlog item, e.g. as user story	Solution idea, prototype, implementation

(Leffingwell 2011), ordered according to customer value or other criteria (Sutherland and Schwaber 2011).

When we started to design a suitable approach to achieve this for our sailing project, it turned out to be necessary to synchronize on some of the terminology that Lean and Design Thinking use. As a result of this discussion, we came across overlaps for the following descriptions:

Table 1 contrasts selected terminology from our perspective: what Lean thinking calls requirements (Reinertsen 2009; Leffingwell 2011), may be what Design Thinking understands as "needs" coming out of a research phase. In both worlds, a persona embodies an archetypical user or user category (Patton 2008; Brown 2009). However, agile methods for requirements engineering (Hildenbrand et al. 2008) such as story mapping usually do not accurately distinguish as-is and to-be descriptions while Design Thinking clearly separates the problem space (as-is scenario) and the solution space (to-be scenario, see Meinel and Leifer 2011, for instance).

Moreover, lean and agile practices such as Scrum suggest decomposing the "solution idea" or "prototype" (Brown 2009) into a "product backlog" consisting of "backlog items" (Sutherland and Schwaber 2011) to answer the question what it actually takes to "bring the product to life" (Pichler 2010). Backlog items can be described as requirements from a user perspective in the form of so-called "user stories" (Cohn 2004). Larger, coarse-grained user stories are often called "epics" (Leffingwell 2011) and correspond to the to-be scenarios used in Design Thinking projects, e.g. prototyped with storyboards (Kelley 2001; Brown 2009).

After synchronizing on language and terminology, we derived our development approach based on the goals and boundary conditions given in the STG project. For instance, the team already decided to visit Kiel Week in June 2011 to meet the sailors and a preliminary shipment of useable software was planned for the German Championships in Travemuende in September 2011.

Figure 2 shows how we planned the transition from Design Thinking-driven user research to lean and agile development practices in later phases of the project. Our intention was to allow for an initial phase in which the team would not "think code" yet but just concentrate on the user:

While this visualization may suggest that agile practices played no role at the beginning of the project that is not the case: In fact, before our first project milestone, Kiel Week in June 2011, the team also maintained a backlog of things to develop and prepare, e.g. interview guides, appointments, and other logistics, with the "sprint goal" of preparing the user- and venue research (Sutherland and Schwaber 2011). Despite not developing any software yet, we used a Scrum-like process to learn as a team, get the most important things done prior to Kiel and make efficient use of the time at the sailing event. With the input gathered during these preparations and at the actual event, we then wanted to agree on our product vision and derive the "real backlog" in order to start continuous development and backlog refinement activities (cp. Fig. 2). That is, later on we wanted to run the project in a rather "standard lean and agile mode", as it was familiar to the team from previous development projects at SAP.

However, we think the process shown in Fig. 2 is only one of many possible ways in which a combination and intertwining of Lean and Design Thinking practices can be realized. In our particular case, it helped us to manage the transition from zero sailing domain knowledge to actually building shippable software for professional sailors and their coaches.

As indicated in Fig. 2, we therefore define three major phases: *Phase 1* includes user research, envisioning, and training of the team. In *Phase 2*, our goal was to merge all the findings from Phase 1 and come out with a stable product vision and a first set of requirements as user stories in the team's product backlog. *Phase 3* would then cover the actual coding and continuous feedback loops with stakeholders.

3.2 User Research, Envisioning, and Trainings (Phase 1)

Since the development team, including the product owner, did not have much experience in the sailing domain, we guided them through research to build up empathy for the potential users and a basic understanding of the domain. Classical *Design Thinking* (Meinel and Leifer 2011) suggests "observation" as the first diverging phase in an innovation project. In our software development domain, this has been extended to "360° Research", referring to observation and interviews with users plus secondary sources like analysts and thought leaders, competitors, analogous and adjacent domains.

In the sailing case, this meant that the team actually talked to sailors and coaches in their natural environment and observed their current behaviors, scrutinized existing "tools" (both on paper and electronic ones) and analyzed their goals and feelings. Secondly, they looked at sailing competitions and similar domains such as show jumping, formula one, and gliding, to name just a few. Third, they informed themselves and plunged into the sailing domain by analyzing books and articles, as well as computer-based sailing simulators.

To get some real-world and direct user experience, the team went to Kiel Week, one of Germany's biggest annual sailing events. Within this week, 15 interviews with various sailors, coaches, and other experts were scheduled and conducted by sub-teams of two members of the team including the developers, product owner, Scrum master, etc. Moreover, we reserved time for the observation of race

Table 2 Overall workshop agenda for synthesis, vision, and backlog elaboration (phase 2)

Workshop Day 1	Workshop Day 2	Workshop Day 3
Synthesis: storytelling and clustering of observations	**Product vision**: statement	**Recap**: Lean/agile development, Scrum and Kanban
Point-of-view: personas for expected users	**Story map**: usage sequence, personas, epics (backbone)	**Overview** of concrete a*gile software engineering practices*
Ideation and p**rototyping**	**User story writing** in pairs	**Working model** and charter, **tool support** required
Vision: product box	**Story map review**, prioritization and "slicing" of map	**Review** of overall workshop results with Marcus (STG)

preparations and actual regattas at the venue site. Last but not least, a full immersion into the topic was achieved by actually sailing as a team in Kiel.

