# The Use of Design Thinking for a Human-centered Requirements Engineering Approach

DISSERTATION

of the University of St.Gallen,

School of Management,

Economics, Law, Social Sciences

and International Affairs

to obtain the title of

Doctor of Philosophy in Management

submitted by

**Jennifer Hehn**

from

Germany

Approved on the application of

**Prof. Dr. Walter Brenner**

and

**Prof. Dr. Lutz Kolbe**

Dissertation no. 4992

Baier Digitaldruck GmbH, Heidelberg 2020

The University of St.Gallen, School of Management, Economics, Law, Social Sciences and International Affairs hereby consents to the printing of the present dissertation, without hereby expressing any opinion on the views herein expressed.

St.Gallen, May 18, 2020

The President:

Prof. Dr. Bernhard Ehrenzeller

## Acknowledgements

I would like to thank all the people who supported me in the last few years in making this thesis possible. I owe them my sincere acknowledgement.

First of all, I want to thank my supervisor Prof. Dr. Walter Brenner for giving me the opportunity to write this dissertation and for providing continuous guidance. His assistance, his expertise, and, most importantly, his support during the past years have been invaluable for the dissertation. I wholeheartedly thank my co-supervisor Prof. Dr. Lutz Kolbe for his advice and valuable feedback on previous versions of this thesis.

Furthermore, I thank all my colleagues of the DT@HSG Teaching Team and fellow PhD students, who I met over the past years. Thank you for all the fruitful discussions, for your insights, and for the friendship. In particular, I want to thank Falk Uebernickel. Without his advice I would have not even considered to start this dissertation and who has been a mentor ever since. I am very grateful for his confidence in me and for his continuous guidance. I wish to acknowledge the administrative and personal support of Barbara Brenner who always had an open ear for me.

A very special mention goes to my friend Ina for continuously proofreading ever since I started writing my first papers back in my first days at university. Thank you for sharing your own experiences, for motivating me, and for just being a great friend.

I am deeply grateful to my family for their continuous encouragement, even though I think they never really understood what my dissertation topic was about. I could have not been more fortunate to have “Oma und Opa” by my side. In so many ways I owe my work to them. Finally, I want to thank my parents. They were, and are, always willing to support and encourage me in any way possible.

Over and above all, Tobias: I am deeply grateful to you for unconditionally standing by my side. It was his love and his patience that helped me get through rough patches, always supported me, and provided me with motivation and joy. I feel extremely lucky to have you by my side.

St.Gallen, September 2020

Jennifer Hehn

# Table of Contents

## List of Figures

## List of Tables

## List of Pictures

## List of Abbreviations

| | |
|---|---|
| AMDiRE | Artifact Model for Domain-independent Requirements Engineering |
| ATM | Automatic Teller Machine |
| BPMN | Business Process Model and Notation |
| DT | Design Thinking |
| e.g. | exempli gratia |
| et al. | et alii |
| GI | Gesellschaft für Informatik |
| i.e. | id est |
| IEEE | Institute of Electrical and Electronics Engineers |
| IREB | International Requirements Engineering Board |
| IS | Information Systems |
| IT | Information Technology |
| KAOS | Keep All Objectives Satisfied |
| RE | Requirements Engineering |
| RQ | Research Question |
| SoE | Sources of Evidence |
| UML | Unified Modeling Language |

## Abstract

Effective Requirements Engineering is one of the most crucial activities in software-intensive development projects. The human-centered approach of Design Thinking is considered a powerful way to complement Requirements Engineering activities when designing innovative systems. While in recent years research has already made great strides to indicate potential benefits of bringing Design Thinking and Requirements Engineering together, it has remained mostly unclear how to actually realize a combination of the two. Hence, the objective of this dissertation is to identify usage schemes of Design Thinking in order to achieve a (more) human-centered Requirements Engineering approach. To accomplish this goal, the dissertation is organized into three research questions. Based on multiple-case studies the first research question aims to understand the value of using Design Thinking for two main activities of Requirements Engineering, i.e. requirements elicitation and specification. The second research question combines and compares Design Thinking and Requirements Engineering artifacts for designing innovative software-intensive systems. Based on an exploratory case study the third research question derives three concrete operationalization strategies on how to use Design Thinking for Requirements Engineering depending on the specific project context. Overall, this dissertation provides a number of important theoretical contributions. First, it offers empirically sound evidence beyond current research findings on how to address common challenges in Requirements Engineering with Design Thinking. Second, it contributes to ongoing research on artifact-oriented Requirements Engineering with an evaluated reference model for the design of innovative and human-centered software-intensive systems. Third, it offers prescriptive knowledge on how to use Design Thinking for Requirements Engineering. In this context the differentiated view on Design Thinking as process, toolbox, and mindset is substantiated and expanded. For practitioners the dissertation offers recommendations on how to improve the effectiveness of Requirements Engineering with the help of Design Thinking depending on the specific project context and objective. Overall, the results of this dissertation provide a solid foundation for many researchers and practitioners because they give a better understanding of *what* Design Thinking represents in the context of Requirements Engineering and *how* both approaches can be combined in innovative software-intensive development projects.

## Zusammenfassung

Effektives Requirements Engineering ist eine der wichtigsten Aktivitäten bei softwareintensiven Entwicklungsprojekten. Der menschen-zentrierte Ansatz des Design Thinking kann dabei die Aktivitäten des Requirements Engineering wirkungsgvoll ergänzen, vor allem bei der Entwicklung von innovativen Systemen. Während Forschung und Praxis in vergangenen Jahren bereits Hinweise für solche gemeinsame Synergien aufgezeigt haben, bleibt der Einsatz von Design Thinking vielschichtig und (noch) umstritten. Weitestgehend ist noch unklar, wie genau eine Kombination beider Ansätze aussehen kann. Ziel dieser Dissertation ist es daher, ein besseres Verständnis zu schaffen, wie Design Thinking eingesetzt werden kann, um ein mensch-zentriertes Requirements Engineering zu erreichen. Dazu ist die Dissertation in drei Teile gegliedert. Basierend auf empirischen Fallstudien analysiert die erste Forschungsfrage den Nutzen von Design Thinking für zwei Hauptaktivitäten des Requirements Engineering, nämlich, die Ermittlung und Spezifikation von Anforderungen. Die zweite Forschungsfrage kombiniert und vergleicht die Artefakte, die aus der Anwendung von Design Thinking und Requirements Engineering im Kontext innovativer softwareintensiver Systeme entstehen. Basierend auf einer explorativen Fallstudie leitet die dritte Forschungsfrage drei Handlungsstrategien ab, die Design Thinking und Requirements Engineering je nach Projektkontext unterschiedlich kombinieren. Die Dissertation folgende wichtige theoretische Beiträge. Erstens bietet sie, über aktuelle Forschungsergebnisse hinaus, empirisch fundierte Belege und gibt Aufschluss darüber, wie allgemeine Herausforderungen im Requirements Engineering mit Design Thinking angegangen werden können. Zweitens trägt sie mit einem evaluierten Referenzmodell für innovative softwareintensive Systeme zur Forschung des artefakt-orientiertem Requirements Engineering bei. Drittens vertieft sie das Gestaltungswissen zur Verwendung von Design Thinking für das Requirements Engineering. In diesem Zusammenhang wird die differenzierte Sicht auf Design Thinking als Prozess, Toolbox und Denkweise konkretisiert und erweitert. Für Praktiker bietet die Dissertation konkrete Empfehlungen, wie die Effektivität von Requirements Engineering mit Hilfe von Design Thinking, abhängig von Projektkontext und -ziel, gesteigert werden kann. Insgesamt vermitteln die Ergebnisse dieser Dissertation ein besseres Verständnis dafür, *was* die Anwendung von Design Thinking im Kontext von Requirements Engineering bedeutet und *wie* beide Ansätze in innovativen softwareintensiven Entwicklungsprojekten kombiniert werden können.

# 1 Introduction

## 1.1 Motivation

A successful information system (IS) anchors in a comprehensive understanding of its users' needs and requirements (Maguire and Bevan 2002). "Getting the requirements right" has always been a desired objective in Requirements Engineering activities and is naturally seen as one of the most significant activities in development projects (Broy 2006; Roberston and Robertson 2013). However, many companies still struggle in this complex endeavor. One of the main reasons is that a lot of factors are vague at the beginning of a project, which makes the task inherently complex and volatile. A survey of IT software projects revealed that 33% of the reasons for project failures originate from insufficient Requirements Engineering (Emam and Koru 2008; Méndez Fernández and Wagner 2014). Further studies have shown that requirements errors may represent 40% of the total project costs; when they are found late in the development process, their correction can make up to 200 times more than when correcting them during the early stages (Venkatesh Sharma and Kumar 2013). Recent research has summarized a variety of challenges in existing Requirements Engineering techniques (Méndez Fernández and Wagner 2014); in particular, actual user needs are disregarded, requirements are invented, are solely based on the requirements engineer's intuition, or they lack creativity (Inayat et al. 2015). Accordingly, the Requirements Engineering community calls to improve the Requirements Engineering process for the development of software-intensive systems (Hickey and Davis 2004; Méndez Fernández and Wagner 2014).

With its growing relevance in agile software development, Design Thinking has gained recognition as a creative problem-solving method, particularly when the real-world problem is complex or "wicked" (Buchanan 1992). Some researchers consider Design Thinking as a "modern form of requirements engineering" (Beyhl and Giese 2016, p.288) addressing some of the aforementioned challenges in current Requirements Engineering practices. Design Thinking leverages interdisciplinary teamwork, a structured approach of ethnographic methods, and fast and simple prototyping cycles to produce innovative solutions in early product, service, and system development processes (Brown 2008; Kolko 2015). This rather diverging nature of problem-solving

is notably different from the more converging ways of Requirements Engineering practices in most software-intensive projects (Harte et al. 2017).

The multi-facetted opportunities of applying Design Thinking for Requirements Engineering are highlighted by several authors. Vetterli et al. (2013) was one of the first who suggested to bring Design Thinking and Requirements Engineering together for developing software applications. Academics with a content-focused view (*what* value does Design Thinking add) have recognized its value in terms of product quality, user acceptance, and process speed, mostly in specific domains like learning environments (Soledade et al. 2013), social innovation (Newman et al. 2015), or health care (Harte et al. 2017). Academics with a process-focused view (*how* does Design Thinking add value), examine usage schemes of Design Thinking with software engineering techniques and agile development toolkits. For instance, authors have investigated the integration of Design Thinking and Scrum (e.g. Häger et al. 2015; Przybilla et al. 2018) and found evidence for higher innovation potential stemming from a combination of both approaches.

Although mainly practice-oriented literature suggests potential benefits of combining Design Thinking and Requirements Engineering, knowledge on how this is actually operationalized and which concrete challenges and benefits to expect still remains unclear (Beyhl and Giese 2016; Vetterli et al. 2013). While Requirements Engineering is a mature discipline with a long research and practice history, there is still limited knowledge about Design Thinking as Yoo (2017, p.v) emphasizes in his call to "advance the intellectual foundation of Design Thinking" for IS research. Little is known, in particular, about the specific impact on Requirements Engineering. A deeper understanding of Design Thinking would enable both communities, Requirements Engineering and Design Thinking, to evaluate its application purpose and potential for discovering and specifying requirements more thoroughly. My research interest arises from the call for a more effective Requirements Engineering and Design Thinking as a promising 'cure'.

This dissertation aims to address this call. The objective of the dissertation is to define usage schemes of Design Thinking for a more effective Requirements Engineering in the context of innovative software-intensive development projects. In particular, the objectives are (1) to understand the benefits and challenges of using Design Thinking to elicit and specify requirements, (2) to create an artifact model, integrating both approaches, and (3) to derive concrete operationalization strategies for Design Thinking and Requirements Engineering.

The context for this dissertation is provided by innovative software-intensive development projects as the unit of analysis. A *software-intensive system* is defined as "any system where software contributes essential influences on the design, construction, deployment, and evolution of the system as a whole" (IEEE1471-2000). In addition, typical software-intensive systems consist of software as well as of hardware (Braun et a. 2014, p. 22). To do a proper integration of Design Thinking and Requirements Engineering justice, this dissertation concentrates on projects with an innovative character.

An *innovative* software-intensive development project is characterized by an unknown project goal and system vision. It has been regularly mentioned that the main purpose of Design Thinking is to solve design problems, especially "wicked" ones (Buchanan 1992), and create (disruptive) innovations that meet people's needs in new ways (e.g. Badke-Schaub et al. 2011; Leavy 2010). Researchers confirm this and recognize the application of Design Thinking particularly in the early stages of software-intensive development projects, when uncertainty about needs is especially high (e.g. Vetterli et al. 2013; Kolko 2015). This implies that during innovative software-intensive development projects design choices are to be made by the project team and project manager.

## 1.2 Research Questions and Research Methods

Derived from the goal and context of the dissertation, the following three primary research questions with separate research methods are followed.

| | |
|---|---|
| **RQ 1** | *What are benefits and challenges of using Design Thinking for the elicitation and specification of requirements?* |
| **RQ 2** | *How does an integrated model of Design Thinking and Requirements Engineering look like?* |
| **RQ 3** | *What are operationalization strategies for combining Design Thinking and Requirements Engineering?* |

Each research question includes one or several studies with substantial standalone contributions. The first research question uses a qualitative approach to describe the benefits and challenges of using Design Thinking to elicit and specify requirements. Building upon these findings, the second research question aims to create a model integrating Design Thinking and Requirements Engineering based on an artifact-oriented approach. Finally, the third research question integrates the previous findings and aims to derive three ways of operationalizing Design Thinking in the context of Requirements Engineering. The following paragraphs describe each research question and the methodological approach used in the studies in more detail.

| | |
|---|---|
| **RQ 1** | *What are benefits and challenges of using Design Thinking for the elicitation and specification of requirements?* |

The first research question aims to understand the benefits and challenges of using Design Thinking for eliciting and specifying requirements. On the one side, practitioners in Requirements Engineering have voiced a plethora of elicitation challenges that might be "cured" by using Design Thinking (Méndez Fernández and Wagner 2014). On the other side, prior literature has mentioned challenges in Design Thinking projects, especially in the context of requirements specification and the transition to software development processes (Beyhl and Giese 2016; Häger et al. 2015). Here, established practices from Requirements Engineering are considered helpful. As literature is scarce, an empirical approach is taken to provide robust evidence for examining this RQ with two separate multiple-case studies (Yin 2011). Both studies stem from a 10-months university course, where teams approached a problem statement provided by a corporate sponsor and guided by the Design Thinking process. To support validity for the findings, empirical data from multiple sources of evidence are collected (Eisenhardt 1989).

The first study presents seven key findings from employing Design Thinking for requirements elicitation practices. Potential benefits of using Design Thinking address elicitation challenges regarding (1) process guidance, (2) stakeholder communication, and (3) requirements quality. First, the process of eliciting requirements, which researchers have criticized to be insufficiently guided (Browne and Rogich 2001), can be steered positively and creatively by the Design Thinking process model, which might even compensate for a lack of professional experience of the analyst. Second, stakeholder communication is considered one of the main challenges in eliciting requirements (Zowghi and Coulin 2005) and can be supported when using low-resolution prototypes to expose tacit knowledge of relevant stakeholders. Third, quality issues can be partially addressed by using Design Thinking. Although Design Thinking also faces challenges in terms of traceability, correctness, consistency, and completeness, it derives a comprehensive set of requirements from a user point of view. The continuous integration of stakeholders throughout the process supports the understanding of changing requirements and has potential to resolve the problem of invented requirements that are not based on real user needs. The findings provide an advanced understanding of the benefits of using Design Thinking for Requirements elicitation practices and give concrete advice for practitioners.

The second study is concerned with presenting six challenges for specifying requirements with Design Thinking. The findings elaborate on three challenges that relate to the output of Design Thinking: (1) User requirements are emphasized, while software and system requirements are neglected, (2) critical artifacts are insufficiently traceable within Design Thinking (i.e. needs, insights, learnings, requirements), and (3) documentation shortcomings limit the use of project results in later implementation stages. In addition, three challenges can be attributed to the Design Thinking process: (4) Project members lack motivation to specify requirements systematically, (5) they perceive resource constraints to be hindering, and (6) they miss adequate tool support for documenting and structuring (implicit) team knowledge. The findings of this article complement findings from the first study to provide a holistic picture of benefits and challenges of applying Design Thinking for two core activities in Requirements Engineering, i.e. to elicit and specify requirements.

**RQ 2** *How does an integrated model of Design Thinking and Requirements Engineering look like?*

The second research question aims at designing a model that integrates Design Thinking and Requirements Engineering on an artifact-level in order to better understand combination possibilities. An artifact-oriented view specifies *what* has to be done while neglecting the particularities of processes and methods which are barely comparable across both approaches and projects (Méndez Fernández et al. 2014). For Requirements Engineering, an existing and evaluated artifact model is used (Méndez Fernández and Penzenstadler 2014a). For Design Thinking, the most relevant artifacts have been developed based on literature, cases from academia-industry collaborations (as evaluated in RQ1), and based on the results of a Delphi study.

The development of a combined artifact-based reference model of Design Thinking and Requirements Engineering is inspired by the design science research approach (Hevner et al. 2004; Peffers et al. 2007). The emergent model has been continuously tested with Design Thinking and Requirements Engineering experts and practitioners to adapt the relevant artifacts and their interdependencies for a comprehensive overview. The resulting model holds 40 artifacts; 16 of these are exclusively assigned to Requirements Engineering, 16 to Design Thinking, and 8 to both (sometimes with differing purpose). It establishes a blueprint of relevant artifacts, i.e. the work results, contents, and dependencies of each approach. The artifact model helps to deduct relevant operationalization strategies as established in RQ 3.

**RQ 3** *What are operationalization strategies for combining Design Thinking and Requirements Engineering?*

The third research question investigates three operationalization strategies for using Design Thinking and Requirements Engineering. They are specified and evaluated in real-world project settings and based on exploratory (longitudinal) case study research (Yin 2011). For each strategy, benefits and challenges are reflected. First, *upfront Design Thinking* is applied to produce Design Thinking artifacts as a basis for performing Requirements Engineering activities. Design Thinking takes the form of a pre-project to identify relevant needs and produce a product vision with key features and functionalities. This approach is particularly suitable when the problem and solution

space is unclear. Second, *infused Design Thinking* utilizes selected methods, and, thus, selected artifacts from the Design Thinking toolbox to integrate them into existing Requirements Engineering efforts. This approach is particularly suitable when commonly used Requirements Engineering practices fail to address specific challenges like lack of creativity. Third, *continuous Design Thinking* integrates activities and artifacts of Design Thinking continuously into existing Requirements Engineering processes with the goal of a seamless transition of artifacts. This approach is often associated with a specific project role and suitable for addressing problems with high uncertainty. The definition of three integration strategies with their benefits and challenges enhances the understanding on how to integrate Design Thinking and Requirements Engineering and gives guidance on when to use which approach depending on the specific project situation.

The third approach is evaluated with rich qualitative data from a longitudinal study in a real-life setting. It illustrates how the continuous Design Thinking approach uses Design Thinking as a process, toolbox, and mindset in combination with Requirements Engineering practices. The findings reveal a morphing nature of Design Thinking from idea formulation to market-ready implementation in an agile project setting. Design Thinking evolves from (1) a pre-project and a process to understand the problem and define a product vision (upfront Design Thinking), via (2) the usage of Design Thinking as a toolbox to complement Requirements Engineering practices (infused Design Thinking), to (3) a manifestation of a human-centered mindset among the entire project team pushing a continuous product delivery (continuous Design Thinking). The findings suggest a human-centered Requirements Engineering approach including principles, methods, and artifacts from both Design Thinking and Requirements Engineering, allowing for a back and forth between the two concepts. In other words: such an integration allows to give Requirements Engineering an arm into “wicked” problems and Design Thinking a seamless transition to subsequent software engineering and quality assurance activities.

## 1.3 Structure of the Dissertation

To address the previously defined research questions, the remainder of this dissertation comprises 12 major chapters. Figure 1 presents an overview of the structure of the dissertation and the content of each chapter.

Chapter 2 provides the conceptual background of the dissertation for all subsequent chapters. First, in section 2.1, socio-technical theory is introduced as a theoretical foundation for the context of the dissertation. Sections 2.2 and 2.3 outline the key characteristics of Requirements Engineering and Design Thinking. Both sections review elements and methods considered essential for the respective approach. Finally, in section 2.4, a brief discussion of related work in the fields of Design Thinking and Requirements Engineering provides further insights into the current research on the topic.

Chapters 3 and 4 build upon this conceptual background and addresses the first research question of the dissertation. The chapters are based on the results of two multiple-case studies. Chapter 3 examines the usage of Design Thinking for requirements elicitation by addressing known elicitation challenges. Chapter 4 focuses on the challenges of specifying requirements through Design Thinking and how Requirements Engineering might help to address these challenges.

Chapter 5 addresses the second research question of the dissertation. It investigates how Design Thinking and Requirements Engineering can be combined depending on their produced artifacts. The chapter reports the results of a design science research inspired study that was conducted based on the experience of academia-industry collaborations. It describes the layers and artifacts in the combined model that can shape a human-centered Requirements Engineering approach.

Chapters 6 and 7 address the third research question and establish operationalization strategies to combine Design Thinking and Requirements Engineering based on the previous findings. Chapter 6 examines three operationalization strategies: (1) upfront, (2) infused, and (3) continuous Design Thinking and their respective challenges and benefits. Chapter 7 extends these findings and strengthens the continuous Design Thinking approach with the results of a longitudinal case study.

Chapter 8 consolidates the results of the previous chapters and synthesizes the main findings of the dissertation. The goal is to present an overall discussion of the results of chapters 3 to 7 with regard to the three research questions.

Chapters 9 and 10 outline the theoretical and practical contributions of the dissertation. Chapter 9 presents the contributions for research. Chapter 10 illustrates the practical implications. They show how the findings may be used to improve the elicitation and specification of requirements for innovative software-intensive development projects.

Chapter 11 acknowledges the limitations of the dissertation and emphasizes potential avenues for future research to extend the findings of the dissertation. It provides an outlook and potential research agenda for studies that intend to delve deeper into using Design Thinking for Requirements Engineering.

Chapter 12 concludes the dissertation and offers a summary.

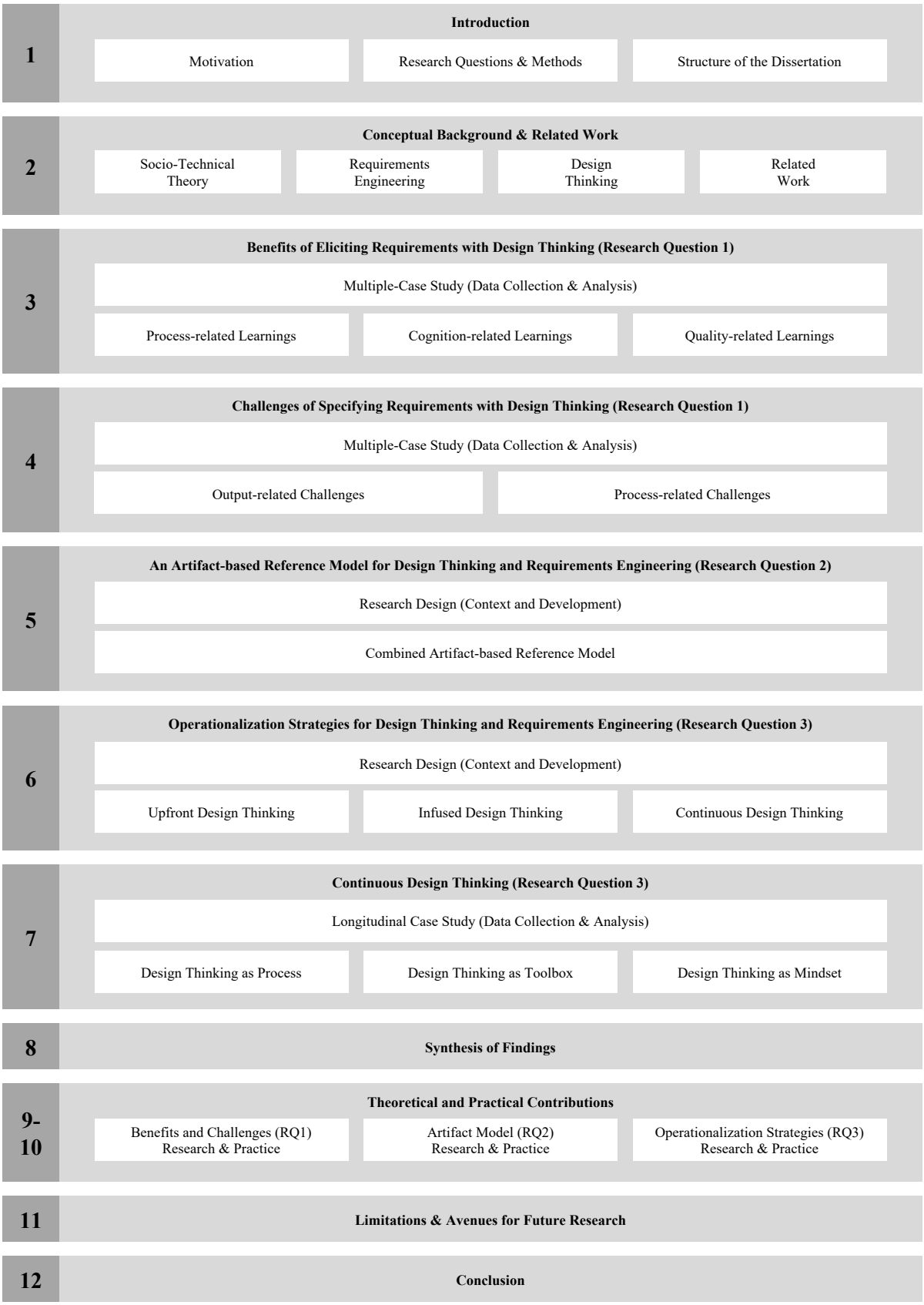



**Figure 1: Structure of the Dissertation**

**(Source: own illustration)**

## 1.4 Overview of Publications

Parts of this dissertation have been published in proceedings of peer-reviewed conferences and/or are accepted for publication. Table 1 provides an overview of these papers and indicates in which chapters the content of these papers has been used.

| No. | Publication | Chapter | RQ |
|---|---|---|---|
| 1 | Hehn, J., Uebernickel, F. 2018. "Towards an understanding of the Role of Design Thinking for Requirements Elicitation – Findings from a Multiple-Case Study," *Proceedings of the 24th Americas Conference on Information Systems (AMCIS 2018).* New Orleans, USA: AIS | 1, 2, 3, 8, 9, 10, 11 | RQ1 |
| 2 | Hehn, J., Uebernickel, F., Stöckli, E., Brenner, W. 2018. "Towards Designing Human-Centered Information Systems: Challenges in Specifying Requirements in Design Thinking Projects," *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2018).* Lüneburg, Germany: AIS. | 1, 2, 4, 8, 9, 10, 11 | RQ1 |
| 3 | Hehn, J., Uebernickel, F., Herterich, M. 2018. "Design Thinking Methods for Service Innovation – A Delphi Study," *Proceedings of the 22nd Pacific Asia Conference on Information Systems (PACIS 2018)*. Yokohama, Japan: AIS. | 1, 2, 5, 8, 9, 10, 11 | RQ2 |
| 4 | Hehn, J., Mendez, D., Uebernickel, F., Brenner, W., Broy, M. 2020. "On Integrating Design Thinking for a Human-centered Requirements Engineering," *IEEE Software, Special Issue Design Thinking*. | 1, 2, 5, 6, 7, 8, 9, 10, 11 | RQ2<br>RQ3 |
| 5 | Hehn, J., Uebernickel, F. 2018. "The Use of Design Thinking for Requirements Engineering – An Ongoing Case Study in the Field of Innovative Software-Intensive Systems," *Proceedings of the 26th IEEE International Requirements Engineering Conference (RE'18).* Banff, Canada: IEEE. | 1, 2, 6, 7, 8, 9, 10, 11 | RQ3 |
| 6 | Hehn, J., Uebernickel, F. 2019. "The Use of Design Thinking for Requirements Engineering – An Ongoing Case Study in the Field of Innovative Software-Intensive Systems," *Lecture Notes: 49. Jahrestagung der Gesellschaft für Informatik*, Extended Abstract. | 7 | RQ3 |

**Table 1: Overview of Publications**

**(Source: own illustration)**

# 2 Conceptual Background and Related Work

The following sections provide the conceptual background of the dissertation.[1] The first section establishes the context for this dissertation with the help of the socio-technical systems theory (section 2.1). The second part introduces the core concepts of Requirements Engineering (section 2.2) and Design Thinking (section 2.3). The last section deals with related work regarding the combination of both concepts (section 2.4).

## 2.1 Socio-Technical Systems Theory

The term 'socio-technical systems' (STS) was coined by Emery and Trist (1960) to describe complex organizational work design that acknowledges the interaction between social humans and technical systems. STS theory highlights the importance of *joint optimization* of social and technical sub-systems in an organization to solve complex issues (Cooper and Foster 1971; Bostrom and Heinen 1977). Four interrelated components should be taken into account when developing new information systems (IS) (Leavitt 1965): the social sub-system with *actors* and *structures*, and the technical sub-system including *technology* and *tasks*. Actors are defined by the employees' capabilities and a shared culture. Structures are defined by project organizations and institutional arrangements, technology by tools and technological systems, and tasks by the required activities to fulfill work or deliver services (Lyytinen and Newman 2008). The interactions between the social and technical systems are inherently recursive as Orlikowski (2000) emphasizes: "Users shape the technology structure that shapes their use." (p. 407)

Understanding the connection between the socio-technical components is critical when introducing new work routines, business processes, or technical systems (Baxter and Sommerville 2011). STS theory has been frequently applied within IS research to investigate IS-induced changes in the organizational context (Lyytinen and Newman 2008). Originating from the workplace, STS is nowadays quite broadly used to describe a variety of other complex settings where technology is deployed, for example in health applications (Whetton 2005) or in the context of smart home-based systems (Sommerville and Dewsbury 2007).

[1] Parts of this chapter have been published in proceedings of peer-reviewed conferences. Please refer to section 1.4 for an overview.

