CHALMERS

The Integration of Design Thinking and Lean Software Development from the Perspective of Product Owners and Scrum Masters

Master of Science Thesis
in the Management and Economics of Innovation Programme

ULRICH FRYE
TINA INGE

Department of Technology Management and Economics
Division of Innovation Engineering and Management
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2013
Report No. E 2013:089

MASTER'S THESIS E 2013:089

The Integration of Design Thinking and Lean Software Development from the Perspective of Product Owners and Scrum Masters

ULRICH FRYE

TINA INGE

Supervisor: INGO RAUTH

Examiner: MARIA ELMQUIST

Department of Technology Management and Economics
Division of Innovation Engineering and Management
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2013

The Integration of Design Thinking and Lean Software Development from the Perspective of Product Owners and Scrum Masters

Ulrich Frye, Tina Inge

Master's Thesis E 2013:089

Department of Technology Management and Economics
Division of Innovation Engineering and Management
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
Telephone: + 46 (0)31-772 1000

Chalmers Reproservice
Göteborg, Sweden 2013

Abstract

Design Thinking as a managerial concept is being adopted by an increasing number of companies in various industries to boost their output of innovative products and services in order to gain a competitive edge on the market. This thesis presents an exploratory case study that was conducted at the global enterprise software producer, *Software Co*, who introduced Design Thinking in their software development organization in 2011. Since 2009, software development at *Software Co* has routinely been carried out according to Lean Software Development and the Agile software development method of Scrum.

The software development teams at *Software Co* are typically assembled in accordance with the Scrum methodology and comprise two managers, an architect, and a number of developers. The two managers have specific roles and responsibilities and are assigned the titles of Product Owner and Scrum Master. The success of integrating Design Thinking with the existing approach to develop software critically hinges on the managerial staff of these development teams. This qualitative study explored how Product Owners and Scrum Masters perceive the integration of Design Thinking with Lean Software Development and Scrum. Specific focus was placed on how Design Thinking practices integrate with Lean Software Development principles and Scrum practices.

Theoretical misalignments were identified between Design Thinking practice elements, Lean Software Development principles and Scrum practices. When investigated in practice, these theoretical misalignments were not perceived very strongly by Product Owners and Scrum Masters. Nonetheless, practical challenges were seen to arise from the integration of Design Thinking with Lean Software Development and Scrum. By exploring theoretical disaccords in a practical setting, this thesis serves as an early step towards the development of theory regarding the successful integration of Design Thinking, Lean Software Development and Scrum.

Key words:

Design Thinking, Lean Software Development, Agile Software Development, Scrum, Product Owners, Scrum Masters

Acknowledgements

This research was conducted with the support and guidance from several individuals to whom we are truly appreciative. Our supervisor, Ingo Rauth, provided on-going guidance, critical feedback, and encouragement throughout this thesis. Also, we would like to extend our gratitude to *Software Co* for their collaboration and granting us access to conduct our interviews.

A special thank you goes to LPS and TH from *Software Co*. LPS provided early insights which formed the foundation of this study, and established initial contacts with interviewees in India for which we are extremely grateful. TH shared his experiences about Design Thinking at *Software Co* which greatly enhanced our understanding of the study's context, and we also acknowledge his efforts to connect us with potential interviewees.

Last but not least, we are very thankful to the employees within the software development organization at *Software Co* who gave their support to our thesis by taking time to participate in our interviews.

Göteborg, 2013
Ulrich Frye
Tina Inge

Table of Contents

1 Introduction

Companies in all industries around the world are under immense pressure to create innovative products and services in order to be competitive in the market. The software industry is no exception to this and some companies have taken to adopting alternative approaches to traditional product development with the goal to develop more desirable offers. *Software Co* is one of the leading enterprise software vendors worldwide and in 2011 began to formally introduce Design Thinking (DT) into their development organization. Although applied for over two years to the software development context at *Software Co*, the understanding of DT and its addition to traditional approaches to software development is still in its infant phase. Thus, *Software Co* was chosen as the organization to study because it is an early adopter of DT in the software industry.

Hildenbrand and Meyer (2012) conducted a pilot case study to understand, primarily, the synergies involved in intertwining DT with Lean Software Development (LSD) and the Scrum process framework mostly from a process perspective. However, a gap in this research was identified as any misalignments between the approaches from an individual Scrum team member's perspective were not studied. Given the influence a Product Owner (PO) and Scrum Master (SM) possess as leaders on a Scrum team in software development companies, it is valuable to understand their perception of how DT integrates with LSD and the Scrum process framework.

The main contribution of this thesis is to provide a theoretical and empirical analysis on how DT, LSD, and the Scrum process framework align from the PO's and SM's perspective. This specific area might be useful for researchers interested in the overall understanding of how DT can migrate from the design industry into the software development industry, similar to how Lean and Agile migrated from the manufacturing industry to the software development industry.

1.1 Case Company

Software Co is a global leader in enterprise software development and associated services with more than 50,000 employees located in multiple countries. Software development takes place in their development labs which are distributed in various countries globally. Their software products are used by more than 200,000 customers around the world.

To create innovative products to serve their global customer base, *Software Co* has adopted DT. DT is defined on *Software Co*'s website as a routine innovation methodology that enables the two halves of the human brain to act in synergy, effectively combining its creative and analytical capabilities. In terms of practical execution, the general DT approach as practiced at *Software Co* is shown in Figure 1 below. The approach illustrated below is highly iterative and allows returning to previous activities or phases should the need be identified based on insights gained at any stage.

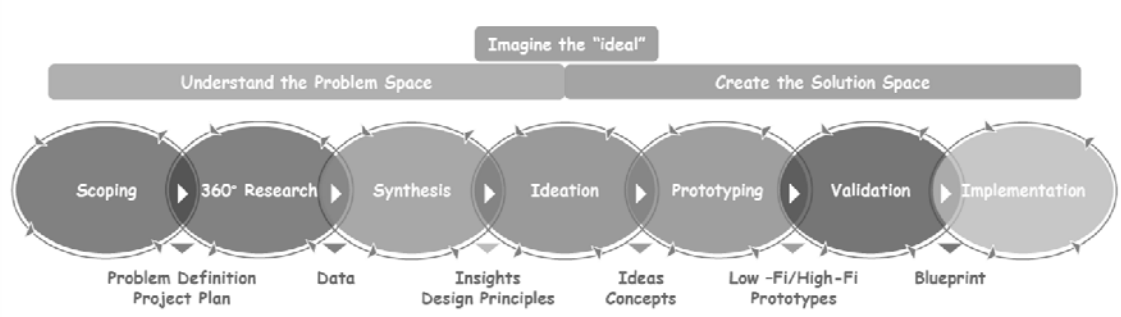

Figure 1: DT process

DT was rolled out at *Software Co* in three different waves. The first wave of DT projects started in 2011 at *Software Co's* facilities in Germany, while the second wave of DT projects commenced in 2012 and primarily involved *Software Co* offices in Germany, India, and China. Finally, the third wave of DT projects started at the end of 2012 and is ongoing.

1.2 Background

DT has risen as a management concept promising increased innovativeness through a user-centered approach to innovation. DT has its origins in the design realm but has entered the management realm and gained considerable acceptance as an approach to product development and problem solving (Hassi & Laakso, 2011). Various companies in different industries have taken to using DT in their product development practices to complement and enhance their existing approaches.

The concept of Lean Thinking originated in the automotive industry and described the production management system at Toyota along with manufacturing techniques (Stone, 2012). It has since penetrated into all areas of management and is primarily concerned with waste and value in organizational operations, with the goal of minimizing the former and only retaining activities contributing to the latter (Poppendieck & Cusumano, 2012; Stone, 2012).

Just as Lean Thinking, the term Agile also came from the manufacturing sector and refers to a producer's flexibility to respond to changes in an unstable environment and customer needs (DeVor, Graves, & Mills, 1997; Poppendieck & Cusumano, 2012). The need for increased flexibility and responsiveness to customer requests was subsequently identified in software development settings and a group of developers created a manifesto based upon a number of agile principles that were specifically adapted to software development (Beck et al., 2001; Highsmith, 2001). Within Agile several methods have been created that aim at making the development process more responsive and flexible (Wölbling et al., 2012).

One specific method of Agile is the process framework of Scrum. The Scrum process framework prescribes the composition of development teams, events in the process, and tangible objects (Schwaber & Sutherland, 2011). Two roles within the Scrum team are of particular interest: the SM and the PO. Individuals filling these roles have specific managerial duties to carry out and principles to adhere to; the primary responsibility of the PO is the functionality of the product and the value it creates for all stakeholders, while the SM is concerned with the execution of the project and smooth running of the process (Schwaber & Sutherland, 2011).

LSD and Scrum have been combined successfully within the software development industry and have become common practice. DT has grown from the design discipline into the management disciple.

These approaches, as separate entities, have been implemented and documented from practitioners' perspectives but there is little evidence of their detailed study in academic literature. There is even less literature, from either a practitioner or academic perspective that focuses on understanding the combination of DT, LSD, Scrum approaches in product development.

Approximately four years ago, *Software Co* introduced an approach to their software development that is founded upon LSD principles, and is executed through the Scrum process framework and in 2011, *Software Co* started an initiative to roll out DT throughout its development organization. Hildenbrand and Meyer (2012) investigated DT and LSD in combination on Scrum teams, but other previous studies on the topic were not evident from a thorough literature search. In their book chapter, *Intertwining Lean and Design Thinking,* Hildenbrand and Meyer (2012) take a holistic perspective to understand if and how these approaches to software development can be combined or whether DT is simply a substitute for LSD on a Scrum team. The authors concluded that these approaches to software development can be intertwined, as opposed to DT becoming a substitute or replacement and suggested a process showing how DT can be intertwined with LSD on a Scrum team.

While the work of Hildenbrand and Meyer (2012) is a valuable early contribution to gain understanding towards the combination of DT and LSD on a Scrum team in software development projects, there are aspects in the study that raise questions and warrant further investigation. The book chapter in question describes the study of a single software development project within a global enterprise software company. First, one single case is only a starting point in understanding if the combination of DT and LSD on Scrum teams are indeed complementary approaches as they were claimed to be. It is therefore questionable if the findings can be considered valid for any other, let alone all, software development projects that utilize these approaches, both within their case company and the wider industry. Second, the investigated case itself does not appear to be quite representative of a real development project as it would generally be carried out within software development companies, because the customer here was a sailing team sponsored by the developing company itself, for a specific event. Naturally, that specific company had a great interest in the success of the project: *"Besides the classical support of a sponsor, SAP is particularly interested in showcasing its technology..."* (Hildenbrand & Meyer, 2012, p. 221). Since the project was a showcase, it may be considered limited in its comparability to software development projects for industry customers within the software industry at large.

Besides the limitations outlined above, Hildenbrand and Meyer (2012) mostly took a process perspective of the combined approaches and the methods within. Whilst this angle is entirely legitimate, it leaves room for expansion. Another interesting aspect for investigation concerns the individuals that are working on the software development Scrum teams and their perception of the combined approaches. In particular, team members with managerial duties and responsibilities for the success of the projects and their understanding of the combined approaches appear to be worthwhile studying because the success of the integration of DT hinges critically on those. As outlined earlier, these individuals in question are POs and SMs.

Based upon working within the LSD and Scrum approach at *Software Co*, experienced SMs and POs are likely to have acquired certain ways of working and gained experience according to the underlying principles and practices of LSD and Scrum. These SMs and POs now have to further adapt

as *Software Co* is rolling out DT throughout their software development with the goal of creating more desirable and innovative software products.

This thesis investigated the combination of LSD and DT on Scrum teams in software development projects, as perceived by managerial staff composed of POs and SMs. Particular emphasis is put on the alignment of underlying LSD principles and Scrum practices with the DT practice elements. Thus, a case study was conducted within *Software Co* to initially explore the topic; the study and its findings are described in the following chapters.

1.3 Purpose

The purpose of this thesis is to contribute to the academic understanding of the combination of LSD and DT on Scrum teams. Given the limited research on the subject, this study primarily builds upon the DT work by Hildenbrand and Meyer (2012). It contributes by investigating the integration of the above approaches from the perspective of managerial staff on Scrum teams who are responsible for the success of software development projects.

1.4 Research Question

Based on the current state of academic literature concerning LSD and DT a need has been identified to further study the combination of these in a practical setting such as software development. This need for understanding is particularly obvious when considering individuals who have significant experience in managing software development projects according to LSD principles on Scrum teams, and are responsible for the integration of DT into this existing approach. In order to explore the topic and gain an initial understanding from the perspective of these individuals leading the LSD and DT activities on Scrum teams at *Software Co*, the following research question is answered by this study:

> *How do the DT practice elements of end user empathy, visualization, and synthesis integrate with LSD and the Agile method of Scrum from the perspective of Product Owners and Scrum Masters?*

1.5 Delimitations

This study investigated the use of DT on software development projects at *Software Co*. Thus, the scope of the study was limited to only cover a single company; within the company, only the software development organization was studied even though DT is being utilized in other parts of the company as well. This was deemed appropriate because this thesis contributes to the general field of innovation management, and DT in the software development organization at *Software Co* was introduced to improve the innovativeness of their software products.

The development organization at *Software Co* is dispersed globally; for the reason of access provided, this investigation only covered development labs in India and Germany. Cultural aspects, both organizational and ethnical, were not considered in this research. The units of analysis were individuals filling managerial roles on software development teams, other team members or staff outside these teams were not considered.

2 Literature Review

This chapter introduces the literature and theories pertaining to this exploratory case study. Overviews are provided about the history and current states of knowledge and practice about DT,

Lean Thinking, Agile, and LSD approaches as well as the Scrum process framework for software development. The above approaches are utilized by *Software Co* in their software development activities, with DT being the latest addition.

2.1 Design Thinking

Herbert A. Simon applied the notion of design as a way of thinking to the design realm in the 1960s, more recently it has been applied to the management realm by both academics and practitioners (Hassi & Laakso, 2011; Simon, 1969). Within the academic realm, DT in terms of how designers think has been discussed and described for over thirty years (Johansson, Woodilla, & Çetinkaya, 2011). However, the application of the way designers think to the management realm is fairly new but growing in importance, partially due to the view that *"design has become too important to be left to designers"* (Brown & Katz, 2011, p. 381).

Johansson-Sköldberg, Woodilla, and Çetinkaya (2013) argue there are the following two discourses on DT: scholarly and management. The scholarly discourse refers to designerly thinking theory to practice and is academically founded in the field of design whereas the management discourse applies to design practices and competences outside of the design field (Johansson-Sköldberg et al., 2013). This study focused on DT elements applied to management roles within the software development industry. Therefore, only the management discourse on DT is discussed as the designerly thinking discourse is outside the scope of this research. However, it is important to note that there are discourses in the design realm that concentrates on the skills and abilities of designers, for example (Simon, 1969), that could be used to further investigate but given the choice to focus on the management realm this is also outside of the scope for this thesis.

2.1.1 Definitions

The term DT does not have a well-defined meaning but is used in a variety of ways and multiple contexts. Moreover, the term is becoming increasingly ubiquitous (Kimbell, 2011). In general terms, DT is thought of as having two main aspects: a user centric philosophy or approach to problem solving (Brown, 2008; Dunne & Martin, 2006; Hassi & Laakso, 2011; Kimbell, 2011; Seidel & Fixson, 2012).

Table 1 lists a selection of definitions given in literature for DT. However, it is important to note that defining DT into a singular definition can be considered a futile effort because there are many different discourses on design and it is inherently individualistic based on the designer as eloquently stated by Johansson-Sköldberg et al. (2013, p. 14) *"to talk about design and leaving the designer out is like talking about music and leaving the musicians out."*

Table 1: Design Thinking Definitions

(Holloway, 2009, p. 51)	*A term used to describe how designers typically approach problem solving*
(Brown, 2008, p. 86)	*A discipline that uses the designer's sensibility and methods to match people's needs with what is technologically feasible and what a viable business strategy can convert into customer value and market opportunity*
(Dunne & Martin, 2006, pp. 512, 517)	*Design thinking results from the nature of design work: a project-based workflow around "wicked" problems*
(Owen, 2007, p. 17)	*The obverse complement to scientific thinking*
(Hassi & Laakso, 2011, p. 53)	*DT is a set of certain practices, cognitive approaches, and mindsets*
(Johansson-Sköldberg et al., 2013, p. 124)	*A way of describing a designer's methods that is integrated into an academic or practical management discourse.*

2.1.2 Processes

As stated above, DT can be described in a general sense, rather than a singular specific definition, as using a user centric approach to solve problems (Dunne & Martin, 2006). However, in both academia and practice, there is attention given to understanding the specific process associated to DT but similar to the definition there is not a singular universal process for DT but rather there seems to be underlying commonalities or elements associated to these processes. While it is important to have a general understanding of processes associated with DT, for this thesis the underlying commonalities or elements of DT mentioned below are the main lens used in answering the research question.

2.1.3 Elements

Hassi and Laakso (2011) conducted an extensive DT literature study of 50 books or articles, of which they used 31, to create a three dimensional framework for the different elements associated to DT. This framework assigned three main DT categories that have related DT elements. These categories are **Practices, Cognitive Approach, and Mindset** (Hassi & Laakso, 2011). This research will focus on the practice category as opposed to the cognitive approach or mind set categories since this is the most relevant category for answering the research question. This exploratory case study is concerned with looking at how individuals work and the tangible approaches they use, as given in the practice category, rather than different styles or ways of thinking given in either the cognitive or mindset categories (Hassi & Laakso, 2011).

Moreover, as stated in the below literature section for LSD and Scrum, these two approaches to software development focus more on the tangible ways of working, activities, and use of specific artifacts. As a result, using the elements given in the below section for the DT practice category was a comparable level of analysis.