Researching openly, without a direct objective for the product design, was also lots of fun for the team and within an impressively short time, they grew both together as a team, and into experts in their new domain.

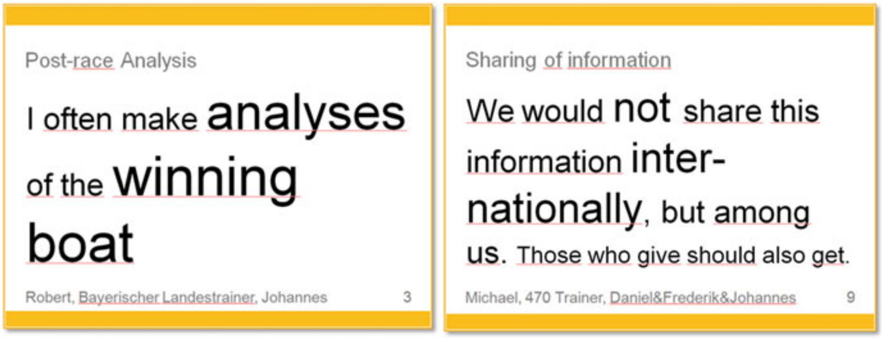
3.3 Synthesis, Vision, and Backlog Elaboration (Phase 2)

Our clothes still soaked with salt water, we returned from Kiel to SAP Walldorf and started synthesizing what we had learned so far as a team. As part of our overall process (see Fig. 2), us coaches suggested conducting a 3-day workshop to develop the product vision, derive a backlog and start development team work. Table 2. summarizes the overall workshop agenda:

Workshop Day 1 – Back in the office, we applied Design Thinking practices such as time-boxed synthesis of key statements by our potential users. Each of the interviews was reported by the respective interviewers from the team in a round of so-called "storytelling"; information about user needs, pain points, and other potential insights was put on **post-its** (see Fig. 3).

In our setup, **storytelling** allowed six minutes per research area and no discussions, with questions being parked on a designated parking lot flip chart. That way, each team member was able to absorb the insights from all other interviews, including the ones they had not attended themselves.

After collecting about 250 of these data points, we clustered them into topic groups. We selected "silent clustering" for efficiency reasons, i.e. the participants put their share of the data points on a pin board and placed related ones close to each other. The overall arrangement could be changed by everyone until the clusters converge. With the highlights of about 20 clusters in mind (the clusters we found included "audio/video support", "trimming", and "tracking the boats", for instance), we developed our first two **personas** as stereotypical users: Tina the professional sailor and Thomas, a coach. The persona descriptions included their age, profession, boat class, background, motivation, and pain points. Design Thinking recommends this synthesis to get from diverging into converging mode and make empathy possible through focused formats.


Post-race Analysis

I often make analyses of the winning boat

Robert, Bayerischer Landestrainer, Johannes 3

Sharing of information

We would not share this information inter-nationally, but among us. Those who give should also get.

Michael, 470 Trainer, Daniel&Frederik&Johannes 9

Fig. 3 Example observations from storytelling in synthesis phase of workshop day 1

With these “flesh-and-blood” users and lots of fresh impressions from Kiel in mind, we were ready to switch on the “solution engine” and finally start to think about solution ideas to address some of the users’ problems. Within three intense days in the field (2 days interviews, 1 day processing), the team had “build up a deep understanding of the sailing domain from a user perspective”, as they mentioned in one of the later retrospective meetings. Now they were able to make the right decisions on the product features most useful and desirable to their personas.

With a structured view on needs and pain points, the first ideas started to spark and the team went into their first ideation session where they developed rough drafts of screens and to-be usage scenarios (cp. Table 1) the user could possibly go through in the future.