Socio-technical system design methods were developed to facilitate the creation of such systems (see Mumford (2006) for a historical development of such methods). Early work focused mainly on manufacturing industries and emphasized a humanistic approach to work as opposed to the mechanistic view with division of labor as promoted by Taylor (1911). Since then socio-technical system design methods had their ups (e.g. in the 70s when labor shortages prompted organizations to retain their staff by all means available) and downs (e.g. in the 80s when lean production techniques and business process re-engineering dominated) (Baxter and Sommerville 2011). The emergence of ethnographic approaches in the late 1980s initiated the importance of socio-technical issues in the design of software-intensive systems (e.g. Blomberg 1988; Heath and Luff 1991), which have been growing ever since. Common methods that support socio-technical ideas include approaches such as participatory design (e.g. Greenbaum and Kyng 1991), empathic design (e.g. Leonard and Rayport 1997), contextual design (e.g. Beyer and Holtzblatt 1999), or human-centered design (ISO 2010). Although differing in some respects, all of these methods follow the same principles of basing design on an explicit understanding of users, their tasks, and the context, in which the system will be used.

*Dissertation context:* The activities of performing Design Thinking and/or Requirements Engineering can be understood as a socio-technical phenomenon affecting organizations in multiple ways. Accordingly, STS theory offers a holistic view on a combination of both concepts. The combined approach of Design Thinking and Requirements Engineering can be interpreted as a design method to support the creation of STS. This requires a change in social and technical systems in terms of technology (i.e. the system to be developed), task (i.e. requirements and needs analysis activities), actors (i.e. project team with their knowledge and capabilities), and structures (i.e. new project framework with a combination of Design Thinking and Requirements Engineering activities). With regard to the social system, actors require applied skills in the areas and intersections of Design Thinking and Requirements Engineering to successfully conduct, lead, manage, and implement a combined approach (task). With regard to technology, the design and implementation of the system is required to provide the technological vehicle to address the users' and stakeholders' needs. Further, an organization needs to transform its organizational structures in a way that it supports and establishes this (new) way of working. Therefore, combining Design Thinking and

Requirements Engineering can be regarded as a strategic capability for gaining insights and design knowledge. Organizations need to analyze and understand how they can use and manage a combined approach in terms of skills, resources, and capacity to achieve the potential advantage of a combination.

## 2.2 Requirements Engineering

Requirements Engineering has been researched and practiced for over 40 years, when professionals recognized the benefits of adopting a systematic approach to discovering and managing those requirements for a software development process (Nuseibeh and Easterbrook 2000). Requirements Engineering constitutes activities within the initial phase of software-intensive development projects. Its objective is to make sure that the intended system meets the needs of its users, customers, and product owners (Broy 2014). The key is to decide on the functionality and the quality of the software to be produced. Requirements Engineering, thus, has to make a decision on the overall functionality, its details in the steps of interaction, the physical user interface, and its various quality attributes.

According to the International Requirements Engineering Board (IREB), Requirements Engineering is defined as "a systematic and disciplined approach to the specification and management of requirements with the following goals: (1) Knowing the relevant requirements, achieving a consensus among the stakeholders about these requirements, documenting them according to given standards, and managing them systematically, (2) understanding and documenting the stakeholders' desires and needs, (3) specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs." (Glinz 2018, p. 18)

### 2.2.1 Requirements and their Classification

A requirement can be understood as "a need perceived by a stakeholder; a capability, or property that a system shall have; a documented representation of a need, capability, or property," (Glinz 2018, p. 17). Requirements are typically specified by natural language, diagrams, or models.

A unified classification for requirements does not exist. The classification of requirements and the respective choice of attributes is dependent on the specific characteristics of the application domain. According to Ebert and Wieringa (2005), one

reason for the lack of agreement on a comprehensive requirements classification is that the discipline of Requirements Engineering itself is not well linked with other software life-cycle activities such as quality assurance or project management. Nevertheless, there is a common understanding that there are three requirement levels, independent from the respective domain: (1) *why* the system under consideration is needed (context/business requirements), (2) *what* functionalities will satisfy this context (functional requirements), and (3) *how* the system is to be constructed (system qualities (also called non-functional requirements) such as security, performance, safety, usability etc., and constraints (Chung and do Prado Leite 2009; Glinz 2007).

*Dissertation context:* Requirements classification plays an essential role in structuring the layers of the combined artifact model in the context of this dissertation. It is explained and incorporated into the artifact model of chapter 5.

### 2.2.2 Key Activities in Requirements Engineering

There are different process models that guide Requirements Engineering activities (e.g. van Lamsweerde 2009; Nuseibeh and Easterbrook 2000). In general, they include four key activities independent of their alignment:

(1) *Elicitation* considering the discovery of user needs and requirements in collaboration with all relevant stakeholders (Zowghi and Coulin 2005).

(2) *Analysis and negotiation* considering the refinement of the requirements in terms of abstraction levels and their classification (Geisberger et al. 2006).

(3) *Specification* considering the modelling and documentation of requirements in a structured form, for example according to standards like the IEEE Std. 830-1998 (IEEE 1998).

(4) *Validation* of requirements ensuring completeness, correctness, consistency, and clarity for the system to be developed and the contexts in which the system will be used (Nuseibeh and Easterbrook 2000).

The activities are usually performed iteratively. Various frameworks exist in literature – including linear, incremental, non-linear, and spiral models – combining the key activities in a notion of a development process model (see Aurum and Wohlin (2005) for an overview).

As a multidisciplinary domain Requirements Engineering makes use of many technological approaches and various techniques for eliciting, modeling, analyzing, and managing requirements (Wiegers 2003). The selection of the specific Requirements Engineering techniques used depends on best practices, the organizational culture of a particular company, and on the application domain. Zowghi and Coulin (2005), for example, provide an overview of various elicitation approaches and classify corresponding methods that can be used.

*Dissertation context:* In this dissertation, all of the above-mentioned activities are considered for interpreting the results. A particular focus is put on requirements elicitation. As elaborated in section 2.4, this key activity is considered to be especially suited for the application of Design Thinking. The empirical results of chapters 3 and 5 support these indications.

### 2.2.3 Artifact-orientation in Requirements Engineering

Artifact-orientation has become a well-accepted approach in software-intensive development processes (Méndez Fernández et al. 2019). An artifact is defined as an "intermediate or final result of system development" (Glinz 2014, p. 9). Typically, artifacts inherit particular properties and can be described using standardized modeling concepts. To explain artifact-orientation, authors regularly contrast it with activity-orientation (e.g. Méndez Fernández and Penzenstadler 2014a). While the former emphasizes *what* to produce, the latter highlights *how* to approach a project, for example by defining the activities and how to combine them. According to Hammerschall (2008), activity-based approaches either neglect artifacts completely, see them as an optional element, or when stated mandatory, they are considered at the project's periphery or they lack interrelations. In the context of a software-intensive development project, artifact-orientation means that the project participants agree on the artifacts to be produced independently from how they are realized. The complexity of a development process is, thus, reduced by predefining a set of deliverables and their structure.

Artifact models provide a framework to structure artifacts and their dependencies throughout the development process. In practice, the first artifact models provided a common understanding on the general contents to be considered in Requirements Engineering, mainly as part of guidelines and checklists like the 'VOLERE Requirements Specification' templates (Robertson and Robertson 2018) or the 'IEEE

Recommended Practice for Software Requirements Specifications (IEEE std. 830-1998)' (IEEE 1998). Those templates, however, neglected interdependencies between contents, which is important when supporting syntactically consistent result structures (Méndez Fernández and Penzenstadler 2014a). Berenbach et al. (2009) were among the first to consider content-related dependencies and described how to model artifacts and key components. They created a measurable reference model that can be tailored to the specific domain application and provided process guidelines. Building on this foundation, several authors have suggested domain-specific (e.g. Silva et al. 2009) and domain-independent artifact models (e.g. Méndez Fernández and Penzenstadler 2014a) with varying claims for comprehensibility. An example for an artifact model in the context of a development process model is the V-Modell XT (Friedrich et al. 2008), which has an artifact model at its core for defining all relevant sub-models and their project execution (Kuhrmann and Hammerschall 2008). The steadily growing interest in artifact-orientation and its differences in interpretation caused Méndez Fernández et al. (2019) to put out a position paper with the goal to standardize the term artifact and syntactically consistent result structures for software engineering "so that researchers and practitioners have a common understanding for discussions and contributions" (p. 2254). They contribute a meta model that provides a description of an artifact that is independent from any underlying process model.

*Dissertation context:* One of the main objectives of this dissertation is the development of an artifact-based reference model for Design Thinking and Requirements Engineering. The construction builds on the current body of knowledge in artifact-based Requirements Engineering.

### 2.2.4 Requirements Engineering in Agile Development Settings

Requirements Engineering and agile methods are often seen at opposite ends. The former is associated with heavy documentation for knowledge sharing. The latter leverages cyclical face-to-face collaboration with customers and developers to reach similar goals (Paetsch et al. 2003). Authors have stressed that the main difference between the traditional and agile approach is not *whether* to do Requirements Engineering, but rather *when* to do it (e.g. Paetsch et al. 2003). In traditional settings, Requirements Engineering focuses on defining all requirements in a specification document before system development starts. In agile settings, Requirements

Engineering is performed repeatedly throughout the development process. This iterative character often leads to a just-in-time model to refine the high-level requirements and derive them into low-level tasks that can be implemented by developers (Schön et al. 2017). Changing requirements, even late in the development lifecycle, can then be addressed appropriately (Vetterli et al. 2013). This circumstance solves some of the mentioned challenges in Requirements Engineering, however, it also produces new challenges which are summarized in Table 2. These challenges will be discussed in chapter 7.

| Challenges | Implications for Requirements Engineering |
|---|---|
| Minimalistic documentation | Requirements in minimalistic documentations are difficult to trace back (Inayat et al. 2015; Heikkila et al. 2015) |
| Problems with customers or users | Hindering access to and communication with customers slows down the process (Inayat et al. 2015; Heikkila et al. 2015; Méndez Fernández and S. Wagner 2016) |
| Neglect of non-functional requirements | User stories mostly focus on system and product features, not non-functional requirements (e.g. security, usability) (Inayat et al. 2015) |
| Inappropriate architecture | Due to short planning time horizon, the architecture might be inadequate or inappropriate (Inayat et al. 2015) |
| Tacit requirements knowledge | Most knowledge often stays tacit as agile practices rely on highly skilled people (Heikkila et al. 2015; Méndez Fernández and S. Wagner 2016) |
| Imprecise effort estimates | Estimates of time and cost are difficult due to the agile project character (Inayat et al. 2015; Heikkila et al. 2015) |
| Difficulties in the prioritization of requirements | Focus on immediate business value as prioritization focus might cause neglect of system related requirements (Inayat et al. 2015; Heikkila et al. 2015) |

**Table 2: Challenges in Agile Requirements Engineering**
**(Source: Hehn and Uebernickel 2018b)**

*Dissertation context:* In the context of this dissertation, Design Thinking and Requirements Engineering are seen from a mainly process-agnostic perspective in order to concentrate on artifacts, i.e. on what has to be done rather than on particularities when or how to do it. For this purpose, the distinction of upfront or agile Requirements Engineering can be neglected. However, for the evaluation of the operationalization strategy of *continuous Design Thinking*, Design Thinking has been analyzed in the context of Requirements Engineering practices in an agile project environment (see chapter 7).

## 2.3 Design Thinking

Design Thinking has become increasingly important in both the academic and the business communities over the past two decades. Industry studies have highlighted this significant development. For example, based on a survey of the Hasso-Plattner Institute (Schmiedgen et al. 2015), over 69% of Design Thinking practitioners and managers identified Design Thinking as one of the major contributors to conduct an efficient innovation process. In a survey of IBM by Forrester (2018), Design Thinking was reported to reduce development and testing time by 33%, equating cost savings of around $1.1M per major software development project. In academia, Yoo (2017) commented on the emerging opportunities for IS academics as follows: “IS scholars can deepen our understanding on the linkage between digital artifacts and human experiences by explicitly embracing Design Thinking practices in their research” and, thus, help to “shape human experiences in a digital world” (p. v).

The opportunities associated with Design Thinking in IS and other areas have helped generate significant interest in this way of problem-solving, which is often referred to as “a human-centered approach to innovation that draws from the designer's toolkit to integrate the needs of people, the possibilities of technology, and the requirements for business success.” (Brown 2012) The roots of Design Thinking date back to the late 1960s, when design academics examined the mental processes that underlie design activities and transformed them into normative guidelines for creative problem solving (Simon 1969). These studies have expanded the scope of design beyond the boundaries of product styling to a way of thinking that can now be universalized for a multitude of disciplines (e.g. management, business, software development, engineering).

On an operational level, Design Thinking is interpreted in three ways: as (1) a process with a sequence of steps according to a prescriptive process framework, (2) a toolbox with a collection of methods for situational support, and (3) a mindset with a set of human-centered principles to be internalized (see Figure 2). While all three modes are interlinked, they result in different conceptualizations on a practical level. As Fraser (2011) suggests, “it takes a combination of the right mindset (being) and a rigorous methodology (doing) that unlocks a person’s thinking, and that one must consider all three of these factors.” (p. 71)

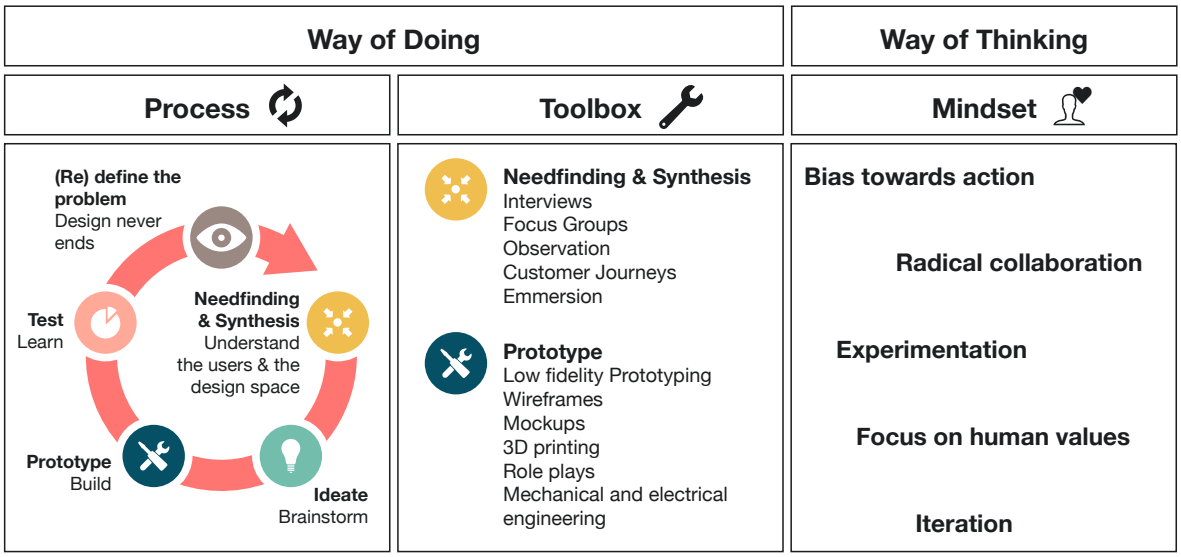



**Figure 2: Design Thinking as Process, Toolbox, and Mindset**
**(Adapted from ME310 and Brenner et al. 2014)**

### 2.3.1 Design Thinking Process Model

The normative Design Thinking process model, as used in this dissertation, is adapted from Design Thinking courses at Stanford University and the University of St.Gallen. It can be divided into two main phases: the *problem space* with (1) problem definition, (2) needfinding, (3) synthesis and the *solution space* with (4) ideation, (5) prototyping, and (6) testing (ME310 2010). Both overarching phases include an interplay of divergent activities (information gathering) and convergent activities (information processing). The problem space demands diligent examination of the problem context by integrating all relevant stakeholders the synthesis of all collected information to a clearly defined point of view, including needs and insights. The solution space encourages the generation of ideas and the creation of prototypes, which can be evaluated and tested with users. The process is repeated several times until a final solution can be presented. Reflection points are carried out during the process wherever necessary as they are crucial steps for adapting to novel information and developing deeper insights. Each cycle stimulates creativity and encourages rapid learning through trial-and-error.

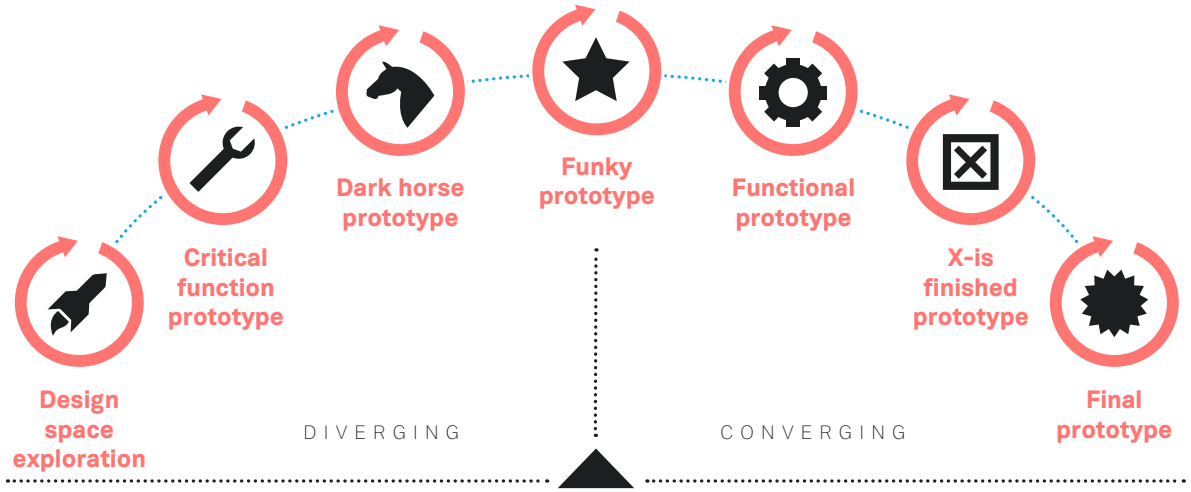



**Figure 3: Design Thinking Macro Process**
**(Adapted from Brenner et al. 2015)**

The iterative process model is integrated into a broader milestone framework (also known as the 'macro-process'). This framework encourages the team to iterate their prototypes in six different prototyping phases, which eventually merge to one final prototype. Divergent activities have the following milestones: (1) Design Space Exploration (to gain an in-depth understanding of the problem space), (2) Critical Function/Experience Prototype (to test and define the core functionalities of the solution), and (3) Dark Horse Prototype (to challenge key assumptions and boundaries with visionary prototypes). Convergent activities have (4) Funky Prototype (to combine the most promising elements into system prototypes), (5) Functional Prototype (to define the system vision and scope of the final solution), (6) X-is finished Prototype (to finalize one single function of the final prototype), and (7) Final Prototype (to integrate the insights and most successful features into one final, high-resolution prototype) (Uebernickel et al. 2015). In the context of software development and engineering, Design Thinking is mainly leveraged as a collaborative approach to develop innovative digital solutions. Design Thinking is about understanding the problem before thinking (too soon) about predetermined solutions or results. Hence, the actual problem analysis gains at least as much, often even more, attention as finding and developing ideas (Brown 2008).

*Dissertation context:* The presented Design Thinking process models have been the guiding framework for eliciting and defining requirements in the (multiple-)case studies as introduced in chapters 3, 4, and 7.

### 2.3.2 Design Thinking Toolbox

A wide range of practitioner catalogs of Design Thinking methods and tools have emerged in recent years (Doorley et al. 2018; IDEO.org 2015; Uebernickel et al. 2015). Design Thinking as a toolbox applies selective (design) methods and techniques for situational support (e.g. storytelling, empathy maps, point of view, brainstorming, ethnographic research, and journey mapping) (Liedtka & Ogilvie 2012). A Delphi study with Design Thinking experts from industry and academia generated a comprehensive list of initial 172 methods, which were distilled to 59 core methods that can be attributed to Design Thinking (for more details see Hehn et al. 2018). Contrary to the process-view, the toolbox offers a more flexible way of using Design Thinking and tailoring it to specific project conditions.

*Dissertation context:* The toolbox of Design Thinking is particularly important in the context of chapter 5 for developing the Design Thinking artifact model. Design Thinking methods and tools provide the basis for defining Design Thinking artifacts. In addition, the toolbox application is discussed as part of the operationalization strategies in chapter 6 and 7.

### 2.3.3 Design Thinking Principles

A growing number of authors stress that the core of Design Thinking goes beyond process models and tools (e.g. Kröper et al. 2010; Martin 2009). They perceive Design Thinking primarily as a mindset or general "design attitude" (Boland & Collopy 2004, p. 3) towards creative problem-solving. This entails the development of empathy, an open-minded and optimistic approach to generating insights and ideas, and the rationality to investigate and fit those ideas in compliance with the context. The main principles are highlighted in the following:

(1) Design Thinking puts the *emphasis on human values* as a starting point and foundation for all related activities (Brown 2008). Understanding what people need and want anchors in a deep empathy for users and is achieved by systematically integrating a variety of stakeholder groups throughout the development process, both through direct dialog and non-obtrusive observation methods.

(2) Solutions are mainly generated through *radical collaboration*, both with users and by composing a multidisciplinary project team that incorporates different functions and departments (Açar & Rother 2011; Doorley et al. 2018).

(3) Design Thinking leverages *abductive reasoning* to constantly generate new information and consider alternative options early on. The abductive nature of this way of working induces a "reflective conversation with the situation" (Schön 1984, p. 76) by looking beyond "what is" and exploring the logic of "what might be" to generate customer and business value (Martin 2009).

(4) Design Thinking emphasizes a *bias toward action*. This means that the preferred ways for gathering insights and feedback from stakeholders are hands-on activities such as experimenting with ideas, building prototypes, and testing them (Doorley et al. 2018; Leavy 2010).

(5) Design Thinking can be seen a "fundamentally exploratory process" (Brown 2009, pp.16-17) that encourages *rapid and iterative learning cycles*. According to the "fail early and often"-principle every iteration leads to further adjustments and new directions in the development process. In the long run, this iterative approach to development is supposed to mitigate risks of not meeting customer needs in the long run (Brown 2009).

*Dissertation context:* In the context of this dissertation, Design Thinking is mainly analyzed on a process and toolbox level as it looks at the artifacts created by Design Thinking and how they relate to Requirements Engineering. Nevertheless, the principles that form the Design Thinking mindset have been kept in mind when interpreting the results. Fostering a likewise mindset is of high importance especially when reflecting upon a sustainable integration of Design Thinking principles into common Requirements Engineering activities (see chapter 7)

## 2.4 Related Work

Researchers have highlighted the similarities between Requirements Engineering and Design Thinking and pointed at the closely related methods and principles of both approaches in the following ways: First, Requirements Engineering, in terms of user-centeredness, has always considered the discovery of user requirements as part of their activities in order to design better systems (Hansen et al. 2009). Second, Requirements

Engineering shares the same goal as Design Thinking in software-intensive development projects, namely, to define the purpose of a system/solution (Jarke et al. 2011). Finally, Requirements Engineering pays attention to the context in which systems are embedded (Kahan et al. 2019), which is similar to Design Thinking taking the problem-context into account.

Prior research has recognized the value of Design Thinking for Requirements Engineering in software-intensive projects, mainly by conducting experiments or case studies in a particular domain (e.g. Soledade et al. 2013; Vetterli et al. 2013; Carell et al. 2018). For example, researchers have pointed out the importance of empathy for defining privacy requirements (e.g., Levy and Hadar 2018), using Design Thinking to design health care applications (Carroll and Richardson 2016; Harte et. al. 2017), or interactive mobile apps (de Carvalho Souza and Silva 2015; Sandino et al. 2013). Using Design Thinking to address common challenges in Requirements Engineering is also highlighted by several researchers. Kahan et al. (2019) point at challenges in Requirements Engineering such as business process focus, systems transparency, distributed requirements, layers of requirements, packaged software, or interdependent complexity. They found that applying Design Thinking raises the awareness about own biases, which can enhance system transparency. They also highlighted that, due to its structured approach, Design Thinking can help to deal with the interdependent complexity of a system. The authors emphasize the benefits Design Thinking could provide for requirements elicitation and stress the "need for a specific process of requirement elicitation that will incorporate the most suitable Design Thinking techniques" (p. 84). Martins et al. (2019) similarly investigate Design Thinking as a means to solve challenges of elicitation when using agile methods. They provide evidence that Design Thinking strengthens stakeholder participation along the core Requirements Engineering activities, while, also indicating a neglect of considering non-functional requirements (except usability requirements).

Academics have also provided first evidence on how Requirements Engineering practices can strengthen the relevance of Design Thinking in software development, for example, by addressing documentation and traceability problems of Design Thinking with Requirements Engineering techniques (Beyhl et al. 2014; Beyhl and Giese 2016). Other researchers provide evidence for a beneficial integration of activities to address

hand-over challenges from Design Thinking to later staged software development activities (e.g., Häger et al. 2015; Przybilla et al. 2018).

*Dissertation context:* Literature on the effective integration of Design Thinking and Requirements Engineering is still in its infancy. Most contributions discussing an integrated view on Requirements Engineering and Design Thinking have so far focused on the integration of isolated methods or principles in particular application domains. This dissertation builds on these findings to achieve a structured assessment of both approaches for a joint result-oriented application.

# 3 Benefits of Eliciting Requirements with Design Thinking

This chapter addresses the first research question of the dissertation and is concerned with understanding the usage of Design Thinking for requirements elicitation and its respective benefits. The chapter presents the results of a multiple case study[2] that investigated the role of Design Thinking for requirements elicitation. This chapter, together with chapter 4, forms the foundation for all subsequent chapters of the dissertation that delve deeper into operationalizing a combination of Design Thinking and Requirements Engineering (i.e. chapters 5-7). Section 3.1 explains the motivation and objectives of the multiple-case study in more detail. Section 3.2 outlines the methodology of the study and provides a detailed description of the data collection and analysis. Section 3.3 reveals the results of the study. Section 3.4 discusses the implications for both theory and practice and shows the study's limitations and an outlook for future research. Section 3.5 concludes the chapter.

## 3.1 Challenges in Requirements Elicitation

Requirements elicitation is an early-stage but continuous activity in software-intensive development projects to determine the needs and requirements of all relevant stakeholders (Hadar et al. 2014). The tasks performed in requirements elicitation can vary but may be broadly categorized into (1) understanding the domain, (2) identifying relevant sources, (3) analyzing stakeholders, (4) selecting techniques, and (5) obtaining the requirements. In reality, this is a complex and iterative endeavor that heavily relies on the communication skills of the analyst, and involves many tasks, with multiple techniques at hand, to perform these activities (Zowghi and Coulin 2005). Thus, the Requirements Engineering community calls for research on improving requirements elicitation as this holds much promise for enhancing the success of software systems (Hickey and Davis 2004). Accordingly, we propose the following research question: *What can be learned from Design Thinking for requirements elicitation in software-intensive projects?*

To investigate our research question, we set up an exploratory multiple-case study, based on twelve cases from three consecutive years in multiple problem domains. We

[2] Parts of this chapter have been published in Hehn, J., Uebernickel, F. 2018. "Towards an understanding of the Role of Design Thinking for Requirements Elicitation – Findings from a Multiple-Case Study," Proceedings of the 24th Americas Conference on Information Systems (AMCIS 2018). New Orleans, USA: AIS

use existing theoretical and empirical knowledge about the challenges of requirements elicitation as a lens to analyze the contribution of Design Thinking.

Academics have generated long lists of challenges in requirements elicitation. Based on a structured literature review (Webster and Watson 2002), we have summarized the main challenges of requirements elicitation practices in terms of process guidance, stakeholder communication, and requirements quality. The results of our analysis are presented in Table 3.

| # | Challenges | Explanation | Key Literature |
|---|---|---|---|
| ***Process Guidance (Process)*** | | | |
| C1 | Lack of process guidance | There is little guidance on selecting the "right" method in different contextual situations. Many techniques developed in theory are not considered sufficiently useful for practice. | Carrizo et al. 2014; |
| C2 | Lack of creativity | The process of eliciting requirements is missing creativity and the application of creativity techniques to develop innovative software systems. | Mich et al. 2005; Maiden et al. 2004 |
| C3 | Insufficient technical translation of needs | Many terms used in the real world are hard to translate into exact technical specifications (e.g. 'user friendliness') and are, thus, easily misinterpreted or prone to 'wrong' execution. | Davey and Parker 2015 |
| ***Stakeholder Cognition and Communication (Cognition)*** | | | |
| C4 | Communication gaps with stakeholders | Stakeholders can encounter difficulties articulating their needs or requirements, e.g. due to cultural differences or a lack of common language between analyst and stakeholder. | Davey and Parker 2015 |
| C5 | Tacit knowledge is hard to grasp | Stakeholders cannot express their needs or requirements clearly because they have only tacit knowledge about them and the topic. | Zowghi and Coulin 2005 |
| C6 | Analyst has limited knowledge/capability | Analysts are not equipped with sufficient expertise (e.g. wrong choice of techniques) or capability (e.g. lack in soft skills) to perform effective requirements elicitation. | Hadar et al. 2014; Davey and Parker 2015 |
| ***Quality of Requirements (Quality)*** | | | |
| C7 | Requirements are invented, not elicited | Requirements don't anchor in 'discovered' needs but are rather invented by analysts or developers. | Karlsson et al. 2007 |
| C8 | Requirements volatility | Requirements change over time due to multiple reasons, e.g. new market trends, feedback from coding reviews, resource constraints, or the influence of new business requirements. | Karlsson et al. 2002; Davey and Parker 2015 |
| C9 | Lack of traceability and interdependency | Requirements are not traceable, especially in complex project environments, and/or their (dynamic) interdependencies are hard to deal with. | Karlsson et al. 2007 |
| C10 | Incorrect, incomplete, and inconsistent requirements | The set of elicited requirements may lack in the "three Cs": correctness, completeness, and/or consistency. | Zowghi and Coulin 2005 |

**Table 3: Challenges in Requirements Elicitation**

**(Source: own illustration)**

## 3.2 Research Methodology

We chose an exploratory study design since our research examines the learning potential of Design Thinking for eliciting requirements. Our aim was to generate generalizable and robust findings. This is why we selected cases that were mainly replications of each other. Therefore, a multiple-case study approach seemed most appropriate (Yin 2011).