2.1.4 Practices

DT practices can be described as *"tangible approaches, ways of working, activities, and the use of specific tools"* (Hassi & Laakso, 2011, p. 55). There are five main elements that are considered by Hassi and Laakso (2011) to be DT practices: human-centered approach, thinking by doing, visualizing, synthesis, and collaborative work style. These are further elaborated on below.

1. Human-Centered
The human-centered approach puts people first by developing empathy for and understanding of the customer and/or users. Observation and ethnography are the key methods used in developing the emphatic understanding. DT is often referred to as customer, user or human-centric design as this is one of the most prominent issues highlighted in DT literature (Hassi & Laakso, 2011).

2. Thinking by doing
Thinking by doing is a tangible and iterative approach that creates knowledge in a practical manner by a rapid, systematic, and iterative process. Thinking by doing involves creating continuous prototyping that helps in ideation. Prototypes are seen as a tool to help both develop and explore ideas instead of being direct representations of the product (Hassi & Laakso, 2011).

3. Visualizing
Visualizing is how one expresses and makes sense of things in DT by using any media outside of words and symbols. Visualizing allows a common understanding of ideas so that they can be discussed and shared (Hassi & Laakso, 2011).

4. Synthesis
A divergent and convergent approach or sometimes referred to as synthesis involves coming to a solution by broadening the scope and then narrowing it by using selection and synthesis. DT typically starts using a divergent approach to allow for multiple alternatives or ideas to be considered instead of taking the initially best idea. The divergent approach is not limited to the beginning but can be used throughout DT. The convergent approach then allows for patterns and relationships to be discovered (Hassi & Laakso, 2011).

5. Collaborative work style
Hassi and Laakso (2011) noted that virtually all DT authors place an emphasis having a collaborative work style in DT. A collaborative work style involves having a wide range of stakeholders. Moreover, collaborative thinking is by thinking outside of your own head by interacting with others (Hassi & Laakso, 2011).

As stated above, DT has started to diffuse into the management discourse from the design world. The basics of DT include empathy and understanding toward the user and a human-centered philosophy. The elements of DT can be classified into three categories: practices, cognitive approach, and mindset (Hassi & Laakso, 2011). These DT categories along with the related elements are summarized in Table 2.

Table 2: DT Elements and Categories according to Hassi and Laakso (2011)

DT Category	DT Elements
Practices	**Human-centered/Empathy** **Think by doing/Action based** **Visualizing** **Diverging & Converging** **Collaboration**
Cognitive approach	*Holistic viewpoint* *Integrative thinking* *Abductive thinking*
Mindset	*Future oriented* *Explorative* *Experimental*

2.1.5 Summary

As explained above and illustrated in Table 2, this paper's focus is on the DT elements in the practices category as summarized in the extensive literature review conducted by Hassi and Laakso (2011). This viewpoint will allow the study to have the tangible ways that POs and SMs work as the level of analysis and to look at their perception of DT from a practice perspective, rather than make a comparison of the associated DT processes or methods.

2.2 Lean Thinking

Lean Thinking has its origin in the automotive production industry in Japan. The term *Lean* was first used in connection with production management practiced at Toyota and has continued to evolve for several decades (Hines, Holwe, & Rich, 2004; Poppendieck & Cusumano, 2012; Stone, 2012). The more specific term *Lean Production* originally described manufacturing techniques (Stone, 2012) and was then expanded to describe *"any efficient management practice that minimized waste, including in product development"* (Poppendieck & Cusumano, 2012, p. 26).

On the most basic level, Lean Thinking distinguishes between waste and value in an organization's operations. As defined by Womack and Jones (1996) in Stone (2012, p. 114), waste is *"any human activity which absorbs resources but creates no value"* and value denotes *"a capability provided to a customer at the right time at an appropriate price, as defined in each case by the customer"*. Moreover, the most basic premise of Lean Thinking is to actively identify and eliminate waste from the firm's processes with the goal of only retaining activities that add value (Stone, 2012). All in all, five basic principles of Lean Thinking were proposed by Womack and Jones (1996) and are listed by Haque and James-Moore (2004, p. 3) as:

- *Specify value*: define value precisely from the perspective of the end customer in terms of a specific product with specific capabilities offered at a specific price and time.
- *Identify the value stream and eliminate waste*: identify the entire value stream for each product or product family and eliminate waste.
- *Make the value flow*: make the remaining value creating steps flow.
- *Let the customer pull the (value) process*: design and provide what the customer wants only when the customer wants it.

- *Pursue perfection*: strive for perfection by continually removing successive layers of waste as they are uncovered.

Poppendieck and Cusumano (2012) elaborate further that Lean Thinking emphasizes foremost the decrease in waste related to resources in terms of time and staff and at the same time promotes the creation of value for the customer and the company through products. Additional benefits of Lean Thinking are those associated with a *"more flexible, iterative, lightweight development process"* (p. 27). Furthermore, value is also increased by adding features or services that are valued by customers but do not contribute to the internal waste, e.g. shortened delivery times (Hines et al., 2004).

2.3 Agile

Similar to Lean Thinking, Agile also has its origins in the manufacturing industry (Poppendieck & Cusumano, 2012) and basically describes a producer's ability to be flexible in order to be successful in an environment of continuous change (DeVor et al., 1997). This need for the ability to respond to changes was identified for and applied to a software development setting when seventeen software experts drafted *The agile software development manifesto* (Highsmith, 2001), which was founded upon twelve underlying principles.

2.3.1 Agile Software Development Principles

The agile software development manifesto was built on the following twelve principles (Beck et al., 2001):

- *"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."*
- *"Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."*
- *"Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale."*
- *"Business people and developers must work together daily throughout the project."*
- *"Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done."*
- *"The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."*
- *"Working software is the primary measure of progress."*
- *"Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."*
- *"Continuous attention to technical excellence and good design enhances agility."*
- *"Simplicity--the art of maximizing the amount of work not done--is essential."*
- *"The best architectures, requirements, and designs emerge from self-organizing teams."*
- *"At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."*

2.3.2 Agile Software Development Methods

Within the overall agile software development approach, several specific methods such as Scrum and Extreme Programming have emerged with the goal of making the software development process more responsive and flexible (Wölbling et al., 2012). In the following section, Scrum is further

elaborated on because it is one of the most used agile methodologies in software development (del Nuevo, Piattini, & Pino, 2011) and has managerial aspects attached to it (e.g. Pries & Quigley, 2011). The Scrum method is utilized by *Software Co* in their development activities.

Scrum

Scrum is defined by its creators as *"a process framework ... used to manage complex product development"* (Schwaber & Sutherland, 2011, p. 3), and has also been described as a method to manage projects that is focused on immediate objectives and deliverables of involved people (Pries & Quigley, 2011). As such, it is not a process or technique for building products but rather a management framework to develop and sustain complex products such as software by employing an iterative and incremental approach (Schwaber & Sutherland, 2011). Scrum is also not a complete methodology since it does not include specific practices. Instead, it aims to optimize the process of software development within the overall value stream (Poppendieck & Cusumano, 2012). The Scrum process framework encompasses various events, roles of members in Scrum teams, and artifacts.

Scrum events are sprints, spring planning meetings, daily scrums, sprint reviews and sprint retrospectives. The primary events are the sprints, which are blocks of time lasting four weeks or less, in which programming takes place and aim to result in working software releases which can be inspected and tested by end-users (Maylor, 2010; Schwaber & Sutherland, 2011). These releases are typically presented at sprint reviews that follow each sprint and allow frequent customer feedback on functionality, which in turn influences subsequent priorities (Rising & Janoff, 2000).

A Scrum team is typically cross-functional and comprised of the development team, the PO, and the SM. The development team is self-organizing with members possessing the necessary skills to deliver the final product. Recommended team size ranges between three and nine members, not counting the PO and SM (Schwaber & Sutherland, 2011).

The PO is responsible and accountable for deciding what needs to be done, i.e. managing the product backlog. By performing his/her duties, the PO's main undertaking is to maximize the value of both the product and the development team (Poppendieck & Cusumano, 2012; Schwaber & Sutherland, 2011). As such, in a larger organizational context, the PO can be seen as the link between the development organization, the wider organization, and external entities. The SM's role is to make the process run as smoothly as possible and continuously improve it along the way by, for example leading meetings and tracking progress.(Rising & Janoff, 2000). Specifically, the SM provides his/her services to the PO, the development team and the organization at large (Schwaber & Sutherland, 2011).

Scrum artifacts serve to make the development process more transparent, make key information visible and thus allow inspection and adaptation. The three types of artifacts are product backlog, sprint backlog and increments. An artifact of key importance is the product backlog which is defined as *"an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product"* (Schwaber & Sutherland, 2011, p. 12). The product backlog is continuously evolving and as such is never complete. It is the sole responsibility of the PO to manage the product backlog, prioritize items in it and ensure they are addressed by the team (Schwaber & Sutherland, 2011).

2.3.3 Summary of Lean Thinking and Agile

As stated above, both Lean Thinking and Agile have their origin in the manufacturing industries and have subsequently made their way into different settings which include the software development industry. The main premise of Lean Thinking is to make the value stream more efficient by eliminating any processes that do not add value. The main idea behind Agile is for an organization to be more responsive in continuously changing environments.

Simply put, both Lean Thinking and Agile put the customers and their satisfaction first but they do so in different manners. While Lean Thinking emphasizes efficiency and planning, Agile builds upon customization resulting from close customer contact and interaction. But as proven in practice, it is possible to optimize Agile operations based on Lean Thinking, especially in the software development industry where this combination has been practiced since the early 2000s. By operating in short and iterative cycles that produce working software, users can be involved frequently to enable identification of customer value and respond to updated requests (Wölbling et al., 2012).

2.4 Lean Software Development

In the early 2000s, Lean Thinking started to diffuse into areas outside of the manufacturing industry such as logistics, military, construction and services. As a result, this delivered proof that in practice Lean Thinking is applicable across various fields (Hines et al., 2004; Poppendieck, 2002). Although Lean Thinking originally related to practices particular to the automotive manufacturing industry, it could be transferred into other settings by viewing it as a set of principles or concepts rather than specific practices (Poppendieck & Cusumano, 2012). Based on this view, Lean Thinking entered the software development via Agile practices (Hildenbrand & Meyer, 2012), because there are many similarities and analogies between their respective elements (Poppendieck & Cusumano, 2012).

By integrating Agile and Lean Thinking, seven principles for LSD were proposed in 2003 and have since become popular in the software industry (Poppendieck & Cusumano, 2012). The seven principles for LSD are (Poppendieck & Cusumano, 2012; Poppendieck.LCC, 2010):

1. *Optimize the whole*: LSD should be founded on a deep understanding of a job that customers would like done and how this job might be mediated by software.
2. *Eliminate waste*: waste is anything that doesn't either add customer value directly or add knowledge about how to deliver that value more effectively.
3. *Build quality in*: continuously integrate small units of software into larger systems to avoid finding defects in the final verification.
4. *Learn constantly*: development is all about creating knowledge and embedding that knowledge in a product; delay irreversible decisions to the last responsible moment, based on best available knowledge.
5. *Deliver fast*: think of software as a flow system where software is designed, developed, and delivered in a steady flow of small changes based on deep understanding of what stakeholders' value.
6. *Engage everyone*: empowering people, encouraging teamwork, and moving decision-making to the lowest possible level by composing multidisciplinary teams encompassing the complete value stream.
7. *Keep getting better*: failure is a learning opportunity and should be used to challenge and improve existing standards based on the scientific method.

These seven principles for LSD do not offer much in the sense of particular roles and responsibilities of individuals. Instead, they focus on how to operate in an organizational setting of software development, while the Agile method of Scrum gives clearly defined roles including the PO and SM. To summarize, LSD focus is on practices, or how to operate in an organizational setting for software development, again this makes the practice category for DT the appropriate category for comparison.

2.5 Synthesis of Design Thinking, Lean Software Development, and Scrum

A visual representation of the above descriptions for the development of Lean Thinking and Agile into the software development industry is shown in Figure 2. The context studied in this thesis project is concerned with DT being integrated with LSD and Scrum.

To understand the integration of LSD, Scrum, and DT, the five DT practice elements as given by Hassi and Laakso (2011) are compared below with the seven LSD principles and the Scrum practice of writing backlogs and engaging customers. The comparison is done to identify any synergies and discrepancies between those.

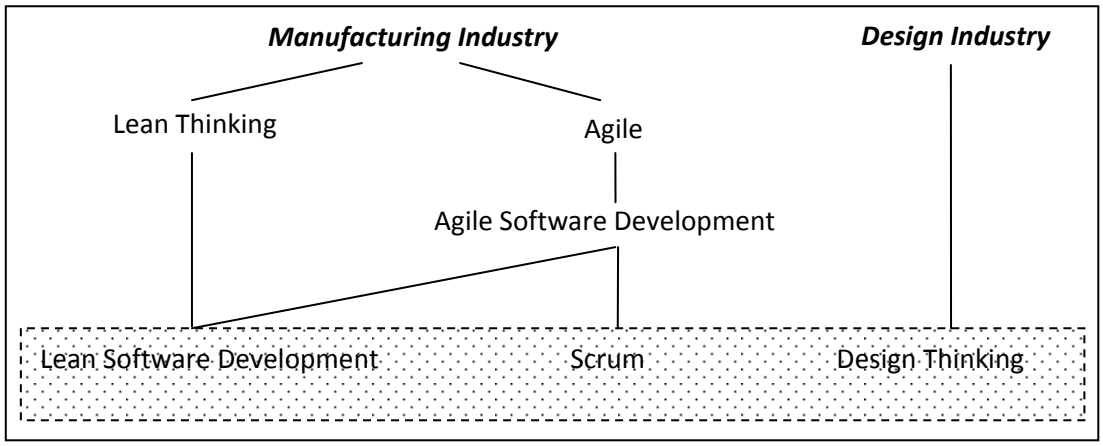

Figure 2: Context of study (shaded area investigated at *Software Co*)

DT has a clear human focus which is practiced by deeply empathizing with the user of the final product. On the surface this aligns well with LSD and Scrum. The first principle of LSD, optimizing the whole, requires a thorough understanding of what job the customer is trying to accomplish. As stated, Scrum is an Agile method and is therefore build on the premise of being flexible and responsive. This requires a good knowledge of what the customer values, which is to be gained through frequent testing of working software releases by end users. However, a discrepancy exists in that DT focuses on empathy to find latent needs while LSD's and Scrum's focus is on the customer satisfaction. A second discrepancy is that when taken at face value, it seems that LSD and Scrum are focused on the customer rather than the end user. Whilst no specific mention of end users is apparent in LSD, Scrum does mention the end user for testing of software releases, but only for the purpose of gaining customer feedback, judging customer satisfaction and adjust priorities for subsequent development which are arguably very different notions than empathizing with humans using a product or service. Nevertheless, owing to a lack of clear definition of the term customer, end users may be included in it. However, the customer and user are rarely the same in the enterprise software development industry and the customer may be unaware of what the actual user does value. Therefore, a potential misalignment exists in that DT places emphasis on finding latent needs

of end users, while LSD and Scrum are focused on customer satisfaction and do not explicitly mention end user empathy.

Collaboration in DT refers to working in multi-disciplinary teams to achieve the desired outcome. This practice matches well with Scrum which calls for self-organizing teams composed of members that bring all necessary skills to the project, and value is put on individuals and interactions. The LSD principle of engaging everyone in the entire value stream to deliver the product is also quite easily equated with the meaning of collaboration in DT, but could be interpreted as having a wider scope.

The DT practice element of visualizing facilitates easier communication, for example through storyboards. On the contrary, Scrum calls for product backlogs as written lists of all requirements and specifications for the product being developed. If DT precedes the actual software development phase according to the Scrum framework, the generated storyboards could be considered a precursor or foundation for the product backlogs. Even though the backlog may build on, or evolve from a storyboard, the process of generating each is quite different. None of the LSD principles is in agreement or in discrepancy with visualizing in DT.

DT is an action based approach, including the rapid creation of prototypes throughout the process. Two of the goals of prototyping are to learn and fail quickly and cheaply. These two are in perfect alignment with the LSD principles of delivering fast, learning constantly, and continuously getting better by considering failure as a learning opportunity. Although it could be argued that failure is waste and as such opposes the second principle of LSD, it reasons that early and cheap failure is still less wasteful than later failures which are more costly. The Scrum methodology aims at producing working software releases, which requires the action of programming. Thus, if working software is analogous to prototypes, then Scrum is also an action based approach and this practice aligns well with DT.

Perhaps the biggest conflict exists between DT and LSD when the DT practice of converging and diverging is considered. It is most apparent during the ideation mode of DT, where the primary objective is to create large amounts of ideas. A large amount of ideas implies by necessity that some, if not most, of those ideas will be discarded. When thinking in Lean terms, discarding anything is a waste and thus the ideation mode in DT violates the underlying fundamental premise of LSD – eliminating waste. Moreover, it can be questioned to what extent the iterative DT practice of converging and diverging ultimately adds value to any stakeholders. Converging and diverging may occur to some extent during the execution in Scrum mode; however, here any changes usually originate from external sources, i.e. the customers, while in DT the internal team itself is required to converge and diverge on their own accord and based on their own research. The comparison of DT, LSD, and Scrum discussed above are summarized in Table 3 for clarity.

Table 3: Summary of DT, LSD, and Scrum

	DT	**Agile**	
		LSD	**Scrum**
Differences	End user empathy	Customer satisfaction	
	Visualizing	N/A	Writing requirements in backlogs
	Synthesis	Waste elimination	N/A
Similarities	Action based		
	Collaboration		

Hildenbrand and Meyer (2012) have investigated the combination of DT with LSD on Scrum teams and have found a number of commonalities in fundamental values. As outlined above, these include the collaboration of multi-disciplinary teams, value for both the business and customers, and their iterative nature. Further, the authors argue that DT and LSD address *"different challenges and aspects in a development project lifecycle"* (Hildenbrand & Meyer, 2012, p. 219). Quite correctly they point out that LSD is concerned with execution aspects such as delivering high quality in a timely fashion. DT on the other hand is focused on what to execute or deliver, i.e. ensuring that the product is indeed of value to the customer.

