



Jumpstarting Scrum with Design Thinking

IWI-HSG: Christophe Vetterli, Prof. Dr. Falk Uebernickel, Prof. Dr. Walter Brenner
HPI: Franziska Häger, Thomas Kowark, Jens Krüger, Dr. Jürgen Müller,
Prof. Dr. Hasso Plattner
SAP: Barbara Stortz, Vishal Sikkha
Report No.: Whitepaper
Chair: Prof. Dr. W. Brenner
Version: 1.1
Date: June 26, 2013



In cooperation with

IT Systems Engineering | Universität Potsdam

**University of St.Gallen for Business
Administration, Economics, Law and
Social Sciences (HSG)**

Institute of Information Management
Müller-Friedberg-Strasse 8
CH-9000 St.Gallen, Switzerland
Tel.: +41 71 224 2420
Fax: +41 71 224 2777

Prof. Dr. A. Back
Prof. Dr. W. Brenner (managing)
Prof. Dr. R. Jung
Prof. Dr. H. Österle
Prof. Dr. R. Winter

Table of Contents

1	Motivation.....	7
2	Scrum and Design Thinking in a Nutshell.....	9
2.1	Scrum	9
2.2	Design Thinking.....	10
2.3	Milestone Concept (in Design Thinking).....	13
3	Common Principles for an Integrated Approach	15
4	DT @ Scrum.....	19
4.1	Methodological Background.....	20
4.2	Operation Mode Overview	21
4.3	Design Thinking Mode.....	26
4.3.1	Activities	26
4.3.2	Roles	29
4.3.3	Deliverables	31
4.3.4	Techniques	31
4.4	Initial Development Mode	32
4.4.1	Activities	33
4.4.2	Roles	34
4.4.3	Deliverables	35
4.4.4	Techniques	35
4.5	Fully Integrated Mode.....	38
4.5.1	Activities	38
4.5.2	Roles	39
4.5.3	Deliverables	40
4.5.4	Techniques	40
5	Integration of DT@Scrum in Organizations – Initial Ideas	41
5.1	Large-scale Projects	45
5.2	Product and Process Standards Compliance	46

5.2.1	Continuous Compliance Model	46
5.2.2	Compliance Requirements during Operation Modes.....	48
6	Validation & Future Work	51
6.1	Possible Extensions.....	51
6.2	Refinement and Validation.....	52
6.3	Current Research Plan.....	54
	References.....	56
	Appendix.....	57
A.1	ME310 Project Milestones:.....	57
A.2	Techniques:	58

Table of Figures

Figure 1: Basic Idea of the Scrum Process	10
Figure 2: Design Thinking Process as taught in HPI School of Design Thinking & ME310	11
Figure 3: Milestones in Design Thinking Projects (ME310 milestones).....	13
Figure 4: Design View vs. Managerial View.....	19
Figure 5: Method components	20
Figure 6: Process embedding of innovation team.....	21
Figure 7: Balance of Design Thinking vs. Scrum Activities within the operation modes.....	24
Figure 8: Design Thinking Mode.....	26
Figure 9: Design Thinking Milestones for the Design Thinking Mode.....	27
Figure 10: Initial Development Mode.....	32
Figure 11: Interactive Paper Prototype	36
Figure 12: Pidoco Workspace and Pidoco Simulation in Sketch Mode	37
Figure 13: Fully Integrated Mode	38
Figure 14: Entry Point with no relevant knowledge of the design space.....	41
Figure 15: Entry Point with existing knowledge of design space and given solution ideas....	42
Figure 16: The rapid converging of the given ideas to one concretized functional prototype	42
Figure 17: Knowledge Flow in Project Portfolios	43
Figure 18: Team Scale up During Operation Modes	45
Figure 19: Product Innovation Lifecycle phases with quality gates	46
Figure 20: Example Project Schedule.....	55

Management Summary

Customers have become more demanding in terms of customization, speed, and involvement with the value proposition of today's services and products. This means that in order to remain competitive, companies need to re-design their existing development processes to constantly capture the customer needs, rapidly conceive innovative, highly customer-oriented solutions, and perform these developments with an increasingly short time-to-market.

Today, almost all products and services are software-driven. The software either provides underlying functions or augments them. For example, there is hardly a product manufactured now that does not have an associated app. Software development in such ecosystems is very complex, and it becomes even more demanding in the context of large enterprise applications. Business software requirements, e.g., for security and accessibility, impose tremendous non-functional requirements, while laws and regulations from around the world further complicate development. These factors have long driven software development methodology towards solid requirements and process efficiency rather than the kind of innovative rapid responses customers now demand.

To rapidly deliver solutions in this flexible environment, many companies have adopted agile software development processes, such as Scrum. But even though Scrum can help to deliver software solutions in a highly dynamic environment with unclear solution requirements, its main focus is not the delivery of radical new innovations. There is, however, a human-centered, prototype-driven innovation methodology called "Design Thinking" (DT), which shares many common principles with Scrum. These principles, amongst others, include an iterative workflow and a flexible process framework. Our hypothesis is that these two methodologies are compatible and that the merger of the two into one integrated *DT@Scrum* approach would result in an agile software development process that could deliver the innovative customer-oriented products and services required by competitive companies.

The paper identifies different operation modes for the described approach depending on the aspired level of innovation, the team composition, and the project size and discusses possibil-

ities to incorporate the new approach into large IT departments of companies. This covers aspects such as distributed teams, scaling of projects, and adherence to internal and external regulations. The operation modes *Design Thinking Mode*, *Initial Development Mode* and *Fully Integrated Mode* are addressed by proposing a set of general process activities, discussing the application of roles and techniques, and proposing techniques to allow a smooth transition between requirements identification, ideation, and implementation.

While the paper provides potential methods of validation, such as initial test projects and experiments, further tests and validation of the proposed activities, roles, and techniques are necessary. The intent of this paper is to make these ideas available for discussion with fellow researchers and all interested and potentially affected divisions within SAP, with the ultimate goal of creating an integrated DT@Scrum process model that will leverage human-centered, creative techniques for requirements engineering and seamlessly move from there to agile development, thus providing a holistic, user-centered software development approach.

1 Motivation

Software engineering, especially for large enterprise applications, is becoming increasingly complex. Customers demand features faster and expect them to be tailored to their specific needs. Standards for business software, e.g., for security and accessibility, lead to a large number of non-functional requirements, while laws and regulations from around the world need to be considered and integrated as well. Especially larger software companies need to introduce additional constraints on their teams and products to ensure high quality software and high availability of the software and its provided services. In order to deliver innovative products with reasonable time-to-market despite these necessary constraints, companies adopt agile software development processes such as Scrum (Schwaber & Beedle, 2001). These agile processes supersede traditional waterfall processes and promote iterative delivery of working software with constant integration of up-to-date customer feedback. While agile processes vastly speed-up development processes, requirements engineering is treated as a black box that the engineering teams need to solve with other techniques.

Over the past years Design Thinking has evolved into a powerful methodology to initiate and implement successful innovation. Its user-centered, iterative concept has been successfully applied in various projects at d.schools, at universities, for example in innovation courses like ME310, by design agencies like IDEO, and - increasingly - in software development companies like Google, Microsoft, and SAP. All of these projects aim at creating products and services that are not only small, incremental improvements of existing solutions, but innovative ideas that provide the maximum benefit for potential end-users.

While both Scrum and Design Thinking excel in their respective application areas, their integration is still subject to research (Hildenbrand & Meyer, 2012). In this paper we present initial concepts for an integrated approach of Design Thinking (DT) and Scrum. We propose a set of activities, deliverables, roles, and techniques that allow for a seamless transition from requirements elicitation using DT to their implementation, using Scrum.

From a methodological point of view, the design science research paradigm fits well to the subject of this paper. It addresses the central goal of designing a new useful product (Hevner et al. 2004). To reach this utility, knowledge and understanding of the problem domain and its solution are developed in the process of repeated construction and application of prototypes. This process enables design science researchers to understand the problem addressed by the artifact and the feasibility of their approach to its solution (Hevner et al. 2004). We will employ this approach to constantly evaluate and adopt our conceived development process model.

Scrum is today most often applied to relatively straightforward engineering tasks that require a high degree of end-user feedback. DT, on the other hand, focuses on solving so-called “wicked problems”, which have no straightforward technical solution. This paper focuses on this problem domain, showing how to unite Design Thinking and Scrum in order to create a new integrated DT@Scrum approach that combines the strengths of both approaches in order to smoothly transfer from the generation of concepts that solve such wicked problems to their technical implementation within the context of enterprise software development.

2 Scrum and Design Thinking in a Nutshell

In the following, we briefly describe the fundamental concepts of both Scrum and Design Thinking. The description of Design Thinking outlines the different understandings that the d.schools, ME310, and literature have about the methodology. Hence, we strongly encourage all readers to not skip the following paragraphs even if they have prior knowledge about Design Thinking.

2.1 Scrum

Scrum is an iterative and incremental agile software development process framework. Its core principle is the idea of performing time-boxed development sprints at whose end a working version of the system under development is created (see Fig. 1). This principle allows continuously validating the matching between the product and the customer expectations and, if necessary, adopting the development by adding, removing, or altering requirements. Scrum teams comprise three main roles. Product Owners collect requirements from the different stakeholders of the system and transfer them into small, implementable units (typically user stories). These units are implemented within the sprints by the development team. A specially trained Scrum Master supports the team by moderating the different meetings and solving problems that potentially impede the work of the team.

The overall requirements for the target system are kept in a prioritized Product Backlog. At the beginning of each sprint, the Product Owner and the development team perform a sprint planning meeting. In this meeting, they discuss the requirements with the highest priority, estimate the required development effort, and determine which user stories will be implemented throughout the sprints. The chosen stories form the so-called Sprint Backlog. Within the sprint, the development team performs a daily stand-up meeting where each member informs their colleagues about current and future work, as well as potential and actual problems that impede their progress. By that, problems can be solved very early in the process and knowledge is evenly distributed within the team. At the end of the sprint, the Product Owner

reviews the implemented functionality and either accepts or rejects the corresponding stories. Based on the results of the review, the efforts for the subsequent sprint can be adopted.

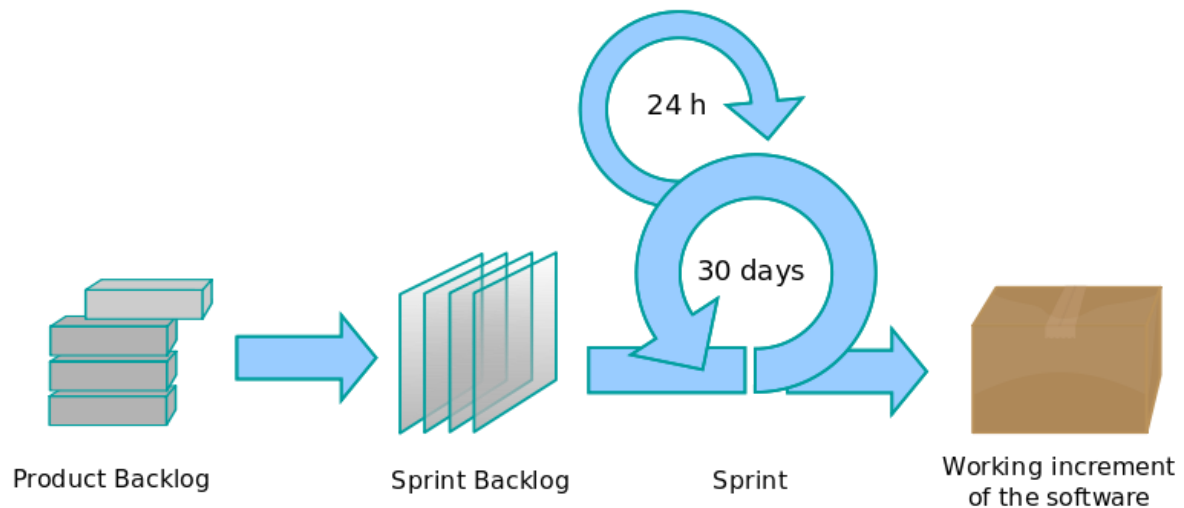


Figure 1: Basic Idea of the Scrum Process (Wikipedia, 2013)

In summary, Scrum provides development teams with a simple framework for incremental software development that ensures the delivery of working software increments after each sprint. These working increments can be used for customer validation and, hence, allow to react to changing requirements in an agile manner.

2.2 Design Thinking

Design Thinking (DT) is a human-centered, rapid prototype based innovation method. Its main idea is to develop a solution in close exchange with stakeholders and target users to ensure desirability, viability, and feasibility of the final solution. The methodology provides a process framework that allows for constant communication between the developing team and the stakeholders and target users. Furthermore, it includes tools and methods that enable the innovation team to gain and collect information, discover hidden aspects or user needs in this information, and communicate their ideas in a tangible manner to collect feedback.

In general a DT project starts with a problem statement or a challenge and an initial *Understand / Needfinding* and gathering of *Instant Expertise phase* that helps collect information

about the project as well as its goals, constraints and the environment of the project. During this phase the Design Team should investigate target users, stakeholders, competitors and possible solutions as well as open questions via a 360° research.

During the problem elicitation phase of DT, the team gets acquainted with the problem domain, investigates existing solutions, and interviews and observes users and stakeholders. The core strength of DT are its tools and techniques that allow not only conceiving solutions for observed or explicitly expressed user “wants”, but also uncovering the underlying needs. A simple example: While users might say, they need a better search engine, further investigations reveal that their real need is to have a better way of finding experts inside the company, regardless of the technical implementation.