The unit of analysis are project teams and the artifacts they produce by applying Design Thinking. Each team approaches a problem statement from a corporate sponsor guided by the Design Thinking process in a university setup as introduced in the research background. We selected twelve cases following the principles of similarity and variation (Eisenhardt 1989). We aimed at similarity in terms of project duration (8 months), an IS-related problem statement, a multi-disciplinary team configuration (mainly master students from business, design, informatics, and engineering), and the knowledge level of Design Thinking. The team members had no or little prior experience in using the presented Design Thinking approach for requirements elicitation. However, they shared the same level of coaching support from experienced Design Thinking experts to elicit needs and requirements to address their problem statement.

All cases were equipped with the same (or a similar) data base, i.e. sources of evidence for our data analysis. Our selection explicitly sought variation in other characteristics such as the corporate sponsor's specific problem statement, their industry, and firm structure. Because the intent of this study is to generate a holistic view on requirements elicitation in Design Thinking projects, variation in context was especially desirable. By balancing the principles of similarity and variation, we identified twelve projects from nine organizations in three consecutive years as appropriate cases (Table 4).

| # | Company Name | Country | Industry Sector | Revenue '16 (B. EUR) | # of employees | # of prototypes; type of final solution |
|---|---|---|---|---|---|---|
| A | Alpha | GER | Pharma | ca. 15 | ca. 50'000 | 33; Chatbot |
| B | Beta | FRA | Pharma | ca. 15 | ca. 110'000 | 32; Smartwatch App, CRM Plug-in |
| C | Gamma | GER | Pharma | ca. 15 | ca. 48'000 | 32; Platform, App, hardware device |
| D | Delta | GER | Financial | ca. 33 | ca.142'000 | 28; App |
| E | Epsilon | GER | Conglomerate | ca. 39 | ca. 155.000 | 44; Platform |
| F | Zeta | GER | Information | ca. .06 | ca. 500 | 44; App |
| G | Eta | GER | Aviation | ca. 5 | ca. 21'000 | 36; Platform |
| H | Theta | CH | Financial | ca. 40 | ca. 53'000 | 23; App |
| I | Theta | CH | Financial | ca. 40 | ca. 53'000 | 33; Platform |
| J | Iota | GER | Automotive | ca. 58 | ca. 84'000 | 46; Web App |
| K | Alpha | GER | Pharma | ca. 15 | ca. 50'000 | 55; Platform, Smartwatch App |
| L | Zeta | GER | Information | ca. .06 | ca. 500 | 52; Platform |

**Table 4: Case Details**

**(Source: own illustration)**

### 3.2.1 Data collection

The authors of this paper were lecturers in the university course with numerous years of experience in applying Design Thinking in corporate settings. The data was collected from October 2013 until June 2016 via multiple sources of evidence (SoE) for the purpose of data triangulation, construct validity, and reliability (Yin 2011). We compiled project documentations (SoE1), physical artifacts (SoE2), and participant-observations (SoE3) for each case. First, project documentations include (a) two text documentations about intermediate results and the final prototype (35-140 pages), (b) a booklet showcasing all produced prototypes and corresponding testing feedback (20-50), (c) a video about the final prototype, and (d) pictures of intermediate work results (ca. 300). We gathered SoE1 because our topic is concerned with requirements elicitation in Design Thinking projects and, thus, we see all documentation generated within the project as an indicator for analyzing our research questions, mainly on a content-basis. Second, we gathered all physical and digital artifacts, i.e. prototypes ranging from low to high resolution (32-52). In Design Thinking, elicited needs and requirements are expressed via prototypes, which appoints them a vital role within our research evaluation. Third, participant-observations include (a) memos of weekly review meetings (28), (b) lecture materials (32), and (c) team performance reports at certain project milestones (6). We collected SoE3 to gain process-oriented insights about applying Design Thinking for requirements elicitation.

### 3.2.2 Data analysis

We applied a team-based research approach and used cross-case analysis as the study's analysis strategy (Yin 2011). All researchers were experienced Design Thinking or Requirements Engineering practitioners as well as trained academics. First, we analyzed all cases separately in reference to the dimensions of Table 3, always keeping an open mind to extend the lens of these challenges (Stake 2005). Specifically, we coded relevant passages and derived themes for requirements elicitation. Each document was independently analyzed by two researchers to avoid subjective interpretation and enhance validity of our study (Yin 2011). To demonstrate rigor, two researchers discussed the results and revised the themes together in an iterative approach. For example, we particularly identified themes that addressed user requirements (e.g. "the solution must evoke positive emotions during usage", or "the bot should be usable by voice recognition") or usability requirements (e.g. "the bot should be easy to setup and use"). As a result of the first analysis round, we assigned a conceptual label to each theme (e.g. "user requirements", "usability requirements", "context analysis"). Second, we linked similar themes together into categories (Stake 2005). Thus, we merged workflow requirements, user requirements, and usability requirements into the category "user requirements" and cross-referenced it with the identified elicitation challenges. Third, we used cross-case analysis to derive the core findings of our study. In this phase, a third researcher acted as a sparring partner to shape the final set of findings and to provide quality assurance and objectivity. This also increased the internal validity of our research (Eisenhardt 1989).

## 3.3 Key Findings

We identified seven key findings from the use of Design Thinking for requirements elicitation; each finding is associated with one of the challenge dimensions process guidance (process), stakeholder communication (cognition), and requirements quality (quality). Table 5 highlights which findings supports which elicitation challenge(s) (grey-shaded). It also provides an evaluation to what extent we perceive Design Thinking to support each challenge based on the results of the cross-case analysis and our experience. A high learning potential is marked by (++), inversely, (--) indicates that Design Thinking is not perceived helpful or suffers similar challenges.

| # | Findings derived from case studies | Case | Elicitation Challenges | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Process | | | Cognition | | | Quality | | | |
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| F1 | Design Thinking provides a guideline for requirements elicitation. | all | x | x | x | x | x | x | x | x | | x |
| F2 | Design Thinking explicitly differentiates between user needs and requirements. | all | x | x | x | | | | x | | | |
| F3 | Design Thinking enhances creativity by creating re-framed system contexts in the problem- and solution domain. | B, C, D, E, I, K | x | x | x | x | | | x | | | x |
| F4 | Design Thinking leverages prototyping as a major method to elicit functional and non-functional requirements already in early project phases. | all | | x | x | x | x | x | | x | | x |
| F5 | Design Thinking enhances the priority of eliciting non-functional requirements with a special focus on user experience. | all | | | x | x | | | x | x | | x |
| F6 | In Design Thinking, context is about the human not the system. | all | | x | x | x | x | | x | x | | x |
| F7 | Design Thinking takes a holistic view on requirements elicitation for a domain-agnostic solution. | A-D, G, I, K, L | | x | | | x | | x | x | | x |
| *Impact on individual Requirements Elicitation Challenges* | | | ++ | ++ | ++ | + | ++ | + | ++ | + | -- | + |

**Table 5: Overview of Findings**

**(Source: own illustration)**

We see high learning potential especially for process guidance and stakeholder communication. A rather mixed evaluation needs to be stated for dealing with challenges in the dimension quality of requirements. In general, we could hardly find supporting evidence to suggest positive impact on Challenge 9 (traceability). Similar to existing requirements elicitation techniques, we found that needs and requirements discovered in Design Thinking projects lack traceability despite the process-oriented way of approaching problems. Our memos highlight the confusion of teams when attempting to recap previous insights and requirements: *"Where did we document it? Did we document it all? How did we get there?*" (Case E). We explain this finding with the working attitude of Design Thinking teams, that, similar to agile techniques, prioritizes doing over documenting to quickly move forward. In that sense, we do believe that Design Thinking can learn from the more formalized way of documenting in requirements elicitation.

### 3.3.1 Process-related Findings

*Finding 1 – Design Thinking provides a guideline for requirements elicitation.* All project teams in our study followed the iterative problem-solving approach of the micro- and macro-process as described in section 2.3. While leaving enough freedom for the

use of specific methods, the structure helps to systematically focus on user needs first and only then turn them into testable solutions. A statement of one project member highlights the usefulness: *"Design Thinking provides us with a helping guideline on what to do next, without forcing us to apply a technique that we didn't find useful."* (Case K) Another one adds: *"It is extremely helpful to have a framework that steers you, but that does not overpower you. It encourages to think about content, not the process."* (Case H). We find that the structured, yet flexible, process framework of Design Thinking could help in overcoming the often-mentioned lack of guidance in conducting requirements elicitation (C1). We firmly believe that integrating process steps of the Design Thinking micro- and macro-model into existing practices will lead to a more effective requirements elicitation. A structure will help the novice to more confidently approach unknown situations and the expert to oversee complex problems more easily. We find that, at least in part, this also helps balance capability and experience issues requirements analysts face. In addition, we see the application of a guiding, yet not rigid framework for requirements elicitation positively influencing most of the other challenges as well. For example, the Design Thinking process has institutionalized the application of creativity techniques (C2) and a structured translation from needs into requirements (C3). Furthermore, with every iteration of the Design Thinking process, the team gains deeper knowledge of the problem domain and empathy with the stakeholders and, thus, collects more detailed needs and requirements leading to further adjustments in order to mitigate the risk of not addressing user needs in the long run (C7, C10).

*Finding 2 – Design Thinking explicitly differentiates between user needs and requirements.* In Design Thinking, an explicit distinction is drawn between discovering needs and deriving requirements. User needs relate to people and express their wants, beliefs, and desires. Requirements are expressions of what is demanded to fulfill these needs with an intended solution. Thus, requirements provide an answer or at least the direction towards an answer for meeting the elicited user need. On a process level, this is defined by the interplay of needfinding, focusing on the human (need-driven), and ideation and prototyping, which concentrates on finding a solution (requirement-driven). Several project teams applied methods like Maslow's need hierarchy or the "Common-to-Qualifier-Framework" for structuring needs and deriving requirements (Cases K, H, I). In Case A, for example, the team identified the following need:

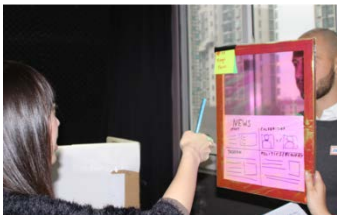
*"Multiple Sclerosis (MS) patients have a great need for an understandable, suitable, and accessible source of MS information."* Based on this need, the team translated and derived several requirements such as: *"The solution must be available in the user's native language,"* or *"Sent content must reach the user in nearly real-time*." Putting our results into the context of established elicitation practices, we see that Design Thinking provides a set of activities and methods to foster the translation of elicited needs into requirements (C3) and to prevent the invention of new requirements (C7) as it establishes an explicit, logical connection between needs and requirements.

*Finding 3 – Design Thinking enhances creativity by creating re-framed system contexts in the problem- and solution domain.* Design Thinking has institutionalized the framing and re-framing of given problems to challenge stereotypical perceptions, encourage learning, and enhance creativity. Case D, for example, visualized their discarded prototypes from early elicitation phases to make failures and learnings explicit and, thus, to inspire continuous experimentation and re-framing. Another team re-framed their given problem statement of *"How does an intelligent collaboration platform in the engineering process for our engineers look like?"* into the vision *"We aim to coordinate knowledge and information in order to enable transparent communication for engineers"* based on their learnings from needfinding and testing (Case E). Eventually, the final prototype looked vastly different from the originally intended platform idea of the project sponsors because the team had created an email-embedded software prototype for knowledge management. During testing users highlighted the ease of use as well as the integration and usefulness in their day-to-day business. Our research suggests learning potential for requirements elicitation practices as Design Thinking mainly addresses process-and method-related (C1-3) and quality challenges (C7, C8, C10). The process of re-framing problems in Design Thinking might be in contrast with the more rational analytic approach prevalent in most elicitation practices. From our point of view, requirements elicitation should be viewed as an explorative dialog with stakeholders and users to obtain questions first to find better answers eventually.

### 3.3.2 Cognition-related Findings

*Finding 4 – Design Thinking leverages prototyping as a major method to elicit functional and non-functional requirements already in early project phases.* The use of (non-technical) prototypes to gain early user feedback is a key component of Design

Thinking. They are used to pursue multiple objectives: (1) they express and conserve the team's elicited user needs, insights, and design decisions; (2) they are used to elicit further needs and requirements through testing; (3) they challenge assumptions and constraints; and (4) they communicate ideas to stakeholders and within the project team. In our study, each case created between 23 and 52 low- to medium-resolution prototypes to express functional requirements in order to determine what should be done and non-functional requirements to examine how the solution should work and look like. Functional requirements were therefore mainly explored through testing form and features of the prototype. Non-functional requirements were mainly discovered through the actual interaction with it. Picture 1-Picture 4 illustrate the variety of prototype forms (from left to right): a low-resolution "Wizard-of-Oz"-prototype tested the interaction with a digital avatar (Case H); a role-play examined an aspired service scenario (Case K); paper-based wireframes evaluated core features of an app (Case L); a user journey assessed daily touchpoints with a banking app (Case I). Meeting memos show a project member's perception, by stating: "It is easier to talk with the help of prototypes and refine ideas to sharpen the overall concept." (Case I) We find that using Design Thinking, which encourages the continuous creation of low-resolution prototypes, can help overcome some of the mentioned challenges in requirements elicitation. Particularly, we acknowledge their potential to enhance creativity of requirements (C2), to translate needs into technical requirements in a more precise way (C3, C10), to ease communication with stakeholders (C4), to expose tacit knowledge (C5), and finally (4) to faster adapt to changing requirements (C8), thereby increasing the effectiveness of requirements elicitation overall.



**Picture 1: Wizard of Oz Prototype**

**Picture 2: Low-resolution Digital Prototype**

**Picture 3: Paper Wireframe**

**Picture 4: Customer Journey**

**(Source: University of St.Gallen, DT@HSG 2012, 2014, 2016)**

### 3.3.3 Quality-related Findings

*Finding 5 – Design Thinking enhances the priority of eliciting non-functional requirements with a special focus on user experience.* Our results suggest that Design Thinking creates an enhanced priority for eliciting user needs and requirements in terms of usability (context of use), workflows (workflows and tasks performed by the user), and user interface (interaction flow or screen layout). However, we also observed a lack of eliciting other non-functional requirements such as security, safety, or performance in our cases. In terms of user experience, Case A distinguishes between functional (given and discovered) and so-called experiential requirements *"to define the emotions that the final concept must evoke in its users,"* i.e. how the solution should eventually feel like. Case G, a B2B project to improve aircraft maintenance by connecting demand and supply on a platform, approached emotional aspects via mood boards to inspire the ease of use and to enhance fast decision-making processes for users in a time- and cost-sensitive high-pressure environment. Moreover, Case D made it its mission to *"make people management a daily experience."* The team identified user experience as one key requirement to guarantee usage on a regular basis. With the help of persona storytelling, they prioritized usability and workflow requirements in every development step to transform *"an improved user experience to an outstanding one."* Relating these findings to existing elicitation practices and challenges, we find that the strong user focus leads to a thorough examination of the user experience, and, thus, to an enhanced priority for eliciting non-functional requirements. In this context, we especially see potential for weakening Challenges 3 (translation) and 7 (invented requirements). While we also need to report the neglect of other (non-functional) requirements, which are essential for later-staged software implementation, we expect higher completeness of requirements (C10) when both methodologies are applied complementarily.

*Finding 6 – In Design Thinking, context is about the human not the system.* Elicitation activities focus on understanding stakeholders and their context regardless of the future solution. This means that they are considered "humans" first and "users" second. Methods to capture context variables are, among many, personas, mood boards, or contextual interviews. In Case A, for example, the team used an empathy suit to mimic limited moving capabilities of an MS patient (Pictures 5 and 6). Thereby the team gained deeper empathy for the daily impairments in their living context first, which was independent from the specific context of the future solution (which was explored at a

later stage). Another team improved their understanding of the main target group, who were doctors, by widening the exploration context to influential parties like hospitals, pharmacies, and patients (Case C). In a similar vein, the team put the focus on the relevant stakeholders of the challenge and created a vision to enable a trustful and efficient exchange between doctors and pharmaceutical companies, stemming from the original problem statement to re-design the go-to-market model for prescription medicine products. We find that this way of approaching problems could contribute to overcoming several elicitation challenges in terms of creativity (C2), communication issues (C4), tacit knowledge of stakeholders (C5), and requirements quality (C7, 8, 10). However, this might also require a profound mindset shift for (some) analysts as the process might feel too lengthy or not focused enough.

**Picture 5: Empathy Suit of an MS Patient I**

**Picture 6: Empathy Suit of an MS Patient II**

**(Source: University of St.Gallen, DT@HSG 2016)**

*Finding 7 – Design Thinking takes a holistic view on requirements elicitation for a domain-agnostic solution.* In general, the application of Design Thinking is not only domain-agnostic in the problem domain (i.e. applicable to a variety of domains) but also in the solution domain (i.e. solutions can be developed via different types of artifacts, only one of them being IS). Subsequently, Design Thinking is able to take a holistic perspective on analyzing the real world, including multiple domains, interacting systems, and stakeholders. We find evidence from several cases that domain knowledge is by far not enough to tackle the given problem statement. All cases show a broad problem exploration with the help of stakeholder mapping, desk research, interviews and observations, benchmarking, and trend research. Notes from a team member in Case A summarize the value of these activities: *"After spending weeks exploring the problem, interviewing and talking to patients and others, I got a much clearer picture of what MS is, the complex problem and dimensions we are dealing with, and, finally, I can see what we need to address their challenges."* We observed that the teams were not constrained by the preconception of software solutions in the early stage. This enabled them to

discover unknown solutions, which, if best for the user, might not have even been digital ones. Overall, we found that most solutions in our cases require an integration of software, service, and/or hardware components to create a compelling user experience. In our case, all solutions combined software and service components and some included a hardware component. The final system of Case C comprised an application for mobile devices, an online platform, and a physical device for doctors' desks. Summing up, we see the potential of Design Thinking in supporting established requirements elicitation practices by looking beyond the system and domain context in order to generate solutions with a comprehensive user experience in a broader service and product environment. We expect a positive influence, mainly for elicitation challenges regarding the lack of creativity, (C2) tacit knowledge extraction (C5), and invented requirements (C7). However, similar to finding 6, this also requires a mindset.

## 3.4 Discussion

The aim of this research is to investigate the learning potential of Design Thinking to boost requirements elicitation in software-intensive projects. Empirical findings on challenges in requirements elicitation practices guided the analysis of 12 cases, which used Design Thinking to elicit requirements. The key findings of this study reveal challenges regarding process guidance, stakeholder communication, and requirements quality. First, Design Thinking offers support for the requirements elicitation process that many researchers criticize to be insufficiently guided (e.g. Browne and Rogich 2001). The guiding framework supports the practical application of methods, which might even compensate for the lack of an analyst's experience. However, as Design Thinking is a team-oriented approach, which strongly relies on the individual expertise of people, limited capabilities can be seen as a critical success factor of Design Thinking as well.

Second, low-resolution prototypes are considered an effective way to ease communication with and expose tacit knowledge of stakeholders and thus, to address cognition- and communication-related challenges that have been raised in various studies (e.g. Zowghi and Coulin 2005).

Third, Design Thinking faces quality challenges, as do current elicitation practices, in terms of the traceability, correctness, completeness, and consistency of requirements. Yet, it emphasizes the discovery of user requirements to derive a comprehensive user

experience. Continuous interaction with stakeholders supports dealing with changing requirements and decreases invented requirements that are not based on user needs. All of this might then lead to generating a solution completely different from what was originally intended. To that effect, Design Thinking can be considered particularly supportive for addressing complex problems as Brown (2008) mentioned. For well-defined problems, i.e. when the real-world problem is known, the application of the entire Design Thinking process might be "over-engineered". Even so, an enhanced mindset for a more human-centered way of requirements elicitation might still prove to be beneficial and "minimizes the risk of delivering a system that does not meet its stakeholders' needs" (Glinz 2014, p.12).

### 3.4.1 Implications for Theory

From a theoretical perspective, this study contributes novel insights to both research on Requirements Engineering and Design Thinking. With regard to research on Requirements Engineering, the understanding of how Design Thinking can support problem solving in Requirements Engineering is advanced. This study answers the calls from various scholars for improving challenges in requirements elicitation (e.g. Zowghi and Coulin 2005; Karlsson et al. 2002; Davey and Parker 2015). Existing literature in this field mostly looked into individual challenges, for example challenges in the context of privacy requirements (Levy and Hadar 2018) or requirements for health care applications (Carroll and Richardson 2016). This study is the first to provide robust empirical evidence about the application of Design Thinking for addressing a comprehensive set of requirements elicitation challenges. Researchers have followed and built on this study to further evaluate challenges in Requirements Engineering in general (Kahan et al. 2019) and elicitation in particular (Martins et al. 2019).

With regard to research on Design Thinking, scholars have been increasingly interested in connecting Design Thinking to software engineering processes and practices (e.g. Lindberg et al. 2011; Corral and Fronza 2018). While related studies have made great strides to develop process models to connect it with SCRUM (e.g. Häger et al. 2015; Przybilla et al. 2018), a structured assessment in the context of requirements elicitation as a crucial step in every development project is still scarce. The findings of the study address this gap in two ways. First, multi-facetted opportunities and benefits of using Design Thinking in early phases of Requirements Engineering are revealed. With

confidence it can be stated that this study contributes to the general understanding around aligning Design Thinking with later staged software development approaches. Second, and more importantly, this study provides evidence suggesting that (methods of) the two approaches might be best applied complementarily in an iterative dialogue to inform and influence each other. Other researchers should feel encouraged to evaluate such an integrated approach.

### 3.4.2 Implications for Practice

The findings can be used as guidelines for companies proposing to adopt Design Thinking and for managers responsible for software-intensive development projects.

A valuable contribution is made for practitioners to avoid aforementioned challenges and gain improved appreciation for eliciting requirements with Design Thinking. Specifically, the findings can act as an aid for using Design Thinking to elicit requirements. Design Thinking provides an approach for eliciting needs and requirements by applying qualitative research methods and producing fast and simple prototypes that converge in innovative solutions. In this regard, Design Thinking not only provides new structures and methods to enhance elicitation efforts but also induces a mind-shift to a more human-centered and explorative way of requirements elicitation.

### 3.4.3 Limitations and Outlook

Despite the careful design of this study, this paper is not without limitations. Biased selectivity should be acknowledged as one limiting factor for document analysis (Yin 2011). Although the analysis was conducted in a near real life setup (real problems provided by real corporate sponsors), it can be argued that the sources of evidence are closely aligned with the specific Design Thinking approach of the University, including its particular assessment policies and Design Thinking philosophy. In addition, all project teams were composed of students being novices in Design Thinking.

Therefore, scholars should investigate project teams in actual organizational settings as organizational factors like resource allocation, organizational stability, or organizational implementation of Design Thinking can be expected to influence the activities of requirements elicitation. Further research should build on our results by deriving concrete propositions from each key finding and quantify their effects. As this study has mainly focused on discussing the *what* of each finding, future research should also

consider the concrete *how*. The distinction of Design Thinking as a process, toolbox, and mindset (Brenner et al. 2014) provides a differentiated guideline to follow.

In general, the access to the vast amount of confidential and detailed data in this study should outweigh its limitations because it led to a first profound understanding of the role of Design Thinking for requirements elicitation.

## 3.5 Conclusion of the Chapter

Effective requirements elicitation is perceived to be one of the most crucial activities in software-intensive development projects. While many scholars and practitioners have pointed out and agreed upon its numerous challenges, others consider the increasingly popular approach of Design Thinking to be the promising 'cure.' This paper provides robust empirical evidence of the role of Design Thinking for requirements elicitation and helping overcome its challenges. Specifically, this paper presents findings in terms of process guidance, stakeholder communication, and requirements quality based on insights from a multiple-case study. In summary, Design Thinking introduces new structures and methods to enhance elicitation efforts and inspires a mind-shift towards a more human-centered and creative way of requirements elicitation. This work provides a better understanding of the multi-faceted potential of Design Thinking for Requirements Engineering in general and requirements elicitation in particular, for both scholars and practitioners.

# 4 Challenges when Specifying Requirements with Design Thinking

Together with chapter 3, this chapter addresses the first research question of the dissertation and is concerned with understanding the challenges to specify requirements with Design Thinking. The chapter presents the results of a multiple case study[3] that investigated such challenges. Section 4.1 provides the motivation of the multiple-case study in more detail. Section 4.2 explains the methodology of the study and offers a description of the data collection and analysis. Section 4.3 reveals the results of the study. Section 4.4 discusses the implications for both theory and practice and points at the study's limitations and an outlook for future research. Section 4.5 concludes the chapter.

## 4.1 The Need to study Challenges in Design Thinking

Many digital and non-digital artifacts evolve during a Design Thinking project. They express the identified user needs, insights, learnings, and design decisions. For time-saving reasons, the project teams capture and specify the mentioned artifacts in a mainly unstructured form, e.g. through post-its, pictures, and low-fidelity prototypes during the process and text documentations on the occasion of specific milestones. This predominantly informal and unstructured mode of specifying requirements can be critical for two reasons, which are strongly interlinked: First, it can hinder the team's ability to reflect and build on intermediate outputs and make decisions for further activities. Thus, it might result in a less desired prototype and, eventually, in a less desired product on market. Second, it also decreases the quality of the outcome when the final prototype is insufficient in explaining (1) certain design decisions made for the prototype and (2) the contextual details, in particular, when handed over to other teams for implementation (often software engineers) that are not able to revisit activities of the Design Thinking team. The lack of information might then lead to the creation of a less desirable product not fulfilling all relevant needs (Beyhl et al. 2014; Beyhl and Giese 2015, 2016). As a consequence, the innovative vision might not be able to develop its full potential because its realization does not conform to the elicited requirements of the Design Thinking team.

[3] Parts of this chapter have been published in: Hehn, J., Uebernickel, F., Stöckli, E., Brenner, W. 2018. "Towards Designing Human-Centered Information Systems: Challenges in Specifying Requirements in Design Thinking Projects," Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2018). Lüneburg, Germany: AIS.

This study focuses on understanding the challenges of Design Thinking teams concerned with specifying requirements. It does not address the specific handover challenges from Design Thinking teams to software engineers, which have been discussed in prior research (Beyhl et al. 2014; Beyhl and Giese 2015, 2016, Häger et al. 2015), but rather focuses on the challenges within Design Thinking. The handover challenge can then (partially) be a result of the aforementioned one. Our research is based on a multi-case study and analyzes the data of five project teams. They approach a problem statement given by a corporate sponsor with the use of Design Thinking in a university context. We investigate the activities of specifying requirements to create solutions and the challenges that the teams face in doing so within the Design Thinking project setup. In this context, requirements are understood as "the descriptions of services that a software system must provide and the constraints under which it must operate." (CS Software Engineering) Requirements can be seen on different abstraction levels, from very high-level statements of service descriptions to detailed specifications of certain functions or algorithms. There is a common understanding to differentiate requirements according to their nature. User requirements reflect the needs, desires, and stated functionalities of a software system. System requirements express services needed for the system to operate successfully from a technological view. And lastly, software requirements specify the architecture of the software system in detail. On a more granular level, the Requirements Engineering community demands requirements that are unambiguous, correct, complete, concise, feasible, understandable, and consistent (Bahill and Madni 2017).

## 4.2 Research Methodology

Our research adopts a multiple-case study approach according to the guidelines of Yin (2011) to address the exploration of the research question: What are the challenges when specifying requirements within Design Thinking projects? The cases derive from a one-year university course where teams take on a design challenge provided by a corporate sponsor and guided by a Design Thinking framework (see Figure 3) as part of the course. All corporate sponsors were either long-standing partners of the university or undertook a comprehensive onboarding process into the Design Thinking program to build up trust with the teaching team. We chose a multiple-case study approach as the cases we present are mainly replications of each other. This way, we expect better generalizable and more robust results for our findings (Eisenhardt 1989).

### 4.2.1 Data Collection

We analyzed five cases referring to five Design Thinking project teams (Table 6). We selected five out of nine projects following the criterion that all the projects had to address digital challenges. All teams consisted of business master students and, depending on the corporate challenge, included students from design, business informatics, and engineering. The students had no or little prior experience in applying the presented Design Thinking framework. However, they shared the experience of the same course content as a common background. To evaluate our research question, we collected multiple sources of evidence consisting of (1) project documentations, (2) physical artifacts, and (3) participant observations (see Table 7).

| Case | A | B | C | D | E |
|---|---|---|---|---|---|
| Team size<br>Corporate Sponsor<br>Type of Industry<br># of employees<br># of prototypes<br>Final prototype | 6<br>Alpha, GER<br>Pharma<br>~ 50'000<br>33<br>Chatbot | 3<br>Beta, FRA<br>Pharma<br>~ 110'000<br>32<br>Smartwatch App and CRM Plug-in | 7<br>Gamma, GER<br>Pharma<br>~ 48'000<br>32<br>Platform and Hardware | 3<br>Delta, GER<br>Financial<br>~ 142'000<br>28<br>Smartphone App | 7<br>Epsilon, COL<br>Financial<br>~ 50'000<br>22<br>Service, Smart-watch, and App |

**Table 6: Overview Cases**

**(Source: own illustration)**

### 4.2.2 Data Analysis

We examined project documentations (SoE1) with the help of document analysis which is the "systematic procedure for reviewing or evaluating documents – both printed and electronic." (Bowen 2009, p. 27) We chose this approach because our topic is concerned with specifying requirements and we see all documentation produced within the project as an indicator for evaluating our case question. Our available data has a broad coverage over the entire project time in multiple settings and scenarios, which makes analyzing documents advantageous for our purpose (Yin 2014). We reviewed 45 documentations and more than 1000 pictures across all cases. We especially looked at the following questions: Which requirements are specified and how? How understandable and complete are they? Are context requirements captured? In addition, we leveraged prototypes as a complementary data source (SoE2). In Design Thinking, prototypes are expressions of elicited requirements and design decisions, which makes them a vital part in the evaluation within our research. To this purpose, we evaluated how and which

requirements were expressed as prototypes. Third, we leveraged participant observations to gain insights about possible challenges (SoE3). The data sources we used were mainly notes and observations from review meetings about process struggles, team dynamics, and content discussions. Three of the authors were lecturers and coaches in this course drawing from multiple years of experience in applying Design Thinking in university-level projects and corporate settings.

| Name | Type | Short Description | Number | Audience |
|---|---|---|---|---|
| **Source of Evidence 1 (SoE1):** Text and Visual Documentations | | | | |
| One-page Summary ("One Pager") | Text and pictures | Short description of project status after each milestone | 6 per case | Corporate Liaison Teaching Team |
| Intermediate Documentation I | Text and pictures | Insights and learnings from first two milestones (35-120 pages) | 1 per case | |
| Intermediate Documentation II | Text and pictures | Learnings from Darkhorse and Funky phases in the form of a booklet | 1 per case | |
| Final Documentation | Text and pictures | Guidelines to understand the final prototype (40-140 pages) | 1 per case | |
| Pictures | Photos | Photos of prototypes and other outputs | >1000 (all) | |
| Videos | Video | Video shows the final prototype | 1 per case | |
| **Source of Evidence 2 (SoE2):** Physical Artifacts | | | | |
| Prototypes | Physical and digital | Low- to high-resolution representations of ideas | 147 (all cases) | Corporate Liaison |
| **Source of Evidence 3 (SoE3):** Participant Observations | | | | |
| Weekly Review Meetings | Lecturer Notes | Weekly one-hour review with each team on progress and content | 28 per case | Design Thinking Teams |
| Weekly Lectures | Lecturer Notes | Weekly 2-3 hour lectures | 32 (all cases) | |
| Regular Performance Reports | One page from lecturer | Assessment of each team with an evaluation of their performance | 6 per case | |

**Table 7: Sources of Evidence**

**(Source: own illustration)**

We chose to draw on multiple SoE with the aim of data triangulation to support construct validity for our findings (Yin 2014). Subsequently, we followed a structured procedure for each SoE. We applied content analysis to identify findings and used cross-case synthesis to retain the challenges to specify requirements. In particular, two of the authors derived the challenges independently from each other by examining the documents. They identified relevant passages and deduced the overarching themes in the topic of requirements specification. Then, the two engaged in discussions about the challenges to iterate and merge their first set of challenges. The third author acted as a sparring partner in the process of shaping the final set of challenges to provide quality assurance and objectivity. Figure 4 visualizes the above described procedure and provides the transition to the next chapter that will conclude the observed practices and challenges in capturing requirements.