A next logical step to build upon the study is to critically assess and investigate any challenges that arise from the combination of DT and LSD on Scrum teams, particularly to the individuals, POs and SMs, which are in charge of running these Scrum teams. Based on their experience in carrying out these roles, they have attained certain practices and ways of working. With the introduction of DT in addition to LSD on Scrum teams, these individuals are potentially facing challenges that originate from the contradicting requirements as outlined above. While the elements may not be mutually exclusive, there could nonetheless be difficulties in adapting to DT and integrating it with LSD on Scrum teams.

2.6 Theoretical Framework

Following the investigation and analysis of the literature, the theoretical framework presented in Figure 3 below has been developed. The focus of the framework lies with individuals rather than the processes or approaches. The individuals of concern are those that take on the roles of SM and PO according to the Scrum process framework. In the previous LSD only approach, they are referred to as LSD manager. With the addition of DT to development projects, they should also adopt the elements included in the DT manager role. The focus of this thesis was placed on the identified misalignments; therefore, the overlapping elements of action based and collaboration were not specifically addressed during the data collection.

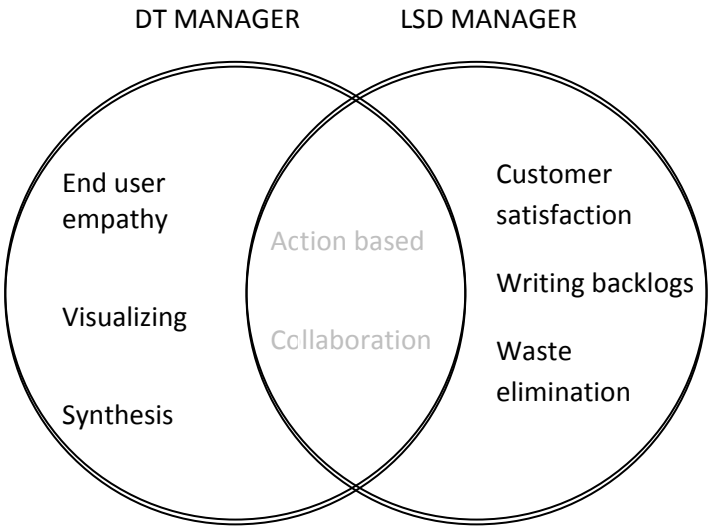

Figure 3: Theoretical framework

Based on the above theoretical framework, the following study investigates the addition of DT to the existing LSD approach on Scrum teams at *Software Co* from the perspective of POs and SMs. The

investigation is concerned with how these individuals perceive the addition of DT and its effect on their way of working. Studied in detail are the theoretical misalignments between the following DT practice elements and LSD principles and Scrum practices***: (1) End User Empathy vs. Customer Satisfaction (2) Visualizing vs. Writing Backlogs (3) Synthesis vs. Eliminating Waste***

3 Methodology

This chapter aims to explain the design and methods used to answer the research question. Moreover, the purpose of the method and sampling structure will be discussed. Also, a description is provided on how data was collected and the validity and reliability of the gathered information is discussed.

3.1 Research Approach

The first step in research is typically to decide between two types of reasoning, deductive and inductive, that are then used to gain a better understanding of the relationship between theory and research (Bryman & Bell, 2011; Saunders, Lewis, & Thornhill, 2007). Deductive reasoning starts with a very broad spectrum of information and converges to a specific conclusion. Wilson defines deductive reasoning as *"developing a hypothesis (or hypotheses) based on existing theory, and then designing a research strategy to test the hypothesis"* (Wilson, 2010, p. 7). Inductive reasoning, on the other hand, is the opposite of deductive reasoning in that the theory is generated out of the research (Bryman & Bell, 2011). Inductive research can be thought of as a "bottom-up" approach to building knowledge; the researcher is required to use observations and data to find patterns and regularities to develop a tentative hypothesis that will lead to general conclusion or theory (Babbie, 2001).

This study used inductive reasoning since the main focus was on DT and its relationship with LSD principles and the Scrum process framework, which is not yet widely covered in academic literature. An advantage of using inductive reasoning in a research study that is exploring a new area is that it is an iterative process that allows the flexibility to go back and forth between the research and theory (Bryman & Bell, 2011). However, there is a risk that an inductive research approach may require time beyond the planned scope of a study and lead away from a narrow and focused study and result in general and broad results that are superficial (Bryman & Bell, 2011). The inductive approach, in contrast to the deductive approach, has a high level of uncertainty since it requires the researcher to generate theories and conclusions out of specific observations and data (Saunders et al., 2007). These risks were mitigated by creating a schedule that included time for iterative work and continuous assessments to insure that the research was not deviating far from the answering the research question.

3.2 Research Strategy

A research strategy is described as the general orientation of the conduct of business research and can be quantitative, qualitative or a mixture of both. Quantitative research focuses on quantification and analysis of data and is more aligned to deductive reasoning, whereas qualitative research focuses on words in order to generate theories and is more aligned to inductive reasoning (Bryman & Bell, 2011). In order to answer the research question for this study, a qualitative strategy was used because the purpose of the study was to contribute to the academic understanding of the combination of LSD and DT on Scrum teams based on inductive reasoning.

3.2.1 Qualitative Research

Qualitative research is, for the most part, inductive with the aim of generating theory or meanings from data collected in the field (Creswell, Hanson, Plano, & Morales, 2007). DT is a relatively new area of study and this research study is focused on understanding how SMs and POs adapt to DT in a specific department at a specific company at a specific point in time. Qualitative research, as opposed to quantitative research, is more aligned to this focus since it takes a more interpretive and naturalistic approach (Denzin & Lincoln, 2005). This makes qualitative research a suitable strategy for this study since there is limited amount of theory to build a hypothesis that could be quantitatively tested.

Qualitative research presents some risks, such as misunderstandings. It is critical in qualitative research that the interviewers ensure they have the interviewees' response instead of depending on their own assumption (Britten, Jones, Murphy, & Stacy, 1995). This was addressed by having two researchers with varied background listen to the interviews, followed by a discussion, and transcription that was reviewed by both researchers.

3.3 Research Design: Exploratory Case Study

The research design selected for this investigation was an exploratory case study. A case study research design allows detailed analysis of a single case. In general, a case can refer to either of a single organization, location, person, or event (Bryman & Bell, 2011). Importantly, a case study should be conducted within its real-life context (Yin, 2003). A succinct definition of a case study is given by Dul and Hak (2008, p. 4): *"A case study is a study in which (a) one case (single case study) or a small number of cases (comparative case study) in their real life context are selected, and (b) scores obtained from these cases are analyzed in a qualitative manner"*. The qualitative manner here implies that no statistical analysis is carried out, but it does not refer to the methods of data collection or measurement (Dul & Hak, 2008).

The type of basic design for this case study is a single case (as opposed to multiple cases) with a single unit of analysis (instead of multiple units). A valid rationale for selecting a single case is when the researchers have access to a previously inaccessible situation which was the case with *Software Co*; in this instance the case is revelatory and a single case is appropriate (Yin, 2003). Further, many business researchers advocate using case study designs for exploratory research where there is a limited amount of theory available and the context is important (Dul & Hak, 2008).

This study was theory oriented, which means that the objective was to contribute to theory development, rather than to contribute to the knowledge of one or more specified practitioners. Within theory oriented research, Dul and Hak (2008) distinguish between three types of activities: exploration, theory-building, and theory-testing, which are to be carried out in sequence. Due to time limitations, this study focused only on the initial activity of exploration, of both theory and practice as illustrated in Figure 4.

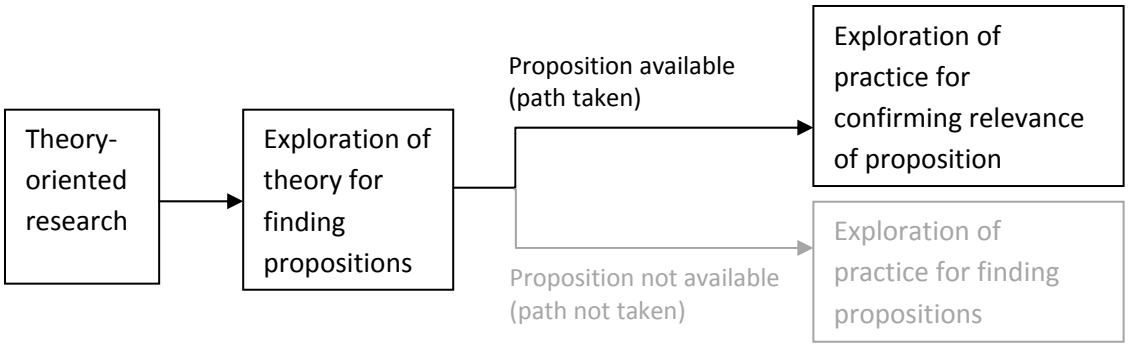

Figure 4: Research design flowchart (adapted from Dul and Hak (2008)

In this thesis, both the theory and practice pertaining to the combination and integration of DT elements and LSD principles were explored. The unit of analysis was individuals acting in certain roles within the development organization at *Software Co*. In order to allow an initial understanding of the topic, the exploratory case study design was selected because it allows a detailed investigation of the topic and to answer the research question. Since the research question is about the perception of individuals in relation to a specific topic, a case study is an appropriate research design as it allows a thorough exploration of the issue. A single case study rather than comparative was chosen due to having gained access to a single company, *Software Co*, early on in the project as well as resource and time limitations. Moreover, *Software Co* provided a unique setting to investigate the integration of DT, LSD and Scrum.

3.4 Sampling

In order to achieve the purpose and address the research question of this study, a purposive sampling strategy was employed within the case. Purposive sampling refers to sampling that is not carried out randomly but rather with the aim to sample participants that are relevant to the research question, and is most commonly used in qualitative research (Bryman & Bell, 2011).

The population for the study's sample was defined as follows: the units in the population had to be experienced in carrying out the role of either SM or PO in the development organization at *Software Co*. In addition, these individuals were required to have been involved in at least one project where DT was used. This minimum criterion of a single DT project was not set to a higher number because DT was introduced at *Software Co* development organization two years prior to this study. Since most development projects have duration in excess of one year, the available pool of respondents who have worked on more than one DT project would have been too small to reach a meaningful sample size.

Once the population of the study was established, snowball sampling was conducted. This strategy was used for two reasons. First, the extent of the entire population was unknown to the researchers, and hence no accurate sampling frame could be determined. Second, initial access to the case was provided by two individuals, who the subsequent sampling was contingent on. Thus, the sampling for this study commenced with two initial contacts with which exploratory interviews were held; one of these two interviewees provided a list of individuals within the population. Further on, interviewees in the population provided further contacts.

3.5 Research Methods

Employing a qualitative strategy implies that the methods used for this study were mainly of a qualitative nature. The one qualitative research method was the semi-structured interview; it is explained along with a description of how they were conducted in the following subsections.

3.5.1 Semi-structured Interview

Semi-structured interviews are a widely utilized method in qualitative research. Since they cover questions relating to specific topics, which are asked in a similar manner to all interviewees in the sample, they provide reasonable comparability of responses. At the same time, the flexibility of semi-structured interviews also allows for new aspects to be discussed as they emerge in the course of an individual interview. Furthermore, emerging aspects can be carried over to subsequent interviews. The emphasis of qualitative interviews in general is to capture the view of the interviewees relating to the topic (Bryman & Bell, 2011).

In the case of this thesis, semi-structured interviews were given preference over unstructured interviews because there was a fairly clear focus, thus allowing the researchers to address more specific issues. Also, semi-structured interviews allow for new themes to emerge that may be considered important by the interviewees. Thus, this method was chosen rather than structured interviews because they were considered too rigid. Thirteen interviews were conducted over the phone with interviewees located either in India or in Germany, while the researchers were based in Sweden. An additional three respondents were interviewed in person during the researchers' visit of the *Software Co* development labs in India. Of a total of sixteen interviews, fourteen were conducted with both researchers present, with usually one leading the interview and the other taking notes and asking additional questions as they arose. Further, all interviews were recorded and transcribed within 24 hours of the interview. The routine employed for interview transcriptions was that each researcher transcribed one half of the interview and then cross-checked the transcription of the other half in order to ensure the quality of the transcription. In three cases, one researcher transcribed an entire interview which was subsequently checked by the other researcher.

3.6 Data Collection

As briefly mentioned above, the data collection for this study started with two exploratory interviews with DT experts at *Software Co*, one in India and one in Germany. These exploratory interviews served two purposes; one being to narrow the topic of the study and the second was to provide some contextual information about the case in terms of the organization and its practices. These two exploratory interviews were semi-structured, and the second one was founded on the first. While these initial interviews mainly informed the researchers about the context and background of the case, knowledge was also gained that contributed towards addressing the research question. Therefore, some aspects of the exploratory interviews are also included in the results presented later on in this thesis.

Following the exploratory interviews, a series of semi-structured interviews was conducted with POs and SMs that matched the criteria for the study's population. In total, fourteen of these interviews were held with three respondents located in Germany and the other eleven in India. Moreover, preliminary findings were presented to and discussed with a group of DT stakeholders at the *Software Co* development labs in India. The outcome from the discussions served as a triangulation means.

3.6.1 Interview Guide

Interview guides were used and complemented with ad hoc questions as needed for both the exploratory and primary data collection interviews. By necessity, the guides for the exploratory interviews differed considerably from those for the actual data collection because a different goal was pursued in each phase. Even the interview guides for the two exploratory interviews were similar only to a limited extent because the second largely evolved from the findings of the first interview.

The interview guide for the primary data collection evolved somewhat after the first interview was held, but no major changes were made. The final interview guide is included in the Appendix and is representative for the majority of the primary data collection interviews, albeit exact phrasing and the order of questions differed between interviews.

When designing the interview guides (for both phases), recommended guidelines regarding the order of questions, type of questions and amount of questions were adhered to. With respect to the creation of semi-structured interview guides, Bryman and Bell (2011) provide the following recommendations, which were adopted by the researchers: logical flow in order of questions, avoiding leading questions, language that is easily understood by the interviewees but not too simple, questions were designed in a way that they can inform the research question without being too specific. The guide was structured into five overall sections as follows:#

1. Introduction of the researchers and the study, assurance of anonymity
2. Factual questions about professional experience
3. Perceptual questions about DT
4. Perceptual questions about the role before and after the introduction of DT
5. Concluding section with an opportunity for the interviewees to raise issues related to the topics that were not covered during the interview

Each section was structured such that questions were arranged in decreasing order of importance, flowed from more factual to more perceptual questions, and also progressed from open towards specific questions. This structuring was done according to the perception of the researchers, which may not necessarily reflect that of interviewees. Furthermore, compromises had to be made where e.g. perceptual questions were deemed very important. In order to keep the interviews flexible, the exact questions and their order varied to some extent between individual interviews; also, spontaneous follow-up and probing questions were asked as they arose during a course of an interview. The interview guide was reviewed and pre-tested with this thesis' supervisor at the Center for Business Innovation, Chalmers University of Technology.

3.6.2 Data Analysis

The collected data were coded and analyzed using the online software Dedoose (SocioCultural Research Consultants, 2013). This particular software was chosen because it allowed real-time online collaboration between the authors and was available free of charge for a limited time. All interview transcripts were imported into the software and codes were applied to relevant excerpts. The first top level code section "Profile & Perception of DT" contains responses that were used to build a context and general understanding of the interviewees' background and perception of DT. The codes under the second top level code of "Differences between DT and LSD" specifically address the main differences identified in the literature that constitute the theoretical framework, and responses

within these codes were used to address the research question. The third top level code "Challenges" was used to identify responses that addressed practical challenges that were mentioned by the respondents to arise from the integration of DT with LSD and Scrum. The code structure is presented below.

- Profile & Perception of DT
  - Definition
  - Opinion
  - Most important aspect
  - Roles & responsibilities
- Differences between DT and LSD (addresses research question)
  - End user empathy vs. customer satisfaction
  - Visualizing vs. backlogs
  - Synthesis vs. waste elimination
- Integration Challenges

3.7 Quality Criteria

Quality criteria are of particular concern in quantitative research. Regarding qualitative research, no clear consent appears to exist amongst researchers about the relevance or exact definition of reliability (Bryman & Bell, 2011). Nonetheless, the main quality criteria pertaining to the conducted study are discussed in the following subsections. In detail described are issues of the study's reliability and various types of validity.

3.7.1 Reliability

In a general sense, reliability in the context of a research study is referred to as whether its results are repeatable. Reliability for qualitative studies can be considered to be either external or internal. External reliability refers to the extent a study can be replicated and internal reliability is concerned with the level of agreement amongst researchers within a team regarding their observations (Bryman & Bell, 2011).

The researchers acknowledge that, due to the nature of the case study, the external reliability is very low. Despite this awareness, there was little though that could be done to improve the external reliability. This is owed to the transient nature of the social setting that was studied at a particular point in time and the single case approach. A replication of the study with the same sample will likely yield different results because the respondents will have made additional experiences regarding the research topic which almost invariably change their perception. Nonetheless, the clear research question and systematic approach should improve the external reliability as much as possible, given the circumstances.

In addressing the internal reliability, the pair of researchers discussed immediately after each interview the main points covered and impressions received. Documenting the main points of these discussions ensured a common viewpoint of the outcome of each interview, especially for later analysis.