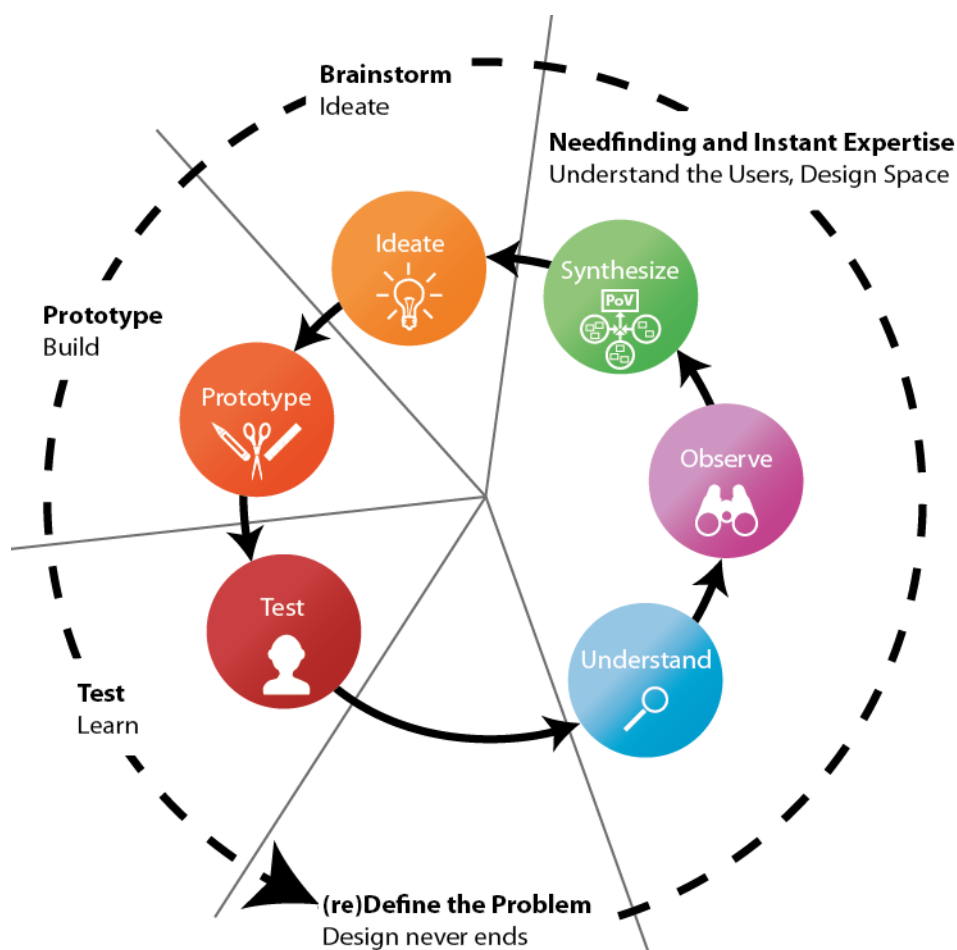


Figure 2: Design Thinking Process as taught in HPI School of Design Thinking and ME310 (adopted from (Plattner, Meinel, Leifer, 2010))

All the information gathered through research, interviews, and observations is synthesized into the team's point of view on the problem. Different techniques to achieve this and to discover meaningful insight from all that information exist. These are among others: Clustering, the Persona, the POV Madlib, the 2 by 2 Matrix and the Venn Diagram (see Appendix for details). As an example, the Persona combines the most important aspects and insights from different interviews and serves as a representative of the target group to the innovation team. Based on this *Point of View* the Design Team *Ideates* aspects of a possible solution and *prototypes* these. To initiate *Ideation* the team should brainstorm on so called How Might We Questions, which can be derived from the team's Point of View. The ideas generated in such a Brainstorming session need to be prototyped in a rapid fashion that focuses on transporting the main idea instead of creating "beautiful" artifacts. Prototypes are a central aspect of Design Thinking, as they allow users to experience a proposed solution or compare different solutions in a tangible manner and not just as an abstract description of an idea. Prototype fidelity can range from low-resolution prototypes like sketches or paper models to high-resolution prototypes like functioning miniature models or fully functional parts of the solution. They can not only be used to validate an idea or solution but can also help to learn and understand the user or certain aspects of the problem for e.g. by simulating certain situations and observe the user in them. Especially in a corporate context, their function as important tool in communication with target users and stakeholders is crucial.

Each prototype will undergo intensive *Tests* with target users. The information gained by testing the ideas needs to be synthesized again. Depending on the outcome of this *Synthesis phase*, the team will start a consecutive Iteration in which it will move on with further ideation to refine the idea or, go back to *Understand* and *Observe* phases to answer open questions and investigate new aspects of the problem.

To summarize, Design Thinking provides innovation teams with a simple set of tools and a process framework that ensures an iterative development in constant exchange with target

users and stakeholders. During the process the prototypes will become more sophisticated and real until a final solution prototype or implementation is reached.

2.3 Milestone Concept (in Design Thinking)

Both Scrum and Design Thinking potentially allow projects to run indefinitely. In a Scrum project, new requirements can be integrated in each sprint, and the DT phases can also be used to almost indefinitely refine a potential solution. To guide teams in their efforts, ME310 adopted the milestone concept that forces Design Teams to break this infinite cycle and converge towards a final deliverable.

With each iteration through the design cycles (compare Figure 1 and Figure 2) the created prototypes provide better insights of the design space and especially in the converging phase (Figure 3) become more sophisticated moving from simple sketches or paper prototypes to wireframes or other forms of software prototyping thus expressing more and more aspects of the pursued solution concept. This forward direction in the process can be supported by prescribing a sequence of certain types of prototypes and other deliverables as done in the global ME310 course or Embedded Design Thinking (Vetterli et al., 2012). Figure 3 depicts the sequence of deliverables as used in the ME310 course and it is important to keep in mind that every type of prototype contains several of the iterations shown in Figure 2.

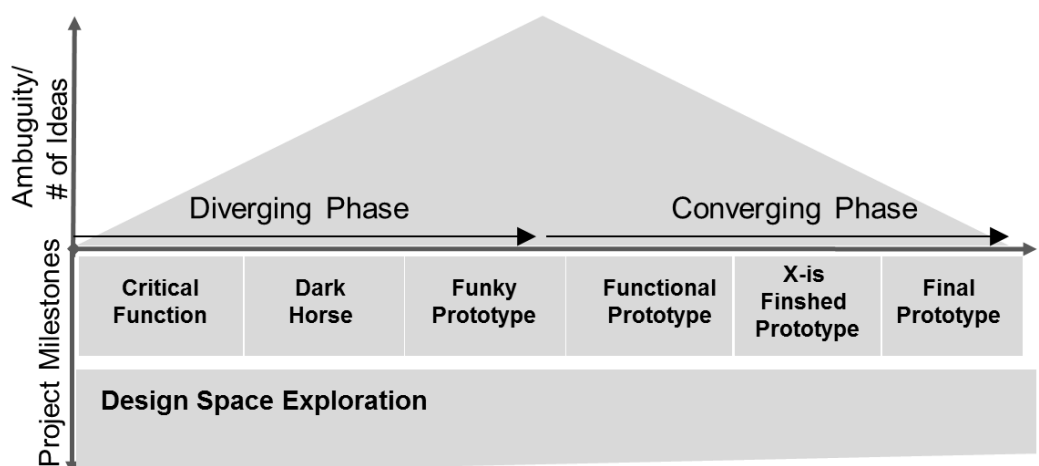


Figure 3: Milestones in Design Thinking Projects (ME310 milestones) (Vetterli et al. 2012)

Every single milestone shown in Figure 3 has a specific didactical goal, which will not be further outlined within the whitepaper but can be read in the Appendix. It is most important to differentiate between the diverging and the converging phase. The diverging phase has its goal in delivering as much ideas as possible whereas the converging phase should extract the most promising ideas out of the diverging phase and converge to a high-resolution prototype, also called the *final prototype*.

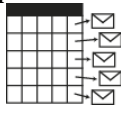
3 Common Principles for an Integrated Approach

The two different approaches, Scrum as well as Design Thinking, imply certain common principles and serve as an adequate base for defining integrated approaches. Scrum results from the overall agile software development. Its main foundation for the understanding of this agile paradigm is linked to the so-called *manifest*. The manifest, an output of a workshop conducted by leading experts in the area of software development, integrates the following four values:

Individuals and interactions* over processes and tools
Working software* over comprehensive documentation
Customer collaboration* over contract negotiation
Responding to change* over following a plan

* That is, while there is value in the items on the right, Scrum values the items on the left more (bold).

The core values are being concretized with the help of the following twelve principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Customer Satisfaction 
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Embrace Change 
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Frequent Delivery 
Business people and developers must work together daily throughout the project.	Interdisciplinary Collaboration 
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Motivation & Trust 

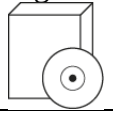
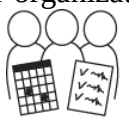
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Face-to-Face Communication 
Working software is the primary measure of progress.	Working Software 
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Sustainability 
Continuous attention to technical excellence and good design enhances agility.	Technical Excellence 
Simplicity--the art of maximizing the amount of work not done--is essential.	Simplicity 
The best architectures, requirements, and designs emerge from self-organizing teams.	Self-organization 
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Reflection 

Table 1: Agile Core Values (Source: agilemanifesto.org (2001))

The different principles serve as frame for the understanding of Scrum as it has been outlined before. When these principles are compared with Design Thinking, it is striking that both approaches largely share a set of common principles. In the following, the authors therefore would like to capture the most important ones with regards to the proposed DT@Scrum process.

Customer orientation

Scrum and Design Thinking are equally oriented towards (potential) customers in development of the final output. Whereas a Scrum process is mainly driven by the goal to develop a

working system, a Design Thinking is agnostic about the output artifacts, but committed to it being customer-oriented. Scrum's Product Owners, collect requirements from the different stakeholders of the system and transfer them into small, implementable units. Design Thinking's design teams create similar small units based on the different situations they have observed when the potential customer was exposed to a partial prototype. They then create small user-stories to analyze and communicate what main trigger points could be used in order to create a new customer experience. Thus, both approaches create context-related user stories wherein the customers as well as relevant external influence factors are integrated. The user stories serve as orientation and guidance in the development.

Iterativity in development

Scrum and Design Thinking both share the focus on an iterative procedure to achieve their respective project target. This results in a high number of loops that are used to validate whether the generated ideas and prototypes meet the needs of (potential) customers. By this means, both methods aim at creating solutions with maximum benefit for the end user and minimizing the cost for doing so by only creating expensive, costly to alter, high-fidelity solutions only after cheap, low fidelity prototypes that represent the general ideas have received positive end user feedback. Furthermore, iterative processes with increasing fidelity of the solution allow halting projects after each iteration with risk only to partial, non-working artifacts. This iterative process resulting in increasingly-high fidelity prototypes (at least in the convergent phase of Design Thinking) might be called the *precedent principle*.

Feedback integration

The different iterations from the precedent principle allow receiving strong feedback directly from the customer. The described iterations focus on involving the future real customer of the solution and create potential solutions at different resolution levels. In every level intensive customer feedback is being captured and integrated in the next loop. Therefore the developed solutions are highly optimized towards serving the identified customer needs.

Pulsing as result driver

The clear pulsing in activities of Scrum as well as Design Thinking enables delivery time forecasting. Although, in Design Thinking the exact output cannot be defined when the pulsing is being defined, the involved parties help to manage the activities adequately in terms of quality and time and at least the timeframe is well-defined by the process. Design Thinking projects use clearly defined milestones, which are communicated beforehand to pulse the design activities.

Scrum teams perform sprints, which are clearly defined in both time and planned output. Thus milestone orientation is a requirement for managing such highly innovative environments and therefore serves as planning tool.

Definition of done in time-limited projects

Both approaches deliver highly relevant and customer-oriented outputs and have per se an output definition of “done”. Hence every output out of these projects reflects the strong goal orientation towards a possible and customer-oriented solution. Though iteration is theoretically infinite, both processes share the principle of convergence within a well-defined timeframe. Especially in organizational context, the realized output with limited length is an added value compared to other approaches.

4 DT @ Scrum

To apply an integrated approach of Design Thinking and Scrum it is crucial to understand the environment of implementation. Companies are mostly business case and project driven and follow managerial paradigms. Therefore, an understanding of the basic differences of a designer's view and a managerial view on the idea generation phase of the innovation process is mandatory. Designers and Design Teams in comparison to managers have a completely different view on the design process for the idea generation phase. Designers see this process as an on-going possibility to always redefine the initial problem and learn from in an iterative way that ends up in a final prototype (Skogstad, 2009). Whereas managers within companies define the design process as a linear decision process and expect that, after the development phase, the process should lead to a market success. This completely different understanding of the opening part of the innovation process creates some conflict potential. Therefore special attention is needed by the diffusion efforts of the innovation throughout different roles within a development process using DT and/or Scrum. The following figure shows the diverging understanding of the design innovation process.