Figure 4 depicts an example of how a sailor could use our software on a mobile tablet device to record and share their experience after sailing while receiving post-race physiotherapy:

This “educated brain dump” was also a perfect warm-up for developing the product vision in the form of a sneak preview software package (called a “vision box”, Highsmith 2009) with a product name, unique selling propositions, and some key features on it. We concluded day 1 with a first rough overall product vision in the form of the vision box and a brief vision statement:

> Sail Better provides easy access to training, trim, and venue data from various sources, allows you to log your own experience and learn from others – fostering collaboration between coaches and sailors to optimize your sailing performance

Workshop Day 2 – with our common product vision and personas already in place, we had an ideal starting point for developing our first product backlog. At this point some great ideas were on the table, but the team including product owner was not even close to having requirements as concrete backlog items to start development yet. Our agile tool of choice to get a full end-to-end picture of the users’ processes and possible backlog items was thus a **user story map** (Patton 2008, see outline and data model in Fig. 5; cp. also terminology in Table 1):

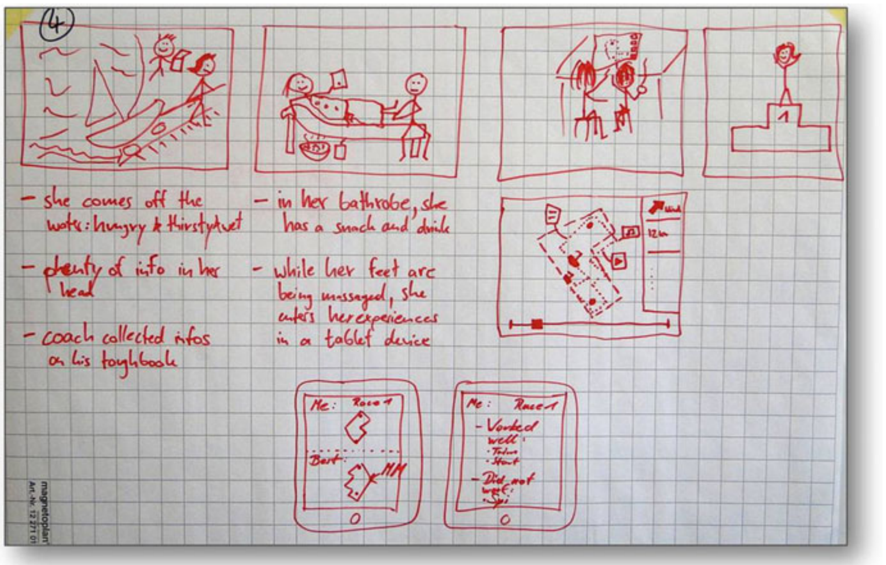

4
- she comes off the water: hungry & thirsty & wet
- plenty of info in her head
- coach collected infos on his toughbook
- in her bathrobe, she has a snack and drink
- while her feet are being massaged, she enters her experiences in a tablet device
Me: Race 1
Best:
Me: Race 1
- Worked well:
- Did not work:
1
magnetoplan

Fig. 4 Example of an early paper prototype for an experience with a mobile application

Fed by the ideas that had come up as prototypes in our day 1 ideation session and based on all the newly acquired knowledge about the daily lives of the user roles, it proved to be surprisingly easy for the team to first come up with an overarching usage sequence for sailors and coaches along the course of one season (i.e. one calendar year), and then fill this "backbone" with insight-based epics and user stories (Patton 2008; Leffingwell 2011). As the product owner put it:

> The team was so well warmed up, that we could write the user stories and immediately fill the story map as basis for our product backlog. This made my job a lot easier at this point in time.

Later, the map served as central reference for the product owner, customers and the team to see what the overall product is supposed to look like, what the current progress to plan is in terms of user stories implemented and remaining work as well as which ideas they want to implement next for which part of the user's daily conduct. Particular user stories are then pulled for sprint planning priority-wise and broken down into smaller stories to fit the sprints (Sutherland and Schwaber 2011; see also Sect. 3.4 and Table 3 for the team's concrete approach to working with user stories and done criteria along sprints).

Workshop Day 3 – on our last workshop day, we focused on *how* the team wants to work with the backlog. After having compiled and ordered a decent set of user stories, i.e. after having an actual backlog, we needed to discuss our **working model** as a team. To facilitate this, we presented and recapitulated existing lean and agile good practices to the team and decided afterwards which approach we start