3.7.2 Validity

Validity is concerned with the integrity of the outcome of research studies, and various types of validity exist which should be addressed for any business research study.

Measurement/construct Validity

Measurement or construct validity is of primary concern for quantitative studies and refers to whether correct operational measures were chosen for measuring a particular concept. Possible ways to improve the construct validity are to utilize multiple sources of evidence (triangulation) and have respondents review the research findings (Bryman & Bell, 2011). The researchers in this study attempted both methods to improve the construct validity. However, due to the scope of access granted to the case, no other information source than personal interviews could be secured. Furthermore, owing to time restrictions both for the researchers and the respondents, validation by the interviewees could not be secured before the finalization of the study.

Internal Validity

Internal validity refers to the validity of a found causal relationship in quantitative research, i.e. whether an assumed independent variable causes change in the dependent variable (Bryman & Bell, 2011). Since this study was exploratory only and qualitative in nature, no dependencies between variables were devised. Therefore, internal validity in this sense is of no concern for this thesis.

In particular for qualitative research, internal validity can be viewed as the level of agreement between theoretical concepts and observations made. Internal validity viewed in this way can be improved by prolonged exposure of the researchers amongst the subjects being studied (Bryman & Bell, 2011). Whilst undoubtedly desirable, this exposure to increase internal validity was not feasible due to the spatial separation of the researchers and the actual social setting of the study.

External Validity

The concept of external validity addresses the generalizability of the study outside its specific context. External validity can typically be enhanced by using representative samples, which requires careful sampling techniques. For qualitative research, the external validity is usually weak due to the use of case studies and small samples (Bryman & Bell, 2011). This weakness applies equally to the case study presented in this thesis because of the relatively small sample size, and its unique contextual setting. But since the study is exploratory, the goal is not for the findings to be generalizable but rather serve as a first step towards a more thorough investigation of the topic. Further, in the view of the researchers, this study could serve as one part in a larger, comparative case study for which the results would achieve a higher external validity.

Ecological Validity

The ecological validity of a research study is concerned with the applicability of its findings to people's everyday reality. Ecological validity is typically compromised by researchers interfering with their respondents' natural environment or routines (Bryman & Bell, 2011). We cannot accurately assess to what extent the interviews carried out for this study have affected the natural settings the interviewees normally work in, since we do not know what they are. However, since the interviews were exclusively concerned with past experiences and the respondents were guaranteed anonymity, we believe that the ecological validity has not been negatively affected by our research design and methods.

4 Empirical Results

The empirical results presented here are the summarized findings from two exploratory interviews and eleven semi-structured interviews with POs and SMs. The exploratory interviews are presented

separately because each covered slightly different topics, and the second interview built on the findings from the first. The remainder of the chapter is presented according to the structured analysis and coding structure presented in Section 3.6.2 and includes aggregated findings from the sample of POs and SMs.

4.1 Exploratory

An initial exploratory interview was conducted with a DT expert from *Software Co*'s consulting organization. This expert has worked as both a DT coach and facilitator at *Software Co* and is a key person in the implementation of DT at the *Software Co* development labs in India. This first exploratory interview was conducted to gain an overview of DT activities at *Software Co* and to build a general foundation of knowledge on the topic. A second exploratory interview was held with another DT expert who is currently building up a DT team at *Software Co* which will be providing DT consultancy services to customer organizations. This second interview was conducted to further explore the topics and issues that had emerged from the first interview and to narrow the focus of the research, which was deemed too broad after the first interview.

4.1.1 First Exploratory Interview

The DT expert first confirmed that the DT projects included in the roll-out waves are carried out in the development organization of *Software Co* only by stating very clearly that *"the projects are from the development organization",* whereas *"I come from a different organization, the consulting organization"*. As a result, the focus of the research was narrowed to include only the development organization at *Software Co* and specifically focus on Wave 2 DT projects that were conducted in either India or Germany.

The interviewee then elaborated on the team structure for DT projects that were run at the *Software Co* development labs in India. The teams are typically composed of six to seven members with varying roles. The roles were then explained as being: Developers, Interaction Designers, Architect, SM, and PO. In the words of the interviewee the teams are assembled as follows: *"the way they assemble the teams now is, they have some developers, some architects, some user interaction designers, SM, which is coming from the lean method, and then they have the PO and sometimes they may use some people in development also. Normally that's how they assemble the teams."* The interviewee then explained that the SM and the PO typically take on managerial roles on the development team: *"the SM is what the project manager is called at Software Co and the PO is the product manager."* Based on this insight that the managers of development teams at *Software Co* are termed SM and PO, the scope of the research was limited to concern individuals working in these roles.

Furthermore, the interviewee described how during these waves *Software Co* has both *"traditional projects"* and DT projects that are worked on simultaneously by development staff members; *"they have their day job and responsibilities and duties. DT is an additional task given to them, and therefore they are trying to do both."* It was then explained that the traditional projects are requirement based and run according to the LSD approach: *"traditional software development projects are requirement based. There are some requirements coming from the customers, which are then taken as specifications by some people somewhere at Software Co and those specifications come to India and here the developers and architects work together and write the code in the Lean method."* DT on the other hand was stated as being a very different approach to run a project, *"DT is*

completely experimental. We know the topic but we don't know the requirements so they have to go explore and go and meet the users do the research with the users." This comparison and the stated differences confirmed that the proposed topic of this study, the combination of LSD and DT, was relevant and worthwhile investigating.

Overall, this first exploratory interview introduced how DT projects are run at *Software Co* and helped to narrow the focus of the research. The findings are summarized below.

1. This interviewee was able to provide direct access to development team managers who were involved in Wave 2 DT projects at the *Software Co* development labs in India. These projects had been running for about one year and hence there would be ample data to be gathered; moreover, since the projects were just started in the beginning of 2012 they were still relevant and fresh in the memory of prospective individuals in the study's population.
2. These projects were primarily conducted in India and Germany and thus the data collection would be limited to these two locations.
3. The most important learning was that the software development teams at *Software Co* are composed according to the Scrum method with POs and SMs as the team managers. This helped to further develop the specific manger role as this thesis' perspective.
4. There is a distinct difference between DT projects and traditional projects at *Software Co*. The traditional projects are run according to the LSD approach, which are considered very different than the Wave 2 projects that use DT. This was interesting since POs and SMs are required to work simultaneously on both types of project. Therefore, it was decided that this research would investigate the difference between LSD and DT approaches from the perspective of POs and SMs.

4.1.2 Second Exploratory Interview

The second exploratory interview was conducted with a DT expert at *Software Co* who works on diffusing DT at *Software Co* into areas outside of software development. This individual gave credence to how DT is being used at *Software Co* in areas outside of software development by stating *"Software Co has a lot of different design thinkers within it, globally. And we have a DT initiative coming from the sales force tribe, we're getting it from teams who are developing custom software for our customers and it's not only for product development which the central initiative was set up to do."* Since, as described by this expert on the matter, DT can be used differently in a multiple of different settings, it was very important for this research to focus on a single department in order to generate a specific contribution. Since the combination of LSD and DT was considered to be of most interest, the product development department was selected as the focus for this thesis.

The interviewee then explained the distinction between Lean and Agile at *Software Co*, as well as mentioning that most projects are run according to these approaches rather than DT. When asked whether most, if not all projects at *Software Co* are run according to Lean, Agile and Scrum principles, the response was: *"That's true. I was afraid you would say that all projects are run according to DT principles – and that would not be true."* The interviewee explained the distinction between Lean and Agile as *"you have to make a difference between Agile, which is, we use Scrum as an Agile methodology, and Lean. Lean has much more of the component of how people work together and trust on teams and not just on upper/middle management."* Furthermore, the respondent clarified that DT is not seen as a replacement to Lean and Agile, but should be more seen as an addition, *"...I*

would be very cautious not to promote DT as something completely new and different, so that people don't go oh my god we've got to throw out everything we just learned and now start new, but there are a lot of similarities that you can use."

The interviewee proceeded to hypothesize that it might be more challenging for POs than SMs to adapt to DT. Underlying this hypothesis is that POs have historically been responsible for the product and have a harder time taking a step back to mentor or facilitate the DT process that involves a shared team responsibility for the product. In arguing for SMs being more equipped to run a DT project the interviewee made the comment that *"SMs are less partial. They can be more, they can I think facilitate; it is the classic problem of being a facilitator and participate at the same time. SMs tend to monitor and support the process rather than feel responsible for products that are happening."* These comments helped to emphasize the importance on understanding the role of POs and SMs on DT projects. There was a clear indication that the roles given from the background in LSD had an effect on how DT projects are managed by the individuals filling these roles.

An interesting aspect concerning Wave 2 projects was brought forward by this interviewee that in stating that they were conducted in both India and Germany. The DT expert referred to a classical problem in which *"we have people in Germany that feel responsible for the product and then in India we have a very, very large, pure development organization. Just because of time and cost it is hard to create projects that involve everyone."* This impact of having an offshore development team in India working on DT projects where the different stakeholders are in another country was seen as a significant issue in the implementation of DT. However, given the scope and access for this research these issues of cultural differences, although potentially significant, were not considered the focus of this study. However, in terms of waste elimination and customer involvement, the spatial separation issue was explored in a logistical context.

Finally, this DT expert brought up two probable challenges for Scrum team managers concerning DT and its combination with LSD and Scrum by saying that *"POs have a very clear idea in their head of what they expect and what they would like the solution to be; part of DT is to involve more people with different backgrounds in that definition phase and because you have a PO who is used to have the responsibility for defining the product it is very hard for him or her to go back to the team and stay open minded and involve everyone openly. I think that is on one hand a real problem. I think the other problem that we have is that Agile and Scrum is there to optimize how fast we produce something with quality."* While the first part of this statement directly addressed challenges for POs, the latter part about the optimization was interpreted to be more applicable to SMs because they are concerned with efficiency and execution. Therefore, this statement substantiated the research direction to study the combination of LSD and DT from the perspective of both POs and SMs since the potential challenges identified in the literature review to them was confirmed by this interviewee.

4.1.3 Summary of Exploratory Interviews

These two exploratory interviews helped to narrow the focus of this research to focus on DT from the perspective of POs and SMs. Both POs and SMs were hypothesized by the two respondents to face challenges related to the implementation of DT. Moreover, it was learned that the projects at *Software Co* used to be primarily based on LSD principles and the Agile method Scrum, and that with the introduction of DT there was an addition rather than replacement effect. Finally, the idea

emerged that having a spatial separation between customers, end users, and DT teams could be problematic. Although there may be a cultural aspect associated to the spatial separation the cultural aspect , cultural difference was decided to be outside of the scope of this study.

The following sections present the empirical findings that were gathered from interviewing both SMs and POs concerning the integration of DT with LSD and Scrum. These interviews were motivated by the outcome of the two exploratory interviews and the literature review.

4.2 Profile of Product Owner and Scrum Master

In each semi-structured interview that was held with both POs and SMs, information was collected to gain a better understanding of their roles and responsibilities for software development projects at *Software Co*.

4.2.1 Roles and Responsibilities

In general, a Scrum team at *Software Co* was described as comprising five overall roles: PO, SM, Architect, Quality, and Developers. An interviewee stated that *"…we have these three or I think overall five roles inside a Scrum team. One is the SM, the other is the PO, third being the Architect, there is also a person involved on the quality side, that is the fourth role, and of course the regular developers. So every team in Software Co, every Scrum team in Software Co will have these five roles."* However, another interviewee stated that on a Scrum team the *"PO, UI designer, and Architect they are the key people."* This interviewee identified a new role, UI designer, which could be considered to be outside the five previously mentioned Scrum team roles, showing that there is to a degree some ambiguity and flexibility on the roles within a Scrum team at *Software Co*. However, all interviewees included both a PO and SM when describing a Scrum team and these roles are the focus of the data collection.

Scrum Master

The main responsibility of a SM on LSD projects at *Software Co* concerns the daily execution of the project. One interviewee plainly described this role by stating *"Execution is my responsibility as a SM"*, while another interviewee echoed this thought by stating that for a SM *"….the focus is more or less on the execution part, meeting the deliverables, getting it validated with the customers at the end."* This same thought was stated again by describing the role of SM as *"the role is to deliver, execute, and meet all the timelines, the processes within Software Co, and also validation with customers at the end."*

In terms of the backlog, which contains the requirements for the project, the SMs' responsibility involves executing the backlog by delivering the requirements within in a certain time frame, usually a sprint. However, the SM does not have much, if any, involvement in gathering the requirements or creating the backlog. A SM stated this by saying *"For me, I just know there is a backlog and I have to get it delivered from the team by this date."* Therefore, the SMs' responsibilities include delivering the requirements in the backlog but not defining or gathering them. Additionally, the SM is responsible for validating these requirements and resulting software releases with the customer at the end of a sprint, but this responsibility does not include questioning if these requirements are the right requirements for the project.

Overall, the role of SM on LSD projects is best described as being a process gate keeper for the project. An interviewee concisely stated this by saying, *"The SM role is basically like a process gate keeper trying to ensure everything is place, and there is smooth functioning of the team."*

At its core, the role and associated responsibilities of a SM for software development projects at *Software Co* does not seem to change drastically with the introduction of DT. However, there were still some notable changes in the SM role. These changes were generally seen in the beginning of a project with less of an impact in the later implementation or execution phases. This is because with the introduction of DT, the overall structure of the Scrum team shifted by becoming less hierarchal and more democratic in nature. The shift to a more democratic team is described by an interviewee as, *"...with DT, all of us did the customer interviews, all of us understood the problem, we were all at the same level when all the problems had been identified."* Another interviewee supported this perception by stating: *"All the roles are the same here (DT). You just go to the customer or the end user and collect the requirements, then come back where you all sit together, brainstorm, do the ideation phase, come up with the design yourself and then implement it. So you'll be on the same page, the entire team will be on the same page."* This idea that introducing DT changed the overall structure of the Scrum team from a team with individual siloed responsibilities to a more democratic team without specific roles was a common theme given by all interviewees.

This change to a more flat team structure without defined roles impacted the role of SM by widening his or her responsibilities to include jobs outside of execution and delivery. An interviewee illustrated this change by stating *"but with DT I think it has changed. You are part of the team, you are already doing it, you are collecting the requirements yourself by observing them (end users), you are validating it with them as a SM also, and you meet the end users."* Another interviewee emphasized this statement by saying *"as a SM, I just know there is a backlog and I had to get it delivered from the team by this date. That is all that used to matter to me (before DT), but now I know in length and breadth the entire backlog and its background."* It is clear from these quotes that SMs' responsibilities did not so much change but instead were widened by participating in defining the problem statement, by being involved in the collecting of requirements, as well the ideation in the beginning phases of the software development project.

Product Owner

The main duty of a PO on LSD projects was described as being responsible for the overall content and product in the project. This high level responsibility was described by an interviewee as *"our (PO) role, as we understand, is that we are overall responsible for the product, so you have a lot of problems, you need to live up to a certain time, you need to deliver a certain scope, you need to take care of your budget, you need to convince customers to buy it."* Another interviewee emphasized this statement by saying *"I (PO) am not a line manager. At Software Co we have a separation, so people are content responsible and other people are people responsible… I am responsible for the content for the team."*

It terms of the backlog and requirements, the PO in LSD projects is solely responsible for working with both the internal and external stakeholders to create the backlog and define the requirements but does not have a daily execution responsibility of the backlog like the SM. An interviewee stated this by saying: *"A PO is responsible for making the backlog items for the product – for the development, and to clarify the requirements and all this stuff. But generally the PO is not really*

responsible for the day to day execution." This backlog responsibility is described in another way by an interviewee as: *"So I am also responsible for defining the so called backlog; so I am defining what needs to be done, and needs to be controlled, what happens after its done, if what the team is doing fits to a customer need, and being overall responsible for the product – that is my role."* Overall, the leadership of a PO on a Scrum team can be defined in terms of their responsibility of defining the requirements, creating the backlog, and passing this on to the members of the Scrum team. An interviewee supported this statement by saying *"leadership, from say the PO, is in terms of having the functional knowledge and giving the requirements."*

The PO, overall, has a high level responsibility by having ownership of the product and content, or essentially the functional knowledge for the entire LSD project. It is the PO's responsibility to create and define the backlog that will be then given to the Scrum team to execute and deliver in the sprints. The PO in a LSD project at *Software Co* was the sole person that worked with internal and external stakeholders including customers to define the problem statement and what requirements are needed, but this sole responsibility shifted with the introduction of DT.

This shift in responsibly with the introduction of DT appeared to be impact the PO role the most by on software development projects. The PO's core responsibility for the overall product and content did not change but due to the previously mentioned shift in the Scrum teams' structure, the PO role was impacted since responsibilities such as gathering requirements changed from being their sole responsibility to a shared responsibility with the introduction of DT. Both POs and SMs interviewed acknowledged this shift by stating: *"…if someone's role is impacted the most I would image it is the PO"* and *"it is a fundamental difference because in the earlier projects (LSD) I, the PO, was the single point of contact…..but here the entire team is equally responsible for getting the requirements."* Another interviewee stated that the PO could be the bottleneck in LSD software development project because they were the single contact for the Scrum team to clarify the requirements, but this changed in DT since *"everybody is at the same level, there isn't this knowledge gap."*

Moreover, the responsibility of working with external stakeholders such as customers or end users to determine the requirements for the backlog in LSD projects in terms of the Scrum team rested solely with POs. With the introduction of DT interaction with external stakeholders became a shared Scrum team responsibility. An interviewee stated that, *"with DT, all of us did the customer interviews, all of us understood the problem, we were all at the same level when all the problems had been identified. So there was no necessity for the PO to explain the problem statement and requirements and get the buy-in from the development team, it was already there."* Another interviewee stated: *"But what has changed is that the ones who participate in DT as a team, all the people who are part of the team, they more or less understand the requirements or what needs to be built into the solution. It's because everybody was involved in the customer interviews and so on, so they really understand what the customer is looking for. Previously this was only known to the PO."* Since this was one of the core responsibilities that belonged to the PO in LSD projects in terms of the Scrum team, an interviewee even described the effect of this shift in responsibility to mean that *"everybody is more or less, I would like to put it, say in a small way a PO because we understand the requirements ourselves."*

Summary of Roles and Responsibilities

In summary, the role of the SMs did not change significantly with the introduction of DT, but some differences were noted as shown in Table 4. In LSD only, the SMs were concerned only with the

project execution, delivery of backlog items and validation of software releases with customers. With the addition of DT, the SMs' duties expanded to also include involvement in problem definition, ideation and requirement collection.