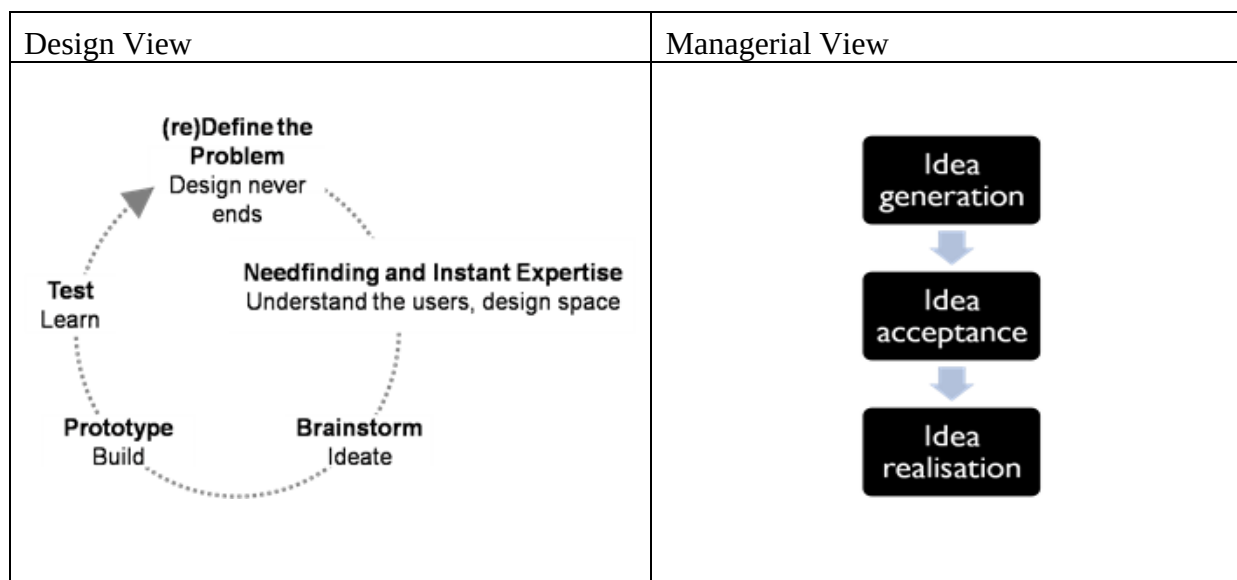


Figure 4: Design View vs. Managerial View (Skogstad, 2009)

A similar difference of thinking can be found with developers who usually see a problem that will be solved exactly once and is then done, while designers encourage early fails and con-

stant redefinition. In order to overcome these differences, managers and developers need to learn to deal with uncertainty especially in the early phases of a Design Thinking project. This includes failing early and often and learning from those mistakes.

DT@Scrum tries to overcome this gap by making both the innovative Design Thinking activities and the software development manageable in a similar manner. Potentially after each sprint, ideally at the end of the different phases, the team and its manager can evaluate the created deliverables and see whether or not the current direction is promising, needs adoption, or should be discarded. The hands-on nature of DT thereby simplifies communication between the two, as testable prototypes are available after each sprint and, hence, management decisions are not only based on PowerPoint slides but real, yet low-fidelity, proofs-of-concept.

4.1 Methodological Background

As outlined in the introduction the authors define Design Thinking as a human-centered innovation method. The institute of information management of the University of St.Gallen has developed the components to define such methods (see, Heym, 1993; Gutzwiller, 1994). The following description of the operation modes will be structured along these five key elements to describe DT@Scrum. The elements are in a relationship to each other as Figure 5 shows:

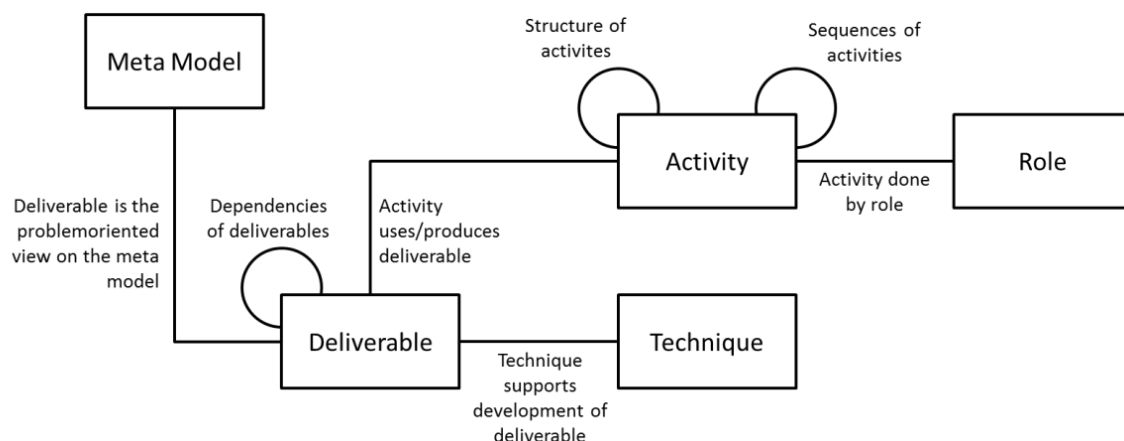


Figure 5: Method components (translated from Gutzwiller, 1994, p.13)

The components are shortly outlined in the following table, whereas for the description of the operation modes the met model will not be part of it:

Activity	An <i>activity</i> is a type of unit which seeks to produce one or more defined result <i>activity</i> is a type of unit which seeks to produce one or more defined result
Role	The activities are being done by people or group of people in certain <i>roles</i> (also called “actors”)
Deliverable	Deliverables are <i>results</i> from the activities
Technique	<i>Techniques</i> are tools to support the creation of the deliverables
Meta model	The <i>meta model</i> is the conceptual data model of the deliverables

Table 2: The five components of a method description

4.2 Operation Mode Overview

The operation field whereas the operation modes can be positioned are between the limits of disruptive innovations and incremental innovations either by teams which are strongly connected to the daily operative processes (fully integrated in organization) or which have an independent working mode (disconnected from organization). Both extremes are visualized in Figure 6. Furthermore the size of the teams matters. Whereas small teams are more agile in terms of readjusting their basic direction, large teams have more power to work on big scale projects.

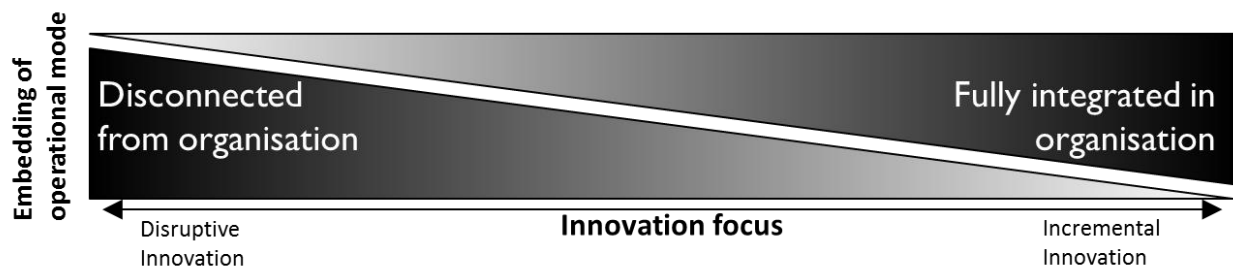


Figure 6: Process embedding of innovation team

The overall process can be split into three operation modes: the *Design Thinking Mode*, the *Initial Development Mode* and the *Fully Integrated Mode*. Each mode consists of one or more sprints to structure the activities of the project team. Each sprint starts with a sprint planning meeting to fill the sprint backlog with respective tasks, define their definition of done and make sure all activities can be finished during the sprint. Each sprint ends with a retrospective. Table 3 provides an overview of the three described operation modes.

	Design Thinking	Initial Development Mode	Fully Integrated Mode
Description	Teams are focusing on developing one clear product vision and a functional prototype to make it tangible (besides a high amount of other low-resolution Prototypes) in order to envision one specific idea	Teams are focusing on different single aspects of the product vision and develop high-resolution software prototypes to further clarify and validate the idea	Straight forward software development process with Design Thinking loops in case of blockers or additional ideas
Qualification	Design Thinking Low-resolution rapid prototyping Milestone understanding sensitivity for result hand-over	High-resolution prototyping Sprint understanding Understanding of implementation context	Design coach: detailed overview of Design Thinking tools handling of ad-hoc Design Thinking activities Experience in working with software developers Development Team: Scrum experience
Risk / Level of Ambiguity	High	Medium	Manageable
Suggested Sprint Duration	1-2 weeks	2-3 weeks	2 - 4 weeks

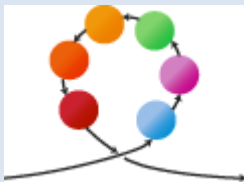
Deliverables (also prerequisites for following mode)	Clear product vision Functional prototype matching the vision Set of high-priority user stories Quantity of low resolution prototypes Documentation of interviews, prototypes, insights Derived requirements (low level of detail) in a Product Backlog	Detailed high-resolution software prototypes of different aspects of the overall software concept Refined user stories and refined Requirements (high level of detail) as base for the product backlog	Software increments Deployment concept Final software product
Embedded Level of Team	Start-up / Intrapreneur	Project	Project
Pattern			

Table 3: Operation Modes Overview

In addition to the different activities of the three operation modes the sprint provides another structuring aspect, in which DT@Scrum activities are planned and executed. The Scrum methodology proposes Sprint Planning Meetings to decide which tasks and activities are covered during the sprint and ensure they can potentially be finished in time. During a sprint, daily stand-up meetings keep everybody informed about their teammate's current tasks and potential blockers. At the end of each sprint retrospectives serve as a validation and adjustment vehicles. What is especially important is the prioritization of the different deliverables, which are being worked out through the sprints. The combined approach of Scrum and Design Thinking can help to decide on a more human-centered base which priorities are being chosen for the next sprint. Furthermore, a planning for Design Thinking activities during a sprint is a new technique that provides the potential to make the creative process more transparent and predictable for teams and management, alike. We henceforth call this type of planning *Design Planning*.

The roles Design Thinking Team and Scrum Team are central in the DT@Scrum approaches and the authors see three different possible team settings. There could be a single Design and Scrum team gradually moving from ideation and prototyping to development of the resulting product. Or the Scrum team could gradually emerge from the Design team as the project moves forward to more development and less design thinking activities. Finally, the Design Team and the Scrum team could be two entirely different teams. Regardless which model is ultimately chosen, DT@Scrum needs to cater to the fact that teams can also change in composition even without an explicit intention to do so. People leaving the company or being reassigned to other projects should not impede the implementation of a product idea.

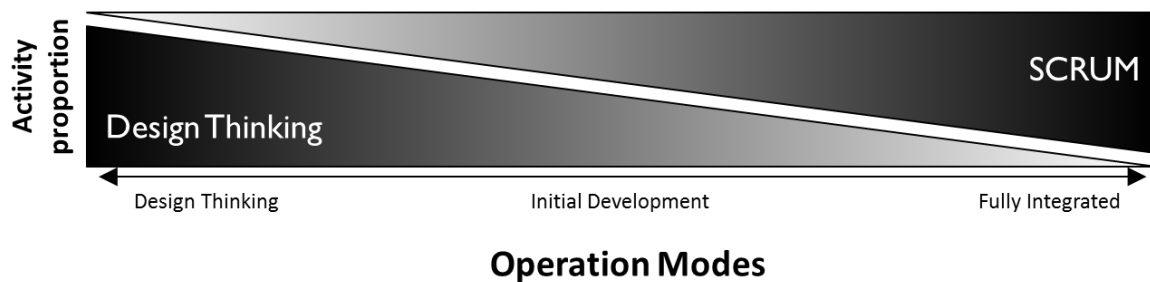


Figure 7: Balance of Design Thinking vs. Scrum Activities within the operation modes

Deliverables of the development sprints play an important role in reaching this target. Each of the described operation modes creates certain deliverables that serve as input for the following phases. Some of the deliverables are persistent and will be updated with each iteration, while others are transient and can be thrown away after usage. Table 4 provides an overview of the deliverables. Regardless of their transient or persistent nature, all artifacts should be captured, ideally in a digital format, to enable teams to understand former Design decisions. This is especially important when team composition changes, but also a valuable aid if the core team remains intact. Tools that support this documentation process in a lightweight manner are currently subject to intensive research (Leifer & Meinel, 2012).

	Output from	Input to	Persistent /Transient
List of Stakeholders and Target Users	Research	Observe	Transient
Interview Guide	Research	Observe	Transient
Information, Insights and Observations	Observe / Testing	Synthesis	Both
Point of View	Synthesis	Ideation	Persistent
Prototypes	Ideation (Observe)	Testing	Both
User Stories	DT Mode / Initial Development Mode	Fully Integrated Mode	Persistent
Product Backlog	DT Mode / Mixed Mode	Fully Integrated Mode	Persistent
Software Increment	Fully Integrated Mode	Fully Integrated Mode	Persistent

Table 4: Overview of persistent vs. transient deliverables

The next explanations focus on the three operation modes, which are being proposed within this whitepaper. In every section the subcategories described in Table 2 are the building blocks of the operation mode. The three operation modes are complementary, which means that it is possible to start from the design thinking mode which is concentrating on a design space which is highly unknown, proceeding to the Initial Development Mode for detailing of single aspects of the solution and finalize the complete journey with the Fully integrated mode whereas Design Thinking helps to overcome potential blockers during the development of a market ready software.