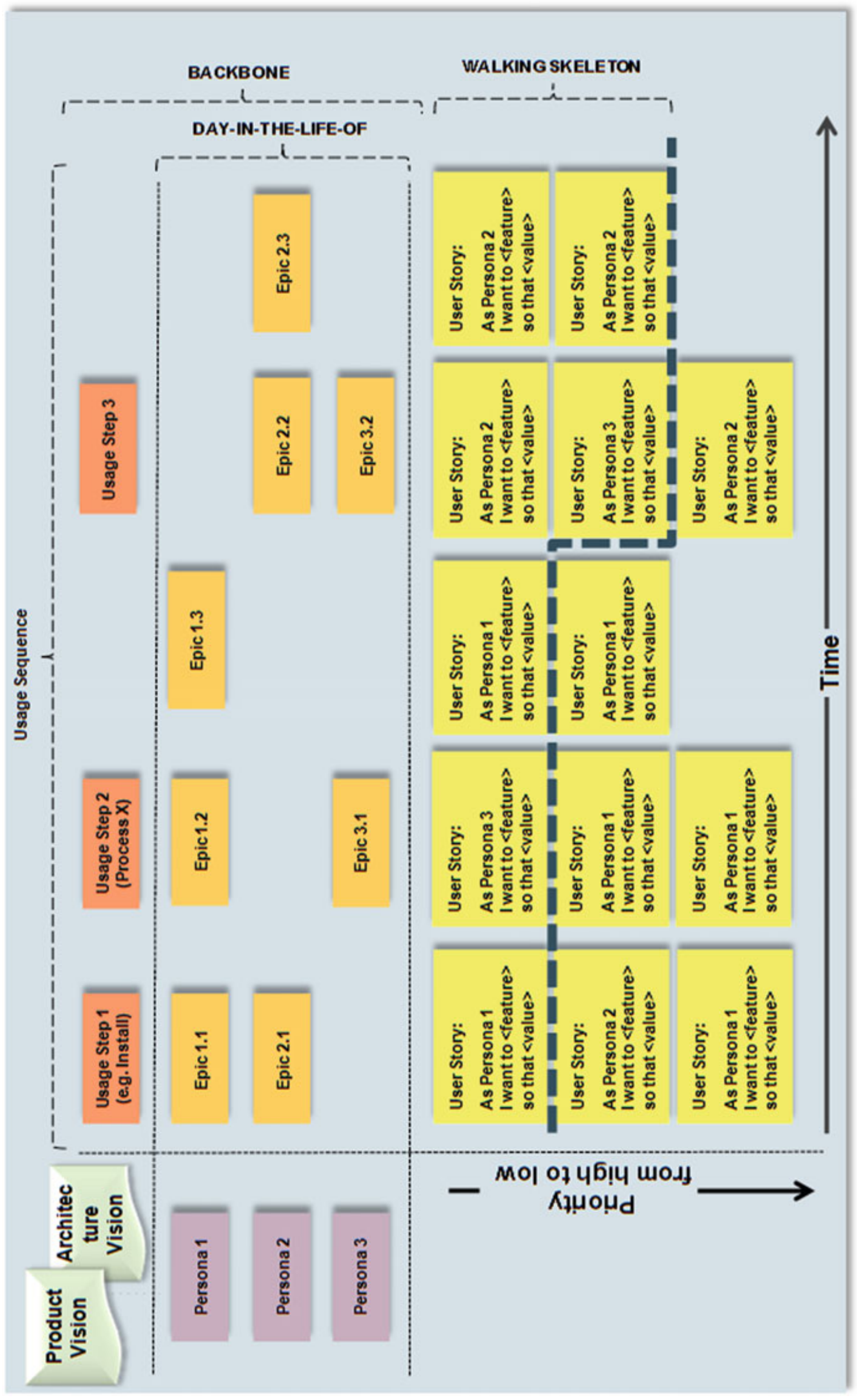

Product Vision
Architecture Vision
Usage Sequence
BACKBONE
DAY-IN-THE-LIFE-OF
WALKING SKELETON
Usage Step 1 (e.g. Install)
Usage Step 2 (Process X)
Usage Step 3
Persona 1
Persona 2
Persona 3
Epic 1.1
Epic 1.2
Epic 1.3
Epic 2.1
Epic 2.2
Epic 2.3
Epic 3.1
Epic 3.2
User Story: As Persona 1 I want to <feature> so that <value>
User Story: As Persona 2 I want to <feature> so that <value>
User Story: As Persona 3 I want to <feature> so that <value>
Priority from high to low
Time

Fig. 5 Constituents and methodology of a user story map (Based on Patton 2008)

Table 3 Three-level approach of implementing user stories to get feedback

Artifact	Description and done criteria
1. Working software	User story fully deployed and integrated into on-demand application so that it can be tested in a browser on different devices
2. Hi-Fi prototype	Digital representation of relevant screen including the application's HTML5 UI design (in Visio or PowerPoint format)
3. Lo-Fi prototype	Paper prototype to communicate the basic idea, data fields, application workflow, etc., e.g. wireframes on A4 paper or comic-style user stories

with, and what our concrete "team parameters" for the Sail Better project would be, e.g. sprint length, time for daily scrum meetings, done criteria for user stories, etc. After discussing Scrum and Kanban as possible process frameworks and highlighting some proven agile engineering practices at SAP, we put the major parameters of our collaboration model such as initial sprint length, time for daily stand-up meetings, time boxes for backlog grooming and retrospectives, etc. on a team charter, "signed it with blood" (only in the figurative sense), and copied them to our team wiki. The team also decided to conduct continuous ideation and prototyping sessions for particular user stories. That is, the team iterates and ideates on certain aspects of the solution that are not yet so well understood as part of our development sprints (also called "spikes" in agile development, cf. Leffingwell 2011). We time-boxed these sessions to 1–2 h maximum and used paper prototyping as primary low fidelity tool to communicate about ideas and get feedback (see Sect. 3.4 and Table 3 in particular).