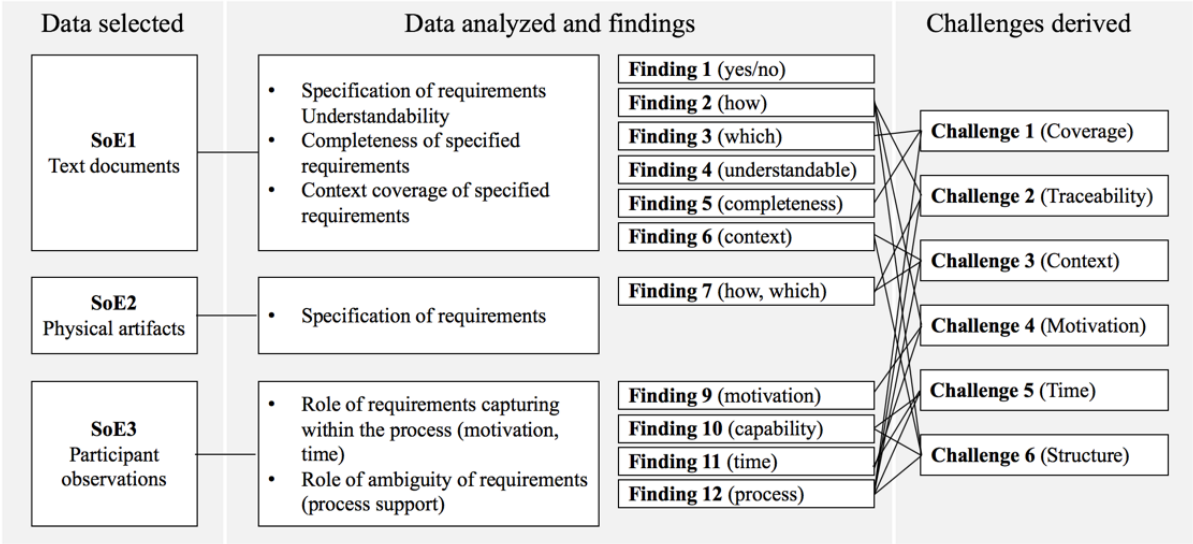



**Figure 4: Convergence of Multiple Sources of Evidence**
**(Adapted from Yin 2014, p. 117)**

## 4.3 Key Challenges

In all cases the teams captured and specified requirements throughout the course. Each team used different ways of specification all of which were generally understandable in form and expression. An explicit listing of requirements can be found in Case A and D, e.g. the final documentation of A states: “Requirements are derived from Alpha’s initial brief, legal constraints in the industry, user interviews, Alpha staff interviews, and various kinds of testing.” The requirements were structured into functional, experiential, and physical requirements. Each specification was further divided into (1) requirement,

(2) metric, and (3) rationale as the example shows: (1) *"The solution must be available in the user's native language,"* (2) *"The solution should be translated to the native language of Alpha's main target markets,"* and (3) *"Our user tests showed us that patients have immense difficulty using a service if it's not offered in the language they're used to."* Furthermore, in Case D pictures of the prototype were used to show functionality and user requirements. The specification structure looks as follows: (1) issue description, (2) user quote, (3) need, (4) requirements derived. The other cases (B, C, E) use expressions like *"features"*, *"functions"*, *"experiences"* and *"implications"* to address the topic of requirements. In addition, prototypes served as a useful way to express requirements. A prototype can take the form of a visual documentation and, thus, should be able to make them explicit in a concise and understandable manner. Not all prototypes were self-explanatory though, and several needed further explanations to comprehend their purpose.

### 4.3.1 Output-Related Challenges

*Challenge 1 (Coverage): Strong focus on user requirements while neglecting software and system requirements.* The better and clearer the requirements, the easier it is to implement them for the intended purpose. During our analysis, we found a strong focus on specifying user requirements, which was one of our questions for SoE1 and SoE2 ("Which requirements are specified?"). Across all cases we see that at least 80% of requirements address users, their needs, insights, and learnings from prototype testing. This finding is supported by notes from review meetings where discussions mainly revolved around user needs and how to transfer them into solutions. For example, a protocol of Case C includes the question: *"How can we transfer the need for trustworthy information sources into a product feature? I don't know how to do that in the best way."* Cases C and E address user requirements only, while documentations of Cases A, B, and D also show implications for software requirements, mainly in the appendix of such documentations. These make up the other 20% of the requirements documentation. The specification of system requirements is neglected completely. This finding is not surprising as the paradigm of human-centric design is the foundation of all activities in design thinking, putting the (potential) user in the role as a recurring sparring-partner for prototype testing. Still, feasibility and viability should be considered as well (Brown 2008). The lens of feasibility demands an exploration of organizational capabilities and technological options to translate the human-centric requirement into

actual products and services. We see potential for Design Thinking to increase the focus on this topic when evaluating prototype options for next steps in the process.

*Challenge 2 (Traceability): Weak links between needs, insights, learnings, and requirements.* Being able to trace requirements back to insights, needs, and learnings from testing helps to revisit previous decisions and to better understand the intended purpose of a solution. All sources of evidence (SoE1-SoE3) point out that achieving traceability seems to be a challenge in design thinking projects. Across all cases, the final documentations showed the most specific and traceable connection between need and (iterated) requirement, however, not always comprehensibly. We want to highlight the following evaluation notes from one lecturer with regard to the final documentation of Case E: *"The final prototype comes a bit out of the blue. [...] Which needs are addressed by which feature of the prototype is unclear."* Notes from review meetings show similar uncertainties when asking: *"Where does this idea come from?"*, *"What is the need?"* (Case B, C) A good example provides Case A, which matches detailed requirements descriptions of a step by step journey of the user going through all screens of the prototype. Links to previous prototypes show the rationale for changing requirements. Case C provides another good example, where the team visualizes the evolution from intermediate to final prototype for a better traceability between need and solution (Figure 5). The team adds descriptions to explain which elements are transferred further and deducts implications for the final prototype. Traceability from final documentation to intermediate project results appears to be an important issue with varying results. We have seen good examples in our cases that can serve as benchmarks in the future. Nevertheless, notes from review meetings of the project have shown the teams' struggle and confusion in the attempt to recap and build on previous findings and specified requirements. We will pick this up again when discussing our observations of Challenge 6.

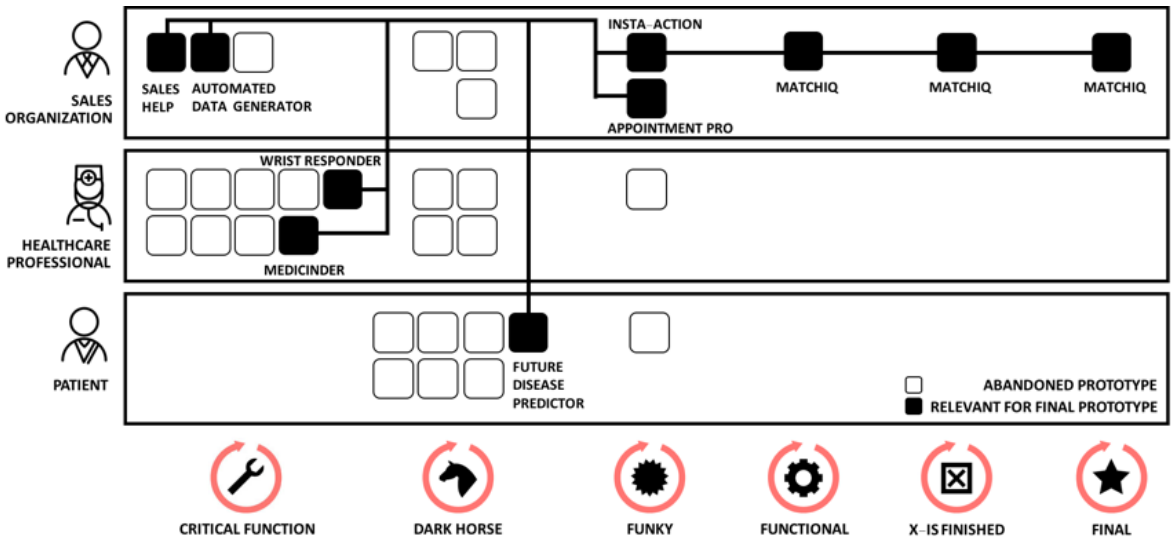



**Figure 5: Evolution from Intermediate Prototypes to a Final Prototype (Adapted from final documentation of Case C)**

*Challenge 3 (Context): Lack of formalizing context requirements.* Contextual information and requirements help to create a holistic view of the entire system's landscape (e.g. in terms of constraints or system goals). Design Thinking leverages a 360-degree approach (by multiple perspectives) to address the problem and solution space of the design challenge and, thus, aims to generate an integrated vision of the challenge and its context. Cases A and E used personas and empathy maps to set the user's context of the solution, e.g. in Case B an element of the platform solution is a status function for doctors in which, similar to a frequent flyer program, they get points that translate into benefits. To put this into context, the team has specified the reason: *"We chose this solution according to the preferences of two target countries, where, unlike in Switzerland, the visit of sales representatives is appreciated as personal and individual attention of company Beta. Thus, novel digitization efforts are harder to argue. This is why we wanted to incentivize to use our platform."* While Design Thinking is virtually predestined to elicit and specify context requirements, we can determine a lack of formalized context requirements in all cases. A consistent way in specifying these requirements according to the context of the three dimensions of desirability, viability, and feasibility is lacking. This missing methodological support causes a lack of information in later implementation stages. Again, this finding can be related to the structural discussions in Challenge 6.

### 4.3.2 Process-Related Challenges

*Challenge 4 (Motivation): Lack of motivation to specify requirements systematically.* Although discovering needs and eliciting requirements is part of the Design Thinking process, documenting them in a systematic manner is found to be challenging. The deeply explorative approach of Design Thinking demands a multitude of sources and tools to create a multitude of artifacts, not yet knowing which ones will be the most relevant in later stages. But relying on implicit knowledge in a sea of analog and digital artifacts may lead to the risk of losing information. In general, the teams in our cases saw less value in specifying the requirements for themselves than for the teaching team and their corporate sponsor. The main motivational factor was the fact that a part of the grading in the course was based on the quality of the prototypes and their written deliverables. In addition, they also mentioned a feeling of obligation towards their corporate sponsor to specify findings for their later usage (A, B, C). To tackle this issue, three teams assigned a specific documentation role within their team. The teams that did not do this (B, E) showed particular weaknesses in the completeness and understandability of requirements specifications.

*Challenge 5 (Time): Lack of time to specify requirements systematically.* The element of available time is a crucial factor in all Design Thinking projects. Evidence of challenges with regard to time is mainly derived from weekly review meetings with the teams. One team member, for example, stated that *"the Design Thinking project is a marathon in the speed of a sprint."* According to sources from review meetings (A-E), team members stated that they were aware that someone should document continuously but failed to do so due to a permanent lack of time. The teams reported to prioritize time for *"real work like conducting user research or building the next prototype"* over specifying requirements in a systematic manner. This triggers a similar observation to Challenge 4 (motivation) where specifying requirements is seen as a timely activity that gets de-prioritized under time pressure for *"more important things."* However, an indication that a more systematic approach could be beneficial, shows the following reflection statement from Case A: *"The fact that we had eight whole months to burn, often allowed us to slip into sluggishness, which really showed in our lackluster results in the winter period, and I feel that our indecisiveness cost us the chance to develop a proof of concept that actually validated ALL, not just most, of our claims and theories about the potential of chatbots in patient support."*

*Challenge 6 (Structure): Knowledge is implicit or captured on Post-its while adequate tool support is missing.* In Design Thinking projects, there is no pre-defined structure on how, when, and which requirements to specify. Hands-on activities like building, testing, and experimenting are the preferred choices to gain insights and feedback from users. In all cases, observed practices for specifying requirements take a multitude of forms: working artifacts are mainly post-its, sketches, pictures, low resolution prototypes, digital shared notes (e.g. on Trello), and a loose collection of word documents in a team share storage. Official documentation purposes are usually addressed by natural language in text form, a high-fidelity prototype, and a final video to provide context. Creating prototypes is an evolving process and a way of "thinking with hands," where the team can reflect on an idea. However, decisions on changes are often not made explicit. This supports the finding that much knowledge stays implicit. Frameworks like personas, service blueprints, and empathy maps are commonly used to synthesize data from field research. However, adequate tools for deriving and specifying requirements are missing. The "informal world of Design Thinkers" (Beyhl et al. 2014, p. 167) (and, in the long-term, also the more formal world of software engineers) might benefit from a non-intrusive tool that fits to the "rough and dirty," explorative working mode of Design Thinkers. The combination of a prototype, showing links to previous prototypes and findings together with natural language specifications of requirements could serve as a first idea into this direction.

## 4.4 Discussion

Prior literature has acknowledged the challenges when specifying requirements with Design Thinking (Beyhl et al. 2014; Beyhl and Giese 2015, 2016). This paper advances the understanding of this problem with data from a multi-case study of five Design Thinking projects. The results of this study show that some of the challenges can be considered as generic problems that also occur in other project related teamwork (especially Challenges 4 and 5). The other challenges, however, suggest a strong association with the specific Design Thinking methodology. In this context, we find Challenge 1 as very IS-specific, whereas the other challenges (2, 3, 6) could also be transferred to other project setups that don't deal with creating and designing IS. Since this study set out to create a holistic picture of the challenges to specify requirements in Design Thinking projects to create human-centric IS, all of the findings can be considered relevant for scholars and practitioners alike.

### 4.4.1 Implications for Theory

The findings contribute to the body of knowledge for Design Thinking in the context of IS development in the following ways. First, this study provides empirical evidence about the challenges to specify requirements with Design Thinking. The paper builds on previous work of Beyhl and Giese (2016) and Häger et al. (2015), who raised documentation or knowledge transfer problems in Design Thinking, and expands these concerns with a more comprehensive set of empirically derived challenges. Several authors call for a sustainable integration of Design Thinking into (early) software development practices (e.g. Przybilla et al. 2018; Vetterli et al. 2013). To achieve this, such a holistic view on the diverse challenges needs to be done.

Second, Yoo suggests that "IS scholars can help Design Thinking practice by inventing new constructs, models, methods, and instantiations." (2017, p. v) This research points to a specific area of future research, i.e. to enhance existing or creating new methods and tools for specifying requirements in such explorative approaches as Design Thinking. The informal way of information gathering can be disadvantageous. A lot of knowledge stays implicit and the decisions by the Design Thinking team are often hard to recap by other parties, e.g. software engineers in the quest of implementing the envisioned prototype (Häger et al. 2015). Methodological support from the more mature discipline of Requirements Engineering, also in terms of classifications and documentation practices, is one promising direction for further research.

### 4.4.2 Implications for Practice

Three main practical implications from our findings can be derived for professionals. Avoiding, or at least being aware of, the aforementioned challenges can be important for successfully applying Design Thinking.

The findings regarding output-related challenges provide an overview of what can be expected when (only) using Design Thinking to specify requirements. The tendency to exclusively elicit and specify user requirements (while neglecting software and system requirements) indicates a specific scope for a suitable application of Design Thinking, i.e. project settings with a strong user-orientation. Thus, project managers in software-intensive development projects receive guidance when to use Design Thinking best, when not to use it all, and what should be done complementarily to Design Thinking. Some of the introduced challenges offer opportunities for common Requirements

Engineering practices and classification techniques to define a comprehensive set of requirements. To reach out to Requirements Engineers to learn from might be a valuable suggestion for Design Thinking teams.

The findings regarding process-related challenges support project managers in balancing resources, time, and activities in software-intensive development projects. The advantages to learn quickly in the early phases with the rapid working style of Design Thinking should be balanced wisely with the demand for more structure and documentation in the later stages connecting to system design and implementation.

In addition, just being aware of all aforementioned challenges, might already help the project team and manager to foresee potential hurdles when using Design Thinking, which, in turn, might result in a more diligent way of working and/or avoiding some of them.

### 4.4.3 Limitations and Outlook

The paper is not without limitations. Yin (2014) stresses biased selectivity as one flaw of analyzing documents for research. Although it can be considered advantageous that our cases were conducted in the same setting, it can also be argued that the available documents are strongly aligned with the university's Design Thinking approach and its specific assessment policies for the course. In addition, the specific team staffing (students from different backgrounds and universities), the form of the corporate briefing, and the teaching team's coaching support should be considered as influencing factors on the derived challenges when transferring our findings to other (corporate) settings. Nevertheless, the entire course setup is crafted in a way that creates conditions that are as "real" as possible. Taking Sun and Kantor's (2016) "three realities"-paradigm (real users, real problems, and real systems) for a naturalistic evaluation in IS projects as a reference, the course does address real problems from corporations (and not fictional ones) and applies real systems (i.e. the Design Thinking methodology with conducting real field research and user testing to create real solutions). "Real users" can be seen as debatable, since the project teams are composed of students who can be seen more as external consultants than internal project members of an organization. The latter ones for instance, often face internal politics or stakeholder management issues that we believe influencing Design Thinking projects in "real corporate settings" to a great degree. Therefore, scholars should feel encouraged to analyze teams in actual

organizational settings, which allows the identification of organization specific challenges. Overall, the access to confidential data in the study should outweigh the limitation because it led to a deeper understanding of the output- and process-related challenges.

## 4.5 Conclusion of Chapter

Design Thinking provides a guiding yet not rigid framework for exploring and specifying requirements in the context of software-intensive development projects. Nevertheless, our research revealed six challenges for specifying requirements, three of them related to the output of Design Thinking: (1) Design Thinking teams focus on the specification of user requirements, while software and systems requirements are widely neglected; (2) the central artifacts of design thinking projects are insufficiently linked with each other and, therefore, traceability is limited or even impossible for further use; and (3) the lack of formalized context requirements is limiting the use of project results in later implementation stages. Furthermore, we discovered three challenges relating to the process of Design Thinking itself: (4) Team members are hardly motivated to specify requirements systematically, (5) perceive the specification of requirements as an extra effort that shortens the already limited work time, and (6) lack adequate tool support for specifying and structuring (implicit) team knowledge. The purpose of this study was to lay the groundwork for creating solutions to these challenges.

# 5 A Combined Artifact-based Reference Model

This chapter addresses the second research question of the dissertation. Based on the previous insights it discusses a model that combines the artifacts of Design Thinking and Requirements Engineering. The chapter presents the results of a study inspired by the design science research paradigm[4] to develop the combined artifact model. Section 5.1 explains the motivation for an artifact-oriented model in the context of this dissertation. Section 5.2 describes the development of the model. Section 5.3 introduces the combined artifact model including a detailed description of the model and the individual artifact types. Section 5.4 analyzes the results and illustrates the implications for researchers and practitioners. Finally, section 5.5 concludes the chapter with a summary of the main findings.

For better readability the terms 'Design Thinking artifact' and 'Requirements Engineering artifact' are abbreviated into 'DT artifact' and 'RE artifact'.

## 5.1 The Need for an Artifact-based Reference Model

Empirical evidence indicates that a combination of Design Thinking and Requirements Engineering appears to be beneficial (see chapters 3 and 4). However, little is still known about how to holistically integrate Design Thinking into Requirements Engineering (Vetterli et al. 2013; Kahan et al. 2019). One difficulty to be taken into account is that there are different views on Design Thinking, ranging from a set of methods, tools, and principles (Brenner et al. 2014). Requirements Engineering in turn is treated as a holistic engineering discipline encompassing various approaches, principles, tools, and even more methods – all to be selected and adopted in dependency to the project situations faced and the software process models chosen (Nuseibeh and Easterbrook 2000). A better understanding of how Design Thinking can be properly integrated into such an environment (and when making use of it makes sense) is essential to leverage the full potential of Design Thinking in such a context. A classification is necessary that allows distinguishing the different facets of both approaches and that explains the occurring combination opportunities.

[4] Parts of this chapter have been published as a preprint and are accepted for publication in: Hehn, J., Mendez, D., Uebernickel, F., Brenner, W., Broy, M. 2020. "On Integrating Design Thinking for a Human-centered Requirements Engineering," IEEE Software, Special Issue Design Thinking.

An artifact-oriented reference model that integrates Design Thinking into a holistic engineering context seems appropriate to accommodate the variety of processes and methods of both approaches. Artifacts determine *what* has to be accomplished (the work products and their interdependencies) instead of *how* it has to be accomplished (the steps that have to be taken) (Berenbach et al. 2009). An artifact is defined by its content, structure, and its notation (syntax and semantics) (Méndez Fernández and Penzenstadler 2014a). Defining a comprehensive view of the "desired" system and its key functionalities and features is an important objective of both Requirements Engineering and Design Thinking. Their produced artifacts are used to support product design and project management decisions throughout the development process and product life cycle (Berenbach et al. 2009). The quality and appropriateness of these artifacts is a key factor for a successful software-intensive system. A model that encompasses the relevant artifacts of Design Thinking and Requirements Engineering can outline the synergies and differences between both approaches. While keeping a consistent structure and terminology, this condensed view focuses on the created work products, their contents, and dependencies and allows to abstract from their particularities of various processes and methods, which would otherwise render a comparison difficult.

The artifact model is supposed to act as a reference model according to Winter and Schelp (2006). Reference models are well-known in IS and Requirements Engineering research. Such a model is a means to capture a conceptual blueprint that can be used as a draft for system design and development (Fettke and Loos 2007). It contains and structures all the artifacts referenced, modified, or created in Requirements Engineering and Design Thinking in software-intensive development projects. In order to be useful, the model should support the re-use of knowledge and should be adaptable to certain situations in an efficient manner (Becker et al. 2007). The aim is to integrate DT and RE artifacts in order to simplify the adaption and configuration (i.e. usage schemes) of Design Thinking for Requirements Engineering.

This study contributes an artifact model, which emerged from several iterations with practitioners and academics. The goal is to establish a reference model that should (1) support the integration of both approaches, Design Thinking and Requirements Engineering, respecting their different "flavors", (2) provide flexibility in the way of working to cope with the various influences in individual project environments and for

organizational needs, and (3) enable a reproducible creation of work products in the context of innovative software-intensive development projects.

## 5.2 Research Methodology

### 5.2.1 Research Design

The development and design of the artifact-based reference model for Design Thinking and Requirements Engineering follows the process as suggested for empirically grounded reference modeling based on Ahlemann and Gastl (2007). The authors have examined a variety of process models and synthesized them to a core basic structure (e.g. Schütte 1998; Fettke & Loos 2004). Figure 6 provides an overview of the five development phases that have been undertaken for the development of the combined artifact model. In analogy with systems engineering and the principles of the Design Science Research (DSR) paradigm, the overall design process is based on a cyclic structure to allow for iterations via feedback loops. At its core, DSR is a well-established research approach that is concerned with the rigorous construction of useful IS artifacts, i.e. constructs, models, methods, or instantiations (Hevner et al. 2004; Peffers et al. 2007).

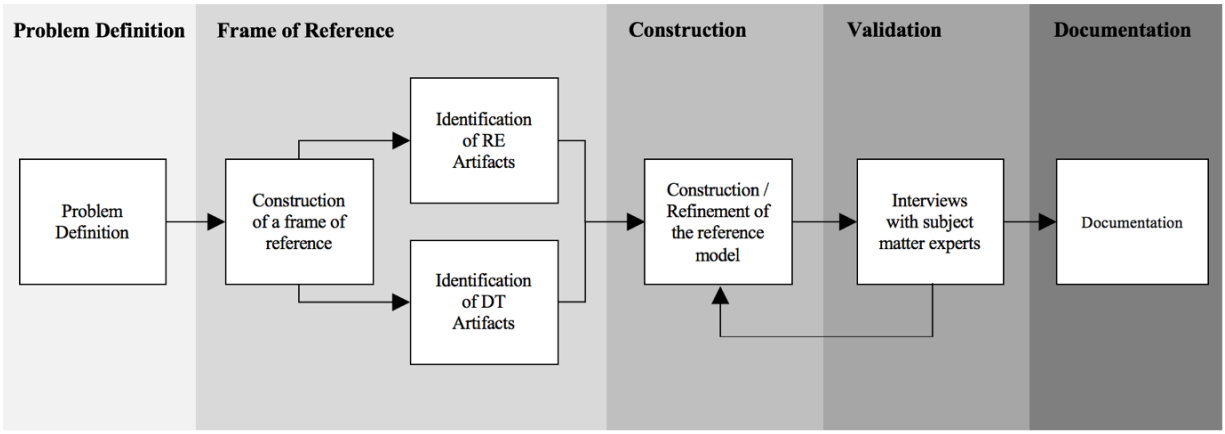



**Figure 6: Development Steps for the Combined Artifact-based Reference Model (Adapted from Ahlemann and Gastl 2007)**

For the purpose of this study, the development process consists of the following five phases (Ahlemann and Gastl 2007):

(1) *Problem Definition*: As outlined in the previous section, a systematic comparison of the concepts of Design Thinking and Requirements Engineering is still missing despite first evidence regarding complementary benefits. This makes the

definition of appropriate usage schemes of Design Thinking for Requirements Engineering inherently difficult. This is both supported by literature (e.g., Vetterli et al. 2013; Kahan et al. 2019) as well as by empirical studies in the context of this dissertation (chapters 3 and 4). Therefore, the development of a combined artifact-based reference model to be applied in innovative software-intensive development projects is regarded as a promising means to solve the problem.

(2) *Construction of a frame of reference*: In the context of this study the frame of reference is defined by an existing and evaluated artifact model for domain-independent Requirements Engineering (also called AMDiRE) (Méndez Fernández and Penzenstadler 2014a). This model has been empirically validated in various academic-industry collaborations over six years of research. Moreover, relevant RE- and DT-related artifacts were identified to assure a comprehensive set of elements in the reference model as Schlagheck (2000) suggests. For a detailed description about this phase see the following sections 5.2.2 and 5.2.3.

(3) *Core Construction*: The initial combined artifact-based model is constructed by mapping the artifacts for Design Thinking and Requirements Engineering. The definition of complementary or overlapping artifacts provides a first version of the model. The construction is based on an exchange between Requirements Engineering and Design Thinking experts as well as on existing literature and research results. For a detailed description about this phase see section 5.2.4.

(4) *Validation*: The reference model is refined with regard to consistency, completeness, and comprehensibility based on feedback from practitioners and academics. The same researchers, who participated in the core construction phase, were involved. For a detailed description about this phase see section 5.2.4. For a detailed description of the resulting artifact model refer to section 5.3.

(5) *Documentation*: The documentation of the research results is done in this chapter of the dissertation. As documentation is a prerequisite for inter-subjective comprehension and validity, it follows the suggested structure of (1) description of the construction process (sections 5.2.1 to 5.2.4), (2) clarification and

description of the model elements with annotations that include theoretical and empirical references (sections 5.3 and 5.4), and (3) documentation of case studies (chapters 6 and 7).

### 5.2.2 Identification of RE Artifacts

The steadily growing interest in artifact-oriented Requirements Engineering approaches (e.g. Méndez Fernández and Penzenstadler 2014a) and the development of respective artifact-based models (e.g. Silva et al. 2009; Friedrich et al. 2008) has been discussed in section 2.2.3. Based on the evaluation of existing research and literature, an evaluated and domain-independent Requirements Engineering artifact model (AMDiRE) was selected as the frame of reference for further modeling activities in the context of this dissertation (Méndez Fernández and Penzenstadler 2014a). AMDiRE is built up of three basic components: (1) the artifact model (in form of a content and structure model), (2) a role model, and (3) a process model. The structure model (Figure 7) provides a logical positioning of the single artifacts in a simplified manner and acts as a connection to the role and process model (Figure 8).