4.3 Design Thinking Mode

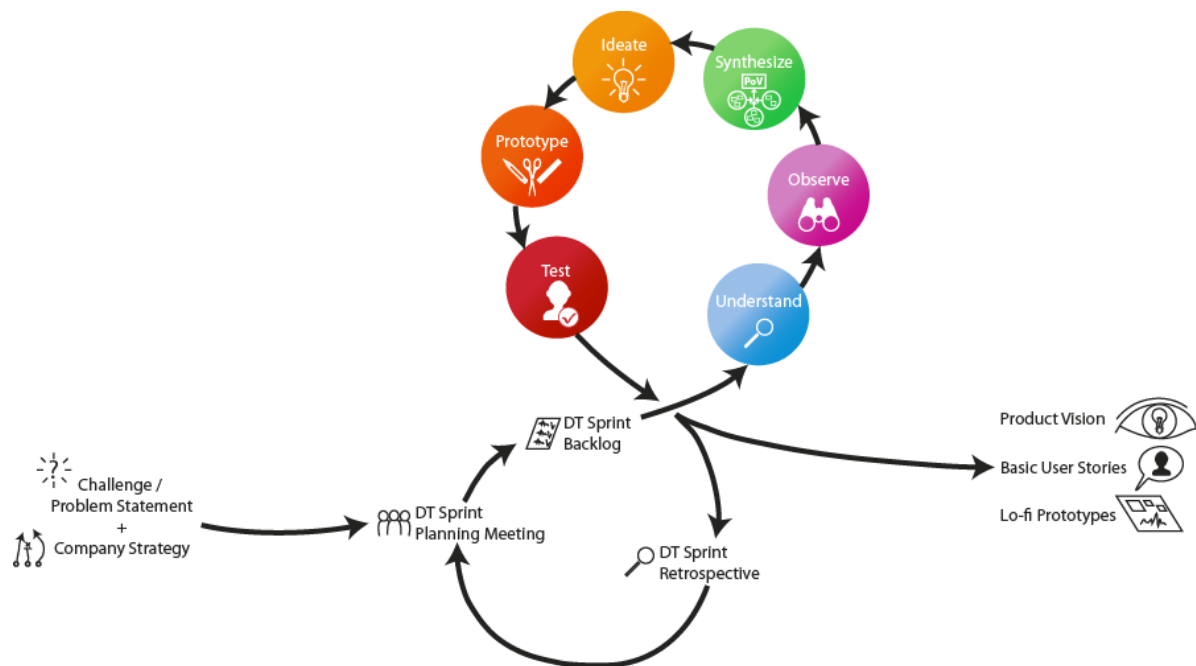


Figure 8: Design Thinking Mode

The Design Thinking Mode uses Design Thinking milestones (compare Figure 3) to explore the project's problem statement and the solution space. During this mode the project team will refine the problem and develop a product vision. The development of a large amount of low-resolution prototypes (mainly in the diverging phase) which result in one functional prototype (in the converging phase) are the main outputs.

4.3.1 Activities

The foundation for the activities of the Design Thinking Mode is the milestone concept (see Appendix), which is being applied in the global ME310 projects or Embedded Design Thinking (Vetterli et al. 2012). In purpose of combining Design Thinking and SCRUM the milestone concept has been adapted and is visualized in Figure 9. The concrete prototypes shown in Section 2.3 also serve a didactical purpose, which may not be required in enterprise settings. Hence, we distilled their core ideas and captured them in this graphic. Teams should ensure that – no matter how long their initial Design Thinking phase ultimately lasts – they created prototypes for each of the presented categories. That means, one showing the critical

function of the developed product (CFP), one that was deemed “un-implementable” or too crazy to pursue (Dark Horse), one that shows the general concept in a rough and easily changeable fashion (FKP) and a prototype that has a rather high resolution and provides both the look and feel of the target solution.

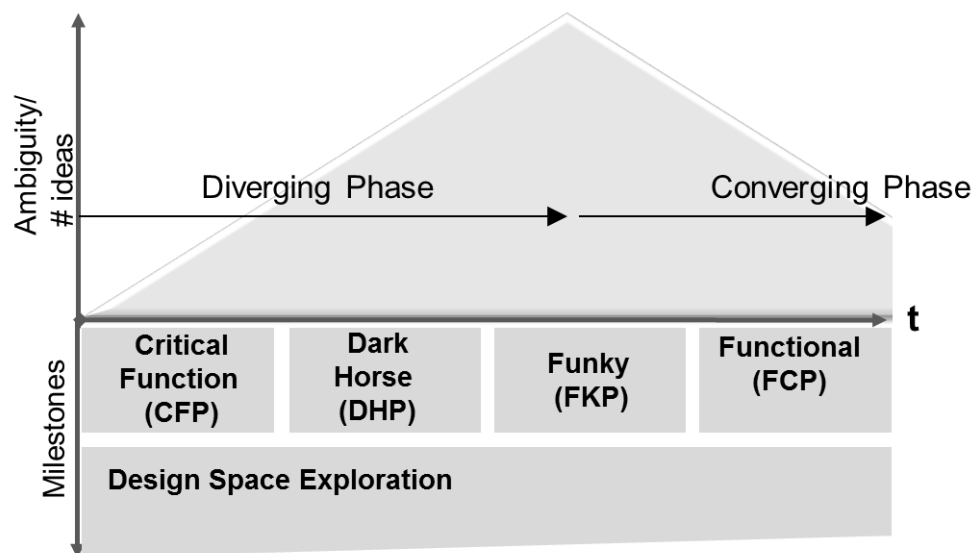


Figure 9: Design Thinking Milestones for the Design Thinking Mode

Before prototyping start, a research plan needs to be created to have a clear overview where and how specific stakeholders are being approached. To cover the essential information, the interview guide serves as outline for interviews. The interview guide is a transient deliverable since the questions will have a different focus in each iteration, but it should of course be captured to later comprehend how certain requirements evolved. The information, insights and observations derived from the interviews are crucial to user-centered development. As such they should be carefully synthesized to uncover hidden correlations and important aspects in the field of interest. The uncovered insights serve as an accompanying deliverable during the following phases and therefore are persistent after a certain maturity level. They should additionally been documented in a backlog to preserve them for the upcoming activities. The converging phase then serves to create one or two functional prototypes, which provide some working functions in it to deliver the best experience possible of the prototype.

Parallel to the described activities, the deliverables of the design thinking activities need to be prepared for usage in the upcoming iterations that belong to the Second or Third operating mode. On one hand the ideated ideas / low-resolution prototypes, user stories as well as the functional prototype are being transferred into the Initial Development Mode. On the other hand a documentation of the non-functional requirements on a high detailing level as well as the description of the functional requirements (backlog) on a middle detailing level should be created. In case teams change from one mode to the next, the probability to create a successful handover can be raised through workshops to deliver the insights, which led to the functional prototype as well as the outlining of the documentation.

On artifact level, user stories are a tool which helps on the design thinking as well as on the development side to be clear of which experience in which form should be delivered at the very end. The user stories build the overall frame of design that integrates the context, the incentives, and the testing environment. A User Story represents a minimal Software requirement often in the form " As a [type of user] I want to [perform some task] so that I can [reach some goal]". User stories on one hand serve as a technique to discuss certain target context for the potential solution, on the other hand it should be defined as a deliverable which accompanies the following development process as orientation point and provider of the main idea. This deliverable is persistent. The collection of all requirements / User Stories for the software is being done within the Product Backlog. As such it provides the development team with an overview of all necessary features. The Product Backlog is a persistent deliverable and is constantly filled, altered, and adopted during each sprint.

The following *Prerequisites* should be present for the sprints within the Design Thinking Mode:

- Company strategy + problem statement
- Access to potential users and other stakeholders
- Design thinking training for the team members

It highly depends on the target product, at which resolution the Design Thinking mode can be left and the teams should start to implement high-resolution prototypes that additionally showcase the feasibility of technical challenges.

4.3.2 Roles

Design Thinking Teams

Design Thinking defines no special roles for an innovation team, but it is recommended to establish *multidisciplinary teams*. As team members with different backgrounds and mindsets will inevitably have different perspectives on the problem, a multidisciplinary team minimizes the chances of missing an important aspect of the problem. In corporate environments the establishments of multidisciplinary teams is very challenging. Practice has shown that environment factors which enable creative working style (working infrastructure, communication behavior, etc.) assures appliance of design thinking on the same level. Additionally, research has shown that sensitizing the teams in terms of which aspect of multidisciplinary is missing in their teams, enables teams to compensate those gaps. The Design Team is responsible for planning and executing the design thinking sprints. A Design team will usually consist of 3-6 people from different areas of expertise as needed for the software under development, e.g. Accounting, Sales Persons, UI Designers, Developers, and Consultants. Ideally, at least one team member who took part in the Design Thinking phase should still be present within the development phase, for example, as the Scrum teams' Product Owner. This role will know about the learnings and insights gathered during the Design Thinking mode sprints and can understand why certain aspects of the functional prototype have been integrated.

(Potential) Users

The users need to be referred to for interviews during research phases. At the beginning of the process, different types of users should be interviewed and/or observed to identify important pain points within the problem space different user groups might have. Further along the process when the team reaches a decision about a target user or user group, the team

should try to secure one or more users of that target group to gain constant feedback in the iterative developing and testing cycles. The User's main task is to be observable in the natural context of the problem space, provide input on the topic and his problems and to give feedback on ideas prototypes and the direction of the process.

Design Thinking Coach

In Design Thinking processes teams are often supported by Design Thinking Coaches. The responsibility of the coach is to moderate discussions, ensure the team is focused on its task and to moderate team dynamic issues, such as conflicts or motivational issues. For the described process this would also be useful during the development phase and thus a process master / process coach should be introduced, that will take over the mentioned responsibilities throughout the process. A process coach might not be necessary the whole time; the authors see the following possible setups:

- The process coach is available to the team at all times and will be called in by the team, if s/he is needed.
- The process coach is with the team at certain times during the process, e.g. once a week, before important meetings, after important meetings.
- The process coach is with the team whenever they are working in the team working space but not at interviews or testings.

The coaches play an important role since they have a methodological process perspective and should be aware of political issues affecting the activities.

Corporate Liaison / Project Sponsor

Corporate Liaisons have a strong interest in the project as they represent the group, which defined the initial problem statement. In the discussed corporate setting, the liaison is the manager who is overlooking the project and serves as primary contact point to the design or development team. The manager also is the main contact point to the Project Sponsor, which has chosen the problem statement referring to strategic decisions of the company. It is crucial

that the manager received Design Thinking training to understand the peculiarities of the methodology and is enabled to support the team as good as possible in their efforts.

4.3.3 Deliverables

This mode generates different low-resolution prototypes as well as one functional prototype, which ideally are being transferred from the Design Thinking mode to the Initial Development Mode. The derivation of non-functional and functional requirements for the development of the product and the generation of an initial set of User Stories are deliverables as well. All the insights should generate a clear solution vision and elaborate why the aspects of the functional prototype has been designed in a certain specific form. The goal of documenting non-functional as well as functional requirements in a documentation is to understand the evolution of certain prototypes. Finally, the teams needs to fill the product backlog with a set of high-level user stories that describe the core functionality of the intended system and allow more detailed prototypes in the upcoming initial development phase.

4.3.4 Techniques

Scrums as well as the Design Thinking methodology include various techniques for different purposes; this section will give a brief overview of techniques and their usage. For a more detailed description of the different techniques refer to Appendix A.

The Design Thinking methodology knows several techniques that help understanding the project environment, the stakeholders, the users and the design space: the 360° research allows to quickly get smart on a topic, user observations and interviews enable the team to understand the user needs and pains, extreme users can help to get a different perspective on the challenge while stakeholder maps enable the team to grasp who is involved in the topic.

When team members work on different tasks or different activities, short Stand Up meetings help to keep everybody up to date. If it is necessary for other team members to get a deeper understanding of what was achieved or to prepare synthesis after research interviews, observations, or testings, storytelling is a good method. Afterwards different synthesis techniques

like clustering or creating a Persona, a Point of View Madlib, or a 2-by-2 Matrix can help to discover or convey insights and findings. When the team has found its current point of view on the challenge, brainstorming on possible solutions generates ideas, which can then be prototyped. During this phase, prototyping is used to understand the users and the challenge, as well as to quickly validate ideas and possible solutions. Therefore, rough prototypes that are fast and easy to build are best for this purpose. This includes cardboard or paper prototypes, for example, of hardware components, sketches of user interfaces, or even role plays of a situation. Testing the prototypes with actual users is essential to understand flaws of the current solution and discover further needs and pains. Testing can be done by observing the user while he is trying out everything and interviewing him afterwards.

4.4 Initial Development Mode

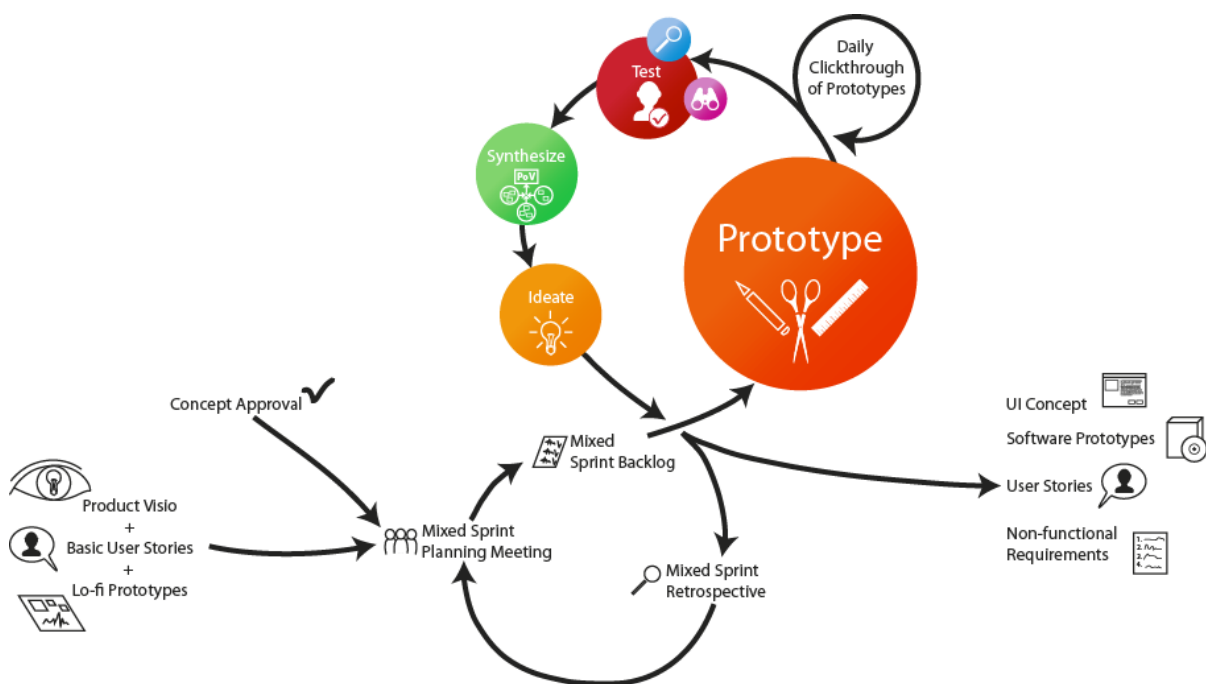


Figure 10: Initial Development Mode

The Initial Development Mode focuses on the different aspects of the functional prototype created during the former operation mode. The detailing of the single aspects to gain high-

resolution prototypes for the different aspects is the main goal. It includes the clear specification of integration in the intended context of appliance. The refinement of the user stories with non-functional and technical requirements is another key aspect of this operation mode. The handover of the different high-resolution prototypes and the learning out of it is being transferred into the software engineering mode whereas software increments are being created step by step.