As a summary of post-Kiel workshop results, we had a clear product vision and backlog which was validated and aligned with our customer, STG. The team had agreed on a working model and our story map helped us to constantly maintain and overview of our requirements and continuously refine the features. Especially the personas (Tina and Thomas) keep us focused on our end users and facilitated communication within the team:

> Hey Chris, do you think Tina would find this radio button intuitive?

Paper prototypes and other cheap artifacts enabled us to continuously receive fast feedback from customers and colleagues right from the beginning and reduced risk tremendously.

One major problem we were facing was the fact that two developers had to go back to Palo Alto (USA) and work from there most of the time. That meant we needed to find suitable meeting times for both locations, had to get a sound station, figure out how to ensure transparency on major decisions and updates also electronically, etc. The main part of the team in Walldorf, however, used the artifacts from Kiel to "decorate" their shared office space and added lots of face-to-face collaboration tools, e.g. whiteboards, a projector, stand-up tables, loads of post-its as well as movable walls. Hence, the team room fosters communication (in terms of quality and frequency) and collaboration by full overall transparency and visual management. You would also often find pairs of developers discussing in the nearby coffee corner.

3.4 Scrum-Based Development and Continuous Design (Phase 3)

Literally on day four, i.e. day one after the workshop described above, the team started elaborating the early prototypes and user stories while receiving continuous feedback by STG.

As indicated in Fig. 2, we planned to get into a regular lean and agile development mode after a first complete delivery of the software at the German Championships in Travemuende. Since we were applying agile methods right from the start (see Sect. 3.2), i.e. maintaining an ordered backlog, working in iterations, etc., the transition from a predominantly Design Thinking-driven working mode to business-as-usual Scrum was very smooth. However, we never stopped experimenting with Design Thinking methods and techniques during Sprints and the team actively requested moderated breakout sessions when they approached new and unclear user stories.

In order to realize regular feedback and gain as many additional insights as possible from our STG users, the team agreed on three-levels of user story completion and according done criteria (cf. Sutherland and Schwaber 2011 and Table 3): if the team could not deliver the story as working software within one sprint, they tried to at least come up with a prototype to evaluate with STG. Sometimes a low-fidelity prototype, e.g. wireframe on paper, was just enough to get feedback. High-fidelity prototypes created with the help of professional user interface (UI) designers and tools such as Visio and PowerPoint were also utilized to refine paper prototype and get additional feedback on the actual look and feel before implementing the screen in HTML5. When stories were deployed and integrated, the users from STG could immediately access the new functionality via the on-demand platform.

4 Evaluation and Discussion

As a result of implementing our process, the team managed to get from zero sailing expertise in June 2011 to a first "shipment" to their customer, Audi Sailing Team Germany, as part of the German Championships in Travemuende in September 2011, i.e. after about 3 months. The feedback on our initial backlog, first prototypes and first software release was very positive and constructive. Due to the fact that we had cheap paper prototypes very early, even during our first team workshop, and continuously evaluated, discarded and/or refined these artifacts, the team received constant feedback from Sailing Team Germany and thus began to "flow".

4.1 Evaluation of Development Process from Empathy to Shipment

After a first delivery of the software in September 2011 in Travemuende, Germany, we collected feedback from the development team, Scrum Master, Product Owner, and STG as customer: most importantly, our customer and the users from STG confirmed that what the team built within the given time frame of barely 3 months went far beyond their expectations (STG 2011):

SAP Sail Better helps us to optimize what we do and it also helps us to innovate.

The Sail Better software has been delivered as an on demand solution with HTML5 UI. Thus, user stories for sailors and coaches could be evaluated in a real-live user environment on the intended devices such as laptops and iPads. At the World Championships in Perth in November, an extended shipment with all major user stories was provided and evaluated by STG: The Sail Better software solution is deemed highly useful to improve the preparations for the Olympic Games in London in 2012 – or as one professional sailor put it in Perth last year (STG 2011):

It is a big advantage over all competitors that we race against [at the world championships].

STG was not only impressed by the software, but also by the methods that have been applied. The product owner (PO) also underlined that the user story map helped him a lot to communicate with both the customer and the development team. Moreover, the PO appreciated that the team was able to nail down the first set of relevant user stories based on their finding from research and the visit to Kiel. Compared to other agile projects, this meant a tremendous boost on the way to development start. PO, development team, and Scrum Master (cf. Sutherland and Schwaber 2011) emphasized that in general the development of the team from having no clue about sailing to being ready to develop after Kiel Week and the workshop was amazing.

Based on our findings and evaluation results, the body of knowledge in both Design Thinking and Lean software product development and our experience with the STG project, we found some first themes to be discussed. The team is generally very positive that the time taken "before actually coding" is deemed useful for their result. All in all, they also claimed that

More projects should have an explicit Lean and Design Thinking coach.