Prior to DT, POs were the only team members who had an overall view of the software that was being developed. As such, POs were responsible for defining the problem statement and creating the product backlog based on requirements they had previously gathered from customers. Furthermore, POs were in charge of being the communication link between the Scrum team and other stakeholders. The role of POs shifted somewhat with the introduction of DT; subsequently, the entire team was involved in collecting the requirements from end users and defining the problem statement. Hence, some of the POs' responsibilities were distributed to the entire team.

Adding DT to LSD projects altered the Scrum teams in such a way that the structure was less hierarchical and decisions were consequently made in a more democratic fashion. In LSD only projects, each team member had clearly assigned duties and tasks to complete but in DT this had changed to a flatter structure without specific roles, at least during the duration of the DT phase.

Table 4: Summary of roles and responsibilities

<table>
<tr><th colspan="2">Scrum Master</th><th colspan="2">Product Owner</th></tr>
<tr><th>LSD</th><th>LSD & DT</th><th>LSD</th><th>LSD & DT</th></tr>
<tr><td rowspan="3"></td><td>Problem definition</td><td rowspan="2">Communication link between stakeholders and Scrum team</td><td rowspan="2">Increase in interaction with entire Scrum team and stakeholders</td></tr>
<tr><td>Ideation</td></tr>
<tr><td>Requirement collection</td><td rowspan="2">Sole ownership for overall product on Scrum team</td><td rowspan="2">Still owns product but more input on decisions from Scrum team</td></tr>
<tr><td colspan="2">Daily project execution</td></tr>
<tr><td colspan="2">Delivery of backlog items</td><td>Defining problem statement</td><td>Increase in sharing of responsibility</td></tr>
<tr><td colspan="2">Validation of software with customers</td><td>Creating backlog</td><td>Increase in sharing of responsibility</td></tr>
</table>

4.3 Perception of DT

In addition to the interviewees profile information, data was gathered concerning the POs' and SMs' opinion of DT and what they considered the most important aspect of DT on software development projects.

4.3.1 Definition of Design Thinking

Interviewees defined their understanding of DT by emphasizing two characteristics: (1) user centric and (2) problem finding. The user-centric characteristic of DT was repeatedly mentioned by interviewees by statements such as *"DT is a technique wherein you look at a problem from the perspective of the end user"* and *"DT is a very user centric approach and it also gives a lot of importance on building quick prototypes and getting it validated, so there is sufficient time to understand the user's perspective and trying to build something for the users and get it validated from them, so if I have to put it in one word it is a very user centric and prototype driven approach."*

The second prominent characteristic that interviewees mentioned when asked to define DT was its focus on understanding or finding what problem was to be solved rather than execution or actually solving the problem. For example, one interviewee stated that *"LSD is more for execution and DT is more for solution perspective where you have to come up with an idea and come up with the solution for it."* While another stated that, *"Lean is something which is more into execution, how you are doing things, how to do it... ...DT is more of learning what to do."* Yet another interviewee eloquently described DT as *"a structured art wherein you creatively try to find out what to develop."* Overall, these were the two main characteristics user-centric and problem finding that emerged when asked to define DT.

4.3.2 Opinion on Design Thinking

When first asked, all interviewees quickly responded with a favorable opinion of DT but this was not given without later mentioning limitations and challenges associated to the approach. The generally positive opinion about DT revolved around comments that pointed out how DT (1) helps to solve the right problems, (2) increases innovativeness, and (3) decreases stress in particular for POs through improved communication, understanding and responsibility sharing.

In terms of having a positive opinion because DT helps to solve the right problems, interviewees made comments such as *"I love this technique because it ensures you are solving the problem that the end users want you to solve, and not solving it the way you think should be solved"* and *"DT and Lean complement each other. Lean focuses on doing the things right and DT focuses on doing the right things."* Interviewees that had positive opinions due to the increase in innovativeness made comments like *"I feel it is very good. It helps us innovate better and it's a win-win situation, both to the development and the customers."* Another interviewee stated they had a positive opinion of DT because it helped to decrease their stress and related pressures by saying *"I think it makes my life a little bit easier. First of all there is lots of knowledge, lots of talking, lots of customer contacts, and lots of exposure to the real business world. And I think these are the gains of DT for me a PO and this was not there so much before."* Another interviewee highlighted this opinion by stating that DT not only decreased the stress on the team but increased the productivity: *"DT with the right mix of creativity and work this was almost like a stress buster, we never thought we were working for some many hours, and that I think leads to a high productivity I feel because the team is motivated and full of enthusiasm there."* A further contribution of DT to make the PO's job easier arose from improved communication and understanding, and being able to share the responsibility around defining the problem statements and collecting the requirements. One interviewee stated that *"…what becomes a lot easier is definitely the fact that people are clear of the requirements. And once people are clear of the requirements, then a lot of pressure is taken off you (PO) and you can focus on the next steps and the next….so that definitely helps because then the pressured is reduced a lot on the PO."*

Limitations

As mentioned above, although the overall opinion of DT was favorable, many interviewees set firm limitations on the use of DT. The limitations included that DT must be done in the right way, should not be considered the solution for all problems, and should be used for abstract rather than mundane problems. The first limitation mentioned is that the DT must be done in the right way. Interviewees made statements such as *"I think it's a really good approach and it always works if it is followed the right way, irrespective of the domain it is applied in"* and *"I would say it is very helpful if you use it in the right way but you really need to find the right structure for that."* Another limitation

to the generally positive opinions about DT was that DT is good, but it is not for everything. Interviewees stated that *"DT is very helpful but it is not the answer for all questions that we have in the software industry."* Moreover, interviewees' opinion of DT is good when it is used in a context where there are abstract rather than mundane problems. For example, one interviewee said that *"DT is really nice when nobody knows what to do, what the problem actually is."* While another interviewee stated very plainly that, *"it is very important or very helpful, especially if you have abstract problems."* The respondents' opinion of DT was by and large positive, but limitations of the suitability of the approach were also pointed out.

4.3.3 Most Important Aspect of Design Thinking

Interviewees were asked to state what they considered to be the most important aspect of DT in relation to their role. The overwhelming response to this question was the focus on the end users. Interviewees clearly stated this by saying, *"I think the most important feature of DT is that you put the end user in the picture and then you start looking at the whole process from his perspective."* Also, *"it was validation also by the customer and end users; that is the main important thing here. Because usually when we also visit customers in other areas or in other methodologies, we visit the customers and not the end users. Here it is mainly focusing on the end users and validating with them"*, and *"for me this is important, the 360 view."* It is clear that both POs and SMs at *Software Co* perceived the interaction and contact with the end user to be the central aspect of DT.

4.4 Design Thinking vs. Lean Software Development

Before presenting the specific data that relate directly to the theoretical framework given in Section 2.6, higher level data trends that emerged during the interviews are presented below.

One interesting observation made by the interviewees was the more direct and hence faster flow of information between the customer and developers on the Scrum team after the introduction of DT. In the LSD projects, communication was stated to pass through three or four instances between the developers and customers, but since DT required the direct interaction of involved persons, a direct link was established to facilitate an accelerated information exchange. This improvement was clearly expressed by one interviewee as: *"the turnaround time was not very good (in LSD), whereas in DT I write an email directly to the business user and get an answer or a document explaining the business process; so it's kind of faster".* It appears that the shift towards user empathy, which necessitates direct contact between Scrum team members and end users, has created the additional benefit of having queries answered directly and faster than in LSD projects. Overall, the focus on end user empathy in DT generated a beneficial side effect in the form of shortening the response time for queries by the Scrum team.

Related to the above data concerning direct contact with end users, was a clear pattern of positive response regarding the commitment of Scrum team members to the DT project. Since the entire Scrum team is involved in gathering requirements from the end user, all team members have a clear understanding and buy in very early in the process, rather than having to be convinced of the benefits of their work. This view was expressed as: *"there is an immediate buy-in from the PO and the development team, they already know that this is exactly the problem which the customer faces and we have to solve this. So the buy-in is better and immediate in the case of DT. It is some more work compared to the traditional approach (LSD) but it is more effective."* Another interviewee described it as: *"since I have been involved from day one when this particular topic evolved and I have interacted,*

I know the background, I know the entire backlog, so for me it is easy to understand what is going on and it really gives me good confidence." From the perspective of both SMs and POs, the benefits of having a committed Scrum team at an early stage outweigh the drawback of more resources and time needed for gaining empathy with the end user.

Another difference between the LSD and DT approaches that emerged in the data was in terms of the point in the project lifecycle where customer feedback is received. Customer focus was considered important both in LSD and DT by interviewees, but the point of time to consult customers differed. In LSD, this occurred after each sprint with useable software releases but in DT feedback was received before any actual software development occurred: *"We also have customer focus in LSD, in the way that we prioritize the backlog, in the way that we do continuous improvement, in the way that we have a useable software after every sprint, and then go to the customer, come back, and things like that. So the focus is the same, but building a low cost prototype for early customer validation is something which is different here."* It thus becomes evident that the timing of customer involvement and feedback has been shifted forward with the introduction of DT into LSD projects.

From a high level view, the data indicates that with the introduction of DT there is a stronger focus on the end user and the entire Scrum team is actively involved in this empathizing.

In summary, the perceived improvements with the addition of DT to a LSD project are an improved response time for queries to and from customers and users, earlier and stronger commitment from all Scrum team members, and earlier feedback from customers. An overall negative aspect of the addition of DT to a LSD project that emerged from the interviews was that DT requires more time and resources to gain initial understanding; however, this disadvantage was considered tolerable when compared to the benefits.

4.4.1 End User Empathy vs. Customer Satisfaction

It becomes clear from the previous section that the focus on end user empathy in DT has profound effects, even on aspects that on the surface seem unrelated. In this section, the interviewees' responses that are more specific to the end user empathy in DT are presented and contrasted with the LSD principle of customer satisfaction.

Terminology

One particular finding that emerged relates to the terminology of *end user* and *customer*. Some of the respondents referred to customers initially, but after follow-up questions prompted by the interviewer, revised their statements and conceded that it was actually end users they referred to. Some statements illustrate the difficulties in expressing the differences: *"In LSD you are kind of, you are talking to customers in a different way. In DT it's kind of similar but it's a more formalized approach I would say."* Another interviewee said *"but the most important thing I am seeing out of DT is the focus on costumer"* and stated within the same answer, seconds later: *"So with DT now we are trying to put more focus on the end user."*

On the contrary, a number of other respondents were very clear on and stressed the distinction between end users and customers, such as the following responses: *"There is really a big difference between customer and end users"* and, *"it is very important to separate between the customer, and by customers I mean the project leads and architects, and really the end users."*

Another respondent seemed very clear on the focus of DT, but somewhat confused about the definition of customer in LSD, as was expressed as: *"I think in LSD it's customer as the customer. I think we also have some end users probably, and some might not be the end users. But here with DT it is strictly end users."*

In essence, the clarity in terminology and meaning appeared to be slightly differing from person to person but after probing by the interviewer the result was quite definite in that for DT, end user refers to the actual people using the final product. Whereas the term customer as used in the LSD approach, can include any person that is representing the organization who may purchase or use the final product.

Contact Persons

After clarifying the initial difficulties in terminology regarding end user and customer, the focus during the interviews was placed on the differences between the end user in DT and customer in LSD, and whether any difficulties arose from these.

In the DT approach, the targeted contact person is much more specified in DT as expressed by one respondent: *"More or less the same people. But what happens is that in LSD you don't mind talking to the IT guys of a customer. But in DT you try to access end users, so there are some differences."* A second interviewee provided a very similar response by saying *"earlier we used to basically talk to the business users or the IT head or the person responsible for the product development department and things like that. But now we're trying to reach out to, say one level down, which is the end users who are actually using the software."* It is clear from these responses that in DT the contact person is specifically the end user.

Very little of the identified theoretical misalignment given in Section 2.6 was actually perceived by the interviewed POs and SMs. Rather, being in contact with end users was an addition to talking to the *"usual customers"* in LSD: *"DT basically focuses more on the end user but I don't see a real conflict between LSD and DT in that sense. You basically focus on the end user, but at the end of the day you also have to discuss with the other stakeholders, like the IT guys at the company and maybe others, like management people as well."* Along these lines, there was some adjustment to be made in terms of the contact person but no conflict arose out of the elementary differences between DT and LSD. As mentioned, communicating with the end user is seen more as an additional step to gain additional and different insights.

One interviewee mentioned the surprise of end users when they were being interviewed by Scrum team members trying to collect information. Because these end users are not accustomed to talk to developers of software they are using, their reaction was described as: *"Why are you asking me these questions? You are from Software Co you need to know what I need..."* Again, this does not relate to any adjustment the PO or SM has to make but is originating externally from the end user being unfamiliar with DT. Although this issue was only expressed explicitly by one interviewee, it is closely related to the larger problem of gaining access to the end user, which is presented later on.

End User Contact before Design Thinking

A number of respondents stated that it was part of their previous way of working to be in contact with specific end users, rather than just unspecified customer staff. In a way, they claimed, they were doing DT before its formal introduction at *Software Co*. As two interviewees independently

expressed quite bluntly, *"for me this is more like common sense, so I thought I've been doing DT all the time"*, and *"I think the whole process is not new to me because since many years we have been looking at what the end users are doing, so DT was not new to me."* However, the approach is much more formalized with the introduction of DT to LSD projects and seems to simplify matters for at least one PO: *"I think with DT it becomes a lot easier because now it is structured."*

Even though several interviewees, POs in particular, stated that they had been in contact with end users during development projects run according to the LSD approach, they appreciate the structure the DT approach provides to the activity. Regardless of the efforts to get in touch with end users were made in LSD or DT approaches, the most pronounced problem faced by all respondents was to gain access to the actual end users which is presented in the next section.

Access to End Users

There was common agreement of all interviewees that getting access to the end users was the most difficult part to achieve in DT (also in LSD, if end users were consulted previously). Although the problem of access was not entirely new to some of them, it was stated to be more pronounced in DT. Whereas previously, gaining access to any customer representative was not easy, getting access to very specific staff amplified the issue. This is reflected in the following statements: *"Getting access to end users is the challenging part, so DT is definitely more challenging."* Also, *"it is not easy to get the right people on the table at the customer, the customer's IT department usually blocks that because they would first like to understand if what you would like to discuss with end users is something relevant for them or not. So you usually need one meeting to convince the IT; then, if you are strong enough you can access the end user. You never go directly to the end user."* Another interviewee stated that *"all these end users have daily work and they do not really have a lot of time. Especially the people in the business area need to do their daily business, and then to discuss with a guy from Software Co how to do the best thing – it is really tough to get these kinds of end users."* This type of statement was encountered again and again throughout the interviewing process, and some respondents even contacted friends in their personal network to gain access: *"Whatever customers we have met is through our personal contacts, so a friend of mine is working in a company so we had to approach him, that person, to get an interview with their end users".*

While this problem of access is not directly related to the theoretical misalignment of DT and LSD, it is a practical challenge that originates from the misalignment. It highlights a very common, practical issue that is being faced by both POs and SMs on Scrum teams at *Software Co*.

Benefit of End User Contact over Customer Contact

The widespread difficulty of getting end user access could logically prompt the question of what the actual benefit is of talking with end users compared to the usual customer representatives. Interviewees, in general, considered the end user contact more beneficial because the outcome is products that are more desirable and ultimately provide more value to both the customer and end users by addressing actual user pain points. Most interviewees could not yet judge the success of their DT efforts in terms of monetary measures; however, their general positive perception was based on the early and continuous feedback received from both end users and other customer representatives.

This view was expressed in a number of statements such as: *"The other positive that is quite obvious is that because you are talking to end users, you are re-validating with the end users, this ensures*

that what you are developing is at the end what the customers wants, so that benefit is definitely there", or *"they give me feedback on what I have done and what I need to do, then I think I'm on the right track."* Regarding the actual customer problems as opposed to the customers' perceived problems, the end user pain points need to be understood: *"It is very important to have this end user empathy also ingrained into the process somehow, so we don't lose the actual requirement, or the actual pain point"*, and *"understanding the customer's pain point is most important. Because that is how you prove to yourself that you are always on the right track. And that is the next very important thing, always validating, building early prototypes, showing and visualizing, to end users."* Also, getting to the pain point requires an altered approach: *"not asking him what does he want, but rather asking him what does he do with the business process and trying to evolve from there into what is it that would address his pain point. So the major difference is not asking the end user what does he want, because then you get a wish list, but rather trying to observe and gain the insights".*

An additional perceived benefit of being in contact with end users is the improved communication and associated reduced information loss, because previously *"at the end of the day what we would deliver was maybe just some 70% of what the customer required, because there was various information filtering at various levels."* While the filtering occurred at both sides of the table, the end user contact certainly reduced such problems as expressed in *"don't always trust what the IT people tell you, go to the end user, and observe what they are doing. Don't always trust that the IT people tell you what the end users are doing."*

Team Empathizing Remotely or in Person

Another difference noted in DT compared to the previous LSD approach was that the end user empathy and understanding should be gained in person by the actual Scrum development team. Previously, it could be a *Software Co* representative collecting information and passing it to the development team; alternatively, some interviewees collected information from end users, but remotely and in a non-structured manner. Both strategies have shortcomings which were, at least to some extent, stated by interviewees to be eliminated by DT.