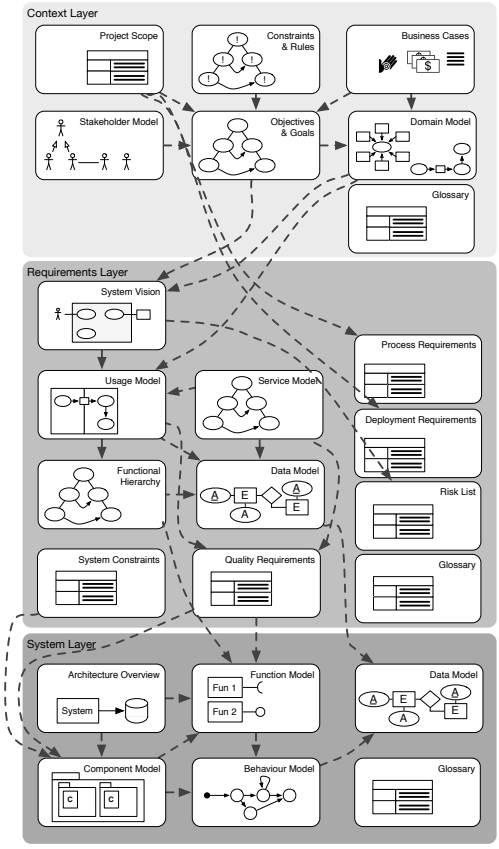



**Figure 7: AMDiRE Model**

**(Méndez Fernández and Penzenstadler 2014a, p. 20)**

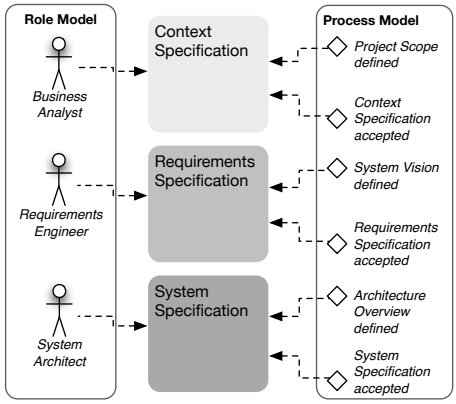



**Figure 8: Overview of Artifact Types, Roles, and Milestones in AMDiRE (Méndez Fernández and Penzenstadler 2014a, p. 15)**

The structure model is used as the reference basis for comparing and integrating DT artifacts, which is why the structure model is explained in detail in the following. The model determines three essential artifact types, where each captures a collection of different artifacts. First, the context layer shows the information relevant for defining the context of the system under consideration, including the overall project scope, stakeholder information, domain model, and underlying goals and constraints. Second, the requirements layer captures the information relevant for defining (user) requirements by taking a black-box view on the system, i.e. primarily requirements from an end users' perspective without constraining, where possible, internal realization concepts of a system. Third, the system layer comprises a glass-box view on internal system realization concepts, including a logical component architecture and a specification of the desired behavior, e.g. via function models. The context and requirements layer include those aspects typically found in RE-related artifacts, while the system layer includes those aspects that address what is known as the solution space and provide the interface between Requirements Engineering and design activities (Méndez Fernández and Penzenstadler 2014a).

The model specifies content items in the problem space and solution space as well as their interdependencies. The problem space produces the document types of context and requirements specification as two main coarse-grained artifacts. The solution space produces the system specification (including domain specific and independent content).

The artifact model of AMDiRE encompasses 24 artifact types, where the 'glossary' and 'data model' are replicated in the model. Each element of the model is described in detail in section 5.3.3.

AMDiRE can be considered particularly suitable as a reference model because it provides a domain-agnostic view on the relevant artifacts in Requirements Engineering. We take the same view for identifying the DT artifacts and developing the combined artifact-based model. In addition, AMDiRE provides a robust basis as it synthesizes several artifact models and evaluates them in different industrial case studies in various socio-economic contexts. According to the authors, the results have been incorporated into the daily Requirements Engineering practices of the companies with whom they have collaborated (Méndez Fernández and Penzenstadler 2014a). This is also the goal of the combined artifact-based reference model – to provide usefulness in practice. In addition, the connection to relevant roles and processes, which can be considered helpful for interpreting the results, is an advantage to other comparable work.

### 5.2.3 Identification of DT Artifacts

In contrast to determining RE artifacts, no artifact model exists for Design Thinking. However, a multitude of practitioner compendia present and summarize Design Thinking-specific methods (e.g. Doorley et al. 2018; IDEO.org 2015; Uebernickel et al. 2015). Gutzwiller (1994) attributes five constituent elements to a method, i.e. *activities*, *roles*, *results* (i.e. artifacts), *techniques*, and a *metamodel*. In this sense, specific results that are in line with a DT-specific artifact can be assigned to each method. A method can produce more than one artifact as output. This definition provides the foundation for determining, synthesizing, and summarizing the artifacts in a DT-based artifact model. Figure 9 presents the development steps.

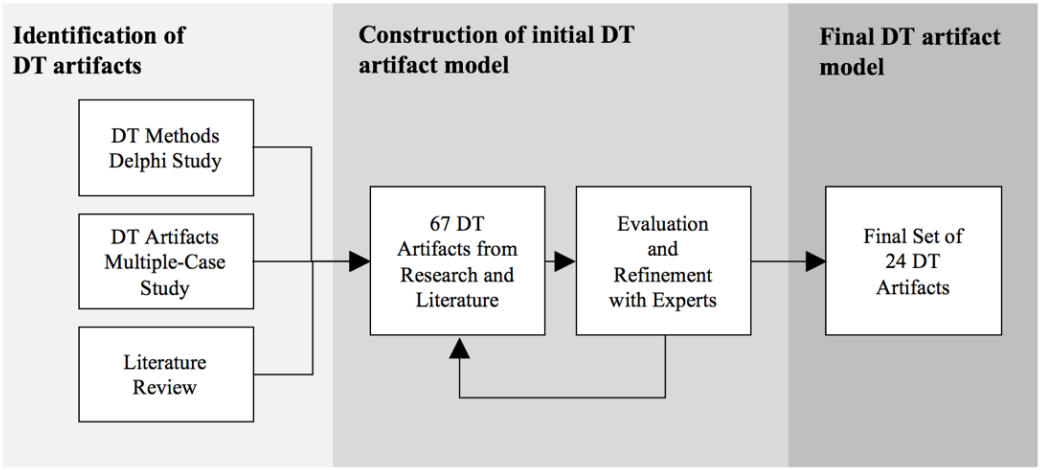



**Figure 9: Development Steps of DT-based Artifact Model**

**(Source: own illustration)**

*Identification of DT artifacts:* Three sources of evidence provide data triangulation and construct validity to identify relevant DT artifacts (Yin 2014). First, the results of a Delphi study about the most commonly used methods in Design Thinking are leveraged (for a detailed discussion of the results see Hehn et al. 2018). After three evaluation rounds and out of an initial collection of 172 Design Thinking methods, the study identified the 52 most relevant ones for designing innovative digital solutions and services. Each method was ranked by its importance and assigned a phase in the Design Thinking process model (i.e. define, needfinding, synthesis, ideation, prototyping, testing). In the context of identifying relevant DT artifacts, the prioritized method list provides a solid foundation to understand which artifacts (as results of each method) are considered highly relevant and how to structure them. The output yielded 47 DT-related artifacts.

Second, data and empirical findings from the multiple-case study presented in chapter 3 are used. This source of evidence can be considered as particularly relevant as the multiple-case study has investigated the benefits of using Design Thinking for eliciting requirements. Similar to the data analysis as described in section 3.2.2, document analysis was performed based on the available data sources (Bowen 2009). This approach is particularly suitable because the topic is concerned with identifying DT-related artifacts and all documentation produced within the project can be seen as a relevant indicator for this endeavor. Accordingly, the analysis yielded 15 DT artifacts on top of the 48 ones from the Delphi analysis. The additional artifacts were mainly related to elicitation activities, for example, ‘hot reports’, ‘field research plan’, ‘testing plan’, ‘constraints’, or ‘need classification’. Eventually, the analysis concluded with 63 DT-related artifacts.

Third, for the purpose of comprehensibility, existing academic literature and practitioner catalogs were examined to not miss out on any important DT-associated artifacts. Two additional artifacts that had not been considered before, were added: the ‘UX model’ and ‘source code’ as part of the method of high-fidelity prototyping. The final set included 65 DT-related artifacts.

*Construction and evaluation of an initial artifact-based DT model:* Figure 10 provides an overview of the initial set of 65 DT artifacts. The artifacts are structured according to the Design Thinking phases from top to bottom. The color assigns each artifact to a Design Thinking phase (green = define, blue = needfinding & synthesis, yellow = ideation, purple = prototyping, red = testing).

The model shows the artifacts in relation to each other, either by denoting dependencies (black arrowhead) or by classifying sub-artifacts (white arrowhead). An example for the former is the artifact 'stakeholder map' that is dependent on the 'design challenge/problem statement'. An example for the latter is the artifact 'secondary research report' that is composed by the artifacts 'trend report' and 'benchmarking report'.

The artifact model was evaluated in unstructured interviews with four Design Thinking experts from academia and industry. All of the experts were required to have either applied or researched Design Thinking methods for a considerable amount of time. Specifically, people were chosen when they had a proven track record of using Design Thinking in the context of innovative software-intensive projects for the past three years. During the conversations with experts some incremental adaptions were made immediately, others were discussed iteratively. Based on the feedback three main findings evolved: First, the completeness of relevant artifacts and their attributions to the Design Thinking phases have been confirmed by all experts. Second, due to initial confusion of the experts, the original structure was adapted for better readability and comprehensibility (as presented in Figure 10). The artifacts were arranged from top to bottom according to the chronological order in which they typically appear in a project. Third, it became apparent that the model was perceived as rather complex, especially in comparison to the reference frame from Requirements Engineering (Figure 7). Accordingly, the model was refined to fit to the frame of reference in terms of granularity of the artifacts.

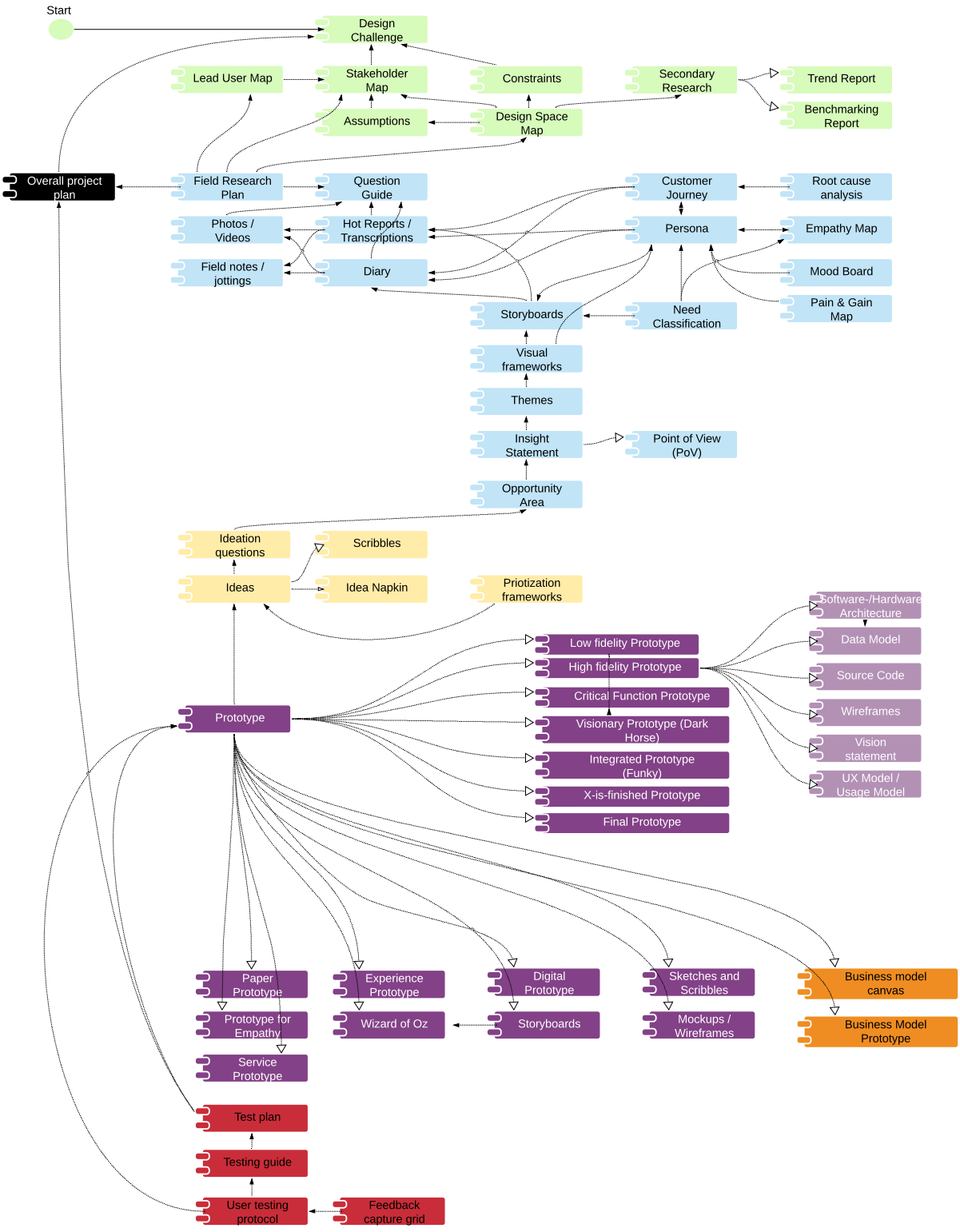



**Figure 10: Version 1 of the Artifact-based Design Thinking Model**

**(Source: own illustration)**

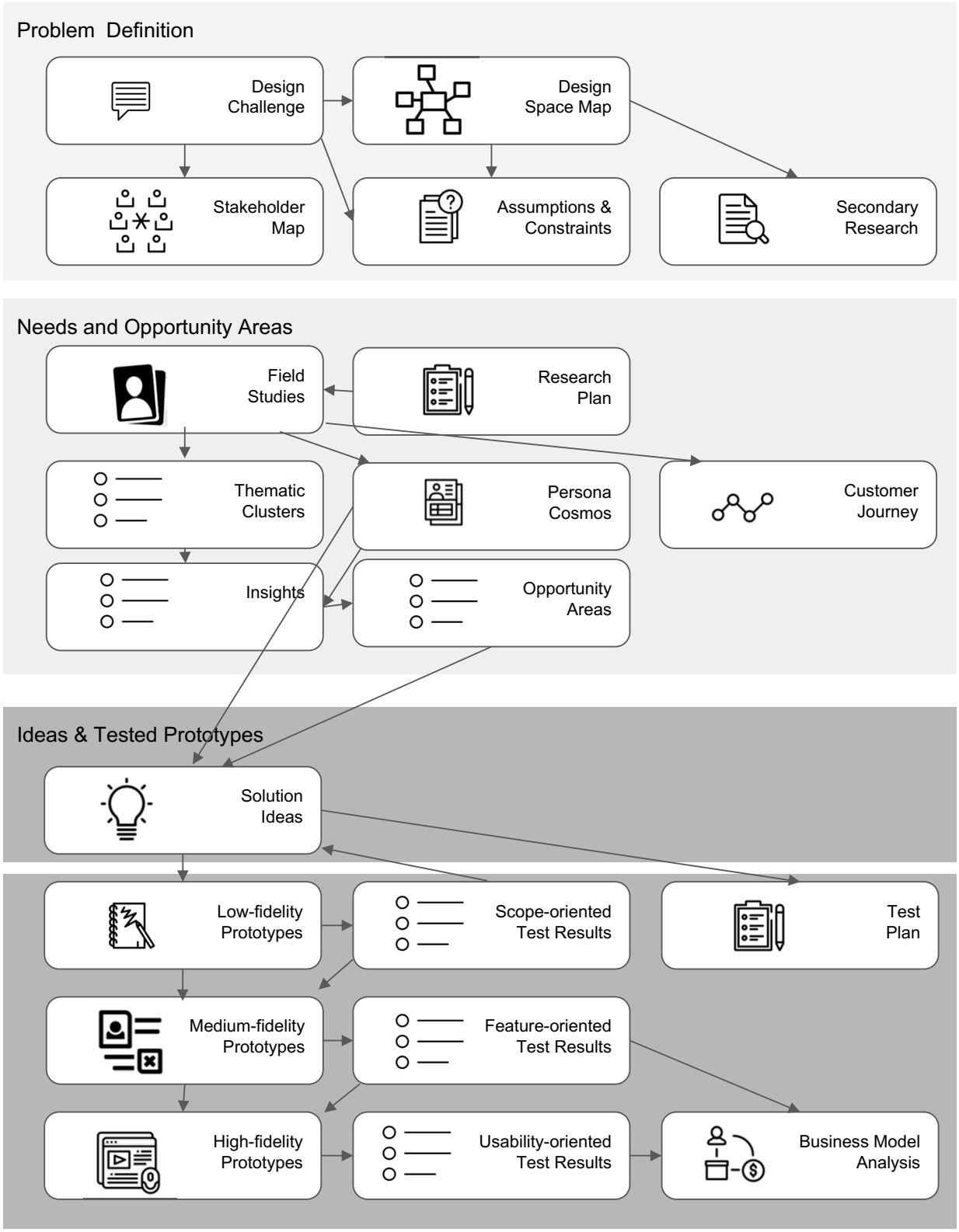



**Figure 11: Version 2 of the Artifact-based Design Thinking Model**

**(Source: own illustration)**

The second version of the model is shown in Figure 11 and encompasses 21 artifacts. With the intent to reduce the number of artifacts, some of the DT artifacts in the initial model were aggregated, for example, the ideation artifacts (including 'ideation questions', 'idea napkins', 'ideas', 'prioritization frameworks', 'scribbles') were synthesized into one single DT artifact named 'solution ideas'. In addition, the aggregation of 23 prototype artifacts into three different content types (i.e. low-, medium-, and high-fidelity) and their respective 'test results' resulted in a significant reduction of complexity as compared to the initial artifact model. The arrows denote input-output relationships between the artifacts. For reasons of simplicity not all interdependencies are depicted in the model.

*Construction of the final artifact-based Design Thinking model:* The revised and final version of the artifact-based Design Thinking model is visualized in Figure 12. It encompasses 24 DT artifacts structured into problem-oriented artifacts (sub-classified into *define, needfinding,* and *synthesis*) and solution-oriented artifacts (sub-classified into *ideation* and *prototype & test*).

Compared to the second version, the following changes were made: (1) Separation of the artifact 'assumptions and constraints' into two distinct artifacts because each artifact specifies different content; (2) elimination of 'research plan' and 'test plan' as their output was considered more important than the actual plan itself; (3) renaming of selected artifacts for more consistency within the model, for example, 'persona cosmos' was re-phrased as 'personas'.


**Figure 12: Final Artifact-based Design Thinking Model**

**(Source: own illustration)**

### 5.2.4 Core Construction and Validation

The steps of the core construction and evaluation of the final combined artifact-based reference model for Design Thinking and Requirements Engineering are presented in Figure 9.

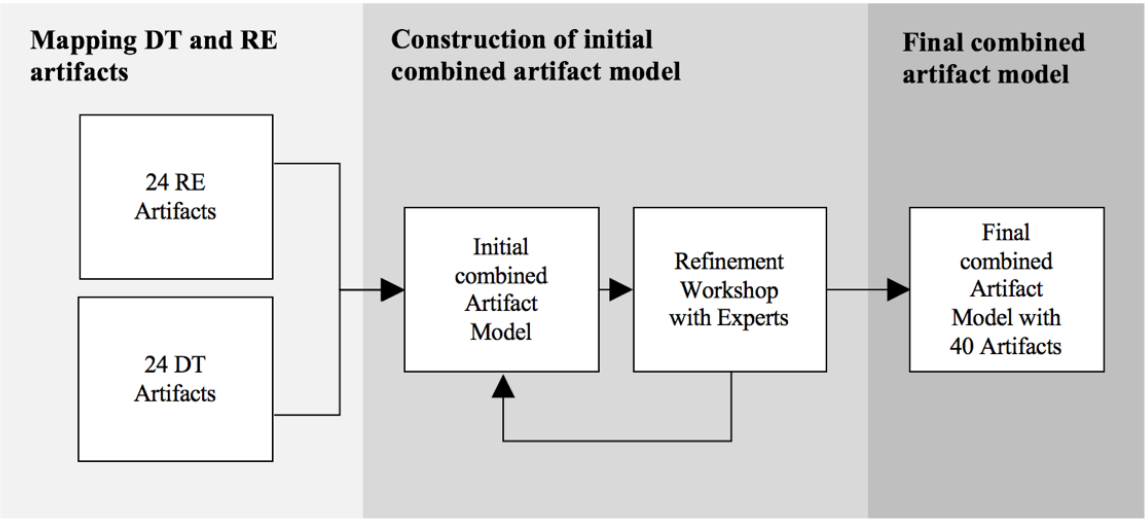



**Figure 13: Construction and Evaluation of Combined Artifact Model**
**(Source: own illustration)**

*Mapping DT- and RE-related artifacts:* The process of mapping artifacts from Design Thinking and Requirements Engineering was performed by two researchers, including one Design Thinking expert and one Requirements Engineering expert. Both experts possessed practical and academic experience (> 5 years) in their respective area of expertise. The mapping process included two main activities. First, all DT artifacts were discussed and assigned to one of the three abstraction layers (context, requirements, system) from the AMDiRE structure framework. Second, the content of artifacts from both Design Thinking and Requirements Engineering was compared and, then either attributed to only one of the two approaches or to both. The comparison was performed with 24 DT artifacts and 24 RE artifacts. Based on these activities an initial combined artifact-based model for Design Thinking and Requirements Engineering was created.

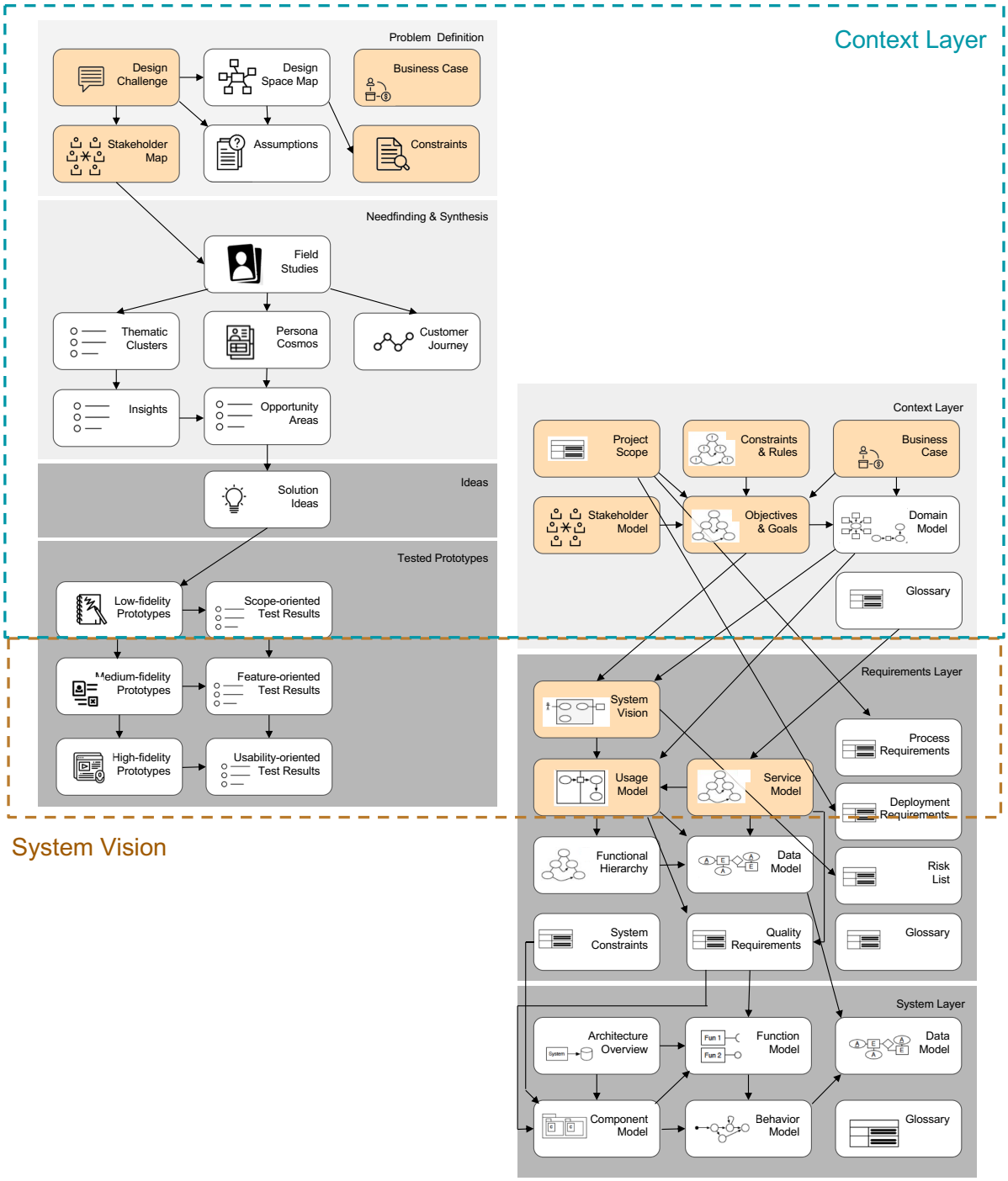



**Figure 14: Initial Version of Combined Artifact-based Reference Model**
**(Source: own illustration)**

*Construction and evaluation of an initial combined model:* The initial version of the model is divided into DT-associated artifacts on the left side and RE-associated artifacts on the right side. The colored artifacts indicate that the content of the artifacts can be created by performing either Design Thinking or Requirements Engineering activities. The position of the DT artifacts indicates which ones belong to the context layer (from top to bottom all until the row of low-fidelity prototypes, scope-oriented test results, and

test plan) and which ones to the requirements layer. There are no DT artifacts that can be assigned to the system layer.

The initial model was evaluated in the form of semi-structured interviews with four Design Thinking experts and three Requirements Engineering experts. Regarding the selection of Design Thinking experts, the recommendation of Ahlemann and Gastl (2007) was followed, i.e. to involve the same persons who participated in the first interviews. As interview partners with knowledge in building software-intensive systems were included in the first round, this knowledge was considered as adequate enough to also evaluate the combined artifact-based model for Design Thinking and Requirements Engineering. One of the advantages of including the same interview partners is that they are already known and their respective statements can be more easily qualified with respect to each other. Regarding the selection of Requirements Engineering experts, it was made sure that they possessed the required knowledge and experience for evaluating the combined artifact model (Ahlemann and Gastl 2007). In this case, expertise in applying Requirements Engineering methods in practice as well as in building artifact-oriented Requirements Engineering models was required. In addition, basic knowledge in Design Thinking was a prerequisite.

The experts were confronted with the artifact model and were asked for feedback regarding the strengths and weaknesses of the model. Furthermore, possible improvements were discussed. Following the suggestions of the experts, the model was reconstructed. For example, feedback regarding (1) structure and (2) artifacts was considered and integrated into a new version of the model. First, in terms of structure, the model was perceived as confusing because it still felt like two models next to each other without clear connections. Therefore, the artifacts were integrated into one clear structure. Second, in terms of artifacts, the number of artifacts was reduced. For example, eight artifacts that can be related to Design Thinking and Requirements Engineering were merged into one artifact while keeping their respective names (e.g. the ‘stakeholder map’ from Design Thinking was merged with the ‘stakeholder model’ from Requirements Engineering and aggregated in one artifact labeled ‘stakeholder map/model’).

Based on the feedback from experts, the initial model was refined and a second version was created that closely resembled the final artifact model as presented in Figure 15.

Subsequently, the previous interview partners were consulted for feedback with the objective to validate, refine, and stabilize the model construction. Following the approach by Lincoln and Guba (1985), this cyclic process was terminated when no more insights could be gained by new interviews. It was concluded that consent was reached among the experts regarding the model's propositions. Eventually, confirming feedback was received to validate the model, especially regarding completeness and comprehensibility.

Controversial feedback was provided about the DT-related prototype artifacts. Some of the experts also regarded them as RE-related artifacts as they can also be created in traditional Requirements Engineering approaches. Finally, the artifacts regaring prototypes were kept as DT-specific in order to highlight their heavy usage as a typical Design Thinking practice in comparison to commonly applied Requirements Engineering practices.

*Final combined model:* Figure 15 illustrates the final version of the combined artifact-based reference model that integrates 40 DT- and/or RE-related artifacts. The model is described and discussed in detail in the next section 5.3. The practical application of the model is indicated in chapters 6 and 7. For instance, the model was used to decide between different operationalization strategies that combine Design Thinking and Requirements Engineering for innovative software-intensive development projects.

## 5.3 The Combined Artifact-based Reference Model

In the following, the combined artifact model is described. It results from the consolidation of the conceptual and empirical contributions that were introduced in the previous sections. In this section, the basic structure of the model is presented first, then the distribution of DT and RE artifacts is elaborated, and at the end a detailed description of each artifact and its relations to other artifacts in the model is provided. The combined artifact model serves to

(1) understand the different abstraction levels (*context*, *requirements*, *system*) in which artifacts can be created, refined and/or decomposed

(2) permit progress control by means of completion levels

(3) enable quality assurance in terms of the artifacts' content for solution-orientation.