Besides this general evaluation of the process, three main "patterns" which we also observed at other projects at SAP have been reconfirmed as findings by our case study:

(1) The *story mapping* technique (Patton 2008) helped us to synthesize and leverage on the findings from user research and bring the "to-be perspective" from Design Thinking into our backlog. (2) Despite the tight time frame, it paid off to spend a considerable amount of time for better understanding the problem domain, concrete user needs and impediments in the development project, i.e. we *stopped to think* at defined points in time. (3) Ideation and prototyping sessions – both, at the outset and within the sprints – enabled us to get fast feedback from our users. Superficially, Lean Thinking might define this as waste, but creating tangible and visual results very early potentially saved us from creating even more waste in the course of the project (see example in Sect. 4.3).

4.2 Story Maps Can Bridge the Gap from Research to Backlog

Story Mapping proved to be a powerful tool to structure the features of a classical prototype into a backlog. Especially in software product development it is important to consider the full life- or usage cycle of a product, and story maps can help raise awareness for details in the sequence. The team and especially the product owner confirmed the story map to be a useful reference in sprint planning and prioritization.

Story maps are structured and rather unemotional: because they follow the user's "Day in a Life" though, they can connect to Design Thinking much better than a classical backlog representation as a flat list. The reuse of the synthesized personas and thereby research data can help the team to empathize with the respective user stories.

Nevertheless our process applied in the STG case study had its biggest weakness at the point where the user story took over the results of our prototyping and product vision exercises. There is one interesting pitfall that has to be analyzed in more detail: As an innovation method, Design Thinking makes a strong point to differentiate between as-is and to-be scenarios and processes from a user perspective. While in "classical" agile requirements engineering (cf. Leffingwell 2011), these two points of reference are often the same in order to support the current process, Design Thinking has the very aim of disrupting the status quo with a true innovation.

For the team in our STG case, this "little" difference turned out to be difficult. Their feedback was that it was unclear what exactly should go on the story map. While they were eager to reuse the artifacts they had developed in the Design Thinking steps, there were two different sources they could draw from: The research data that described as-is processes and the future scenarios described in the prototypes. However, in agile development, there is only one map.

Our impression is that story mapping can be used for both, but with a clear distinction. A story map that assembles an overview of the current process can be a possible research artifact, but not automatically a basis for an innovative product and the respective backlog. If the story map is supposed to feed the backlog, it has to emerge from the usage scenario of the future solution. To make this transition possible, the respective prototypes themselves must already have a strong emphasis on the process (mockups are not enough), on the way they will influence the user's daily conduct in the future. On the other hand, it may help to enrich the story map to create more empathy. The use of visuals and pictures, for example, may be a way to improve the logical connection between prototypes and the story map. If we put descriptions of usage processes into the prototype and emotions into our story maps, they are likely to work even better.

4.3 Take Your Time, Stop to Think

Design Thinking suggests to spent sufficient time for observing and conducting user research to better understand the underlying problem space, customer needs, and develop true empathy for future users. This enables teams and organizations to build products that are desirable, feasible, and viable. Moreover, iteration in Design Thinking accepts the fact that a complete restart might be required due to insightful feedback from users (Brown 2009; Ries 2011).

Lean Thinking, on the other hand, reserves time to analyze which processes directly contribute to customer value, which ones are non-value-adding but nonetheless necessary, and which ones are deemed obvious waste. Retrospectives in Scrum (Derby and Schwaber 2006) and other methods, such as A3-based problem solving (Rother 2010) give guidelines on how to achieve continuous improvement, i.e. "sharpening the axe" efficiently and sustainably. In our STG project, for instance, we also conducted regular retrospectives in order to streamline the team processes.

Bottom line, both thinking schools reserve a reasonable amount of time for the implementation of their core values such as identifying value-adding activities and eliminating waste in recurring processes (in Lean Thinking) as well as developing empathy for user needs and pains by taking a user perspective and investing in exhaustive research and continuous prototyping (in Design Thinking). One could say that they both "stop to think", i.e. stop in order to think. In both cases, the time spent for thinking in these respects is well invested. The successful shipment of the "Sail Better" software to STG is one more proof point.

4.4 Innovation Needs Some Waste

"Waste" is a term from Lean Thinking and lean production systems coined at Toyota and Porsche, for instance. Waste denotes activities and processes that do not add direct value for the end customer in the sense that value is defined as "something the customer is willing to pay for" (Womack et al. 1990; Poppendieck 2002; Poppendieck and Poppendieck 2003; Womack and Jones 2003).