The first issue of an *Software Co* representative external to the Scrum team collecting information is information loss on the *Software Co* side rather than the customer side. An interviewee described this loss in LSD as *"in the earlier development model (LSD), there is a solution management team or somebody somewhere who would go and interact with the end user, do some market research and let the PO or somebody know that this is a requirement and this is what needs to be done, and then the PO would start drilling down on those topics ... But with DT the team goes and talks to people directly and tries to understand their concerns , tries to observe how they are doing things, what is it they are lacking or missing. This direct interaction and observation helps everyone understand the process, and everyone is on the same page when we come back, and we don't have to start all over again when we try to create backlogs and realize these backlogs into solutions."* The outcome of the co-located empathizing by the DT team appears twofold; as mentioned previously, no information is lost in the transmission and the team buy-in is positively affected, as is also reflected by this statement: *"But now every single person knows what the benefits the customer is going to get out of what he or she is developing. So that is a very big, very nice advantage of going out and observing the end users."*

One interviewee, who was in contact with end users before the formal DT introduction, mentioned that *"the biggest difference is the emphasis of doing this in person. So either face to face, or at least have a video connect and have this discussion, which brings a lot of human angle to it, therefore we can see the end users and know: is he happy, is he not happy."* In a few cases interviewees reported to have had no other option than to gain end user insights remotely, despite trying to follow a formal DT approach. This was an outcome of spatial separation from the end users and travel budget restrictions. In these cases, DT was seen as being far less effective: *"we lost the empathy when we did it remotely. We have to actually see them, talk to them. They were different when we met them in person than on the phone. They had a lot of things to discuss and to show when we were in person, it was good. We were able to observe them which we weren't before."* On the other hand, another PO stated the necessity of this approach as *"you don't have the travel budget, you need to make a few web sessions or calls, and so you need to have a kind of pragmatic approach."* Ideally all end user research is done in person, but in reality it may not be feasible.

These accounts highlight the importance of doing the end user research in person and that it is being done by the Scrum team, the persons who will create the product for the users. On the other hand, some downsides and desired prerequisites were stated for the Scrum team travelling to the customer such as: *"The entire team can go and talk to customers but my only concern would be, the people who are going to customer interviews should be knowledgeable. At least they should go with some basic knowledge of the industry or line of business of the customer they are talking to."* The need for having a deep knowledge of the customer was pragmatically dealt with as: *"It is very important that we have a stable team going to the customer, it makes no sense to always mix the team that goes to the customer"*, because *"we tried this out in different ways but we stepped back because it was too confusing that always different people go to different customers."* From these statements it is evident that this particular respondent had established an *empathy sub-team* within the DT team.

Which End Users and How Many?

Another theme that emerged during the interviews within the topic of end user empathy revolved around questions of which are the right end users to talk to, how many are needed and how much time should be spent on it. These issues are rather practically oriented as opposed to having theoretical relevance in perspective of the framework for this thesis; nonetheless, they were considered important by the respondents.

The impact of whether the researched end users are actually the right ones was most profoundly expressed by this statement: *"a huge question where you can you can fail with DT is: do you really talk with the right end users? And this is really tricky, so this is one thing where DT is a problem."* This shows that the risk here is, of course, that a great product is being built but for users who may not be associated to customers who buy it, hence it carries commercial repercussions.

Related to the quality of end users researched is the quantity, i.e. how many should be researched in order to get good enough insights to build a good product. One interviewee suggested that establishing certain guidelines such as: *"you should have some criteria in place for DT, minimum of this many interviews, this many customers and so on – there should be a minimum criterion. Once we do that and we have good enough interviews and good enough insights, then it gets easier. But if you couldn't get that many appointments with end users, then my personal feeling is that you shouldn't*

go forward." This can be very much regarded as a statement concerning the confidence in the gathered data from a quantitative viewpoint which parallels the previous paragraph in terms of confidence in the quality of end user data.

The consideration of how many interviews or observations are sufficient has to be practically balanced with budget and time restrictions, a point touched upon before. Of course the respondents were aware of the increased time required to communicate externally: *"perhaps you have more contact with customers and end users. So to manage that is perhaps not new, but you spend more time on it."* Realizing this requires adjustments in planning projects, such as *"you need to manage it, you cannot always make a department trip with the full team to a customer. If you interview two people, you cannot go there with five people. You have to balance it. And you can also not go to twenty or fifty customers; you don't have the travel budget."* It appears that the need to find the right compromise between quantity and quality of end users researched, and the way to do it, is there and has been recognized. However, a final solution does not seem to be in place as yet.

4.4.2 Visualizing vs. Writing Backlog

This section presents the findings that relate to the topic of visualizing in DT in comparison to the LSD practice of writing out requirements in product backlogs.

The common pattern that emerged from the interviews was that the DT practice element of visualizing is not in conflict with writing requirements and product backlogs in LSD. Differences are perceived nonetheless, but in a positive way that facilitates the creation of better products: *"this is a big difference to the old world where we were always writing this huge specification, word documents with hundreds of pages. With this new process you are working more with prototypes, power points, more with paper mock ups, so this mock up style is much, much more important than the specification."* This view was shared by many respondents: *"I think actually visualizing the problem and then trying to see what we want to do, where we want to reach by the process of visualization is more important than just listing backlog items."* An additional benefit was seen in that visualizing facilitates communication: *"Building storyboards and prototypes can help the message to come across and drive the ideas with executives."*

Much rather than being misaligned, the visualizing in DT was seen as a precursor to backlogs, just as DT is usually preceding actual software development at *Software Co*, which is still executed in the usual LSD mode, as the following statement illustrates: *"we have made better experience when you are in a research phase with DT than in a development phase. So this is because you change many things then, new things come in, this creative phase fits much better if it's only for the research phase. DT is very questionable when the development phase has started; I am against that – that is not working"*. The visualized findings are then translated into written requirements, which are still necessary for the development in LSD mode: *"the PO has the responsibility to map all this unstructured information we have in the form of diagrams, maps, and designs into structured requirements. It has made my work easier, because as you go through the journey the process is more or less clear in your mind of what is expected and was it not, so just putting this down in form of a document is not extremely difficult."*

Realizing that the visualizing can effectively precede writing of requirements in backlogs creates a shared view of its benefits. Moreover, the backlogs were stated to have in some cases evolved from the visual artifacts: *"we made a very rough prototype of what the problem is and where we want to*

go to. It was still not a solution. But from all this then, we designed the backlogs and there are no more missing building blocks." The complementarities were even expressed directly: *"they complement each other".*

The potential disaccord between DT and LSD that originated from the DT practice elements of visualizing and writing product backlogs was not perceived by the interviewees as a significant misalignment. Much more, the addition of visualization to their work was seen as a benefit that helps problem identification, facilitates communication, and makes the creating of backlogs easier.

4.4.3 Synthesis vs. Waste Elimination

The third theoretical misalignment between DT and LSD concerned the DT practice element of synthesizing. This DT element is in theoretical conflict with the underlying LSD principle of waste elimination. Owing to the centrality of waste elimination in LSD, this identified discrepancy was confirmed by most interviewees in general. However, wastefulness was perceived not only in the synthesis DT element, but also in other aspects. A few interesting statements that do not relate to a specific aspect of DT are: *"DT is a good approach, but it involves a lot of waste. In LSD you are told to eliminate waste. So you try to be as efficient as possible." "Our experience is that DT is less efficient and less effective."*

However, these views were not shared by all respondents, some of which had come to the conclusion that some waste is required to achieve the goals of DT, as was reflected by the following statements: *"LSD brought in some efficiency and some speed to product development but still the products where not innovative. So the desirability was still not taken care of and probably that is the element that DT will bring into the whole process." "Things are very different; LSD is for efficiency as compared to DT stands for innovation. If you try to enforce efficiency in a process many times you might not get innovative outputs." "There will be some product waste which will be generated by DT, but this is not actually a waste; this is a waste that is required."* The majority of respondents acknowledged some inherent waste in DT, but at the same time conceded that it was necessary in order to get a higher quality output.

Furthermore, some statements expressed a view that waste production in DT was too strong a terminology and preferred to rephrase it a bit more moderately such as: *"at some point you do feel that whether a particular step is necessary or not necessary, but I don't know if we should call it waste", and "I wouldn't call it waste. But yes, it definitely takes some efforts."*

Presented below are some individual themes within the topic of waste elimination that emerged during the interviews.

Necessary Waste

As briefly touched upon above, a number of respondents had the perception that DT does generate some waste, but considered this waste necessary to create better offers for their customers or end users. For example, one interviewee stated: *"We have more ideas; we try to discuss and brainstorm, and meet again with customers. Yes, this really takes a long time, but I think in the end we build a better product." "You have to be very patient, that's what I found out, but if you are patient and you do it in a proper way the returns are also good."* Also, *"at the end DT does ensure that what you are developing is something that will solve a user problem."*

Not only was the point mentioned of increasing the quality of the output, but another aspect discussed was that some waste earlier in the process might reduce the waste at later stages. Because if it is ensured that the right thing is being developed, there will be less adjustments or re-work later on. *"Definitely, it takes more time and I think that is the radicalness of the DT approach and we need to factor in that more time is being spent on that. But I think that in turn it might end up giving you better ideas that could be beneficial in later phases of development. We might actually come up with ideas that take less time and we can complete in less time."*

Another statement returned to the previous point of team commitment and satisfaction; since everybody is involved from the start, the increased motivation actually increases the execution efficiency: *"It requires a little more of time, but then I feel it is more effective. It is not just one person's brainchild but everybody would have put their mind into trying to realize this particular requirement. So in that way, time wise, it is more time consuming but at the end of the day everybody is satisfied with that we have come up with a very good proposal."*

In the end it appears to come down to the requirement of finding the right balance, i.e. how much waste is tolerable for the greater goal. As one respondent summarized, *"I think that you need to balance it. I am open to spend some waste and spend some time for generating some ideas, but from my experience it is not always the best if you expect it from the group sitting in a room together and doing post-its."* The second part of this last statement leads seamlessly on to the next two themes, which revolve around time being wasted in discussions among teams with too many, and unsuitable participants.

Team Discussions

Ideation in DT involves converging and diverging as repeatedly mentioned previously, which is typically performed in a group setting and members holding active discussions. These discussions were perceived to not always be as fruitful and efficient as desired, as is reflected by the following statements: *"A lot of discussions are happening which is good in some way but you spend a lot of time in doing that." "The entire team needs to be there and discuss for long and all those things. So sometimes it feels like (laughs) there is a lot of waste of time."* Not all interviewees took it with as much humor though, but rather had already thought of ways to improve the perceived flaw: *"This is a big disadvantage from my perspective. This was one of my main proposals to make DT more efficient: to prepare such DT sessions, not putting the people one full day together in one room and think something is working out. It is even better to have two or three hours of DT session and all people come prepared by having some own time for their ideas and bring it on the table and then working together. That is how meetings are efficient, you should work together, you should exchange, you should discuss that is out of the question. But for me this is somehow missing from our current DT. We only write things on post it and this is always an inefficient way."*

The discussions are considered necessary but as practiced so far, were considered too unstructured to yield good results in reasonable time. Related to the time taken on discussions is the next theme of team composition and size, i.e. which people should be involved and how many of them.

Team Size and Composition

The theme presented here is concerned with how many people and what roles should be on a DT Scrum team. Having large teams around ten members was considered too many for reaching

agreement. Moreover, having people with insufficient experience or unsuitable personalities was seen as being detrimental to the outcome and efficiency.

The main points about Scrum team sizes in terms of DT were expressed as: *"It was very difficult to come to a point because of so many different people sitting in a room. It would be easier and much faster in a classical approach where you have two or three main contributors who do research and then do storytelling to the others. In a bigger team you need more investment and I didn't make the experience that this was better." "The biggest difference or challenge is to integrate the full team and working in prototypes and so on but very often I had the feeling it was not efficient or effective in relation to what we find out."*

Concerning the roles and personalities, the following emerged: *"PO, UI designer and Architect are the key people. I never ever attend any meeting where the quality people, the knowledge management or the translators or whoever made any significant contribution to the project. I think this is the main misunderstanding that you should involve all of these roles in a DT project. For me this is a major misunderstanding, these people are not that creative to generate or create a new project. I made the experience that on three teams that there was zero, no contribution and for them it was more or less boring and a waste of time and at the end they stepped out."* Also, *"to put a full team of ten or so together is too much waste because not all of them contribute in the same manner. Not everyone is feeling that creative to contribute and that leads to frustration and that is blocking the team."* These statements indicate some form of frustration with the composition of the team, and that having unsuitable participants actually inhibits the productivity and creativity of other team members.

Ideas not Technically Feasible

One of the aspects of DT is to generate ideas that push existing boundaries with the aim to find more radical solutions. Owing to this encouraged creativity, several interviewees mentioned that ideas had to be abandoned because they were technically not (yet) feasible, and hence had to be deemed waste generated in the process. "*The challenge is that you might come up with different ideas, and when you go and implement them you might face a lot of issues and then you basically have spent a lot of time coming up with your ideas but you can't execute them."* The previous statement referred to the challenge of realizing early when an idea may not be pursuable at the time and not to spend more time on it than necessary. The next statement equates straight to the aspect of waste elimination, rather than describing a challenge: *"waste elimination came into picture during the ideation where we had some wild ideas. We imagined many things out of the box and when we wanted to prototype them, we could not do everything because of our own constraints in knowledge, time, landscape, licensing, system issues, etc. There we thought maybe we shouldn't have invested too much on thinking on things like cloud, mobile, and all those things."*

Design Thinking Eliminates Waste

The last and somewhat unexpected theme that emerged was that DT actually eliminates waste in some aspects, rather than create it, as was deduced from the literature analysis. A pattern became apparent where interviewees deemed DT and LSD to be a perfect fit because the early use of DT eliminates the need for re-work in later stages and avoids unnecessary work by doing the right thing from the start.

One interviewee stated the following: *"I would say it is a perfect fit. By showing prototypes to the end user in very short iterations you don't have to write this huge specification so I would say it fits really*

well but you need to bring it in the right order." Another statement reiterated the correct order of activities intertwined with the topic of having the right personnel on DT teams*: "It is good if you can start with the DT before you bring the developer in place because this is also how you start to structure your product."*

The benefits of being able to answer any question early and exclude undesired product features based on the outcomes of DT were also mentioned: *"It is also good to know early that this feature doesn't add any value"*, leading to the following benefit: *"Almost all the questions were answered by customers or we were able to validate our proposals and we knew we were on the right track when we started developing after the first two or three months delay. And hopefully we are able to make up for the delay in the following months where everybody was clear of the requirements and everybody is just ready to go. So I think it kind of balances out in the entire development cycle."* Specific mention of the elimination of rework occurred as follows*: "This is a very important point to note, that the re-work was less and sometimes in LSD the re-work is there. This is what we have experienced that we get the requirements through a channel and sometimes when the product finally reaches the customer and the customer sees it, there is of course some kind of re-work that is required in the earlier model which we didn't experience here."*

The view that developers do not add value to the DT activities was certainly not shared by all interviewees; in fact, this aspect was appreciated by a number of interviewees as exemplified by this statement: *"Each of us became an expert on the topic but previously this was not there, there would be information lost at each level, until it comes down finally to the developer so the developer has the task of tracking back again when there is a gap that he has identified, which was a waste of time."*

4.5 Challenges to Design Thinking

Finally, data was collected on the challenges associated with the introduction of DT. Interviewees listed four main challenges to DT at *Software Co* for software development projects. These four challenges included: (1) End user involvement (2) Convergence (3) Success Measures (4) Adaptation of POs to DT.

Overall, the end user involvement generated by far the most comments in terms of challenges for the interviewees in terms of integrating DT into software development projects. The challenges related to end users revolved around access to end users and observation techniques. Convergence and synthesis of data collected from the end users and customers was also mentioned by the interviewees as a major challenge. The latter two challenges were not as prevalent but still mentioned as significant obstacles to DT in software development projects at *Software Co*.

Access to end users or customers, as well as getting useful information, was seen as a major problem for Scrum teams using DT. In a general sense, just getting the access to these end users was seen as a major issue. An interviewee stated that *"getting access to end users is the challenging part, and then getting some meaningful insights out of them is certainly challenging, so DT is definitely more challenging. But as I said, if you do it in a proper way, the returns are also good*" and *"to have access to experts or even then normal users, who are using something in a day to day environment was a bit of a challenge."* To deepen this problem of getting access to end users, another interviewee mentioned that the global nature of *Software Co* and the common issue of being located in different parts of the world, away from the end users, is a major issue when using DT: *"for DT to really work, you are always expected to meet the real end user who will use our software. And this, not only in*

Software Co but in any company, is really very difficult, because Software Co is producing software for different varieties of industries and a lot of users. For example, our software is used in North America or even in Europe. And for me it is impossible to meet one of those guys so that's the other, not drawback, but I would say challenge, because DT expects you to do this – it is not always possible." The interviewee mentioned that using phone or video to solve this problem is more of a *band aid* solution to this problem because *"when you do it over video sometimes the person is not 100% engaged in the discussion" and "we lost the empathy if we do it remote."* This issue was brought up by another interviewee stating *"we have budget restriction and travel is not possible so we end up interviewing customers over the phone for DT which is totally not acceptable because you have to see what they are doing and try to understand their process. Such budget restrictions, if it is there then I don't think DT is going to be effective."* Another interviewee mentioned the added stress this challenge brings by stating "*since we have direct customer contact and we have to talk to them, sometimes we have to wait and since we are India and most our customers are in Europe and the US, we did not have travel budgets to meet them directly sometimes, so we had remote calls at all times being in India and so these are all for me a little bit stressful."* Obviously, given the common issue of having the producer of software work in offshore countries from the end users presents a large obstacle for DT since at its core it involves observation of the end user to build empathy.