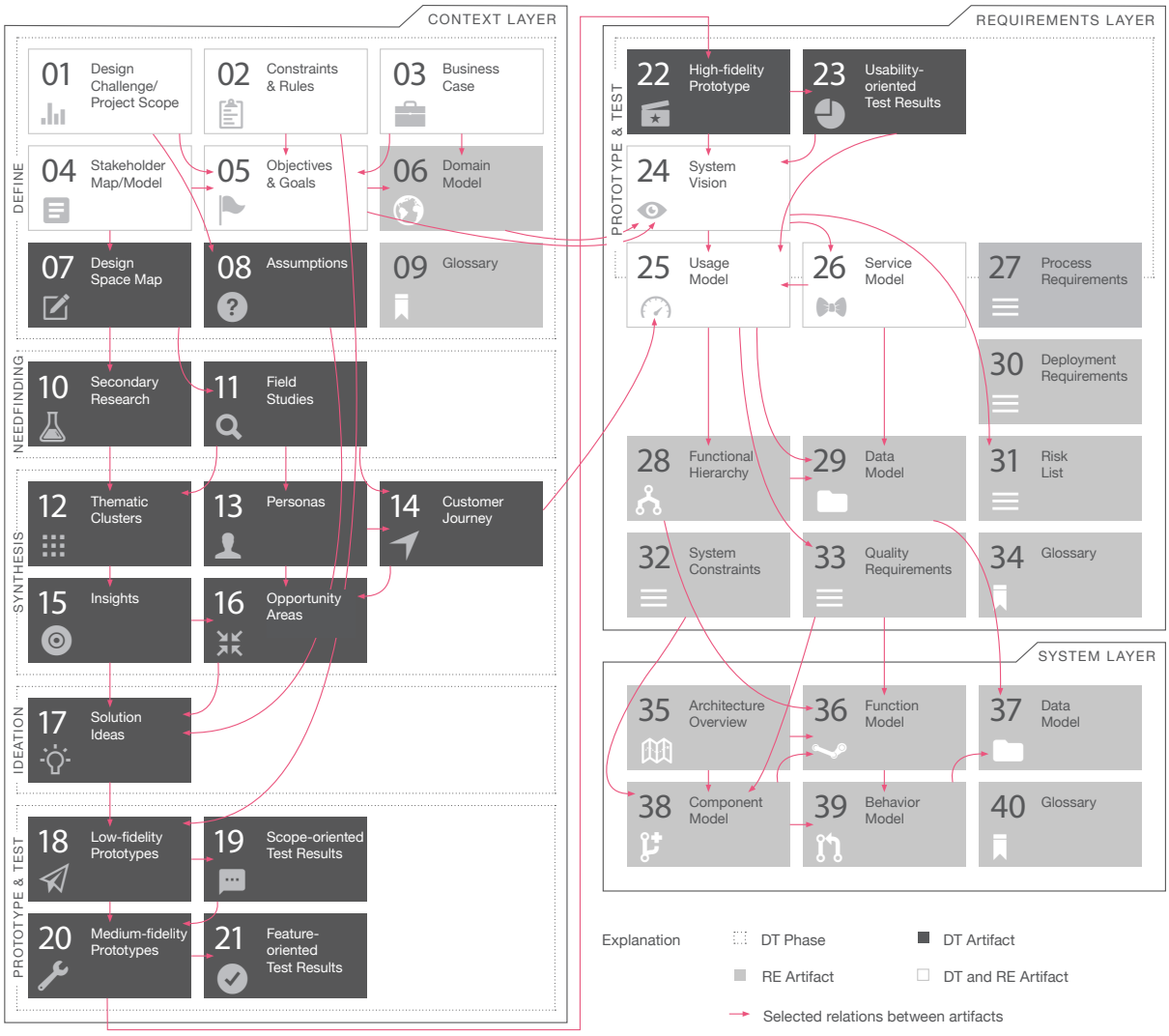



**Figure 15: Final Combined Artifact-based Reference Model**

**(Source: own illustration)**

### 5.3.1 Overall Structure

The overall structure of the model consists of three layers (context, requirements, system) each capturing a collection of relevant artifacts from Design Thinking and/or Requirements Engineering. All artifacts are denoted in rectangles including the name of the artifact and a number. Associations depict relations between the artifacts, however not exhaustive, for reasons of reducing visual complexity. The Design Thinking phases are integrated into the overarching structure of three layers to provide a sub-structure for organizing the DT artifacts. The phases are marked through a dotted line and can be found in the context and requirements layer. Table 8 summarizes the elements used to compose the artifact model.

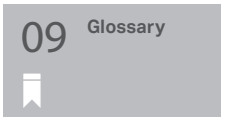
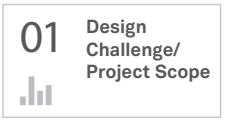
| Visual Representation | Description |
|---|---|
|  | The folder box denotes the layers *context*, *requirements*, and *system* as the overarching structure of the artifact model |
|  | The dotted line indicates the Design Thinking phases (*Define, Needfinding, Synthesis, Ideate, Prototype, Test*) for means of comprehensibility |
|  | The dark rectangle denotes a DT artifact including the artifact name, a number in the artifact model and an icon. |
|  | The grey rectangle denotes an RE artifact including the artifact name, a number in the artifact model and an icon. |
|  | The white rectangle denotes a combined artifact (DT and RE artifact) including the artifact name, a number in the artifact model and an icon. |
|  | The arrow denotes a unidimensional relation between artifacts. It expresses an input-output relationship. |

**Table 8: Overview of Elements in the Artifact Model**

**(Source: own illustration)**

The *context layer* describes why the system is needed. The context is the part of the "real world" that affects the requirements for the system and, thus, the system itself. This comprises the business context as well as the operational context that (physically) surrounds the system under consideration (Braun et al. 2014, p. 26). Context artifacts cover the information relevant to define the context and include, for example, the overall project scope, stakeholder information, a domain model, and assumptions of the project team, and underlying goals and constraints. Information within this layer is mainly documented via natural language.

The *requirements layer* encompasses what is necessary to operate in this context. On this level, the interaction between users and the system is specified by taking a black-box view without constraining internal realization concepts (Méndez Fernández and Penzenstadler 2014a, p. 14). Requirements artifacts capture the information relevant to

define the conditions or capabilities the system should exhibit at its interface (Braun et al. 2014, p. 26). They include, for example, the system vision, high-fidelity prototypes, a usage and behavior model, and the function hierarchy as entry point for the system layer. Information within this layer is documented using both natural language and conceptual models (data, function, goals, scenario).

The *system layer* includes information on how the system is to be realized. On this level, a glass-box view on the system is described in terms of structure and behavior. System artifacts include, for example, a logical component architecture and a specification of the desired behavior, e.g. via function models. (Méndez Fernández and Penzenstadler 2014a, pp. 14-15). Information within this layer is documented using both natural language and conceptual models (data, function, behavior).

### 5.3.2 Distribution of Artifacts

The combined artifact model consists of three artifact types that encompass 40 content items with various relations. Out of all artifacts, 16 can be associated with Design Thinking, 16 with Requirements Engineering, and 8 with both (see Figure 16). The latter can be further distinguished into artifacts with similar semantics but different purpose (3 out of 8). These include the design challenge/project scope (#01), the business case (#03), and the objectives and goals (#05). The main reason for their different purpose is that in Requirements Engineering these artifacts have a convergent nature while in Design Thinking they can be considered as open because they provide the opportunity for a broad context exploration. Detailed information about all content items and the underlying concepts and differences are provided in Section 5.3.3.

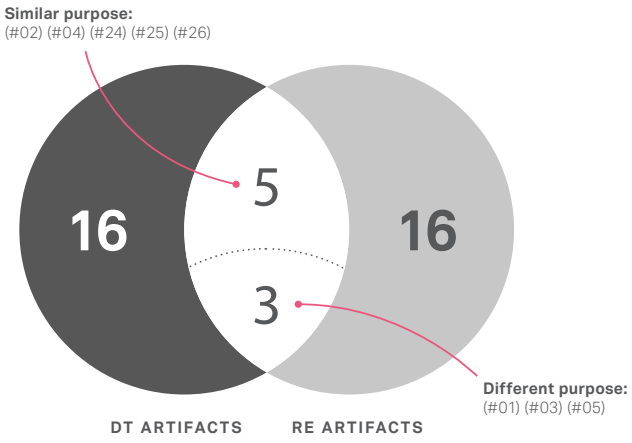



**Figure 16: Overall Distribution of Artifacts**

**(Source: own illustration)**

The distribution of artifact types according to the specific layers in the artifact model is depicted in Table 9.

| Layer | DT | DT and RE | RE | Total |
|---|---|---|---|---|
| Context | 14 | 5 | 2 | 21 |
| Requirements | 2 | 3 | 8 | 13 |
| System | 0 | 0 | 6 | 6 |

**Table 9: Layer-related Distribution of Artifacts**
**(Source: own illustration)**

The model positions the majority of artifacts within the context layer (21). The majority of DT-related artifacts can also be found here (14 DT only and 5 DT & RE artifacts). Next to the data model (#29, #37) the glossary (#09, #34, #40) is an RE-only artifact that can be found in all layers. This artifact type is revised based on the specific layer objectives. Starting in the context layer, the design challenge/project scope (#01) defines the relevant problem and primary scope of a project. Within this realm, the stakeholder map/model (#04) captures the most relevant stakeholders and their relationships. They provide one important rationale for the requirements and goals of the system (#05). The domain model (#06) contains context information and constraints (#02) about the operational environment connecting it to the requirements layer. DT artifacts complement and expand these mainly RE-related artifacts with a broad and human-centered perspective. For example, field study results (#11) and insights (#15) help to frame the project scope (#01) and inform specific use cases and scenarios (#25, #26) as defined in the requirements layer. Low- and medium-fidelity prototypes (#18, #20) are mainly leveraged to better understand stakeholder needs and system context.

The requirements layer contains five DT-related artifacts (two DT only and three DT & RE artifacts) and eight RE artifacts. The system vision (#24) denotes the general concept and idea of the intended system. High-fidelity prototypes (#22) are a way to visually enrich the system vision (#24) and to illustrate the key functionalities and general form of interaction (app, desktop solution etc.). Agreed upon by the relevant stakeholders, a system scope, i.e. major features and use cases as well as its constraints (#32), is specified. A service model (#26) defines the services the system shall offer complementary to the use cases defined through a use case model (#25). User-visible

system functions are structured in a functional hierarchy (#28) which is the entry point into the system layer.

The system layer holds six RE artifacts and none of them are related to DT. While the context and the requirements layers include the information aspects that are typically found in DT- and RE-related artifacts, the system layer includes the items addressing what is known as the solution space and providing the interface for Requirements Engineering into design activities. In the system layer the functions of the functional hierarchy (#25) are related to components (#38), a functional model (#36), and their internal behavior (#39), which also provides the basis to identify the data model (#37).

### 5.3.3 Description of Artifacts

The structure and information of the description of each artifact type are illustrated in the following overview:

- *Artifact Name:* Captures the name of the artifact type and references the assigned number (#) within the artifact model. If the artifact can be attributed to both Design Thinking and Requirements Engineering, different descriptions for both approaches (e.g., Design Challenge and Project Scope) are marked by a slash (/). In this case, the description for the Design Thinking-related artifact is provided first and the Requirements Engineering expression second.

- *Description:* Describes the content, main elements, and the structure of each artifact type (e.g. the individual slots of a use case description or the types of elements in a context model).

- *Purpose:* Contains a brief description of the purpose of each artifact type. This description includes the rationale why artifacts of this type have to be created and why they are relevant in the problem analysis or development processes. For the artifacts that relate to both Design Thinking and Requirements Engineering, an additional paragraph is included to highlight similarities and/or differences.

- *Interdependencies:* Summarizes the relationships between the artifacts with regard to their content within the artifact model. The description differentiates between the input that artifacts receive from the content of other artifacts and the output that they provide for other artifacts in the artifact model.

- *Notation:* Suggests appropriate documentation and specification techniques for each artifact (e.g. natural language, Unified Modeling Language (UML) class diagrams, model-based documentation).

- *Example:* Provides a picture and short example case from practice to complement the description. Additional information in the text box can contain exemplary process steps to create the artifact or evaluation criteria for checking the quality of the created artifact.

#### *5.3.3.1 Design Challenge / Project Scope (DT and RE Artifact #01)*

*Description:* The 'design challenge / project scope' describes the business problem that defines the starting point in every design process for eliciting needs and requirements (IDEO LLC. 2012, p. 20).

*Purpose*: It provides direction and guidance for all further problem analysis tasks and development efforts. Although the semantics of this artifact type are similar to both Design Thinking and Requirements Engineering, their purpose can differ significantly. In Design Thinking, the design challenge provides a direction, but not yet an intended solution (Uebernickel et al. 2005, p. 88). It rather inhabits an exploratory character, i.e. to open the problem space and leave room for unpredicted discoveries that might even lead to a potential re-definition of the challenge at later project stages. In Requirements Engineering, the project scope pursues a convergent objective, i.e. to set a clear statement of intent of a potentially resulting project (Méndez Fernández and Penzenstadler 2014a, p. 37).

*Interdependencies:* The design challenge / project scope is the starting point of each project and, thus, provides input for project-related objectives and goals (#05), assumptions (#08), process requirements (#27), and deployment requirements (#30).

*Notation:* This artifact is usually specified by natural text.

*Example: Design Challenge / Project Scope*

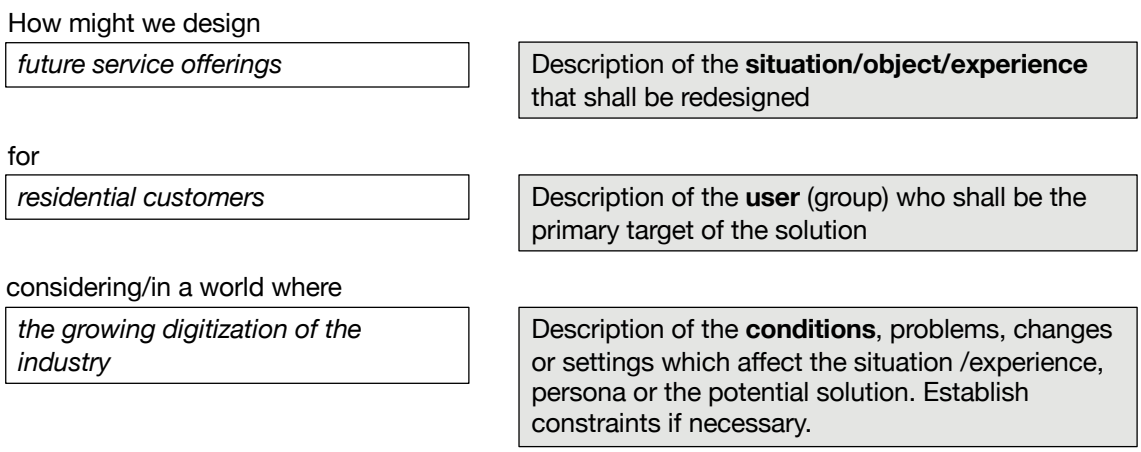



**Figure 17: Example of a Design Challenge**
**(Source: University of St.Gallen, DT@HSG 2014)**

Example of a template for a design challenge. The formulation usually starts with the phrase "*How might we"*, the primary target group, and the context or pre-conditions. Two examples from two different energy companies in Germany: *"How might we design future service offerings for residential customers considering the growing digitization of the industry?"* and *"How might we design a smart meter based added value product or service for households and business customers?"*

#### *5.3.3.2 Constraints and Rules (DT and RE Artifact #02)*

*Description:* 'Constraints and rules' are restrictions and fixed design decisions that have an effect on the system design and implementation and must be obeyed or satisfied. Constraints might be interdependent and can be of legal, cultural, technical, physical, and environmental nature (Méndez Fernández and Penzenstadler 2014a, p. 37). Constraints do not provide any additional capability to an intended system; they rather control the way in which one or more capabilities are to be delivered.

*Purpose:* Constraints and rules limit decision-making in projects. Establishing them helps to run and manage the project within the intended business and technical restrictions (IDEO LLC. 2012, p. 19). However, too many constraints, all of them reasonable, can also make a development impossible (Hull et al. 2011, p. 111), which is why they should be examined carefully. The limiting factor of constraints and rules in Design Thinking is often explicitly challenged, for example via low-fidelity Dark Horse

prototyping (Uebernickel et al. 2005, pp. 36-37) to challenge those constraints and think beyond the given limitations.

*Interdependencies:* Constraints and rules are closely associated with the solution's objectives and goals (#05) and can provide input for the creation of low-fidelity prototypes (#18).

*Notation:* Constraints and rules are usually specified by natural text.

---

*Example: Constraints and Rules*

---

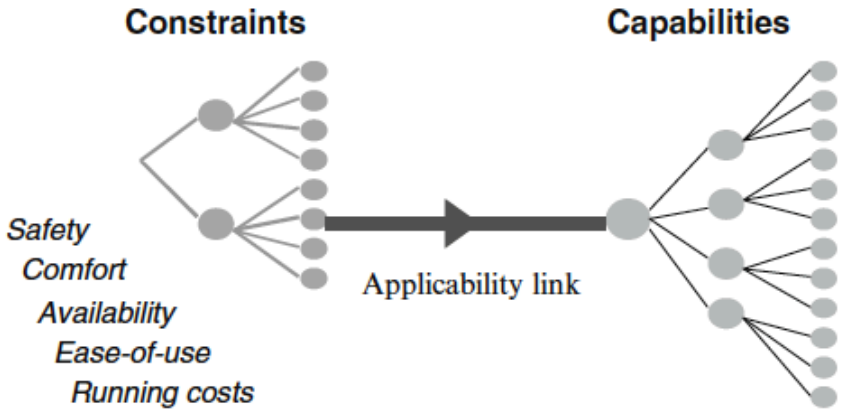



**Figure 18: Connecting Constraints and Capabilities**

**(Source: Hull et al. 2011, p. 111)**

Constraints are often related to safety, comfort, availability, ease-of-use, and running costs that affect the capability of a system. Business constraints are for example related to the project schedule ("the final deliverable needs to be finished by…"), the team setting ("we need these areas of expertise in the project"), or the human and financial resources available ("we have to run this project with a budget of …"). Technical constraints are for example related to a specific programming language ("Java has to be used"), operating system ("It must work on Linux"), or specific (open source) libraries. (Hull et al. 2011)

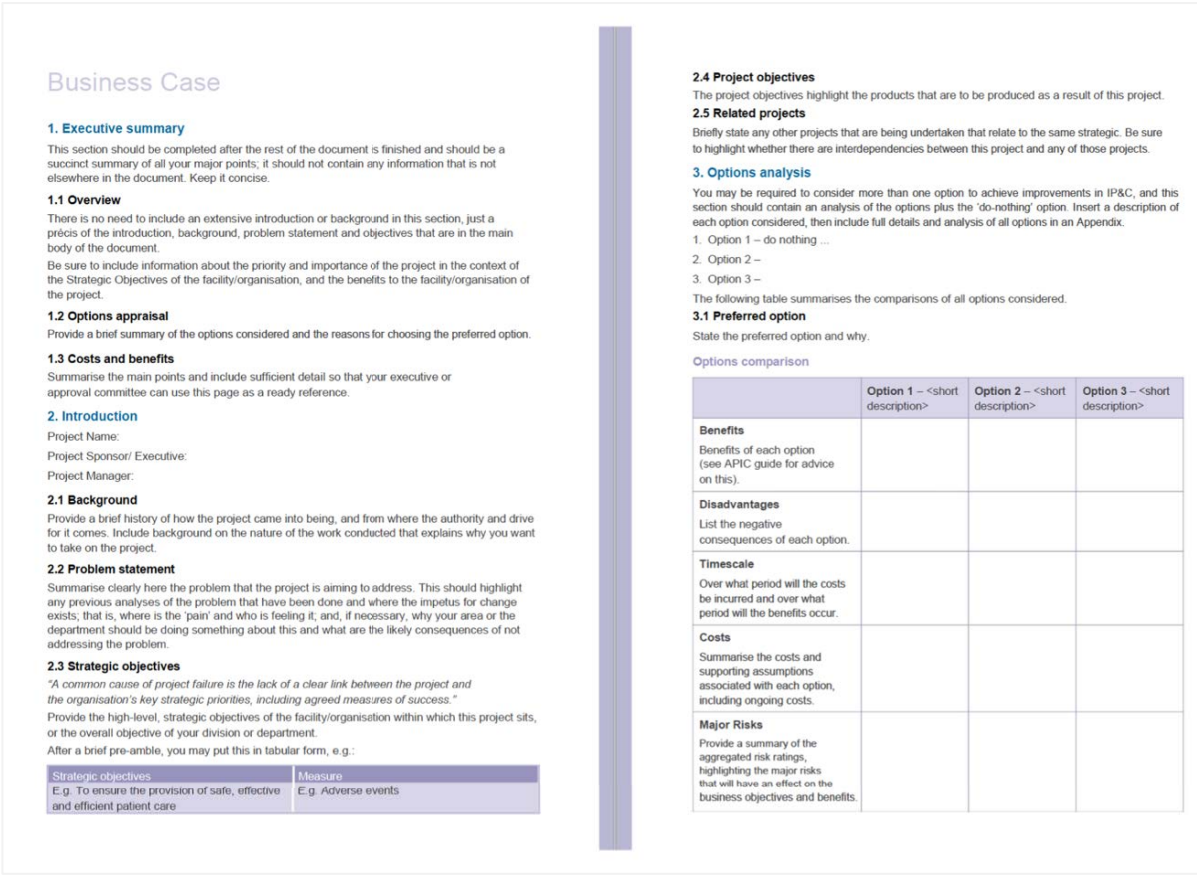
---

#### *5.3.3.3 Business Case (DT and RE Artifact #03)*

*Description:* The 'business case' provides the argumentation for a design project and typically includes an executive summary, a problem analysis, several solution options, a project description (timeline, resources, budget), a cost-benefit analysis for all

proposed options (including the calculation of no action at all), risks, and deduced recommendations (Penzenstadler 2017a).

*Example: Business Case*



Business Case

**1. Executive summary**

This section should be completed after the rest of the document is finished and should be a succinct summary of all your major points; it should not contain any information that is not elsewhere in the document. Keep it concise.

**1.1 Overview**

There is no need to include an extensive introduction or background in this section, just a précis of the introduction, background, problem statement and objectives that are in the main body of the document.

Be sure to include information about the priority and importance of the project in the context of the Strategic Objectives of the facility/organisation, and the benefits to the facility/organisation of the project.

**1.2 Options appraisal**

Provide a brief summary of the options considered and the reasons for choosing the preferred option.

**1.3 Costs and benefits**

Summarise the main points and include sufficient detail so that your executive or approval committee can use this page as a ready reference.

**2. Introduction**

Project Name:

Project Sponsor/ Executive:

Project Manager:

**2.1 Background**

Provide a brief history of how the project came into being, and from where the authority and drive for it comes. Include background on the nature of the work conducted that explains why you want to take on the project.

**2.2 Problem statement**

Summarise clearly here the problem that the project is aiming to address. This should highlight any previous analyses of the problem that have been done and where the impetus for change exists; that is, where is the 'pain' and who is feeling it; and, if necessary, why your area or the department should be doing something about this and what are the likely consequences of not addressing the problem.

**2.3 Strategic objectives**

*"A common cause of project failure is the lack of a clear link between the project and the organisation's key strategic priorities, including agreed measures of success."*

Provide the high-level, strategic objectives of the facility/organisation within which this project sits, or the overall objective of your division or department.

After a brief pre-amble, you may put this in tabular form, e.g.:

| Strategic objectives | Measure |
|---|---|
| E.g. To ensure the provision of safe, effective and efficient patient care | E.g. Adverse events |

**2.4 Project objectives**

The project objectives highlight the products that are to be produced as a result of this project.

**2.5 Related projects**

Briefly state any other projects that are being undertaken that relate to the same strategic. Be sure to highlight whether there are interdependencies between this project and any of those projects.

**3. Options analysis**

You may be required to consider more than one option to achieve improvements in IP&C, and this section should contain an analysis of the options plus the 'do-nothing' option. Insert a description of each option considered, then include full details and analysis of all options in an Appendix.

1. Option 1 – do nothing ...
2. Option 2 –
3. Option 3 –

The following table summarises the comparisons of all options considered.

**3.1 Preferred option**

State the preferred option and why.

Options comparison

| | **Option 1** – <short description> | **Option 2** – <short description> | **Option 3** – <short description> |
|---|---|---|---|
| **Benefits** Benefits of each option (see APIC guide for advice on this). | | | |
| **Disadvantages** List the negative consequences of each option. | | | |
| **Timescale** Over what period will the costs be incurred and over what period will the benefits occur. | | | |
| **Costs** Summarise the costs and supporting assumptions associated with each option, including ongoing costs. | | | |
| **Major Risks** Provide a summary of the aggregated risk ratings, highlighting the major risks that will have an effect on the business objectives and benefits. | | | |

**Figure 19: Example of a Business Case Document**
**(Source: own illustration)**

A business case can have different forms and degrees of detail. The following elements are considered essential so that all the relevant information is included: (1) the reasons why the project was initiated, (2) the options to address the problem (this can include specific solutions but also options on how to address the problem), (3) the expected benefits and limitations, (4) the projected timeline and budget needed to complete the project, and (5) major opportunities or risks when undertaking the project.

*Purpose*: The business case is used to rationalize the design project and used to convince a decision-maker or potential project sponsor. Similar to the design challenge / project scope (#01), the artifact exists in both Design Thinking and Requirements Engineering. However, its purpose and realization may differ. The main objective of the 'business

case' in Design Thinking is to evaluate the available budget regarding project execution, i.e. resources and timeline. As the actual outcome is usually not specified when applying Design Thinking, it more or less recommends whether to use Design Thinking or not. The project scope in Requirements Engineering can be much narrower and may have concrete solution options in mind.

*Interdependencies:* The business case is usually intended to satisfy the statement of intent from the design challenge/project scope (#01) and rationalize the goals stated under objectives and goals (#05) (Méndez Fernández and Penzenstadler 2014a, p. 37).

*Notation:* The business case is usually specified by natural text in a business case document.

#### *5.3.3.4 Stakeholder Map / Stakeholder Model (DT and RE #04)*

*Description:* A 'stakeholder map/model' is a list of all relevant stakeholders (internal and external) for the project (Glinz and Wieringa 2007, p. 18). A stakeholder is defined as "an individual, group of people, organization or other entity that has a direct or indirect interest (or stake) in a system" (Hull et al. 2011, p. 7). The respective interest can stem from using the potential system, developing it (the project team), having a financial interest in the solution (in terms of revenue, costs, sales), being negatively affected it (in terms of cost or potential harm), being responsible for the system (manage, introduce, operate, or maintain), or constraining the system (as regulators). The types of stakeholders are frequently expressed via the roles they inhabit in a project. They typically include the project sponsor or client, the project manager, the product manager, other (senior) decision-makers, investors, end users, customers, operators, product disposers, sales and marketing, or regulatory authorities (Hull et al. 2011, p. 96; Glinz and Wieringa 2007, p. 19). Each stakeholder group can be assessed according to their impact on the project (high, medium, low) or the risk incurred by neglecting a specific group (critical, major, minor). In complex cases, the 'stakeholder map/model' can be complemented with each of the stakeholder group's goals, their relationships, and dependencies including conflicts and information flows.

*Purpose:* The stakeholder map/model helps to identify key internal and external stakeholders as sources of requirements. Based on the list of stakeholder roles and their identified priorities, the project team can determine concrete interview partners

(Stickdorn and Schneider 2012). In addition, the stakeholder overview provides guidance for effective project management and communication. The purpose and realization of the stakeholder map/model is similar for both Design Thinking and Requirements Engineering.

---

*Example: Stakeholder Map/Model*

---

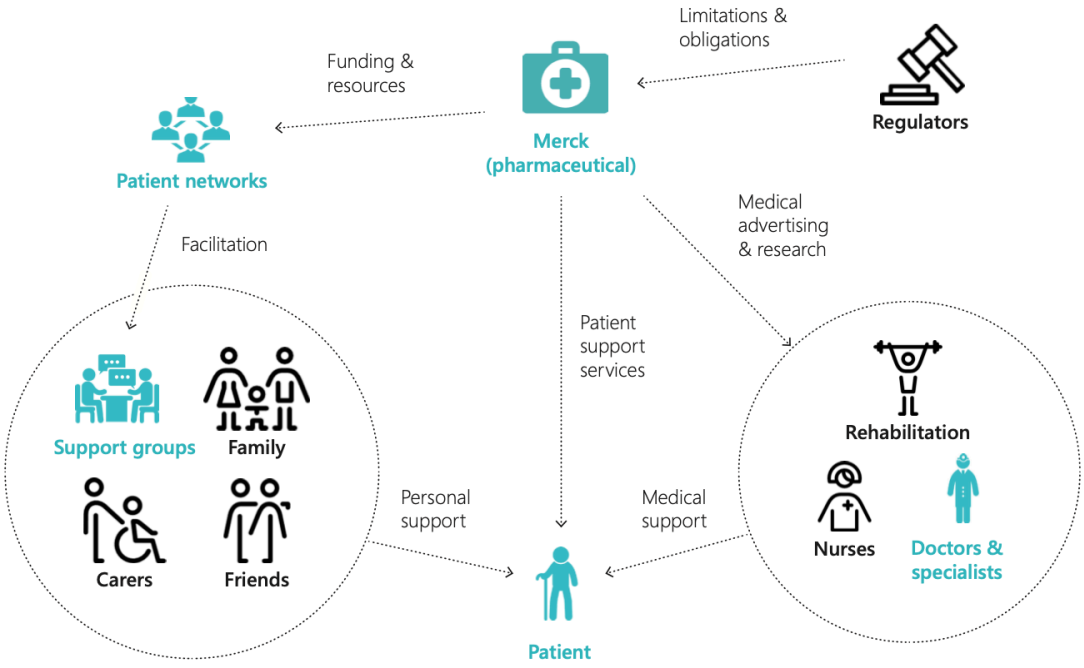



**Figure 20: Example of a Stakeholder Map**

**(Source: University of St.Gallen, DT@HSG 2016)**

The stakeholder map in Figure 20 depicts the relevant stakeholder groups around a patient as a target user of a pharmaceutical company for designing a new patient support program (PSP). The map included anyone with a significant input such as the personal support network of the patient (family, friends, care-givers, and support groups) or the medical support network like nurses, doctors and specialists and rehabilitation facilities. Each of the stakeholder group was assessed in terms of potential input and impact on the project (e.g. "personal support network: further input about patients, also possible users; large impact on project outcome").

---

*Interdependencies:* The stakeholder map/model receives input from the design challenge/project scope (#01) and provides input for phrasing the objectives and goals (#05) and for the creation of the design space map (#07). The user groups defined in the stakeholder map/model are particularly relevant for designing the specific user

interaction with the intended system (e.g. as part of the usage model (#25) (Méndez Fernández and Penzenstadler 2014a, p. 37).

*Notation:* The stakeholder map/model can be specified by natural text in form of a list or a diagram or via a UML actor hierarchy.

#### *5.3.3.5 Objectives and Goals (DT and RE Artifact #05)*

*Description:* 'Objectives and goals' are prescriptive statements of intent regarding business, usage, or system goals issued by a stakeholder (van Lamsweerde 2009, p. 260). Goals are related to each other in terms of conflicts, constraints, or support and can be phrased on different abstraction levels from high-level (e.g. strategic results) to low-level (e.g. specific technical aspects of the target system). Types of goals can be quality-related (non-functional goals like usability, performance, security, accuracy, reliability, e.g. "serve the customer quickly"), optimization-specific (increase/decrease of measurable items), behavioral (functional goals, e.g. "provide cash"), and also anti-goals (e.g. safety hazards, security threats) (Rolland and Salinesi 2005). As opposed to requirements that are objectively measurable/decidable (e.g. "The user interface must support the languages…"), goals are often hard to measure (e.g. "easy to use").