4.4.1 Activities

The preceding Design Thinking Mode is defining the main activities in this mode since a functional prototype resulted as one of the final deliverables. Detailing is now at forefront of activities in terms of identifying single design aspects out of the functional prototype. These single design aspects are being developed and tested within SCRUM sprints and result in single high-resolution prototypes each being part of the overall software concept. All prototypes should be tested with target users for maximum user satisfaction, the information gained from the testing, should be used to further refine the implementation in the next sprints. During this mode the team can test the UX concept with software prototyping tools and explore the technical feasibility of required functionalities with first development efforts. The complete software concept with its single high-resolution prototypes should be already tested as integratable in the intended context of application.

As outlined already in the Design Thinking Mode, the Product Backlog is the central part of documentation of all the findings during the single sprints. Hence, refinements of high-level user stories that are already part of the product backlog are required during Initial development mode. Furthermore, the increasing fidelity of the prototypes likely uncovers new non-functional requirements that should be added to the Definition of Done of the respective user stories.

Sprint Prerequisites and Deliverables

Prerequisites for the sprints within this Initial Development Mode:

- Company strategy + problem statement
- Access to potential users and other stakeholders
- Design thinking and SCRUM training for the team members
- Clear Product Vision
- Initial set of high-level User Stories
- Functional Prototype that resulted from the DT phase
- Pool of low resolution prototypes

4.4.2 Roles

Scrum Master

The so-called Scrum Master makes sure that the Scrum team follows the process structure. For our merged process the Process Master will also be responsible for the team's adherence to Design Thinking tasks, as such he would be a Design Thinking Master and a Scrum Master as one person.

Product Owner

The Product Owner is the representative of the customer. He is responsible for filling the backlog with user stories and for prioritizing them.

Scrum Team

The Scrum Team is responsible for the planning and execution of the development sprints. A Scrum team usually consist of 8-10 developers drawn from areas of expertise as needed for the software under development, e.g. Backend developers, front end developers, database experts, UI developers, etc.

(Potential) Users

In this mode the users play a central role especially in terms of testing the software increment during test phases. As well as in the previous mode the users should be identified during the low-resolution prototype testing phase of the diverging phase.

Corporate Liaison

Referring to 4.3.2., the corporate liaison is the strategic partner of the Diverging Design Thinking Team and the SCRUM Team and therefore serves as one core stakeholder of the projects. It should also facilitate communication with other sections of the company to avoid duplication of efforts, enable synergetic effects between teams, and allow reuse of existing software.

4.4.3 Deliverables

The deliverables in this mode are strongly focusing on the software prototypes that need to be delivered. Therefore one key deliverable are the *UX* prototypes tested with end-users which are being based on the non- functional and functional requirements derived from the refined user stories. Additionally the *Backend Spikes* created during the sprints to show feasibility of required functionality and the documentation of prototypes and feedback in the backlog are deliverables of this mode. Finally, a Clear specification of integration within the company context needs to be created. This includes identifying dependencies to other, already existing systems or potential for reuse of existing software components in the final implementation.

4.4.4 Techniques

The following techniques are an extract of the core techniques needed in the Initial Development Mode:

Lo-fidelity UX prototyping techniques

To create fast UI prototypes simple sketches on paper provide a great tool. They can be used to test different arrangements of the content, the navigation between different pages or interaction concepts. Paper prototypes can also be "interactive" during user testing if one person

of the design team manually changes the UI by drawing additional content or adding and moving pieces of paper around. The following Figure 11 shows an interactive UX paper prototype, that uses different prepared sheets of paper as widgets inside the application window, post it's for highlights and overlays and also lets the user draw and write inside the application window.

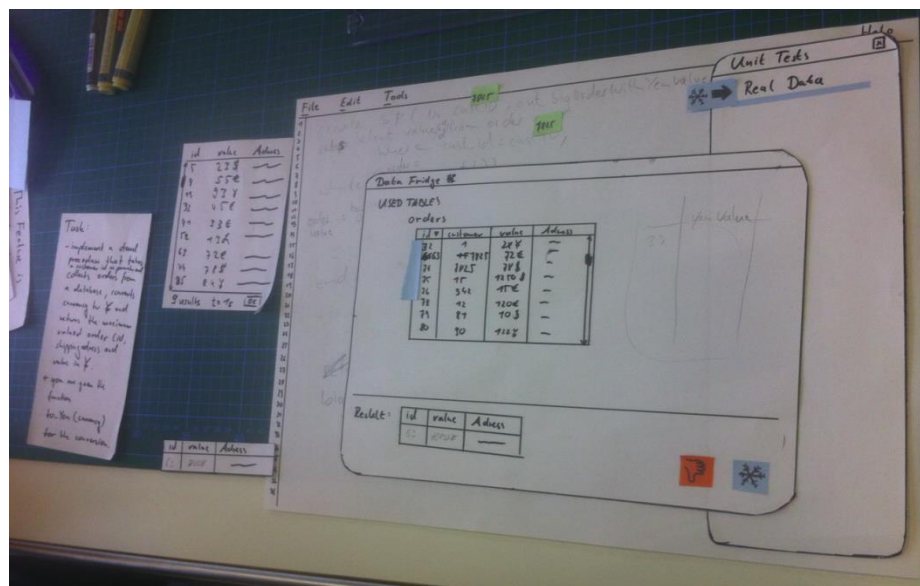


Figure 11: Interactive Paper Prototype

Mid- and Hi-fidelity UX-Prototyping techniques

Wire framing is a technique that uses simple sketch like widgets and controls to build a user interface prototype. Various tools like pidoco¹ or gomockingbird² exist, that provide the user with a variety of building blocks to build screens or even clickable prototypes. The following picture illustrates the pidoco user interface and a simulation of the designed page.

¹ <https://pidoco.com/>

² <https://gomockingbird.com/>

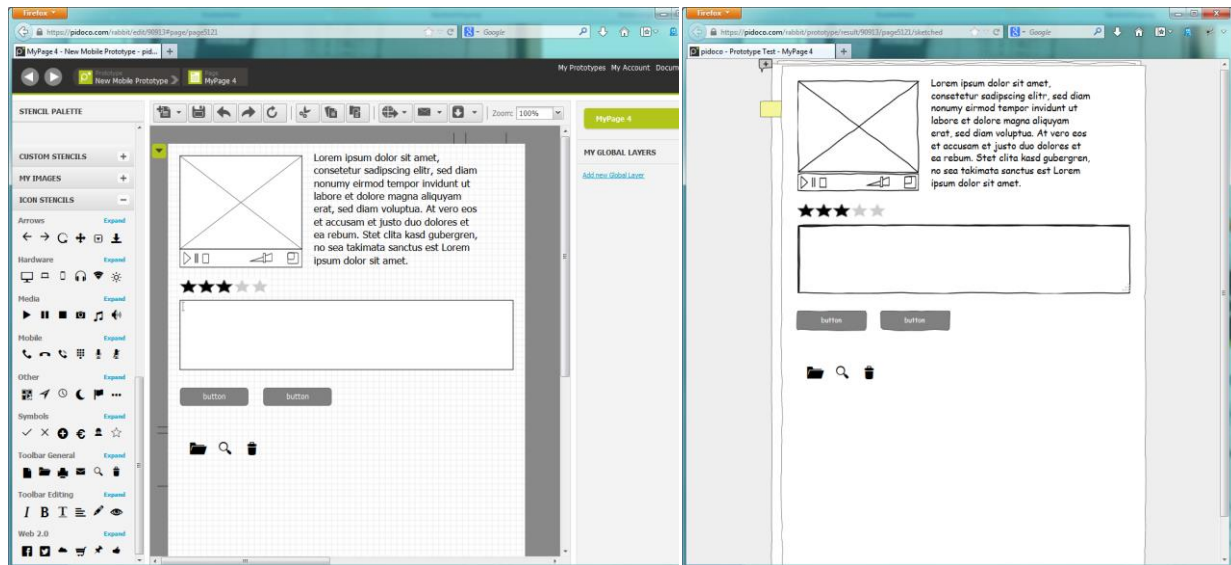


Figure 12: Pidoco Workspace (left) and Pidoco Simulation in Sketch Mode (right)

In cases where a more sophisticated UI prototype is required (e.g. to discuss progress with management) tools like Keynotopia³, which enable clickable Keynote / PowerPoint UIs or fast HTML prototypes can be used.

User story mapping is a technique that helps teams to understand the functionality of the system under development, identify holes and omissions in a backlog, and plan releases that deliver value to user and business. The User Story Map arranges the main activities from left to right in an order that makes sense, e.g. in a workflow. Task centric User Stories are also arranged from left to right under the activity they belong to. Tasks that can happen in parallel will be placed vertically under one another.

³ <http://keynotopia.com/>

4.5 Fully Integrated Mode

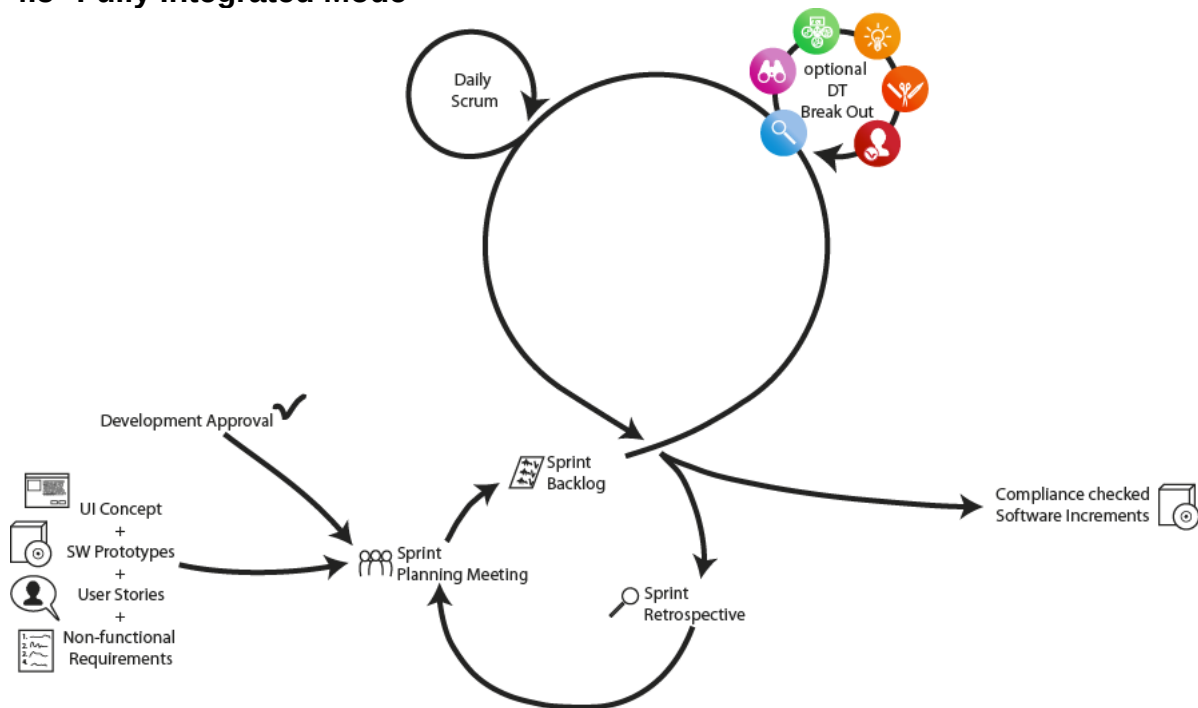


Figure 13: Fully Integrated Mode

The Fully Integrated Mode mainly complies with a scrum development process, enabling the team to work towards a final product in incremental steps. In case of blockers in the development process design thinking tools will be initiated, hence the design thinking appliance is ad-hoc and a close observation of the development process is needed to quickly react with the adequate design thinking tool. The roadmap for the development has been generated in the Initial Development Mode.

4.5.1 Activities

The activities are based on the software development / engineering method and follow the Scrum pattern which has been implemented. During this mode the team focuses on development of software increments as well as deployment and maintenance concepts. In case features are not well enough defined or problems arise the team can choose to include short Design Thinking bursts in the activities to refine the feature idea or find solutions to the problem. Thus Design Thinking in this mode, compared to the other two modes, is not focusing on creating insights in front of the software development process. It is rather creating ad-hoc insights and different solutions to overcome some impassable blockers.

The sprint duration in the Fully Integrated Mode is being geared by the standard development cycle and should therefore last between two to four weeks, depending on the team's preferences.

Prerequisites for the sprints within this Fully Integrated Mode are:

- Clear Product Vision
- Prioritized list of detailed user stories
- List of Non-functional requirements (see PIL Product standards)
- Documentation of proof-of-concept implementations

4.5.2 Roles

Scrum Master

The so called Scrum Master makes sure that the team follows the process structure. For our merged process the Process Master will also be responsible for the team's adherence to Design Thinking tasks, as such he would be a Design Thinking Master and a Scrum Master as one person.

Product Owner

The Product Owner is the representative of the customer. He is responsible for filling the backlog with user stories and for prioritizing them.