Innovation approaches such as Design Thinking, on the other hand, foster "real" brainstorming and the creation of ideas in large quantities. Many of these ideas might be discarded right away or later in the process based on relentless user feedback. That is, they literally end up in the "waste bin". Moreover, Design Thinking also suggests rapid and cheap prototyping to get feedback in order to "fail early and often" (Brown 2009, p. 87). Again, many of these prototypes will end up in the waste bin after they served their purpose of leveraging feedback on existing solution approaches or inspiring even better ideas.

Now, despite all the waste, can our two thinking schools intertwine to solve problems for the software industry that neither one could on its own? We think so, especially in standard software development processes. Just imagine this simple example:

A development team combines Design Thinking and Scrum. They spend a serious amount of time for user research, brainstorming, ideation, paper prototypes, etc. – similar to our team developing “Sail Better” for STG. Let us say 20 percent of their ideas make it into the final product which becomes a blockbuster – i.e. in Lean Thinking terms, ideation inferred 80 percent waste. On the other hand, another team does it the “textbook agile way” with some requirements workshops, user story writing, and backlog grooming. They realize 80 percent of the initial backlog, but the product completely fails on the market despite efficient development processes.

In Lean Thinking terms, the latter implies close to 100 % waste, since no customer is willing to pay for the software eventually (see also the “Lean Startup” approach by Ries 2011).

5 Conclusion and Outlook

To conclude, let’s summarize again *why* the combination of the two thinking schools is so promising and *how* it helps software development teams to come up with an innovative product vision and derive requirements in a yet unknown domain:

While *Lean Thinking* and agile practices are meant to help us building products right, i.e. in-time, in-quality, etc., *Design Thinking* can help us to build the right product based on a valid customer problem in the first place. Both thinking schools address responding to desire at the customer side and working efficiently as a team, however with different emphasis: a lean development project, for example, expects the agile team and the product owner in particular, to already have a product vision ready. But where does that vision come from? Design Thinking, on the other hand, provides various methods to build empathy with end users and other inspiration sources for developing a solution idea or product vision as well as prototypes for what the product should actually do for the end user. But after that, it lacks a clear framework of the roles, artifacts, tools, and necessary steps that can take the vision from prototype to a shippable product.

In our concrete case, the STG project, we learned that understanding and developing empathy for the customers’ context and experience enables the team to elaborate and choose the right backlog items and user stories to be developed. This process is mainly driven by Design Thinking practices. Iterative development and thus fast feedback reduces project risk and ensures efficient delivery and shipment of the software. The development and delivery process of Sail Better is mainly based on common Lean and Agile practices.

Besides the positive feedback from the team and the customers on both the approach and the solution, three main findings evolved from our case study: (1) *User story maps* can serve as a tool to bridge the gap from the empathy and insights gained with Design Thinking practices and the backlog to build the solution, (2) both thinking schools suggest to reserve a substantial amount of project time, i.e. stop to think, for implementing their core values, such as developing empathy by means of observation, user research, etc. as well as eliminating waste by reflecting the value-add created by particular processes, obvious waste, and according analysis and improvement activities. (3) Innovation and great solutions do not come for free and it may require a little "waste" due to scoping and prototyping from a Lean perspective in order to gain fast feedback and succeed eventually. Moreover, several core values of the two thinking schools are almost identical and hence facilitate intertwining the respective methods and techniques: e.g. cross-functional or multi-disciplinary teams working in iterations, continuous and fast feedback loops ("inspect and adapt") as well as a clear focus on customer value and desire.

Our future research on Lean and Design Thinking will include a multi-case study across 10–50 so-called "early adopter" projects at SAP that implement Lean and Design Thinking. In order to come closer to an integrated theory of both efficient and innovative software product development with Lean and Design Thinking principles and practices we need to better understand the contingencies in different project settings (Van de Ven 1985). Particular process design challenges include the smooth and efficient information and artifact flow from divergent and convergent thinking in iterative problem scoping and solution development. As we already experienced in the STG project, a clear distinction but smooth transition from as-is to to-be scenarios is critical to maintain this information flow. Another research trajectory revolves around scaling and embedding Design Thinking in a larger lean and agile enterprise software development setting. By this overall research agenda we try to address that managing the transition from traditional waterfall-like development and scaling reliable innovation and efficient delivery for up to 100 development teams working on one complex solution for various customer segments will be one of the major challenges in the software industry.

Acknowledgements We have to thank: Stefan Lacher, Jochen Guertler and his Innovation Center team, as well as Marcus Baur as main representative for Sailing Team Germany.

References

Anderson, D. J. (2010). *Kanban*. Sequim, WA: Blue Hole Press.

Beck, K. (1999). Embracing change with extreme programming. *Computer, 32*(10), 70–77.

Beck, K., Beedle, M., Bennekum, A. V., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). Agile manifesto.

Blau, B., & Hildenbrand, T. (2011b). Product line engineering in large-scale lean and agile software product development environments – Towards a hybrid approach to decentral control and managed reuse. *Presented at the 6th international conference on availability, reliability and security*, Vienna.