*Purpose:* Objectives and goals can provide direction for problem analysis and system development tasks. In addition, they offer early identification and resolution of conflicts and they justify the rationale of a requirement. The artifact can exist in both Design Thinking and Requirements Engineering. If given in Design Thinking, the list contains mainly high-level business goals and objectives provided by the project sponsor to keep outcome and specifics open for exploration. In Requirements Engineering they can be much more precise according to the mentioned modeling and description techniques.

*Interdependencies:* The artifact objectives and goals gains its input from the content of the design challenge/ project scope (#01), from defined constraints and rules (#062), from the analysis in the business case (#03), and the stakeholder groups in the stakeholder map/model (#04). And it provides the foundation for developing a domain model (#06), system vision (#24), and usage model (#25).

*Notation:* Objectives and goals' can be specified by natural text in diagrams or via goal graphs like KAOS (keep all objectives satisfied).

*Example: Objectives and Goals*

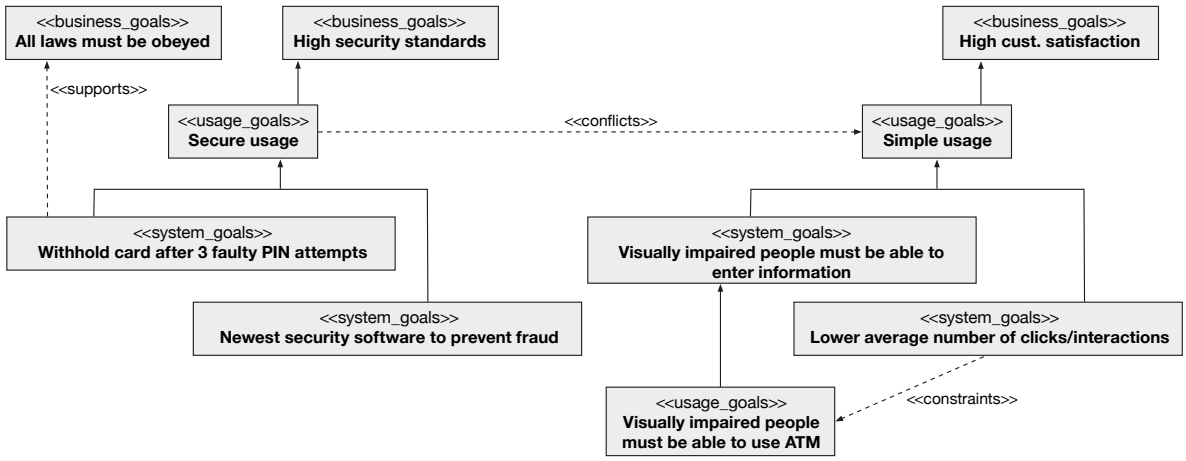



**Figure 21: Example of Objectives and Goals**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 21 shows an example of a goal model tree for an automatic teller machine (ATM) including business, usage, and system goals. Each system goal is connected to a usage goal and each usage goal to a business goal. A hierarchical decomposition of goals into sub-goals is undertaken with system goals. The model also shows three possible dependencies between the goals: (1) To *support* each other in achieving a goal (e.g. "all laws must be obeyed" is supported by the system goal "withhold card after 3 faulty PIN attempts"), (2) to be in *conflict* with each other (e.g. "secure usage" and "simple usage"), and (3) to act as a *constraint* for another goal (e.g. the system goal "lower average number of clicks/interactions" is constrained by the usage goal "visually impaired people must be able to use ATM") (Penzenstadler 2017b, p. 31-42).

#### *5.3.3.6 Domain Model (RE Artifact #06)*

*Description:* The 'domain model' is a conceptual model composed of all the topics related to a specific problem. The domain objects are not software objects but a representation of real-life conceptual objects. The model captures the fundamental business entities, their attributes, roles, and relationships, and the constraints that influence the problem domain (Ambler 2018; Penzenstadler 2017c, p. 3).

*Purpose:* The domain model ensures an understanding of the landscape of business entities in the problem area and can be used to solve problems related to that domain. It contains the vocabulary of the domain, thus, allowing the model to be communicated to non-technical stakeholders (Ambler 2018).

*Interdependencies:* In the artifact model, the domain model receives its input from the business case (03#) and the objectives and goals (05#). It provides the basis for the glossary (#09, #34, #40), the system vision (#24), and usage model (#25).

*Notation:* The domain model can be specified by UML activity diagrams or Business Process Model and Notation (BPMN).

---

*Example: Domain Model*

---

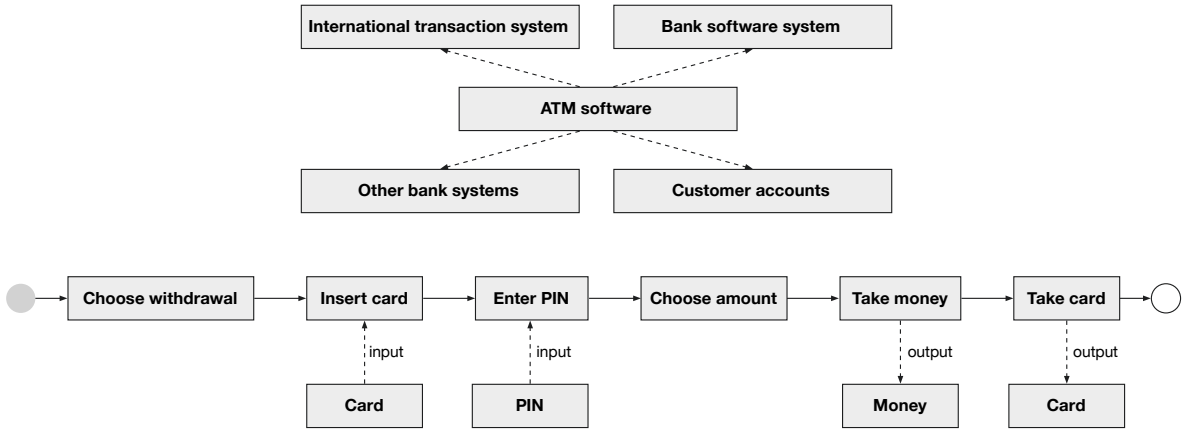



**Figure 22: Example of a Domain Model**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 22 shows a simplified domain model for an ATM. It visualizes associations between the domain classes “ATM software” and other domain classes like “other bank systems” or “international transaction system”. In addition, the figure shows a business process model for money withdrawal from an ATM.

---

#### *5.3.3.7 Design Space Map (DT Artifact #07)*

*Description:* The ‘design space map’ depicts an overview of what the project team knows and what they do not yet know in the context of the project (IDEO LLC 2012, pp. 26-27). Knowledge gaps provide research directions on which the team needs to

focus, get answers, and do primary research in the needfinding phase. The ‘design space map’ is usually created in an analog form with sticky notes on the wall.

*Purpose:* The design space map helps to structure the exploration phase. It provides a common understanding of the design challenge in a multidisciplinary team and sparks relevant discussions about the project dimensions at the beginning of a project. The design space map acts as a navigator for further activities and evolves over the duration of a project in which new knowledge is added.

---

*Example: Design Space Map*

---


**Picture 7: Example of a Design Space Map**
**(Source: University of St.Gallen, DT@HSG 2015)**

The design space map of a project with a German insurance provider is shown in Picture 7. The design challenge (“revolution in CIO dialog”) is put in the middle of the wall, while the green post-its provide an overview of the relevant dimensions to be looked at (e.g. motivation, communication, CIO, reporting structures, trends). This design space map also included an overview of the organizational structure as an indication for performing further elicitation and needfinding activities.

---

*Interdependencies:* The design space map derives input from the design challenge/project scope (#01) and the stakeholder groups depicted in the stakeholder map/model (#04). Its output informs the elicitation efforts that result in secondary research reports (#10) and field studies (#11).

*Notation:* The design space map/project scope is usually specified in natural text and kept in form of post-its visible on a project room's wall.

#### *5.3.3.8 Assumptions (DT Artifact #08)*

*Description:* The project team collects their 'assumptions' about the given project. These typically include hypotheses that something is possible to achieve (within certain constraints such as time and cost), that something works because of particular conditions, and that people trust, think or aspire certain things (Dam and Siang 2017).

*Purpose:* As a delivering artifact the team gains a list of assumptions and hypotheses that they can explore and test. The list also provides a first overview of possible team biases. Making these explicit can help to validate them in research, but also to challenge them in ideation. The former helps to reflect on what they still need to know about the topic and, thus, specifies a starting point to brainstorm ways to reduce the risk they pose (IBM Corporation 2018). The latter provides a chance to re-consider concepts and come up with solutions the team has never considered possible (Dam and Siang 2017).

*Example: Assumptions*

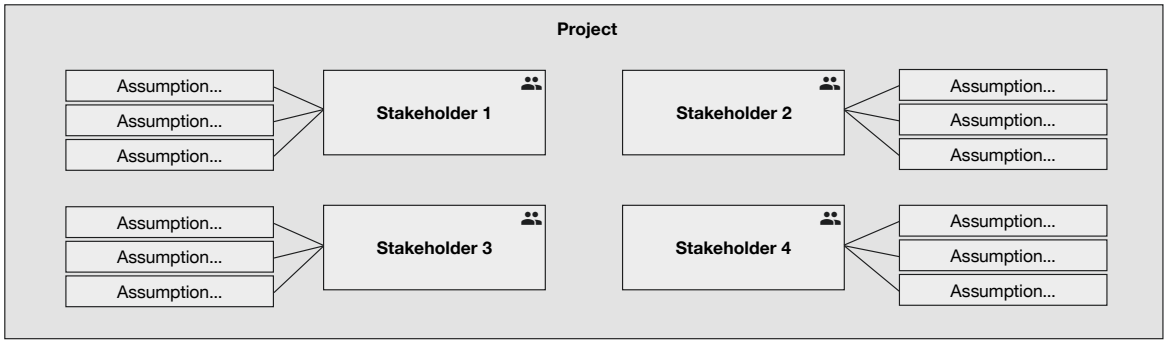



**Figure 23: Example of an Assumptions Document**
**(Source: own illustration)**

Figure 23 shows the assumptions that are associated with specific stakeholders. Accordingly, the team lists these assumptions, e.g. "Ultra-High Net Worth Individuals require personalized services" or "Business travelers work in flight to save time". These explicit hypotheses can be tested to see if they are right or wrong.

*Interdependencies:* Assumptions are derived from the design challenge/project scope (#01) and the stakeholder groups in the stakeholder map/model (#04). They influence and inform the generation of solution ideas (#17) and low-resolution prototypes (#18).

*Notation:* 'Assumptions' are usually specified by natural text and put into a list.

#### *5.3.3.9 Glossary (RE Artifact #09)*

*Description:* The glossary is a list of all relevant business or technical domain-specific terms (including abbreviations, synonyms, and descriptions) to which project members may refer to. The key elements of a glossary are terms (typically nouns), definitions (in an unambiguous way), aliases (interchangeably with the primary term), and related terms (references to similar terms) (Méndez Fernández and Penzenstadler 2014a, p. 37).

*Purpose:* A well-defined glossary ensures consistent usage of project-relevant terms throughout the entire development life cycle (Yilmaztürk 2005, p. 394). This artifact can, thus, facilitate learning about a new domain, enable clear communication among stakeholders when used consistently, and prevent unnecessary misunderstandings.

---

*Example: Glossary*

---

**Glossary**

| Term | Abbreviation | Description | Synonym | Comment |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Figure 24: Example of a Glossary Structure**
**(Source: own illustration)**

Figure 24 shows an exemplary structure for a glossary including the term, its abbreviation, a concise description, possible synonyms, and additional comments. Unlike a dictionary, a glossary contains only the expressions and terms that are unique to the respective business domain. However, it is also recommended to specify terms like "customer" as it can be understood differently by different business stakeholders.

---

*Interdependencies:* The glossary can receive its input from the stakeholder map/model (#04), the domain model (#06) and the design space map (#07). It recurs in the requirements layer (#34) and system layer (#40) as more terms are added over the course of the project.

*Notation:* The glossary is usually specified by structured natural text.

#### *5.3.3.10 Secondary Research Report (DT Artifact #10)*

*Description:* The 'secondary research report' summarizes various sources of information, insights from existing market research, and other knowledge about the given subject domain. Common information sources are market and benchmarking reports (internal and external), sales reports, internal databases, government statistics, and different media such as articles from newspapers, journals, or research studies from universities. To find out what is new, this can also include reports about innovations in a particular area or inspirations from similar settings in other industries (IDEO.org 2015).

*Purpose:* The secondary research report supports the project team to clarify research questions and gain an initial understanding of the context of the challenge (Collins 2010). It provides a solid foundation of knowledge and helps to focus on the right questions to be asked during any subsequent primary research (needfinding) (IDEO.org 2015).

*Interdependencies:* The secondary research report receives input from the design space map (#07). It mainly provides input for thematic clusters (#12) and insights (#15) in the context of the problem space.

*Notation:* The secondary research report is usually specified by natural text in a text document.

*Example: Secondary Research Report*

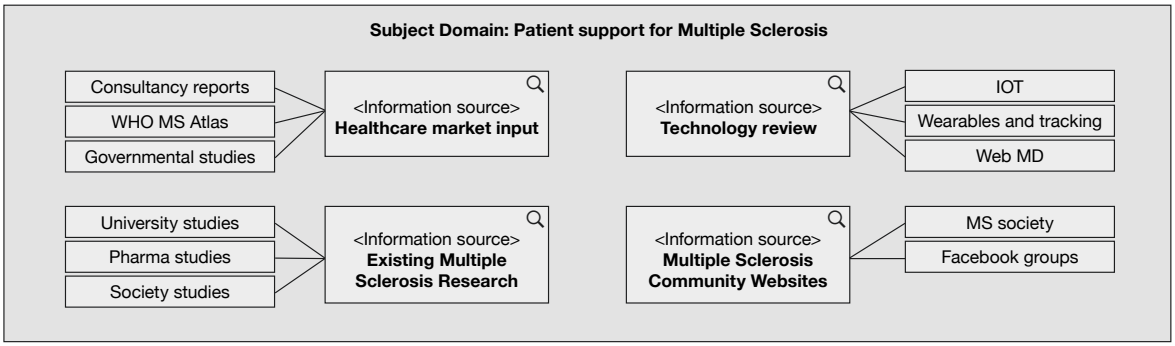



**Figure 25: Example of a Secondary Research Report Structure**
**(Adapted from University of St.Gallen, DT@HSG 2016)**

The example in Figure 25 illustrates a structure for secondary research reports about the topic: *"How does the future patient support program for Multiple Sclerosis patients look like?"* Existing information about "patient support" (e.g. trends, as-is patient journey, patient types, legal boundaries) and "multiple sclerosis" (statistics, types, course of disease, treatments) were used. For example, healthcare market inputs, multiple sclerosis research reports, and documentations about technological innovations (e.g. IoT in healthcare, wearable sensors, activity trackers) were relevant information sources.

#### *5.3.3.11 Field Studies (DT Artifact #11)*

*Description:* 'Field studies' are a collection of raw data (including user statements, observations, pictures, videos) from interviewees with identified stakeholder groups (IDEO.org 2015). Transcribing interviews on post-its creates a comprehensive picture of the real stories of users the interviewer was engaged with. It visualizes the key conclusions, quotes and observations from an interview in a way that allows it to be easily shared with other people and to be used actively in many of the subsequent steps in the project.

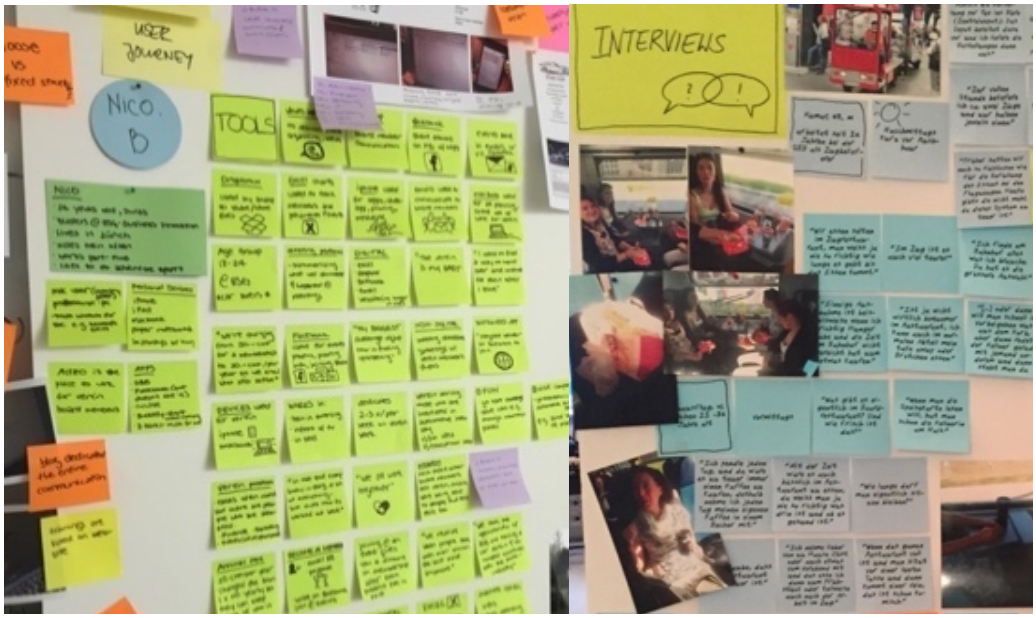
*Purpose:* Field studies help with sharing the interview among the project team. The visual story gives the team a common understanding of the raw data and empathize with the interviewees. In addition, it transforms an interview into single building blocks (min 25 post its) which can be used for analyzing, filtering, grouping, and other subsequent

methods of data processing. This usually leads to an exchange of opinions in the team fostering the ability to define patterns and relationships between single user stories.

---

*Example: Field Studies*

---




**Picture 8: Examples of Transcribed Field Studies**
**(Source: ITMP 2018)**

Picture 8 illustrates two field studies from a project for a German software provider (left) and a railway company (right). The boards summarize key information about the interviewees and their key statements in a way that allows for easy sharing with other people. All team members can approach and discuss the collected data points in order to gain empathy and share findings from different interviews. The field studies provided the basis for analyzing needs, pain points, requirements, and insights of the target stakeholder groups.

---

*Interdependencies:* Field studies are guided by the identified stakeholder groups in the stakeholder map/model (#04) and the knowledge gaps in the design space map (#07). They provide input for defining thematic clusters (#12) and creating personas (#13) and customer journeys (#14).

*Notation:* 'Field studies' are usually specified by natural text (on post-its) and complemented with pictures and videos.

#### *5.3.3.12 Thematic Clusters (DT Artifact #12)*

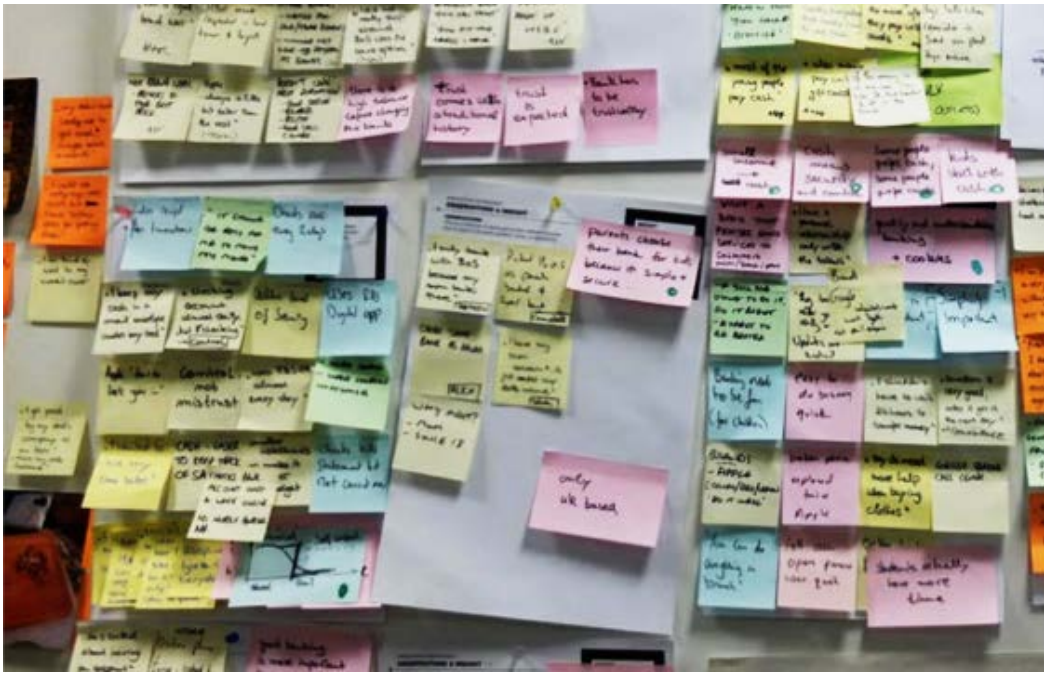
*Description:* 'Thematic clusters' are a group of user statements, observations, or other findings from primary and secondary research that represent a specific subtopic of the project content (IDEO LLC 2012, p. 43).

*Purpose:* Thematic clusters provide a first overview of relevant topics within a given subject domain. They help the project team to recognize patterns and structure research data (e.g. link similarities, uncover contradictions, find exceptions). The process of grouping data enables shared understanding and strengthens team knowledge.

---

*Example: Thematic Clusters*

---



**Picture 9: Example of Thematic Clusters**
**(Source: ITMP 2015)**

A3 sheets of paper provide the frame for clustering data points in form of statements and observations on post-its. Picture 9 illustrates thematic clusters from interviews with students about new banking services. To draw up the clusters, each team member selects statements and post-its from the field studies that appear most important to them. By grouping them the thematic clusters are defined. Each cluster is assigned a title explaining the theme of the cluster (e.g. "Bank of Mum and Dad").

---

*Interdependencies:* Thematic clusters are composed by the data from field studies (#11) and the content of secondary research reports (#10). They inform the generation of insights (#15).

*Notation:* Thematic clusters are usually specified by structured, natural text.

#### *5.3.3.13 Personas (DT Artifact #13)*

*Description:* 'Personas' are archetypes of people that represent a specific stakeholder group relevant to the project (Uebernickel et al. 2015, p. 125). They are fictional characters that are derived from the interviewees' behavior and attitudes representing the different types that might use the intended service or product in a similar way. The description of a persona should be focused on the specific project context and include information like behavioral patterns, attitudes, goals, needs, pain points, and a demographic profile (Cooper et al. 2014, pp. 82ff).

*Purpose:* Personas help to design solutions from a user point of view. They facilitate the understanding of (potential) users' needs, behaviors, motivations, and frustrations and provide alignment for discussing design decisions. The narrative structure of personas and their resemblance to real people supports empathy within the project team so that the goals and tasks of personas help to determine what a product should do and how it should behave (Cooper et al. 2014, p. 79).

*Interdependencies:* Personas are mainly derived from the data of field studies (#11). They provide input for defining customer journeys (#14) and developing opportunity areas (#16) for generating new solution ideas (#17).

*Notation:* Personas are usually specified by natural text.

*Example: Personas*

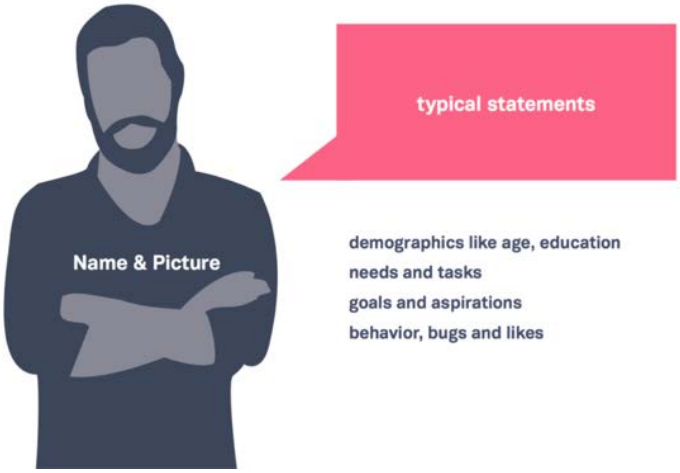



**Figure 26: Example of a Persona Template**
**(Source: own illustration)**

Different personas represent different types of users with a certain set of motivations, behaviors, attitudes, goals, and frustrations toward a current product or system. Figure 26 shows a template for describing such a persona. The following steps are undertaken to create a persona: (1) specification of behavioral and attitudinal variables that emerged from the interviews, (2) mapping of interviewees along these dimensions, (3) identification of significant behavior and attitudinal patterns, (4) synthesis of key characteristics from relevant interviewees, (5) check for redundancy and completeness, (6) detailed description of the designated persona type (Cooper et al. 2014, p. 79).

#### *5.3.3.14 Customer Journeys (DT Artifact #14)*

*Description:* A 'customer journey' is a visual representation of the experience of a customer (e.g. their specific activities, tasks, and touchpoints) when interacting with an organization, product, or service (Doorley et al. 2018, p. 7f). From the customer's individual standpoint, the journey depicts the detailed steps, key moments, and their connections of an experience, usually in a chronological order.

*Purpose:* Customer journeys help the project team to look beyond the existing solution and consider the customer's entire experience to identify areas for improvement or with innovation potential (Doorley et al. 2018, p. 7f). They offer a systematic analysis of challenges, pain points, what the customer liked and loved during the experience. They

also give a deeper understanding of interaction points between different ecosystems. In addition, the creation of a journey can help the project team to gain empathy for the steps a customer goes through.

*Interdependencies:* Customer journeys get their input from field studies (#11) and personas (#13). Their output informs the creation of insights (#15), opportunity areas (#16), and use cases for the solution depicted in the usage model (#25).

*Notation:* Customer journeys are usually specified by natural text and sequence and activity diagrams.

---

*Example: Customer Journey*

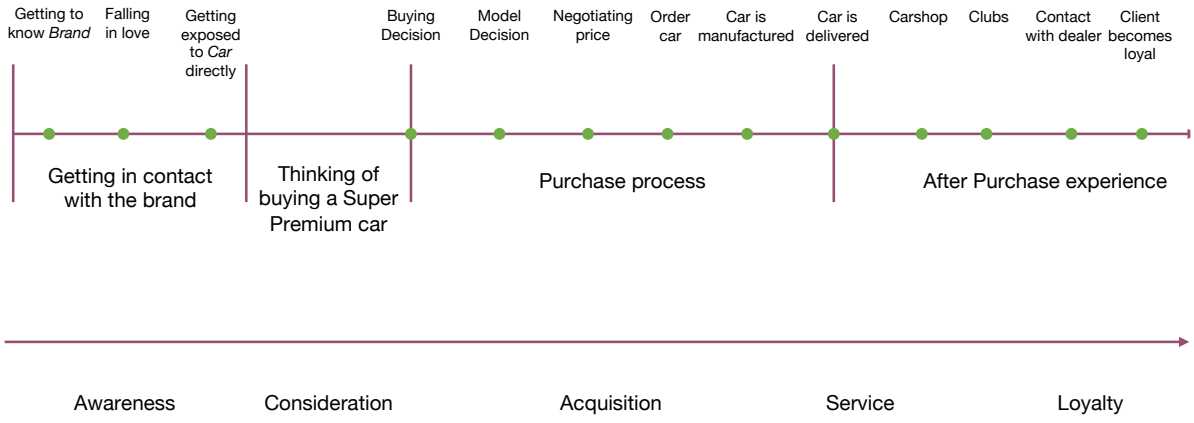



**Figure 27: Example of a Customer Journey**
**(Source: University of St.Gallen, DT@HSG 2019)**

Customer journeys can take different forms depending on the specific project needs and existing journeys. Figure 27 shows a simplified customer journey when buying a premium car. When creating a journey, the project team maps the stages and actions along a timeline (e.g. “getting to know the brand”, “order car”, “contact with dealer”). The team then marks the pains and the needs of the customer in each individual stage based on the findings from observations and interviews. Each stage is looked at and prioritized from both a supply and a demand point of view (Doorley et al. 2018, p. 7f).

---

### *5.3.3.15 Insights (DT Artifact #15)*

*Description:* 'Insights' ("aha moments") are findings that occur when the project team can suddenly link and interpret certain observations made (IDEO LLC 2012, p. 45). The team will then understand what lies at the heart of the behavior and statements they have observed. An insight is usually expressed in one sentence to explain why something is happening.

*Purpose:* Insights extrapolate individual user stories into overarching "truths". They are produced as a result of synthesis and interpretation and provide the basis for generating new solution ideas. Insights can stem from different findings, for example the project team suddenly detects something surprising that has always been missing, or they finally understand why people are behaving in a certain way, or they can suddenly explain a contradiction.

---

*Example: Insights*

---


**Picture 10: Example of Insights**

**(Source: ITMP 2018)**

Picture 10 illustrates several insights about healthy food. The insights relate to thematic clusters as they describe the finding behind a certain theme (e.g. "Eating is not about food, but about the location" or "Health is an individual reality"). They are phrased as an overarching truth connecting different thematic clusters or as a need statement from the user's point of view (e.g. "Health-conscious student needs a way to reduce plastic when shopping but his financial budget does not always allow it").

---

*Interdependencies:* Insights get their input from field studies (#11) and thematic clusters (#12). They provide the basis to define opportunity areas (#16) and create new solution ideas (#17).

*Notation:* Insights are usually specified by structured natural text.

#### *5.3.3.16 Opportunity Areas (DT Artifact #16)*

*Description:* 'Opportunity areas' describe "fields", in which the team sees the potential for innovation based on the insights and needs found in primary research (IDEO LLC 2012, p. 46-47). It is rare that only one opportunity area would be identified; instead it is usually a group of areas.

*Purpose:* Opportunity areas are the catalyst for generating new ideas as they define specific directions for next steps. At the same time, they offer high strategic value because they indicate the innovation potential for the future. Those opportunity areas often go beyond the project assignment itself.

*Interdependencies:* Opportunity areas are formulated on the basis of thematic clusters (#12) and insights (#15) as well as personas (#13) and customer journeys (#14). They provide the basis for generating new solution ideas (#17). The boundaries between the formulated insights (#15) and the opportunity areas are often blurred. The actual difference consists mainly in the way they are articulated, i.e. the formulation of opportunity areas is rather action-oriented, while the insights describe the status quo or a desired future state.

*Notation:* Constraints and rules are usually specified by natural text.