Scrum Team

The Scrum Team is responsible for the planning and execution of the development sprints. A Scrum team usually consist of 8-10 developers drawn from areas of expertise as needed for the software under development, e.g. Backend developers, front end developers, database experts, UI developers, etc.

Design Thinking Expert

The Design Thinking Expert manages a great repertoire of tools, which help the development to team to quickly react if blockers are stopping the development process. The expert helps

the team especially to create totally different perspectives to generate new ideas. In this mode the responsibility of the coach is as well to moderate discussions.

(Potential) Users

In this mode the users play as central role in terms of testing the software increment during test phases.

4.5.3 Deliverables

The Fully Integrated Mode focuses on creating working and implemented software. Hence, all developments should be potentially shippable by the company. This includes that the software adheres to certain product standards (see Section 5.25.2 about details how to achieve that in an agile manner) and is deployable. That means, the teams should also create, or at least keep in mind, a strategy of how their software can be delivered to the end user. With regards to mobile apps, this is rather straightforward, but when developing on-premise software that integrates with existing landscapes, the team needs to explicitly reserve time to create a deployment strategy. Finally, developers should not only blindly implement the given stories but be open-minded about potential improvements. Hence, their own ideas or suggestions should also be captured, and, if applicable, be transformed into user stories for upcoming sprints.

4.5.4 Techniques

The specific techniques of this mode refer to software development techniques and in this case especially to Mid- and Hi-fidelity UX-Prototyping techniques and Lo-fidelity UX prototyping techniques (compare 4.4.4). The companies' standard development tools are to be used along with potentially beneficial technologies that have not yet been employed.

5 Integration of DT@Scrum in Organizations – Initial Ideas

DT@Scrum strongly depends on the context, which is surrounding the process. As shown in Figure 6, the embedding of the project team within the operational processes of the company, strongly influences the innovation focus.

Depending on the knowledge of the design / development teams concerning the design space or the solution, the adequate choice of the entry point needs to be discussed. As the operation modes outlined already are three operation modes that define one software development journey moving from high ambiguity to manageable ambiguity and from little knowledge about the design space to very deep knowledge of the design space. In organizations different levels of knowledge of design spaces can be observed and therefore project leader have to decide which entry point is the most adequate for a specific project. The following visualization will show the two most important versions of project entry points and the consequences on the antecedent and the following Mode.

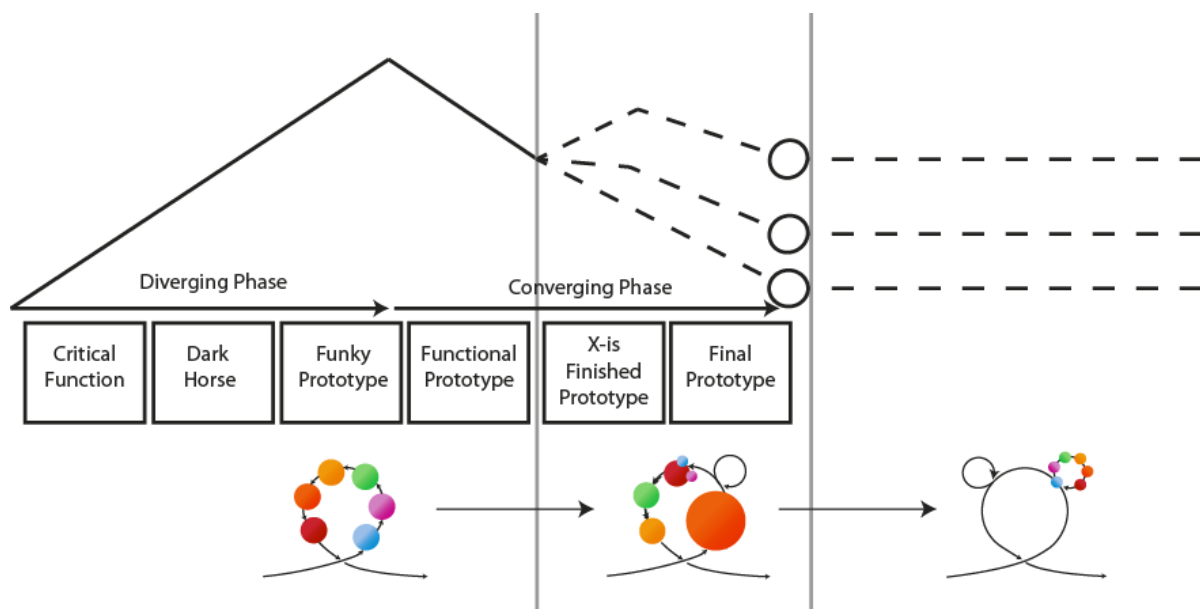


Figure 14: Entry Point with no relevant knowledge of the design space

The visualization shows the situation when no relevant knowledge of the design space is given and concrete ideas for possible solutions are missing. Hence the operation mode Design Thinking Mode would provide a entry starting point to gain a massive knowledge of the de-

sign space and come up with concrete ideas and at least middle resolution functional prototype to deliver a clear product vision. Additionally this entry point would provide a documentation as well as the Backlog which is needed to pursue with the next operation mode Initial Development to detail the single aspects of the found solution. As it is visualized the Design Thinking Mode would take about 60% of the total time.

The second main entry point is being visualized in following figure:

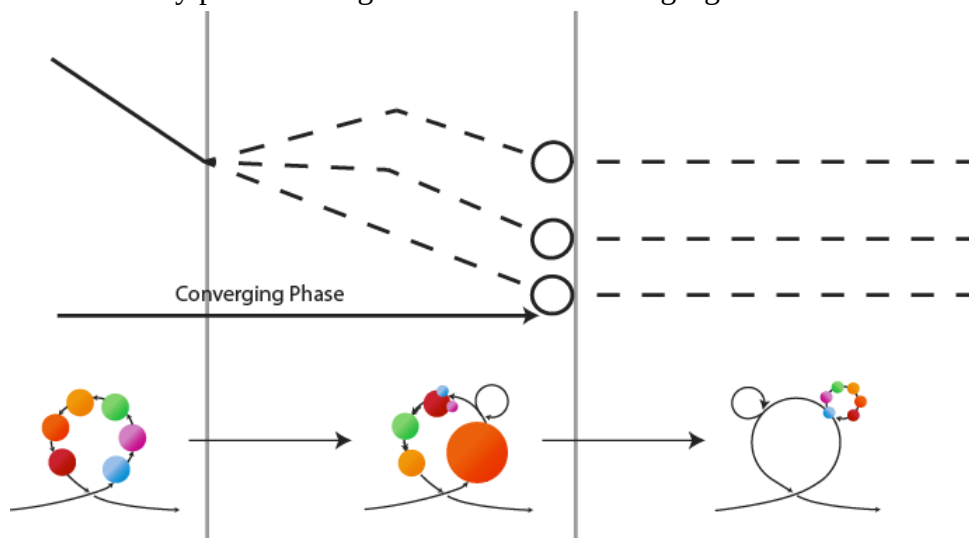


Figure 15: Entry Point with existing knowledge of the design space and given solution ideas

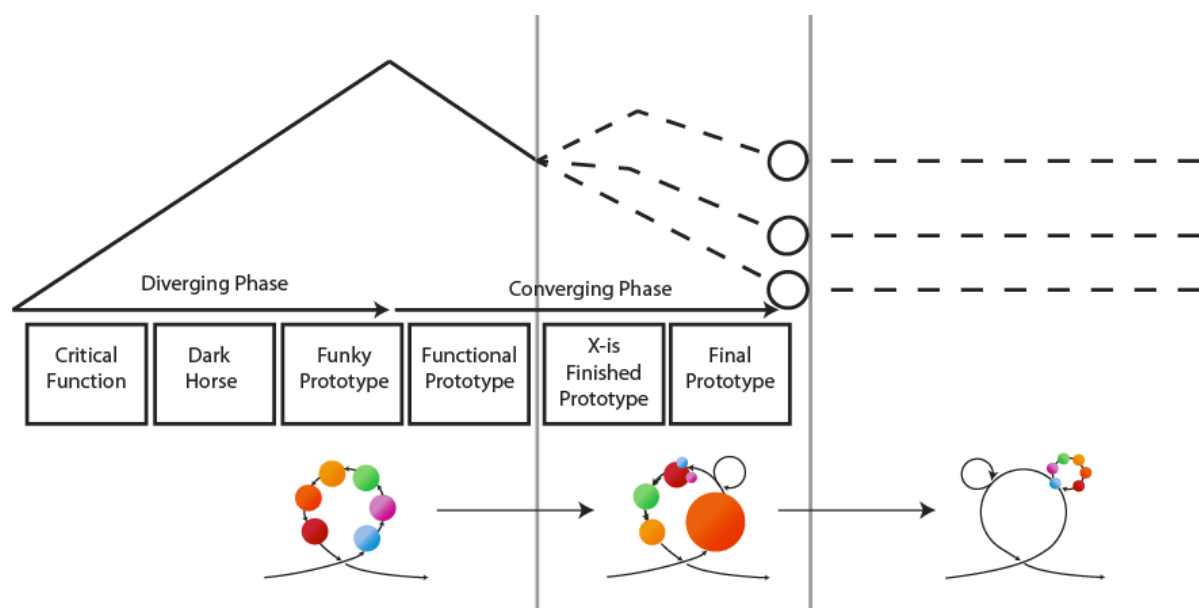


Figure 16: The rapid converging of the given ideas to one concretized functional prototype

The situation here is different in terms of knowledge of the design space and already given ideas for solutions. Therefore there is no need of pursuing all the milestone phases like in leads directly to the Initial Development Mode whereas the intensive detailing of the single aspects in terms of high resolution prototypes serves as focus of activities. The length of the Design Thinking Mode is much shorter in this situation and more time is left for the specification of the single components.

Although the context of implementation is decisive from a project perspective, the importance of managing all three operation modes is crucial for a sustainable strategy in developing outstanding software regularly from an overall project portfolio perspective.

The following figure will demonstrate the knowledge flow from one mode into another and how this is linked to completeness of projects respectively total knowledge about a certain design space, because at the end of the day this is nearly the only process step which counts for business.

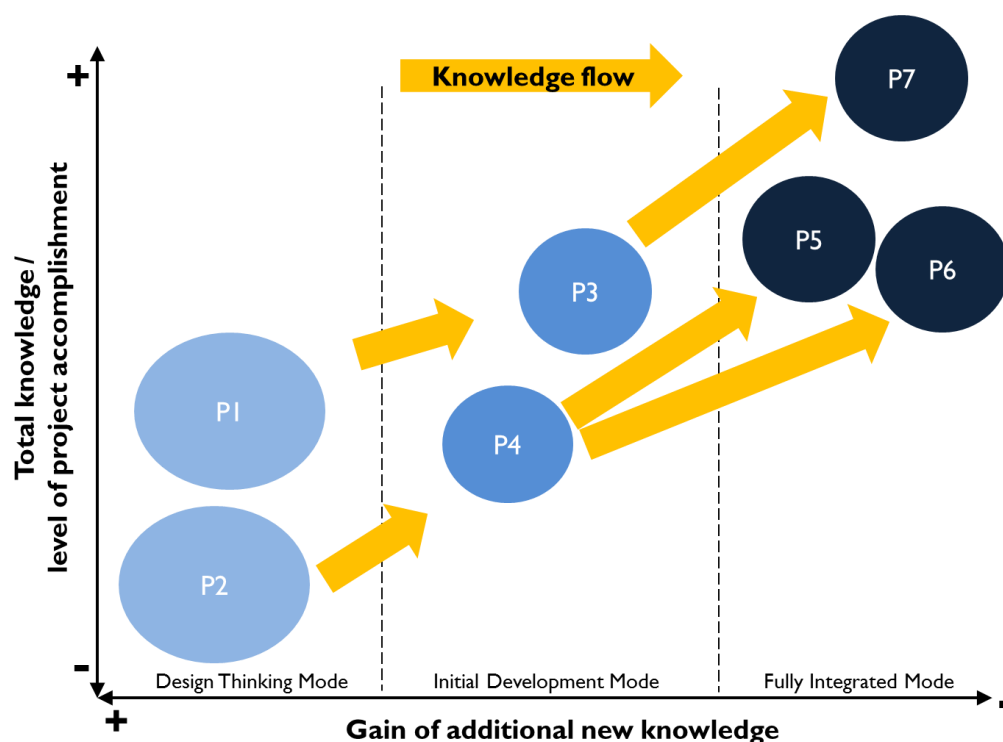


Figure 17: Knowledge Flow in Project Portfolios

The x-axis visualizes the three operation modes whereas the y-axis stands for the total knowledge and the level of project accomplishment. The Bubbles within the visualization stand for different Projects (P_1 - P_7). The amount of gained new knowledge about a design space is bigger in the Design Thinking Mode (DT) than in the Fully Integrated Mode (Fully) due to the intensive Design Thinking activities which are related to this mode. Therefore the knowledge flow goes from the Design Thinking Mode to the Initial Development Mode (Initial) until it reaches the Fully Integrated Mode. The total knowledge reaches the highest level at the Fully Integrated Mode due to the continuous knowledge flow which needs to be maintained from previous modes in a well-managed project portfolio. From a project portfolio perspective the level of project accomplishment needs to be at maximum level as well in the Fully Integrated Mode.

In the following, we discuss potential challenges for the integration of DT@SCRUM into organizations like SAP. Two main challenges arise. Firstly, large companies need to coordinate hundreds of development teams in parallel and ensure interoperability between the different teams. If two teams within the company need to collaborate with each other within a project, general guidelines for roles, artifacts, and techniques need to be in place to minimize the communication overhead between the teams.

Secondly, all of SAP's products need to fulfill a multitude of non-functional requirement in various areas. This includes performance, accessibility, documentation, deployment, globalization, security, usability, usage of third party and open source components and information lifecycle management.