A further two issues associated to the end user and DT were, first, ensuring that the team was interacting with the right end user and, second, being able to scale these observations into a solution that is applicable to several *Software Co* customers and that does not just solve problems specific to a single customer. As one interviewee stated, "*a huge question where you can you can fail with DT is do you really talk with the right end users and this is really tricky so this is one thing where DT is a problem."* This thought was then extended to include the importance of talking to multiple end users. Given the need for *Software Co* to market their products on a large scale*, "it is very important that you get a lot of end users, and end users from different companies because you need to extract from all the inputs that you get, you need to extract all the aspects and come to a kind of product and if you don't really have the right end users, or the wrong type of end users, then you do it right for this end user but you are not successful in the market because this is a very specific end user that has a very specific process that does not fit to other customers for your target market."* While another interviewee stated the same thought by saying *"so one drawback that is there is that when you talk to different end users, each one has a different idea. And at some point of time, because we are a global company and we have to do things which are very generic, you will have to bring this one level higher to generalize this process. And when you start generalizing it, this is the time when you would need to discuss with more end users, to see if what you are generalizing is something that others will like as well, not just the 10, 15, 20, 25 end users you have discussed this with. So if we can overcome that challenge, I think then yes, of course we will build a lot better product."*

Once access to the right end users are gained and useful insights are gathered, the next major challenge seen to DT was both the ability and time required to converge this data into a useful problem statement. For example, interviewees highlight the challenge in converging the data by stating *"you get a lot of data and then to come up with an execution plan based on that data requires a lot of synthesis which is a challenge, like that is something that we encountered*" and *"I think the challenge would be that if I am talking to ten or so customers, if I have ten different views of the problem statement, then how would I decide"* and *"it was definitely difficult to get a very clear convergence of all the observations"* and *"I think it is the synthesis phase in the DT which is very*

grueling." Even beyond the challenge of converging large quantities of data generated from DT, interviewees found it a challenge to meet the time requirements due to the prolonged synthesis phase. As one interviewee stated, *"to come back do the development and finish it within the time lines that was the major challenge" and "the challenge is that you might come up with different ideas, and when you go and implement that, you might face a lot of issues and then basically you have spent a lot of time coming up with your ideas but you can't execute them."*

A third challenge for DT from the perspective of POs and SMs was seen to be how to define if the resources spent on DT result in any additional value. As an interviewee said, *"so more or less that everybody can say that we got some new insights... ...and later, when you build the software and it goes out, there is no traceability."* Since there is a big push concerning DT at *Software Co* there was a feeling that everyone claims success but there is no way to trace this success. This lack of traceability is tied to the next challenge that DT is good but it is not a fix-all solution. An interviewee tied this challenge to the fact that there is a heavy push behind DT at *Software Co*: *"since DT is so much in the limelight, everybody wants to be a part of it. So even without considering whether my project or process is fit for a DT project, people embark on the journey and then later run into issues."* The same issue is stated again by an interviewee as *"DT is good, it can fix a lot of problems, but it is not a solution for everything, which somehow seems to be the interpretation, or misinterpretation, at least for the time being."* Another interviewee proposed that criteria should be in place for using DT to prevent projects from moving forward that are not really suitable for DT by stating: *"you should have some criteria in place for DT, minimum of this many interviews, this many customers and so on. Once we do that and we have good enough interviews and good enough insights, then it gets easier. That's what I mean to say. But I mean for somebody that couldn't get that many appointments with end users and all those things, then my personal feeling is that you shouldn't go forward."*

Finally, a challenge specifically concerning the adaptation of POs to DT was mentioned by interviewees. The PO role, as stated above, encounters the most change when DT is used on a software development project. As one interviewee stated, *"I know that the POs feel that the DT people are invading their territory. They have been functional experts, who knew everything about the product and whose opinion counted the most, but now everybody seems to be an expert and everyone is claiming that I meet with this customer and that customer and this is how we should do it and so on. So I don't think it is so simple for the PO. I think if adapted promptly, it can make their life much easier. Otherwise it can mess it up, they might be completely demotivated after it and they might question is "why do they still exist in the team because everyone is sort of a PO."* It is clear that this challenge is very much dependent on each PO and how they adapt to DT. But it was, nevertheless, mentioned through the interviews. Another interviewee stated as much by saying *"a lot of different people who always question if you are doing this, what is the role of the PO? And sometimes people asking such a question are POs themselves and sometimes other roles, so in general it is not just a perspective, at least to start with. Everybody gets this feeling. Later, this might get stronger or it might go away, depending on how the project runs and is handled, but to start with everybody has that question. Even I had the same question."* The data clearly shows that the challenge in adapting to DT is most prevalent for POs, but this is a challenge for some and not all.

Of the four perceived challenges related to the introduction of DT to development projects at *Software Co*, two are directly related to the theoretical misalignments between DT, LSD, and Scrum. These challenges are associated with the DT practice elements end user empathy and synthesis. The

remaining two challenges do not appear to be linked to the identified misalignments, but do carry practical implications for *Software Co*.

5 Analysis

The following analysis compares the empirical data with the theoretical framework to answer the research question, *How do the DT practice elements of end user empathy, visualization, and synthesis integrate with LSD and the Agile method of Scrum from the perspective of Product Owners and Scrum Masters?*

5.1 Lean Software Development, Scrum, and Design Thinking

In the following section, the empirical results concerning the integration of LSD, Scrum, and DT presented in Section 4.4 are analyzed using the theoretical framework presented in Figure 3.

The main topics that will be further analyzed below include the three DT practice elements given in the literature section that include (1) end user empathy, (2) visualization, and (3) synthesis. These DT practice elements will be compared to LSD and the Agile of method of Scrum to better understand their integration from the perspective of POs and SMs.

5.1.1 End User Empathy

Comparing the theoretical framework to the empirical data concerning end user empathy, several themes emerged. These will be analyzed in detail in the following sections: (1) Terminology of End User and Customer (2) Purpose of Contact - Empathy versus Satisfaction (3) Access.

Terminology

One pattern that emerged from the empirical data concerning LSD was the importance of customer satisfaction. However, the empirical data showed that when speaking about LSD there is not a firm definition for the term customer. LSD literature does emphasize the word customer based on the first principle of LSD: *"founded on a deep understanding of a job that customers would like done"* (Poppendieck & Cusumano, 2012, p. 28). However, the question remains as to whom to develop a deep understanding for as a result of the vagueness behind the terminology used in LSD literature. This vagueness or lack of specific definition concerning the term customer in literature (Hildenbrand & Meyer, 2012) was reflected in the empirical data in that the interviewees conceded that in terms of LSD the customer could be any contact person at the purchasing organization, whether it be for example an end user or IT manager.

On the contrary, the empirical data showed a solid trend concerning the terminology of end user in DT, with all respondents stating the focus of DT was exclusively the end user, i.e. the person using the actual product. Hence, for DT the prominence and specificity given to the end user in DT theory (Hassi & Laakso, 2011) was confirmed by the empirical data in the case studied. Overall, it was seen that the terminology concerning end user and customer was ambiguous concerning LSD projects but specific in DT.

Empathy vs. Satisfaction

As mentioned above, the terminology surrounding end user is explicit both in the empirical and theoretical data concerning DT. On the contrary, the terminology surrounding customers in LSD is vague both in the empirical and theoretical data.

Since the term customer given in both in LSD literature and the results is unspecific, it could include end users. The Agile method of Scrum, on the other hand, does suggest the involvement of end users specifically in software development for testing of software releases which in turn results in customer feedback (Maylor, 2010; Rising & Janoff, 2000; Schwaber & Sutherland, 2011). However, this implies that development activities have taken place and produced working software before end users are consulted. This shows that the focus in LSD and Scrum is more on satisfaction than on initially emphasizing to understand the latent need as shown in DT.

The results show that a key difference in DT projects at *Software Co* is the timing and purpose of involving the end user or customer. This became evident by the account given where in one DT project the end users were surprised to be asked questions and visited by *Software Co* developers prior to being shown any software which they could test. The developers were not interested in just validating working software but gaining empathy. As a result it is clear that there is a difference both in the literature and practice concerning the DT practice elements of empathy and the emphasis on validation of working software or customer satisfaction given in both LSD and Scrum.

However, this distinction between DT and LSD/Scrum was not entirely clear in the results. At least half of the POs stated they actively sought contact with end users to understand their latent needs before DT was formally practiced in the development organization of *Software Co*. In a way they conclude, quite correctly, that they already practiced this element of DT, even before they were trained in it. In a sense, these POs had modified and expanded upon the practices of LSD and Scrum on their own behalf. Rather than gaining customer satisfaction or validation by gathering customer or end user input after working software had been developed as recommended by e.g. the Scrum Guide (Schwaber & Sutherland, 2011), they had altered both the timing and purpose of end user involvement to the pre-development phase to better understand the end users latent needs – in essence, they were empathizing as summarized by Hassi and Laakso (2011). For these particular individuals, the early end user contact was not new and hence from their perspective little adjustment was necessary with the addition of DT. However, the formal introduction of DT and being trained in it provided confirmation on their practice of involving end users early on to discover their latent needs rather than just at the end of sprints to validate working software. Furthermore, they were appreciative of the structure and legitimacy DT added to their approach of managing software development projects.

Across the entire sample a unified view was communicated that being in contact with end users to gain empathy provided benefits compared to the previous approach. In the previous approach, customers would provide a list of requirements, i.e. request features to be included in software releases in line with the Agile software development principles (Beck et al., 2001), which underlie the Scrum methodology. By seeking early and continuous end user feedback, the DT approach was considered to provide and build confidence in that what the Scrum teams did was indeed the right thing for the end users. Highlighted in the data and literature was the importance that empathizing has to be done in the right way to have the optimum effect: rather than asking for customer wishes, their actual work and goals have to be understood, which has a central position in DT theory and practice, as was identified in the comprehensive literature review by (Hassi & Laakso, 2011).

DT's practice element of empathizing with the end users, as opposed to satisfying customers, was perceived to result in better and more desirable end products, thereby ultimately increasing the

value for the customer – both the organization and individual users. Therefore, this aspect of user empathy in DT seems complementary to the LSD approach, rather than being in disaccord. Since LSD and Scrum are built around customer satisfaction and only practicing activities that add customer value (Poppendieck & Cusumano, 2012), empathizing with end users was seen to be a beneficial extension but not replacement for customer satisfaction.

Access

Although, as analyzed above, the empirical data shows that DT's practice element of end user empathy integrates fairly well with the customer satisfaction in LSD, the issue concerning access was seen as a hindrance to the alignment of end user empathy and customer satisfaction.

As mentioned above, one of the biggest perceived challenges in the introduction of DT to software development projects at *Software Co* was gaining access to end users. This issue is practical in nature but could be considered an outcome of the difference between end user and customer terminology previously mentioned. Prior to the introduction of DT, gaining customer access was not seen as that great of difficulty because it was not specific on whom the contact should be. Rather, it could include various roles outside of the end user which were part of well-established contact networks.

The first issue in gaining access to end users is that they are busy with their usual work within their own company and no allowance is made for talking to software developers from an outside organization. End users typically have internal contact persons, typically someone from the internal IT department, whose duties include communications with software vendors. Also, an existing network may provide for contact with e.g. the IT department at the customer organization, but end users are not typically part of the existing networks and hence getting access to these requires an extra step by contacting the usual contact person first, who may then, if convinced of the benefit, establish the connection to the end users. The statement given by one interviewee, *"you never go directly to the end user"* summarizes the difficulties encountered succinctly.

The challenge of gaining access to end users is seen as an integration issue and supports the theoretical framework in showing that there is a misalignment between customer satisfaction and end user empathy because of the inherent difference in end users and customers. Because the end user is clearly specified in DT, there is no alternative person for gathering data from. Management staff, who supervise end users, were sufficient for serving as customers in LSD, but are not appropriate in DT. Because the wrong person would be empathized with, the result could potentially be a less desirable end product. This problem may however reduce over time; with the DT approach maturing, POs and SMs should be able to build up networks among end users, and end users will become accustomed to being involved in pre-development activities for enterprise software development. As a result, the perceived and actual difficulties in getting access to the end users could reduce in severity.

In summary, the introduction of DT and the specification that POs and SMs should gain end user empathy (Hassi & Laakso, 2011), has made their work somewhat more complex compared to before. The looser term customer in the previous LSD and Scrum approach (Poppendieck & Cusumano, 2012; Schwaber & Sutherland, 2011) allowed more flexibility in whom to talk to. Logically, if a larger pool of subjects is available, the probability for success is higher. Moreover, easier access is granted from persons at the customer organization whose job it is to interact with software vendors.

5.1.2 Visualizing

The centrality of writing requirements in product backlogs given by the Scrum process framework (Schwaber & Sutherland, 2011) was identified as a misalignment to the DT practice element of visualizing, which promotes using anything except for written words and symbols (Hassi & Laakso, 2011). At first, this theoretical misalignment does appear to exist in practice at *Software Co*. However, this misalignment was perceived as a positive rather than negative influence in the way the POs and SMs carry out their work. In fact, visualizing was in general considered more important because it facilitates communication and problem definition, which are the main benefits of visualizing that Hassi and Laakso (2011) identified.

By specifically integrating DT into existing LSD and Scrum practices, visualizing was perceived as a complement rather than a replacement to writing product backlogs in Scrum. The interviewees reflected that visualizing helps them to know what exactly needs to go into the written backlog, and thereby it eliminates wrong things being written into the backlogs. Furthermore, a problem is defined in a more comprehendible manner from visualizing that was said to be easier translated into written items in the backlogs. A unanimous opinion existed about the order of the two practices; in order to be beneficial visualizing has to precede not replacement backlog writing, which subsequently is followed by development execution.

Visualization, as opposed to the written backlog, involved the entire Scrum team. In LSD projects, the written backlog was created and owned by a singular person as prescribed by the Scrum Guide (Schwaber & Sutherland, 2011), whereas in DT visualizing is conducted by the entire Scrum team. POs in LSD projects had, in most cases, the sole ownership of the backlog and would dispense the knowledge to the team as needed. However, interviewees noted that when using visualization practices, there was a shift due to the entire Scrum team's involvement. The entire Scrum team was said to be on the *same page* as a result of visualizing and had the full scope in terms of the problem to be solved.

The theoretically identified potential misalignment between the DT practice element of visualizing and the Scrum practice of writing product backlogs was clearly rejected by the interviewees. Rather than being in conflict, these two complement each other well if performed in the order of visualizing followed by backlog writing.

5.1.3 Synthesis

In this section, the findings relating to the DT practice element of synthesis and the LSD principle of waste elimination are analyzed within individuals themes that emerged during the interviews. The individual themes include: Necessary waste, Scrum team discussions, Scrum team size and composition, ideas not technically feasible, and DT eliminates waste.

Overall, there was no unified view among the POs and SMs, i.e. responses included both confirmation that DT is wasteful and rejection of the proposition. Where interviewees considered DT being wasteful, it was not only due to the practice of synthesis, but a variety of aspects which are analyzed in detail below. Even though a number of respondents did not think DT creates waste overall, they conceded that some waste was produced but was inherent in the approach and necessary when viewing the bigger picture of gaining the right solution.

Necessary Waste

In general, POs and SM agreed that DT generates waste, but overall this was considered tolerable and in fact required to achieve the greater goal of building products that are innovative and of higher quality, essentially providing more value to the customer. When looking at this aspect from a pure LSD perspective, it becomes a matter of a trade-off between waste and value and finding the right balance between those. Waste as a LSD principle is defined as *"anything that doesn't either add customer value or add knowledge about how to deliver that value more effectively"* (Poppendieck & Cusumano, 2012, p. 28). The results of this case study show that the interviewees were happy to generate waste in DT as long as the customer value increased. The timing of waste was also commented on; since DT at *Software Co* is typically done in the early phase of a development project, and therefore any waste associated with DT is created early as well. That being the case, it was perceived as easier to rectify issues earlier rather than later, and thus waste was considered to be better early in DT than later in the execution. Yet, this goes contrary to literature that states in DT diverging should be utilized not only at the beginning, but throughout (Hassi & Laakso, 2011). A softer aspect that was perceived to a limited extent was that DT activities require more time but pay off positively later because the team is more motivated which in turn increases execution efficiency. However, it is important to note the empirical data shows that tracking the connection between waste generated by DT and customer value is a difficult process and frequently overstated due to the importance given to DT at *Software Co*.

Scrum Team Discussions

The primary source of waste in DT was considered to be wasted time due to extended discussions among the development team during the synthesis phase. This point was strongly felt by the respondents and is strongly tied to the DT practice element of synthesizing. Because the entire Scrum team is usually involved to select and synthesize findings (Hassi & Laakso, 2011), it takes significant time which was considered wasteful by the respondents. Because this theme was perceived as problematic, a number of interviewees had come up with suggestions to improve this aspect, such as providing more structure rather than very open discussions.

Interestingly, the theoretically identified main source of potential waste in DT's synthesis practice was ideas that were generated but later on discarded. The empirical results suggest, however, that the excessive amount of time required to achieve convergence among the team was seen as the primary source of waste.

Ideas not Technically Feasible

As mentioned, the ideas that are generated during the synthesis phase and later discarded are, according to the theoretical framework, the main origin of waste in DT. Although this was not perceived as the primary source of waste in DT, it was mentioned during the interviews. Some respondents stated that the ideas that were generated were very broad in scope, as is encouraged in DT (Hassi & Laakso, 2011). However, after spending some time with developing the ideas it became evident that, for example, consideration toward the readiness of the technology was not given and hence the ideas had to be abandoned. Since DT allows for, and indeed encourages wild ideation, there is a risk that a track is pursued that does not lead anywhere due to them Scrum team not being able to implement the idea. This is a violation of the underlying LSD principle of waste elimination as defined by Poppendieck and Cusumano (2012).