Blau, B., Hildenbrand, T., Xu, Y., & Fassunge, M. G. (2011a). Incentives and performance in large-scale lean software development – An agent-based simulation approach. *Presented at the 6th international conference on evaluation of novel approaches to software engineering (ENASE 2011)*, Beijing.

Brown, T. (2009). *Change by design: How design thinking transforms organizations and inspires innovation*. Harper Business.

Chow, T., & Cao, D. B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software, 81*(6), 961–971.

Cohn, M. (2004). *User stories applied: For agile software development*. Boston: Addison-Wesley Professional.

Derby, E., Larsen, D., & Schwaber, K. (2006). *Agile retrospectives: Making good teams great*. Raleigh: Pragmatic Bookshelf.

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of Management Review, 14*(4), 532–550.

Hevner, A., March, S., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75–105.

Highsmith, J. A., & Highsmith, J. (2009). *Agile project management: Creating innovative products*. Boston: Addison-Wesley Professional.

Hildenbrand, T., Geisser, M., Kude, T., Bruch, D., & Acker, T. (2008). Agile methodologies for distributed collaborative development of enterprise applications. *International conference on complex, intelligent and software intensive systems. CISIS 2008* (pp. 540–545), Barcelona.

Kelley, T. (2001). *The art of innovation*. London: Profile Books Ltd.

Kelley, T. (2008). *The ten faces of innovation*. London: Profile Books Ltd.

Kniberg, H. (2007). Scrum and XP from the Trenches. *InfoQ Enterprise Software Development Series*.

Larman, C., & Vodde, B. (2008). *Scaling lean & agile development: Thinking and organizational tools for large-scale Scrum*. Boston: Addison-Wesley Professional.

Larman, C., & Vodde, B. (2010). *Practices for scaling lean and agile development: Large, multisite, and offshore product development with large-scale scrum*. Addison-Wesley Professional. Upper Saddle River, NJ: USA

Leffingwell, D. (2011). Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional. Upper Saddle River, NJ: USA.

Martin, R. C. (2008). *Clean code: A handbook of agile software craftsmanship*. Upper Saddle River: Prentice Hall.

Martin, R. L. (2009). *The design of business: Why design thinking is the next competitive advantage*. Harvard Business School Press. Boston: USA.

Meinel, C., & Leifer, L. (2011). *Design thinking: Understand – improve – apply*. Berlin/ Heidelberg: Springer.

Patton, J. (2008). The new backlog. AgileProductDesign.com.

Pichler, R. (2010). *Agile product management with scrum: Creating products that customers love*. Amsterdam: Addison-Wesley Professional.

Poppendieck, M. (2002). Principles of lean thinking. *OOPSLA Onward*.

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An agile toolkit*. Upper Saddle River: Addison-Wesley Professional.

Reinertsen, D. G. (1997). *Managing the design factory: A product developer's toolkit*. New York: Free Press.

Reinertsen, D. G. (2009). *The principles of product development flow: Second generation lean product development*. Redondo Beach, CA: Celeritas Publishing.

Ries E. (2011). *The lean startup: How constant innovation creates radically successful businesses*. Crown Publishing Group. London: UK.

Rother, M., & MyiLibrary. (2010). *Toyota kata: managing people for improvement, adaptiveness, and superior results*. New York: McGraw Hill.

Sailing Team Germany Uses. (2011). *SAP Sail Better*. youtube.com.

Schnitter, J., & Mackert, O. (2011). Large-scale agile software development at SAP AG. *Evaluation of Novel Approaches to Software Engineering*, *230,* 209–220. doi:10.1007/978-3-642-23391-3_15

Schwaber, K., et al. (1995). Scrum development process. In *OOPSLA business object design and implementation workshop* (Vol. 27, pp. 10–19). Austin: TX.

Smith, P. G., & Reinertsen, D. G. (1992). Shortening the product development cycle. *Research Technology Management, 35*(3), 44–49.

Sutherland, J., & Schwaber, J. (2011). *Scrum Guide*, http://www.Scrum.org/scrumguides/ (cit. 2011).

Van de Ven, A. H., & Drazin, R. (1985). The concept of fit in contingency theory. *Research in Organizational Behavior, 7*(3), 333–365.

Womack, J. P., Jones, D. T. & Daniel, R. (1990). *The machine that changed the world*. Free Press. New York: USA.

Womack, J. P., & Jones, D. T. (2003). *Lean thinking: Banish waste and create wealth in your corporation*. New York: Simon and Schuster.

Yin, R. K. (2008). *Case study research: Design and methods* (Vol. 4). Thousand Oaks: Sage publications, Inc.

Yin, R. K. (2011). *Applications of case study research* (Vol. 34). Thousand Oaks: Sage Publications, Inc.