We outline how our process model could incorporate these non-functional requirements without placing large bureaucratic burdens on the development teams. Furthermore, we show that the defined roles, deliverables, and techniques are a means to foster interoperability between different development teams within SAP.

5.1 Large-scale Projects

Larger software projects like the development of a complex ERP solution require a large number of developers possibly split into several development teams. Solutions to solve this problem already exist, for example the Scrum of Scrums or Meta Scrum. In this technique the individual Daily Scrum of all Teams is followed by a Daily Scrum of Scrums with an ambassador from each team, who will give a progress report from his team and take back important information to his team members. If necessary this technique can be used on multiple levels. Ambler (Ambler, 2009) or Larman and Vodde (Larman, 2008) (Larman, 2010) give examples and case studies on how agile processes can be scaled for large project teams and explain appropriate techniques.

It is the authors' point of view that a similar scale up of design teams for the Design Thinking and Initial Development Modes would not be helpful. Instead, a regular Design Team of four to eight people will work on the project during the Design Thinking Mode. During the Initial Development Mode the team can split into multiple mixed teams working on different projects that follow a product idea from the DT Mode or the Design Team can split into multiple sub teams working on parts of the product vision derived during the DT Mode in one project. The teams or sub teams will then hand over their concept and requirements to or evolve into even more development teams, which will facilitate the development modes.

The following figure illustrates the flow of project teams during the modes:

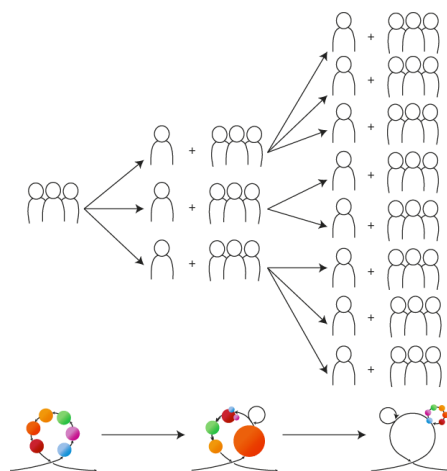
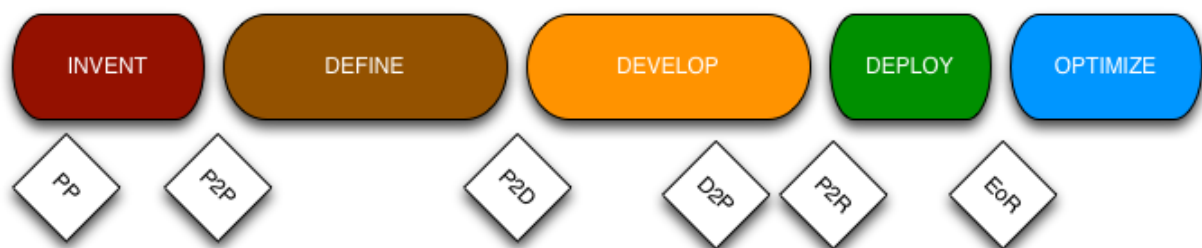


Figure 18: Team Scale up During Operation Modes

5.2 Product and Process Standards Compliance

To account for the previously mentioned product standards that its software needs to fulfill, SAP introduced the Product Innovation Lifecycle (PIL) and its successor, Idea-to-Market (I2M), as a means to guide development teams in their efforts. In a nutshell, PIL and I2M define Quality-Gates (Q-Gates) at certain points within the development process to determine whether or not development can be taken to the next phase. By that, it is sought to ensure that products comply with legal regulations, SAP-internal product standards, and general product quality measures (e.g., ISO regulations) before being handed over to customers. An overview of PIL phases and the ensuing Q-Gates is shown in the figure below.



*Figure 19: Product Innovation Lifecycle phases (rounded squares)
with quality gates (rhombi)*

With the Q-Gate mechanism, PIL and I2M try to ensure comparable quality of all software projects at the end and beginning of the respective phases. In-between Q-Gates, the teams are self-responsible for maintaining the quality standards. By that, an inherent risk of ignoring some standards, either intentionally or as a result of lack of knowledge, is present within the projects. This can result in increased overhead late in the different development phases, e.g. through refactoring, code reviews, or documentation.

5.2.1 Continuous Compliance Model

Hence, we propose to supersede the stationary Q-Gate model and introduce a process that promotes continuous quality awareness. The definition of done (DoD) for tasks within the Scrum process plays an important role in this model. This simple, yet highly efficient, instrument adds further, non-functional constraints to the user stories and prevents teams from

declaring “it works” before all criteria in the DoD are fulfilled. But how can we balance the plethora of product standards with the requirement for agility in the development process?

Reviewing the existing PIL Product standards, two basic categories can be identified:

a) Standards that should be automatically monitored for compliance

This include, for example, performance, but also a variety of accessibility and deployment standards. By using automated performance tests based on real or realistic data sets, it should be prevented that any alteration of a project’s source code leads to violation of the given standards. With regards to accessibility, various standards can be monitored by source code analysis (e.g., checking that links should have descriptive texts or their URL on mouse over events). Finally, deployment of the system should grow continuously with the project and become an integral part of the development process. This technique, known as Development Operations (DevOps, see Appendix), has a strong emphasis on automation of application deployment and thereby enables all project members to create running versions of the project at any given point in time.

These ideas currently lack tool support. However, the huge potential for time savings by not being able to violate basic standards without being notified should be incentive enough to drive development of such tools. See also Future Work for an outline of required tools.

b) Standards that need manual review

While some of SAPs product standards can be monitored for compliance automatically, others require manual review either by fellow team members or specialized departments within the company. This includes usage of open source and third party libraries, which require review by legal staff. Documentation and the general development process should be reviewed either by project management experts or other developers. For other standards, such as usability, a continuous validation process is also a desirable solution. Checking whether guidelines are fulfilled upon declaring a user story finished should be mandatory for each develop-

er and not postponed until later reviews as the original developer might not be present later on and larger violations rapidly lead to costly refactoring within the software.

While these standards require manual review, feedback about system parts that need further attention could also be captured using smart tooling. For example, heavily used code parts can be identified in an automated fashion (Panchenko et al., 2009) and should be regularly checked for up-to-date documentation. Similarly, open source licenses can be checked for incompatibilities (Keivanloo et al., 2012).

In addition to the currently missing tool support, general acceptance within the company obviously is a major issue. Continuously tracking whether the systems under development fulfill the given quality standards consumes developer time. As with Test-Driven Development, this approach pays out in the long-run, when only minor or no additional work is required to ship projects to test customers or roll them out as real products. An internal test project that continuously monitors its product for adherence to the product standards is therefore highly desirable in order to provide case-study evidence about the expected increase in confidence about the developed product and the decrease of necessary reworks towards the delivery dates.

5.2.2 Compliance Requirements during Operation Modes

In the following we discuss the impact of the different operating modes on product standards compliance. While activities in all modes should generate “complying” artifacts, the subset of standards that need to be considered differs vastly. This is also an important aspect with regards to the aforementioned tool support. If, for example, an alert about missing translations keeps popping up in early development phases, developers are likely to ignore it later on, as well. Hence, the tools should be able to determine between the different stages and, e.g., not provide alerts for missing translations but for code segments that potentially hinder translation later in the process.

Design Thinking Mode

Software-related standards can be ignored in this mode due to the lack of existing software. It is, however, crucial that decisions made during this phase can be reconstructed in later phases. Hence, it should be possible to trace back resulting software requirements to single interviews, brainstormings, or user feedback sessions. By that, it can be ensured that only features desired by the customer or user will be implemented in the final product.

As the vision of the desired target product becomes clearer, teams also need to be able to specify the nature of the resulting software. Depending on this nature, a set of standards that the *final* product needs to comply with should be chosen. That should free developers of determining which standards potentially apply to their project later in the process.

Initial Development Mode

In creating a software prototype, teams materialize the generated product vision but do not implement certain elements that are not necessary to transport the concept to potential test users. However, as prototype maturity increases, teams will also start to create proof-of-concept implementations for resulting technical challenges (e.g., database queries, interaction schemas, etc.). The experiences gathered during these steps should be documented in a manner that even if the team changes, efforts are not repeated in later sprints. Hence, meaningful documentation standards (not necessarily the existing ones) need to be established.

As the prototypes mature over multiple sprints, the target architecture of the solution starts to evolve. Given the respective elements of the architecture, it also becomes clearer, which necessary (due to legal reasons) and meaningful (due to company policy) standards are applicable to the project. The prototyping phase therefore should be used to refine the chosen standard set of the DT phase.

Fully Integrated Mode

When “real” product development starts, the continuous compliance model discussed in Section 5.2.1 can be employed. Based on the previously identified applicable project standards –

but, of course, also newly discovered ones – the project needs to be checked for compliance at every given check-in. This immediate feedback mechanism enables developers to revert standard violations right away and thereby prevent costly reworks later in the process.

6 Validation & Future Work

The presented approach is a working proposal and therefore does not claim completeness. In particular, the described development model is mainly focused on projects that target consumer applications. Projects with a strong technical focus, i.e., ones that do not try to solve a „wicked“ but an engineering problem are beyond the current scope of our work. Furthermore, we assume that every chosen project idea can be implemented as a software system and solutions that are of a non-technical nature are out of scope in our considerations. In the following, we outline possible future extensions of our proposal and present initial ideas for validating the effectiveness of the process both in academic and in industry settings.

6.1 Possible Extensions

The process presented so far aims to provide an agile software development approach close to the customer. In order to constantly collect feedback from the customer, software increments and the final product need to be deployed and updated fast and in sync with the development sprints. In order to achieve closely coupled development and deployment cycles, the transition from development to operation of the software should be focus of future research.

As described the development phases of the process use User Stories as requirement artifacts. Hildenbrand and Meyer (2012) found that User Story Mapping is a helpful Technique to collect and structure these User Stories when moving from prototyping to development. However the User Stories need to be created without information loss and should ideally build on artifacts from prior phases. Future research will investigate the use of artifacts like the PoV as a basis for User Stories, as well as techniques to create these User Stories from insights and findings.

As described in Section 5 projects in large software development companies face a large set of functional and non-functional requirements for the product itself as well as for the development process. Product development processes in such companies require audits at certain points in the process in order to ensure compliance with these requirements. The authors feel, that a large amount of these requirements could be ensured with the help of software tools,

thus easing the preparation and execution of such audits. For example AnalyzeD (Kowark and Plattner, 2012), a platform developed at the HPI can help to ensure team setups or indicate the adherence-to or negligence-of the software development process by collecting and analyzing artifacts of virtual team collaboration, such as emails, wiki entries, tickets, bug reports or commits. Future Research will investigate the use of AnalyzeD for this purpose as well as other techniques and tools that help to ensure compliance with for example accessibility requirements.

Teams from the Design Thinking Research Program⁴ established at Stanford and the HPI investigate, among others, tools to support globally distributed Design Thinking Teams. Tools like Teleboard⁵ enable distributed teams to jointly work on e.g. synthesizing tasks or brainstormings and, additionally, keep track of information collected throughout the process, thereby allowing tracing the life cycle of ideas. Other helpful tools that enable distributed teams to work together and share information as well as document findings and decisions during the process need to be investigated and evaluated.

6.2 Refinement and Validation

In order to validate the methodology introduced in this paper, the authors plan to setup a software engineering course that is based on the principles and setup of Stanford University's Mechanical Engineering 310 (ME310), a course that has been the focus of research studies investigating factors that influence the performance of design teams (Carleton & Leifer, 2009). The course projects are collaborations between Stanford, international partner universities, and corporate partners. The interdisciplinary student teams tackle product innovation challenges that were proposed by the partner companies. Each project aims at delivering a mature and functional prototype to the corporate partners. In general, the projects are split into three phases:

⁴ http://www.hpi.uni-potsdam.de/forschung/design_thinking_research_program/programm.html

⁵ <https://tele-board.de>

- Make it up: teams investigate the problem space, focus on a problem and start exploring the solution space for this problem
- Make it real: teams validate and refine their ideas with the help of realistic prototypes
- Make it happen: teams create, test and refine their final functional prototype

An initial project was started in October 2012 and features collaboration between SAP, HPI, and the University of St.Gallen. This initial test project aims at evaluating both the overall teaching approach and parts of the process as described in this paper. AnalyzeD (Kowark & Plattner, 2012) is used to collect virtual collaboration artifacts of all team members and all tools the team chose to use. The collected artifacts will then be analyzed regarding, team collaborations, working patterns, code reuse and overall tool usage. Furthermore we will observe and interview the participants at different points in the project and prepare short questionnaires for them to evaluate our teaching approaches as well as the used techniques and tools. Further projects with other industry partners should follow.

Apart from this course setup, the authors will setup experiments to evaluate techniques tools and methods in a more controlled setup. For example the design planning technique as described in Section 4.2 should be evaluated regarding its usefulness and criteria that enables the design team to make a good plan for the given timeframe. Research questions would be:

- Does design planning help to structure and plan Design Thinking activities in a given timeframe?
- Does the design team's level of experience influence the usefulness of the plan?

A number of more and less experienced design teams will then be given the same challenge and time frame and are asked to plan their activities upfront. Other experiments should include the evaluation of User Story Mapping, different software prototyping tools and the influence of design coaches in the development teams.

In addition the authors aim at using this whitepaper to initiate discussions with experts from the design thinking and the agile software development world. Furthermore interview guides

and questionnaires will be prepared to gain more information and insights on the integration of Design Thinking and agile software development processes. All the user and expert feedback thus collected along with the information gained from experiments will be used to constantly refine and improve the course and our process model.

6.3 Current Research Plan

In conclusion, the authors' research plan for the nearby future includes four main research activities: using the white papers as a mean to trigger discussion and refinement, investigating and evaluating product standards and tooling to support compliance, investigate how working in sprint during DT activities influences teams and their performance, and setting up test projects.