The difficulty for POs and SMs working with DT is then to realize as early as possible that an idea is not feasible, be it for technical or other reasons. Perhaps some technical validation could take place concurrent with prototype validation with users to avoid spending too much effort on ideas that have low success chances.

Scrum Team Size and Composition

This theme emerged only partially in the data but, nonetheless, a pattern was observable. None of the literature reviewed gives recommendations about the size of teams for DT, but the point is discussed regardless since there was an observable pattern in the data. The Scrum Guide (Schwaber & Sutherland, 2011) suggests team sizes between three and nine members; at *Software Co*, the team sizes seem to be towards the upper end of this suggestion and beyond. The large team size was seen to contribute to the large amount of time taken to come to an agreement, contributing to the theme discussed above. From this perspective, team sizes at the lower end of the range for recommended Scrum team sizes, were perceived to perhaps align better with DT's practice element of synthesis.

Another aspect mentioned was the appropriate roles on a DT team, both in terms of experience and personalities. Issues were described that related to these aspects in that some people did not contribute meaningfully to the DT activities and in the worst cases were even perceived to block the creativity of other team members. However, this theme is outside of the scope of the study since it is more related to the DT categories of cognitive approach and mindset as per Hassi and Laakso (2011).

Design Thinking Eliminates Waste

Perhaps the most striking pattern in the data was DT association with eliminating waste, rather than creating it. While the waste discussed above is created on a smaller scale in individual steps of DT, a view of an entire software development may actually reveal that DT, if done properly, might reduce waste in a larger sense. In fact, DT was stated to be a perfect fit with LSD in this context. By creating a little waste early in the process, pursuing the wrong track for longer is avoided and hence any correction is less costly if detect earlier. This observation also reiterates the importance of taking a sequential approach, i.e. DT followed by LSD and Scrum approaches.

In addition to the opportunity to detect wrong paths early on, DT was stated to eliminate unnecessary product features being developed. Previously it occurred occasionally that certain software features were included in a product but turned out to be unnecessary or not desired by customers, thus essentially being waste. Moreover, the early use of DT was mentioned to minimize the need for re-work of software releases by seeking early user feedback before actual development. Unnecessary features were found to be one of the main causes of waste in software development (Poppendieck & Cusumano, 2012), and it appears that DT has the potential to at least reduce this type of waste in the software development industry, if done at the right point in time.

6 Discussion of Results

The foregoing analysis showed that the three theoretical misalignments between DT practice elements, LSD principles, and Scrum practices were perceived by PO and SM to be only partially misaligned. These partial misalignments did create challenges for PO and SM. These findings are discussed in the following section.

6.1 End Users

The specific terminology of end users in DT leaves little room for deviation and as such was welcomed because it provides clear guidance on who to talk to at the customer organization. But rather than be a replacement for the loosely defined term of customers in LSD and Scrum, end users are seen as additional people that contact must be sought with because the decisions at the customer organization are still to be made by the previous contacts (e.g. management or IT head). Despite this misalignment in terms of terminology between DT, LSD, and Scrum, no real practical conflict was perceived to emanate from it. From this point of view, the theoretical misalignment does not impede the practical integration of the three approaches.

The focus on empathizing with the end users in DT was seen as helping the development teams to create more desirable and innovative products, which ultimately would lead to customer satisfaction as focused on in LSD and Scrum. By improving the end users' work experience through well designed and functioning software, those users should by simple logic be more productive. Higher productivity in turn satisfies supervisors and managers in the customer organization; following this chain of reasoning, DT increases customer satisfaction which originates from empathizing with end users.

Specific practical challenges were evident which originate from DT's emphasis on customer empathy. These include development teams being required to empathize with end users remotely due to spatial separation and budget restrictions, considerations of which and how many end users to empathize with in order to be able to develop scalable offers and getting access to end users in the first place. The reviewed literature does not offer much guidance on how to overcome these challenges and thus they will need to be dealt with pragmatically over time. Time however, given the competitive nature of the software industry, is scarce and a resolution of these issues is to be sought sooner rather than later. On the other hand, these challenges (and benefits) of end user empathy would also apply to other settings and industries, i.e. when it is integrated with product or service development approaches other than LSD and Scrum. Perhaps a look towards organizations who have adopted DT in other industries could provide some valuable advice.

6.2 Visualization

The DT practice element of visualizing was perceived as well aligned with LSD and Scrum, contingent on DT being done before any backlogs are being written. Visualizing helps POs to get a clear picture of what the actual problem is and how to solve it, and it is only logical that writing out the technical solution is much easier when that clear vision is there. Owing to the entire team's involvement in the visualizing practice, communication is improved within the development teams. No challenges were identified that are the direct outcome of the DT practice element of visualizing and therefore it appears a perfect complement to writing backlogs, without any hurdles to their integration.

6.3 Synthesis

The DT practice element of synthesizing was perceived to have the potential to contribute to overall waste elimination in a software development project. The extent of improvement is (at this stage) unknown and its assessment is closely linked to establishing some success measures and KPIs which should be in place, but are currently lacking at *Software Co*.

Nevertheless, interviewees perceived that DT does create some waste in certain parts and the entire picture may not be very obvious in the daily business and hence any wastefulness may be felt quite

strongly. Also, the bigger picture and overall waste reduction can only be judged when the project has been completed, which was not the case for all DT projects the interviewees had been involved in. Perhaps, certain aspects of DT can be fine-tuned in the future in order to further reduce the perceived waste creation associated to DT, such as smaller DT team sizes.

A practical challenge that is the direct outcome of the DT practice element of synthesizing is the convergence of gathered data. POs stated that given the nature of DT the amount of data collected increased dramatically and it was difficult to decide the correct path to follow. In other words, how should the PO decide which end user observations were valid and scalable. Given the increase in team size, PO and SM felt perceived this to be a challenge since it is difficult to involve all team members equally when converging data.

As mentioned in the Data Collection, PO and SM perceived four challenges: (1) End user involvement (2) Convergence (3) Success Measures (4) Adaptation of POs to DT. The first two perceived challenges, as described above, are showed to be direct results of the theoretical misalignment between DT practice elements, LSD principles, and Scrum practices. The perceived challenge of end user involvement is directly related to the theoretical misalignment between end user empathy vs. customer satisfaction. The perceived challenge of convergence is directly related to misalignment of synthesis vs. waste elimination. However, the perceive challenges of success measures adaptation of POs to DT are indirectly related to the theoretical misalignments in that they are influenced by but not a direct result of the theoretical misalignments.

7 Conclusion

In order to answer the research question, *How do the DT practice elements of end user empathy, visualization, and synthesis integrate with LSD and the Agile method of Scrum from the perspective of Product Owners and Scrum Masters?,* the theoretical framework presented in Section 2.6 was used to analyze the collected empirical data.

The DT practice elements of end user empathy, visualizing, and synthesis stated in the literature review by Hassi and Laakso (2011) were identified to be in theoretical misalignment with the LSD principles and Scrum practices of customer focus, writing backlogs and waste elimination, respectively. This thesis investigated these theoretical misalignments in practice by studying the integration of DT with LSD and Scrum at *Software Co*. The conclusions concerning the integration of these DT practice elements on LSD and Scrum projects from the perspective of POs and SMs can be stated as:

1. The notions of end user empathy and customer satisfaction were perceived to be neither clearly aligned nor contradicting. The DT practice element of empathizing was seen as contributing to the development of better products, which in turn increases customer satisfaction; therefore, end user empathy positively effects customer satisfaction. The specific DT terminology regarding end users negates the ambiguity of the term customer in LSD and Scrum, and thus increases clarity. On the other hand, end user contact is an addition to the usual customer contact and creates practical challenges in terms of access, logistics, budgets, and quality and quantity of end user data.

2. The theoretical discrepancy between visualizing in DT and the Scrum practice of writing backlogs was clearly rejected. These practices were perceived to align well, provided that visualizing from DT precedes Scrum practice of writing backlogs in LSD projects. Visualizing was seen as more important because it improves communication and facilitates both problem definition and backlog writing. Rather than being in conflict, visualizing and backlog writing complement each other.

3. Neither misalignment nor alignment could be clearly concluded concerning DT manager's focus on synthesis and LSD manager's focus on waste elimination. Noteworthy is that DT aspects other than synthesis were perceived as wasteful. Sources of waste in DT were considered to be long discussions as an effect of large team size, sub-optimal composition of teams, and generation of technically unfeasible ideas. When viewed overall, DT was accredited with the potential to reduce overall waste on software development projects by reducing the amount or necessary re-work and eliminating unnecessary software features.

The above conclusions relating to the theoretical misalignments between DT practice elements and LSD, and Scrum principles and practices provide initial insights into how POs and SMs perceive the integration of DT with LSD and Scrum. These initial insights contribute to the development of new theory regarding the successful integration of DT, LSD, and Scrum by clearly showing that the theoretical misalignments are not perceived very strongly in practice by POs and SMs. By critically studying the integration from the perspective of individuals that are central to its success, this study complements previous research on the topic conducted by Hildenbrand and Meyer (2012) who explored the integration from a process perspective. Aside from the specific findings listed above, a number of areas for future research have emerged from conducting this exploratory case study. These are briefly described in the following section.

8 Areas for Further Research

DT is a relatively new approach to areas outside of design, and has yet to be extensively studied within the management realm, particular within software development settings. The relative newness of DT combined with the investigative nature of this exploratory case study leaves significant room for discussion and expansion and some topics are suggested below.

Areas for Further Research	
Product Owner	• Shift in a PO's role and responsibilities with the introduction of DT • Shift is viewed favorably or critically depending on the individual PO • Experience of the PO, individual personality, and/or cultural background might affect how the PO views this shift
Success Measures	• Correlation of DT activities to market success • PO and SM role in communicating the correlation of DT to market success to motivate the Scrum team
"Right way"	• DT is good if done the "Right way" • What is meant by the "Right way" for DT
Synergies	• Study the practice elements, Action Based and Collaboration, that showed a theoretical alignment
DT Categories	• Study other the two other categories in Hassi and Laakso (2011) DT literature review: Mindset and Cognitive approach.

9 Sources

Babbie, E. (2001). *The practice of social research* (9 ed.): Wadsworth/Thomson Learning.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). Principles behind the Agile Manifesto. Retrieved 21 February, 2012, from http://www.agilemanifesto.org/principles.html

Britten, N., Jones, R., Murphy, E., & Stacy, R. (1995). Qualitative research methods in general practice and primary care. *Fam Pract, 12*(1), 104-114.

Brown, T. (2008). Design thinking. *Harvard Business Review, 86*(6), 84.

Brown, T., & Katz, B. (2011). Change by Design. *Journal of Product Innovation Management, 28*(3), 381-383. doi: 10.1111/j.1540-5885.2011.00806.x

Bryman, A., & Bell, E. (2011). *Business Research Methods* (3 ed.). Oxford: Oxford University Press.

Creswell, J. W., Hanson, W. E., Plano, V. L. C., & Morales, A. (2007). Qualitative Research Designs Selection and Implementation. *The Counseling Psychologist, 35*(2), 236-264.

del Nuevo, E., Piattini, M., & Pino, F. J. (2011, 15-18 Aug. 2011). *Scrum-based Methodology for Distributed Software Development.* Paper presented at the 6th IEEE International Conference on Global Software Engineering (ICGSE).

Denzin, N. K., & Lincoln, Y. S. (2005). *The Sage handbook of qualitative research*: Sage Publications, Incorporated.

DeVor, R., Graves, R., & Mills, J. J. (1997). Agile manufacturing research: Accomplishments and opportunities. *IIE Transactions, 29*(10), 813-823.

Dul, J., & Hak, T. (2008). *Case Study Methodology in Business Research* (1 ed.). Oxford: Elsevier.

Dunne, D., & Martin, R. (2006). Design Thinking and How It Will Change Management Education: An Interview and Discussion. *Academy of Management Learning & Education, 5*(4), 512-523.

Haque, B., & James-Moore, M. (2004). Applying lean thinking to new product introduction. *Journal of Engineering Design, 15*(1), 1-31. doi: 10.1080/0954482031000150125

Hassi, L., & Laakso, M. (2011). Making sense of design thinking. In T.-M. Karjalainen, Koria, M. & Salimäki, M. (Ed.), *IDBM papers vol 1* (pp. 50-62): Helsinki: International Desgin Business Management Program, Aalto University.

Highsmith, J. (2001). History - The Agile Manifesto. Retrieved 21 February, 2012, from http://www.agilemanifesto.org/history.html

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). Berlin Heidelberg: Springer.

Hines, P., Holwe, M., & Rich, N. (2004). Learning to evolve: A review of contemporary lean thinking. *International Journal of Operations & Production Management, 24*(9/10), 994-1011.

Holloway, M. (2009). How tangible is your strategy? How design thinking can turn your strategy into reality. *Journal of Business Strategy, 30*(2/3), 50-56.

Johansson-Sköldberg, U., Woodilla, J., & Çetinkaya, M. (2013). Design Thinking: Past, Present and Possible Futures. *Creativity and Innovation Management, 22*(2), 121-146. doi: 10.1111/caim.12023

Johansson, U., Woodilla, J., & Çetinkaya, M. (2011). *The Emperor's new cloth or the magic wand? The past, present and future of Design Thinking.* . Paper presented at the 1st Cambridge Academic Design Management Cconference.

Kimbell, L. (2011). Rethinking Design Thinking: Part I. *Design and Culture, 3*(3), 285-306. doi: 10.2752/175470811X13071166525216

Maylor, H. (2010). *Project management* (4 ed.). Harlow: Financial Times Prentice Hall.

Owen, C. (2007). Design thinking: Notes on its nature and use. *Design Research Quarterly, 2*(1), 16-27.

Poppendieck, M. (2002). Principles of Lean Thinking.

Poppendieck, M., & Cusumano, M. A. (2012). Lean Software Development: A Tutorial. *Software, IEEE, 29*(5), 26-32. doi: 10.1109/MS.2012.107

Poppendieck.LCC. (2010). Lean Software Development. Retrieved 21 February, 2012, from http://www.poppendieck.com/

Pries, K. H., & Quigley, J. M. (2011). Using Agile's Scrum in embedded software development. *Embedded Systems Design, 24*(9).

Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. *Software, IEEE, 17*(4), 26-32. doi: 10.1109/52.854065

Saunders, M., Lewis, P., & Thornhill, A. (2007). *Research Methods for Business Students* (4 ed.): Prentice Hall.

Schwaber, K., & Sutherland, J. (2011). The Scrum Guide.

Seidel, V., & Fixson, S. (2012). Adopting'Design Thinking'in Novice Multidisciplinary Teams: The Application and Limits of Design Methods and Reflexive Practices. *Journal of Product Innovation Management, 30*(6).

Simon, H. A. (1969). The sciences of the artificial. *Cambridge, MA*.

SocioCultural Research Consultants. (2013). Dedoose. Retrieved 12 May, 2012, www.dedoose.com

Stone, K. B. (2012). Four decades of lean: a systematic literature review. *International Journal of Lean Six Sigma, 3*(2), 112-132. doi: 10.1108/20401461211243702

Wilson, J. (2010). *Essentials of Business Research: A Guide to Doing Your Research Project*: SAGE Publications.

Wölbling, A., Krämer, K., Buss, C., Dribbisch, K., LoBue, P., & Taherivand, A. (2012). Design Thinking: An Innovative Concept for Developing User-Centered Software. In A. Maedche, A. Botzenhardt & L. Neer (Eds.), *Software for People* (pp. 121-136). Berlin Heidelberg: Springer.

Womack, J. P., & Jones, D. T. (1996). *Lean Thinking: Banish Waste and Create Wealth in your Corporation*. New York: The Free Press.

Yin, R. K. (2003). *Case Study Research: Design and Methods* (3 ed. Vol. 5). Thousand Oaks, CA: SAGE Publications.

10 Appendix: Interview guide

This interview guide is representative for the interviews with POs and SMs. It should be noted though that the exact phrasing of questions as well as their order differed between interviews. Also, additional follow-up questions were typically asked during the course of an interview.

Introduction

Is it okay to record the interview?

Introduction of the researchers.

Ensured anonymity and confidentiality.

Experience & Background

What is your title and role at *Software Co*?

How long have you worked in this role?

How many development projects have you worked on?

- How many involved in DT?

Design Thinking

What is your definition/understanding of DT?

What is your opinion about DT?

How did you react to the idea of potentially doing DT projects?

LSD and Scrum combined with DT

With the introduction of DT in development projects, have projects changed compared to having only a LSD approach? How?

- Is it correct that one main difference is having requirements vs getting requirements?

What is the most important aspect when carrying out the SM/PO role in a LSD project?

- Is it different in DT? If so, how?

Are there any particular challenges when carrying out the SM/PO role in a LSD project?

- Is it different in DT? If so, how?

Of the following LSD and Scrum elements, which is the most important for you?

- Customer focus
- Listing requirements in product backlogs
- Waste elimination
- Having defined success measures

Do they conflict with the following DT elements?

- End user empathy (vs customer focus)
- Visualizing (vs listing requirements in backlogs)
- Synthesis: converging/diverging (vs waste elimination)
- No clear success measure (vs defined success measures)

Has your role and responsibilities changed when using DT on development projects?

If so, how?

Did you have to adjust your way of working when using DT on development projects?

If so, how?

Conclusion

Given your role as PO/SM, is there anything else you would like to add about working with DT, LSD and Scrum approaches?

Do you have any criticism on DT or what could be done better?