With this white paper the authors will reach out to fellow researchers as well as SAP developers and managers. In a first round, the white paper will be shared with fellow researchers at HPI and HSG and selected SAP employees. The authors will then engage in discussion with these researchers and employees, and develop a questionnaire based on their input. In a second round, the authors will send out the white paper together with the questionnaire to further SAP developers and managers in order to collect ideas and suggestions for refining and validating the process framework and discuss the ideas.

Together with SAP and non- SAP developers, the authors currently evaluate the SAP product standards. For this purpose a small web tool was set up that allows users to rate the standards as either useful or not, and add comments as well as suggestion on how the standard could be automatically tested and which application types should comply with it. Additionally, the authors will research tools that support standard compliance at SAP and in general. Based on the results of the evaluation and the research, the authors will develop a questionnaire that will be send out to more SAP developers in order to verify the results. By doing so, the authors aim at providing a reduced set of standards together with possibilities to automatically verify their compliance.

To verify that an organization of Design Thinking activities in sprints and the usage of scrum sprint planning tools have a benefit for the design teams, the authors will setup experiments with Design Thinking teams, that try to answer the following research questions:

- RQ1: What level of detail is needed when planning DT activities?
- RQ2: Does the design team benefit from planning and organizing DT sprints, or would that just help management?
- RQ3: Are planning meetings and planning poker applicable to DT sprints?
- RQ4: How much DT experience is necessary to make realistic plans and assumptions?

In order to test team setups, the different operation modes (Design Thinking Mode; Mixed Mode (=Initial Development Mode) and Development Mode (=Fully Integrated Mode) as well as the handover from one mode to another the author will setup test projects with small development teams at SAP. We currently aim at project duration of 3 months for the first test run. The following figure shows a possible time schedule for such a project.

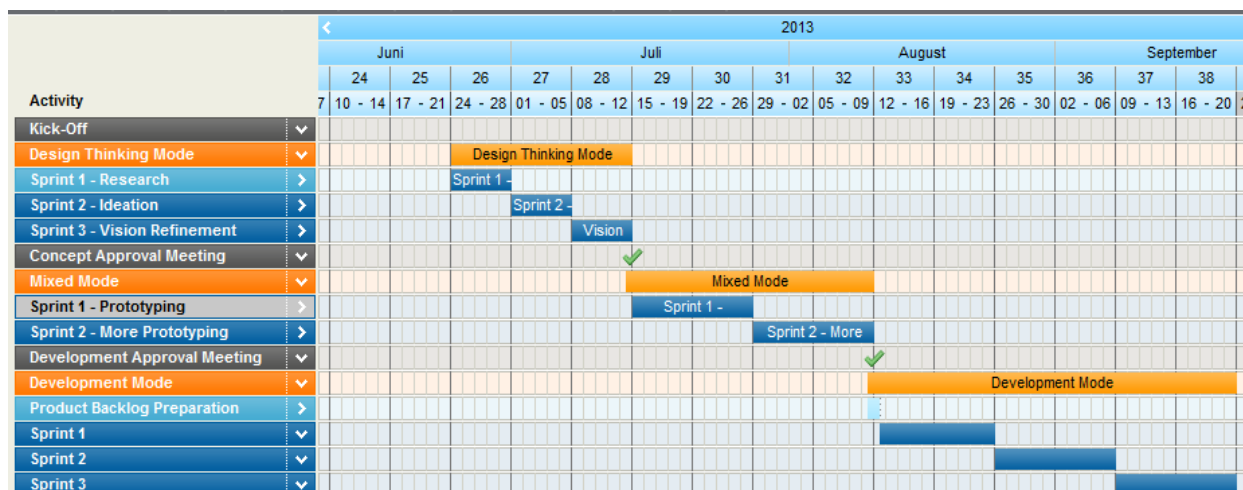


Figure 20: Example Project Schedule

The results of the presented research activities will be used to refine the proposed process framework.

References

- Ambler, S. W. (2009). *The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments*.
- Beck, K. et al. (2001). *Manifesto for Agile Development*. Retrieved on the 30th of January 2013 from: <http://agilemanifesto.org/>
- Carleton, T., & Leifer, L. (2009). Stanford's {ME310} course as an evolution of engineering design. *Proceedings of the 19th CIRP Design Conference—Competitive Design*.
- Gutzwiler, T. (1994). *Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen*. Hochschule St.Gallen für Wirtschafts-, Rechts- und Sozialwissenschaften: Dissertation.
- Hevner, A.R., S.T. March, J. Park and S. Ram (2004). *Design Science in Information Systems Research*. MIS Quarterly, (28)1, pp. 75-105.
- Heym, M. (1993). *Methoden-Engineering. Spezifikation und Integration von Entwicklungsmethoden für Informationssysteme*. Dissertation University of St.Gallen No. 1433.
- Hildenbrand, T., & Meyer, J. (2012). *Intertwining Lean and Design Thinking: Software Product Development from Empathy to Shipment*. In A. Maedche, A. Botzenhardt, & L. Neer (Eds.), *Software for People* (pp. 217–237). Springer Berlin Heidelberg.
- Keivanloo, I.; Roy, C.K.; Rilling, J. (2012). *Java bytecode clone detection via relaxation on code fingerprint and Semantic Web reasoning*. Software Clones (IWSC), 2012 6th International Workshop on, vol., no., pp.36,42, 4-4 June 2012
- Kowark, T., & Plattner, H. (2012). AnalyzeD: A Shared Tool for Analyzing Virtual Team Collaboration in Classroom Software Engineering Projects. *The 2012 International Conference on Frontiers in Education: Computer Science and Computer Engineering*.
- Larman, C., & Vodde, B. (2008). *Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum*. Addison-Wesley Professional
- Larman, C., & Vodde, B. (2010) *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison-Wesley Professional
- Panchenko, O.; Mueller, S.H.; Zeier, A. (2009). *Measuring the quality of interfaces using source code entropy*," *Industrial Engineering and Engineering Management*, IE&EM '09. 16th International Conference on, vol., no., pp.1108,1111, 21-23 Oct. 2009.
- Plattner, H., Meinel, C., & Leifer, L. (2010). *Design Thinking: Understand - Improve – Apply*. Springer Berlin Heidelberg.
- Plattner, H., Meinel, C., & Leifer, L. (2012). *Design Thinking - Measuring Performance in Context*. Springer Berlin Heidelberg.
- Schwaber, K. & Beedle, M. (2001). *Agile Software Development with Scrum* (1st ed.). New Jersey: Prentice Hall PTR.
- Schindlholzer, B. (2013). *Methode zur Entwicklung von Innovationen durch Design Thinking Coaching*. Universität St.Gallen: Dissertation
- Skogstad, P. (2009). *An unified innovation process model for engineering designers and managers*. Stanford University: UMI. PhD-Thesis.
- Vetterli, C., Uebersnickel, F. & Brenner, W. (2012). *Initialzündung durch Embedded Design Thinking — Ein Fallbeispiel aus der Finanzindustrie*. In: Zeitschrift für Organisationsentwicklung 2, pp 22-31.
- Wikipedia. (2013). *Scrum*. Retrieved 12th of January from <http://de.wikipedia.org/wiki/Scrum>

Appendix

A.1 ME310 Project Milestones:

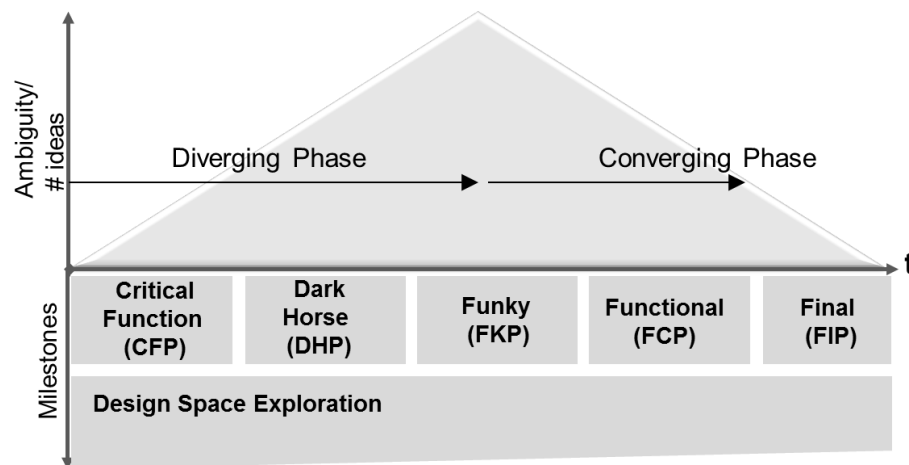


Figure A.1: Milestones of global ME 310 Design Thinking Projects and Embedded Design Thinking Projects (Vetterli et al. 2012)

The **Critical Function Prototype** is a low-fi prototype that aims at trying to implement a critical function of the product/service, thus letting the design team discover what is necessary to and how to implement the critical function.

The **Critical Experience Prototype** is a low-fi prototype that tries to convey a real enough product or service experience to users discover insights about the experience the users would like to have.

The **Dark Horse Prototype** is a low-fi prototype that enables the design team to follow a crazy, unrealistic or risky idea. The prototype can help to create new more realistic ideas, and will enable the design team to gain new requirement insights.

The **Funky Prototype** is a mid-fi concept prototype that aims to provide an overview of the whole system by bringing together first hacked parts. The system does not yet need to be fully functional or even nice to look at it but it should show convey first features and functions. This prototype is about deciding what functionalities and features the final solution must en-

tail. It also forces the design team to consider and resolve technical issues, that have not been face yet.

The **Functional System Prototype** is a mid-fi prototype takes the idea of the funky prototype one step further. It aims at giving an outlook of the final system with all features and functions necessary. This prototype will not look polished but enable users to test out the intended functionalities and the experience they will have when using the product. It forces the design team to bring together all features and ideas they want to provide to the user.

A.2 Techniques (see also B. Schindlholzer, 2013):

360° Research is a technique to get a lot of information about a topic in a very short time. This technique is often used to start out the project and get familiar with the topic / challenge at hand. It can also be used when a change of direction occurs in the process. During 360° Research the team or team members will investigate the topic via internet, books and other media, determine possible stakeholders and users, research competitor products and basically try to learn everything they can from their desk.

User Observation is a technique that helps to understand the user and his surroundings and helps the team to notice workarounds and problems that the user himself might not be aware of. This technique requires one or two team members to accompany a user during his work, his topic related tasks or during the testing of a prototype. The user is observed, ideally even recorded in video and audio. Usually this is done as a quiet observer, however it might be appropriate to let the user explain what he is doing and thinking or ask him about specific observations in order to fully understand what could be seen.

User Interviews are a technique to gain information about the topic under research and to gain an understanding of the user and his problems and needs. User interviews should be done in groups of two or three team members, where at least one is taking notes and one or two are interviewing. If possible interviews can be recorded in video or audio. In preparation of the interview an interview guide should be developed with the whole team to make clear

what information the team wants to gain, what the most important questions are and to make sure the interview time is used best for the team. This can be done by brainstorming questions and then grouping them into themes, thus identifying important themes and questions.

Extreme Users, for example very old or very young people or people that have very little or very much contact with the topic / problem or product the team is researching, can give a totally different perspective on things which the team might not get from interviews and observation with users from the respective target groups. Their workarounds and problems will be more noticeable but will often apply to other users as well. Therefore interviewing and observing extreme users can help to find and understand user problems.

Storytelling is a technique to get the whole team updated on information collected during observations and interviews. For this technique the team member(s) that collected the information share their experience by telling the team how the interview / observation went. The other team members take note of important aspects on post-it's and collect these on a whiteboard. Thus the information is written down for synthesis and all team members are up to date.

Clustering is a technique that helps to group different ideas on post-it's by all team members, when groups have been formed they will be named respectively forming a cluster. This technique helps to identify topics and aspects of the challenge in the information collected.

A **Persona** serves as a typical target user whose problems should be solved and whose needs should be addressed. In a physical representation as a cardboard figure or a drawing or picture he can be with the design team during ideation and prototyping phases. To form a persona the team should collect often heard problems, wishes and needs together with common schedules and activities from target users. These can be put onto the persona in form of Post It's or notes next to the picture.

A **Point of View Madlib** helps the design team to formulate their point of view on the system. The technique requests the team to write down a variety of PoVs.

A **2 by 2 Matrix** or a **Venn Diagram** are tools that help to think about and explore relationships between things, people or pieces of information. A 2 by 2 Matrix simply has two spectra, thus forming 4 quadrants in which information can be placed. Very full or very empty quadrants can give a hint to importance or open aspects. A typical example for a 2 by 2 matrix could be a competitor landscape.

A Venn Diagram is a diagram with overlapping circles. Each circle represents an aspect of the topic or problem, overlapping areas show that more than one aspect is addressed; the sweet spot in the middle addresses all topics. In a Venn diagram with three circles 7 areas will form in which information can be placed. Again very full or very empty areas can be interesting.

The **How Might We question** (HMW) is a technique to formulate the current PoV of the team in a question that triggers ideas. HMW questions typically have the following structure:

How might we help [person] to do [something]?

A good HMW question is short and precisely describes the problem the team currently wants to solve.

Brainstorming is a technique to collectively generate ideas about a topic or problem. For ideation purposes brainstorming is usually started on a HMW question. Different brainstorming setups exist. One example is to start out with 5 silent minutes in which each team member writes down his ideas, afterwards the ideas are presented and the team collaboratively generates more ideas based on these. Another possibility is to generate ideas collaboratively the whole time. However the brainstorming is facilitated the following rules should be adhered to:

- Only one person speaks at a time.
- Don't judge ideas, instead build on existing ideas and improve them.
- Go for quantity; collect a large amount of ideas during a brainstorming session.
- Encourage wild ideas, because good ideas are usually close to crazy ideas.