*Example: Opportunity Areas*


**Figure 28: Example of an Opportunity Area**

**(Source: ITMP 2015)**

Figure 28 shows an exemplary formulation of an opportunity area in the context of enhancing decision-making for employees of an agricultural company. The project team took a holistic perspective on all the thematic clusters and insights to define opportunity areas. The team askes themselves: "Where do we see opportunities for innovation?" and aggregated some of the insights into opportunity areas. The formulation should sound like an appeal and a call for action. It can be useful to provide a description of the exact pains, needs, and goals for each opportunity area as shown above. The opportunity areas are complemented with so called "How might we"-questions, in order to offer concrete directions for ideating solutions in the specific opportunity area (Berger 2012).

#### *5.3.3.17 Solution Ideas (DT Artifact #17)*

*Description:* 'Solution ideas' are a list of concrete ideas in the context of the given project. Techniques that are often applied to achieve specific solution ideas are brainstorming, brainwriting, six thinking hats, power of ten, and other creativity methods to generate a large quantity of diverse ideas (Uebernickel et al. 2015, pp. 138-

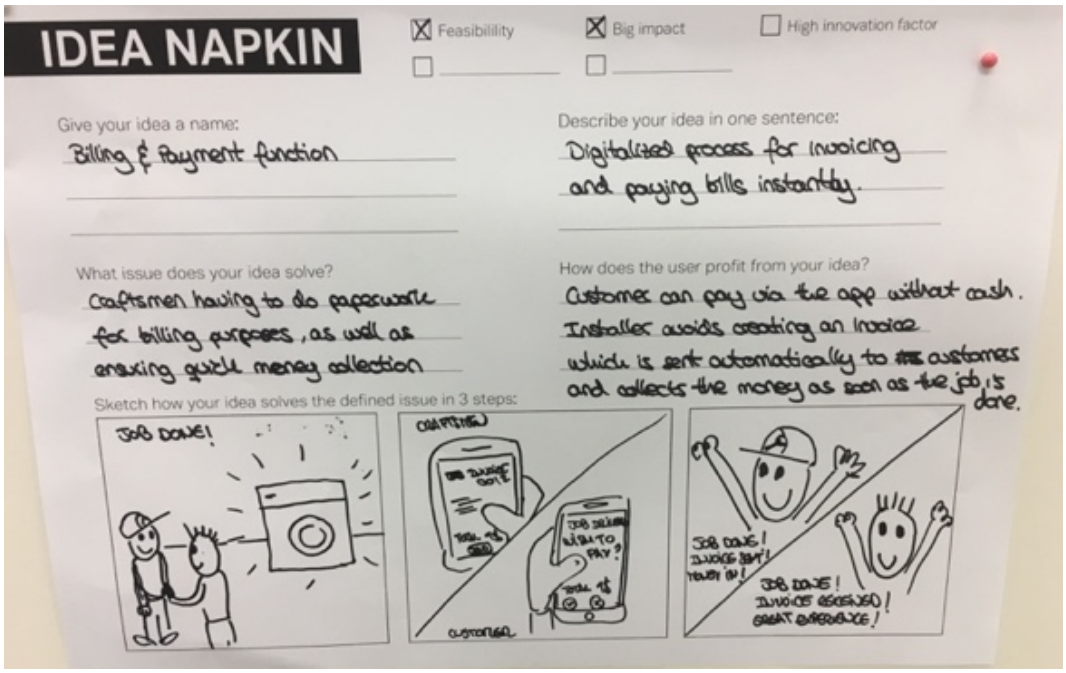
144). A strong focus lies on listening and building on each other's ideas to leverage collective thinking.

*Purpose:* Solution ideas provide specific features and concepts on how to solve a given problem statement.

---

*Example: Solution Ideas*

---




**Picture 11: Example of a Solution Idea**

**(Source: ITMP 2017)**

Example of an idea napkin for a German utility company to improve processes for craftsmen. The idea napkin is a tool that helps to spell out each solution idea to better understand and concretize them for better group evaluation. Ideally, all idea napkins are stuck to a wall so that everybody can see them. The napkin prompts team members to provide a name for the idea, to describe it in one sentence, to think of the problems solved, and to highlight the benefits generated by the idea. The idea napkins provide a certain level of comparability of the solution ideas, when discussing next steps.

---

*Interdependencies:* Solution ideas typically address a selected opportunity area (#09) and provide the basis for building a variety of prototypes (#18, #20, #22) and testing them with users and customers (#19, #21, #23) in order to define the system vision (#24).

*Notation:* 'Solution ideas' are usually specified by natural text.

#### *5.3.3.18 Low-fidelity Prototypes (DT Artifact #18)*

*Description:* 'Low-fidelity prototypes' are tangible and testable artifacts that demonstrate the key functionalities of an idea rather than its visual appearance (Shneiderman et al. 2018). They can take different forms, for example paper prototypes to demonstrate screens, features, and functionalities, role plays and Wizard of Oz prototypes to test interactions and behavior, or storyboards to narrate and test a user experience through a series of images (see Uebernickel et al. 2015, pp. 156, 160-163; Doorley et al. 2018, p. 26). The selection of different prototyping methods can be subsumed under the common principles of *right*, *rapid*, and *rough* (Crai 2003). Depending on what a team aims to achieve with the prototype, e.g. to test an experience or certain features with customers, to convince decision-makers, or to communicate an idea within the project team, the team should choose the *right* and most suitable prototyping technique. Prototypes should never take too much effort in the early project phase. Quickly implementing an idea or *rapidly* iterating it based on feedback is an effective way of bringing ideas to life efficiently. This way of rapid implementation has a bearing on a prototype's level of detail which brings out crude prototypes with the watchword "just good enough" to answer the questions at hand. Once the idea has been tested with a couple of iteration loops, the 'low-fidelity prototypes' can be further developed in more detail.

*Purpose:* Low-fidelity prototypes bring ideas to life and support early-stage learning by providing the fastest-possible way to iterate ideas (Liedtka and Ogilvie 2011). Costs and effort are extremely low, which allows the project team to explore various ideas at once. In addition, this type of prototyping stimulates group ideation and communication within the team and with other relevant stakeholders. Since it doesn't require special skills (e.g. programming or design), multiple people can be integrated into the idea-formulation and design process. Low-fidelity prototyping is particularly suitable during the early stages of a project, when the topic is still abstract or in the process of forming. It finds its boundaries when visually complex and complicated operations are necessary to convey. The further the team gets into the design process, mid- and high-fidelity prototypes (#20, #22) are considered more suitable (Cooper et al. 2014).

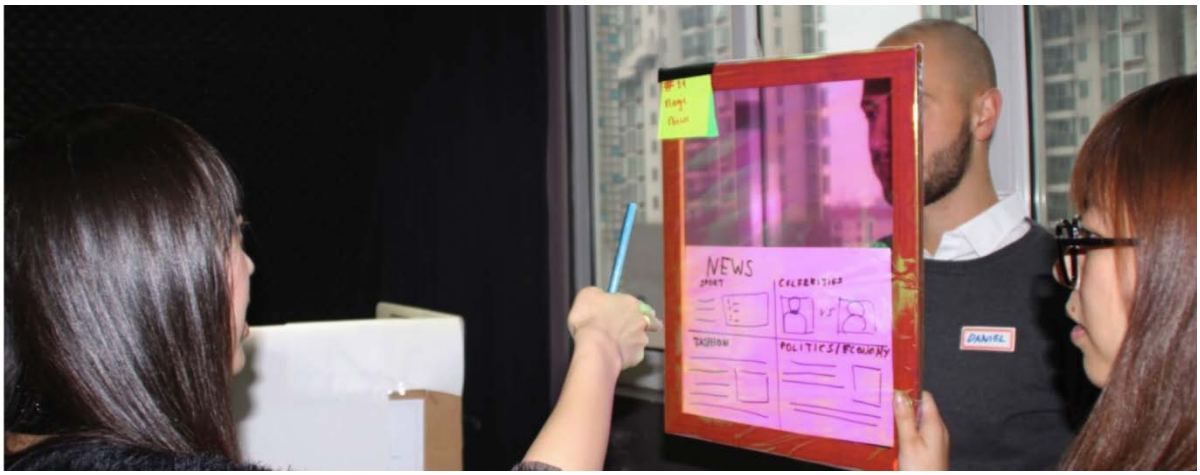
*Interdependencies:* Low-fidelity prototypes realize solution ideas (#17) in a quick and easy way. They provide the input to gain scope-related test results (#18) to create medium-fidelity prototype (#20).

*Notation:* Low-fidelity prototypes can be expressed in different forms, most often in a paper-based format.

---

*Example I: Low-fidelity Prototypes*

---

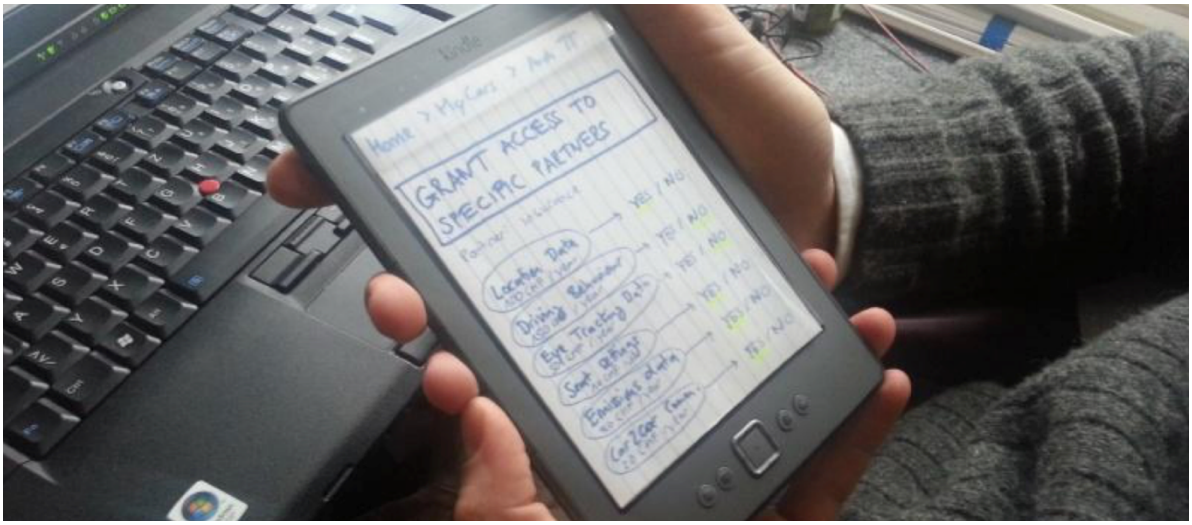



**Picture 12: Example of a Wizard of Oz Prototype**
**(Source: University of St.Gallen, DT@HSG 2016)**

A Swiss Bank needed to test user reactions to their avatar for online banking advisory. Instead of coding, a person acted as the avatar, starting to speak when the user pushed certain buttons as shown in Picture 12. The Wizard of Oz prototype can also be extended beyond physical prototypes. For example, a vending machine can be tested without building the mechanics by using a person hidden inside it to deliver purchases (Doorley et al. 2018, p. 27).

---

*Example II: Low-fidelity Prototypes*




**Picture 13: Example of a Paper Wireframe Prototype**
**(Source: University of St.Gallen, DT@HSG 2016)**

An information provider was rethinking their services for the independent automotive aftermarket. The prototype, as shown in Picture 13, visualized the key features of the new service to gain access to a wider partner network of garages. This idea was realized in form of paper wireframes with several sheets of paper (representing different screens) fixed to an actual device. This way the prototype became interactive by integrating a facilitator during the testing session whose task included to jump to the right screens depending on the user's chosen strategy.

#### *5.3.3.19 Scope-oriented Test Results (DT Artifact #19)*

*Description:* 'Scope-oriented test results' are gathered based on the feedback from users. The basic concept of an idea is in focus of the evaluation. A feedback framework can be used to process feedback interviews and structure test results. It helps the team to decide what to keep, what to refine, and what to drop (Uebernickel et al. 2015, p. 210).

*Purpose:* Scope-oriented test results help the project team to challenge the current value proposition and show whether customers' needs and expectations are met appropriately. Based on the test results the project team can refine their prototype(s) and also the understanding of the people for whom the team designs. In these early stages, the project team usually not only considers the feedback about the solution but also uses the opportunity to gain more empathy for their target group. Thus, scope-oriented test results also include new insights about users and novel inspiration for the solution scope.

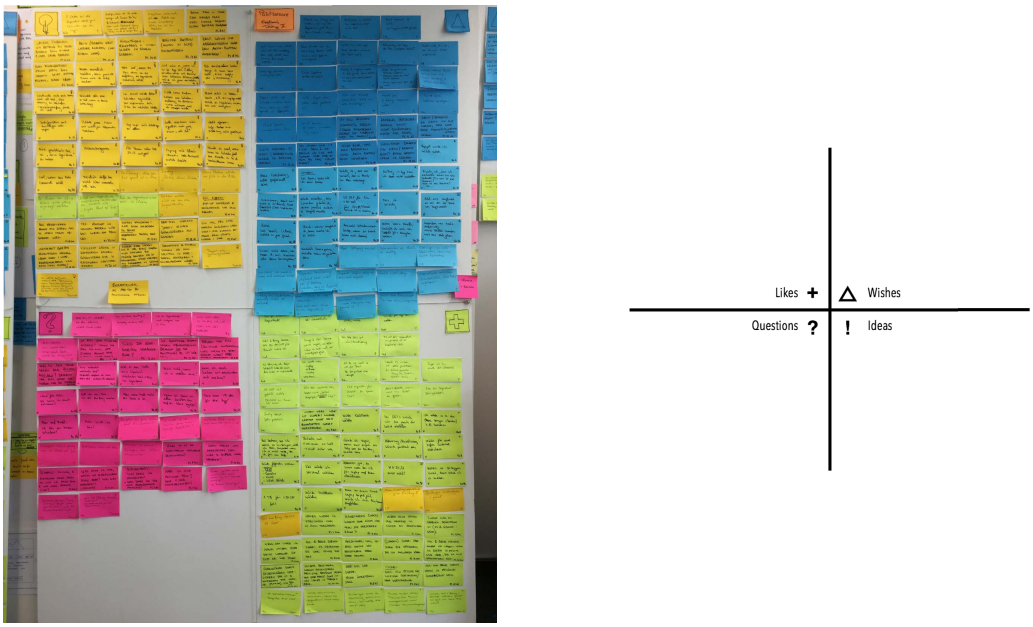
*Interdependencies:* Scope-oriented test results are linked to low-fidelity prototypes (#18). Their content provides input for medium and high-fidelity prototypes (#20, #22).

*Notation:* Scope-oriented test results are usually specified by structured natural text.

---

*Example: Scope-oriented Test Results*

---




**Picture 14: Example of a Feedback Capture Grid**
**(Source: ITMP 2018)**

Picture 14 illustrates an example of transcribed scope-oriented test results from an international insurance company. The feedback capture grid was used as a framework to discuss the test results on the general concept and scope of several tested low-fidelity prototypes. The grid arranges the feedback according to four dimensions: (1) *plus* (green post-its): aspects that received positive feedback, (2) *delta* (blue post-its): constructive criticism and change suggestions, (3) *question mark* (pink post-its) questions that have come up and should be addressed, (4) light bulb (yellow post-its): ideas from interview partners during feedback. The grid helped the project team to decide which prototypes should be changed in the next iteration, which new ones should be built, and which ideas did not require further development.

---

### *5.3.3.20 Medium-fidelity Prototypes (DT Artifact #20)*

*Description:* A 'medium-fidelity prototype' is a non-technical prototype with the key features of the target product or service. Similar to the low-fidelity prototype (#18) it is

still rudimentary in order to test these features in a basic design language. A somewhat realistic and click-through prototype can be achieved through hyperlinking PowerPoint or Keynote slides or by connecting wireframes with specialized digital prototyping tools such as Prott (https://prottapp.com/) or POP Prototyping (https://marvelapp.com/pop) (Babich 2017).

*Purpose:* While early stage, low-fidelity prototypes (#18) are useful to inspire new ideas, medium-fidelity prototypes are mainly used to test and refine existing solution ideas (#17) (Hartson and Pyla 2012). In comparison to low-fidelity prototypes (#18) they usually take more effort to build, yet also provide a much more realistic representation of the envisioned behavior and user interface (Hartson and Pyla 2012).

---

*Example: Medium-fidelity Prototypes*

---


**Figure 29: Example of a Medium-fidelity Prototype**
**(Source: University of St.Gallen, DT@HSG 2016)**

Clickable wireframes for a German insurance provider are visualized in Figure 29. The team used a combination of PowerPoint slides and Prott to interlink the slides for interactivity and swiping functions. The app was created to improve the feedback culture in the organization. Managers could provide short feedbacks in real-time on the performance of their employees. As a form of visual feedback, not just text but also emojis could be sent.

---

*Interdependencies:* A medium-fidelity prototype gets its input from solution ideas (#17), low-fidelity prototypes (#18) and their respective scope-oriented test results (#19). Its content provides the basis for generating feature-oriented test results (#21) and creating high-fidelity prototypes (#22).

*Notation:* Medium-fidelity prototypes can be expressed in different forms, most often in a digital format.

#### *5.3.3.21 Feature-oriented Test Results (DT Artifact #21)*

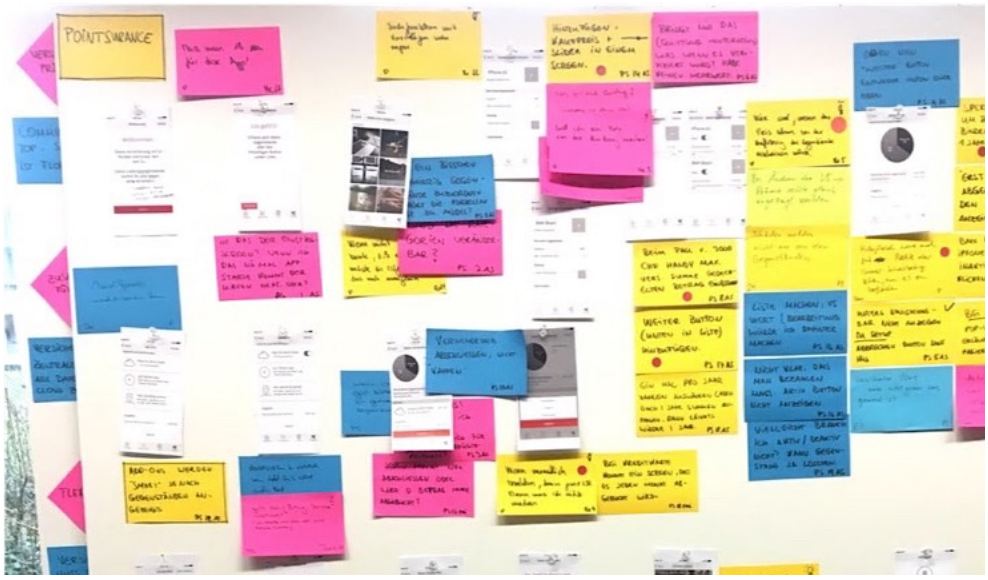
*Description:* 'Feature-oriented test results' encompass the aggregated feedback from users and other relevant stakeholders regarding key features and functionalities of the prototype. They validate the customer's expectations of a solution and its added value before cost-intensive implementation begins (Hartson and Pyla 2012).

*Purpose:* The results provide indications on where to refine the concept of the idea and a direction for further adjustments that have to be made in upcoming iteration loops. Feature-oriented test results also help to prioritize functionalities for the implementation timeline. Initial obstacles can be eliminated and challenges can be identified and tackled at an early stage.

*Interdependencies:* Feature-oriented test results are most often linked to medium-fidelity prototypes (#20). Their content provides input for creating a high-fidelity prototype (#22) and specifying usage and service models (#25, 26).

*Notation:* Feature-oriented test results are usually specified by natural text.

*Example: Feature-oriented Test Results*




**Picture 15: Example of Feature-oriented Test Results**
**(Source: ITMP 2018)**

Picture 15 shows an example of transcribed feature-oriented test results from an international insurance company. The team had created and tested a clickable medium-fidelity mockup with PowerPoint and Prott. Printing out the screens and assigning relevant feedback to each wireframe (on post-its) helped the team to proceed with the relevant features and revise the prototype. Color-coded post-its supported the team in distinguishing between new ideas and positive and negative feedback.

#### *5.3.3.22 High-fidelity Prototypes (DT Artifact #22)*

*Description*: A 'high-fidelity prototype' shows how the final system will look and feel (Walker et al. 2002). A project team usually creates such a prototype when they have a profound understanding about the key features and functionalities, the appearance, and user experience (Beaudouin-Lafon and Mackay 2003). A high-fidelity prototype includes all the content in a realistic interface and interaction design. In the context of software-intensive systems, digital prototypes are the most common used form. A variety of specialized software allows the project team to build a visually rich and detailed prototype with interactive effects and complex animations (e.g., InVision, Axure, Adobe XD) (e.g. Babich 2017).

*Example: High-fidelity Prototypes*

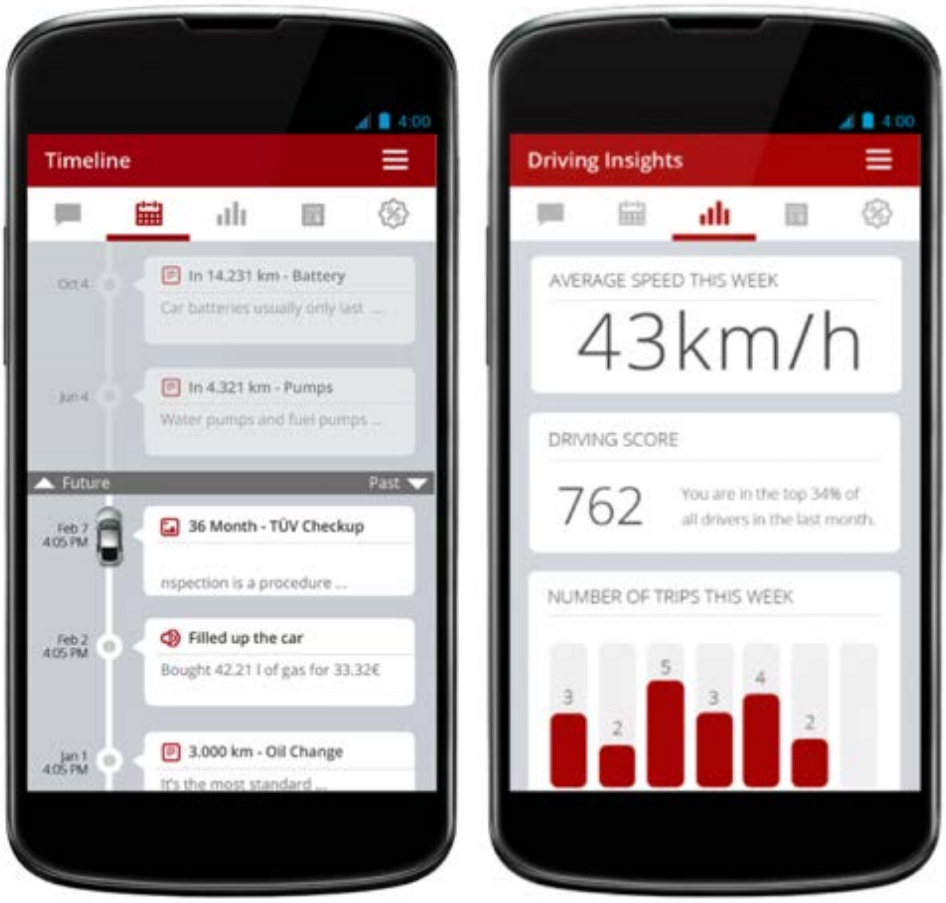



**Figure 30: Example of a High-fidelity Prototype**
**(Source: University of St.Gallen, DT@HSG 2014)**

Figure 30 shows a high-fidelity prototype that helps car owners manage their cars in a convenient way. The solution allows customers to manage their repair history, receive repair reminders with optional offers, receive vouchers from partner companies, as well as understand driving behavior and total driving costs. By including relevant stakeholders – repair shops, gas stations, car service providers and insurance companies – the solution is a one-stop personal assistant for every car owner. The functional high-fidelity prototype portrays the look and feel of the intended final solution.

*Purpose*: High-fidelity prototypes help the project team to gain meaningful feedback for usability testing. As these prototypes often look and feel like real solutions, users are more likely to interact with them naturally, just like with a real product (Babich 2017). This kind of prototype is also suitable to gain buy-in from clients and internal project

stakeholders. The prototype offers a clear vision of how the solution is supposed to work. However, due to the detailed work that has gone into the prototype, the project team might also feel reluctant to change designs based on negative feedback and less likely to fully explore the design space (Walker et al. 2002).

*Interdependencies:* A high-fidelity prototype gets its input from solution ideas (#17), low- and medium-fidelity prototypes (#18, #20) and their respective scope-and feature oriented test results (#19, #21). Its content provides the basis for generating usability-oriented test results (#22) and specifying the system vision (#24).

*Notation:* High-fidelity prototypes can be expressed in different forms, most often in a digital format.

#### *5.3.3.23 Usability-oriented Test Results (DT Artifact #23)*

*Description:* 'Usability-oriented test results' provide information about the user's interaction with a product with the goal of assessing the usability of that product. Typically, a test report includes a background summary (what was tested and why), the methodology (how was the test conducted and with whom), test results (quantitative and qualitative metrics like satisfaction, task completion rates, average time taken, or comments), findings (positive and negative), and recommendations on how to proceed (Cooper et al. 2014, pp. 70-71).

*Purpose:* Usability-oriented test results, at their core, provide information about the effectiveness of ideas; they do not necessarily establish completely new ideas. Usability testing emphasizes the measurement of how well users can operate specific tasks and the respective problems they encounter in doing so. The results provide areas for improving issues of understandability and point at directions for refining particular design elements and interaction mechanisms (Cooper et al. 2014, p.143).

*Interdependencies:* Usability-oriented test results are most often linked to high-fidelity prototypes (#22). Their content provides input for the system vision (#24) and usage model (#25).

*Notation:* 'Usability-oriented test results' are usually specified by structured natural text. Visual content like screen shots or video clips can support the explanatory power of the findings.

*Example: Usability-oriented Test Results*


**Figure 31: Examples of Usability-oriented Test Results (Source: ITMP 2018)**

The usability test report about a new digital platform of a German utility organization included a walkthrough by users, a track and test discovery-process and an emotional journey of testers, track and test functions and a functional journey (incl. usability and navigation), and qualitative feedback on usability and content to identify major bugs and conceptual mistakes. The heat map in Figure 31 led to the following finding and recommendation: "Users pay a lot of attention to the value proposition bubbles and often attempt to click them repeatedly, even though they are not clickable. Currently, the only possible interaction is the registration. Recommendation: Value proposition bubbles should be interactive and/or provide more information on the underlying functionality to provide a pleasant entry point to the platform." (ITMP 2018)

#### *5.3.3.24 System Vision (DT and RE Artifact #24)*

*Description:* The 'system vision' is the specification of how an information system is to fit into the business context while supporting pre-defined restrictions and goals (Méndez Fernández and Penzenstadler 2014a, p. 37). It is usually agreed upon by all active stakeholders and defines the big picture of the system vision (Monk and Howard 1998).

*Purpose:* The system vision serves as a means for agreeing on what the solution is about. It eases communication with stakeholders about the scope and the goals of the final

system. While the purpose of the system vision is similar to both Design Thinking and Requirements Engineering, its realization might be different. In Design Thinking it is usually comprised of a high-level natural text specification for the solution including its main intent and goals. It is usually phrased in 2-3 sentences. It is often accompanied by a medium-or high-fidelity prototype (#20, #22) and a high-level description of the business model of the solution (see example below). In Requirements Engineering the system vision is often expressed via rich picture.

*Interdependencies:* The system vision summarizes the business context from artifacts like business case (#03), stakeholders (#04), goals (#05), and the domain model (#06). In addition, it takes the form or input from the high-fidelity prototype (#22). The content of the 'system vision' provides the input for the usage model (#25), service model (#33), and risks (#31).

*Notation:* The system vision can be specified by rich picture, prototype, or structured natural text.

---

*Example: System Vision in Requirements Engineering*

---

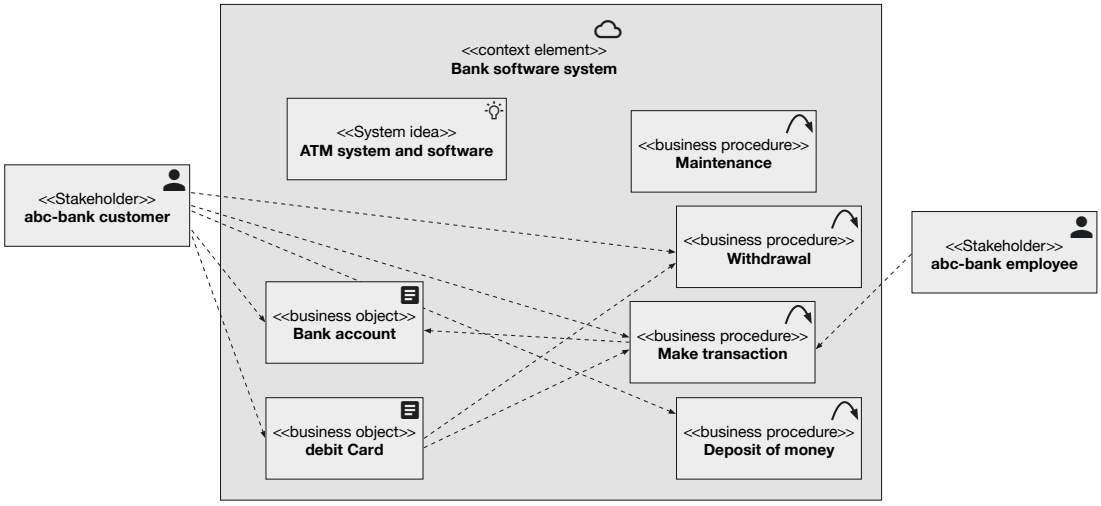



**Figure 32: Example of a System Vision in RE**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 32 illustrates a system vision of an ATM ("ATM system and software") that denotes the system border, the most central business procedures like "withdrawal", "make transaction", "deposit of money", the affected stakeholders, and business objects. The visualization is done via rich picture (Monk and Howard 1998).

---

*Example: System Vision in Design Thinking*

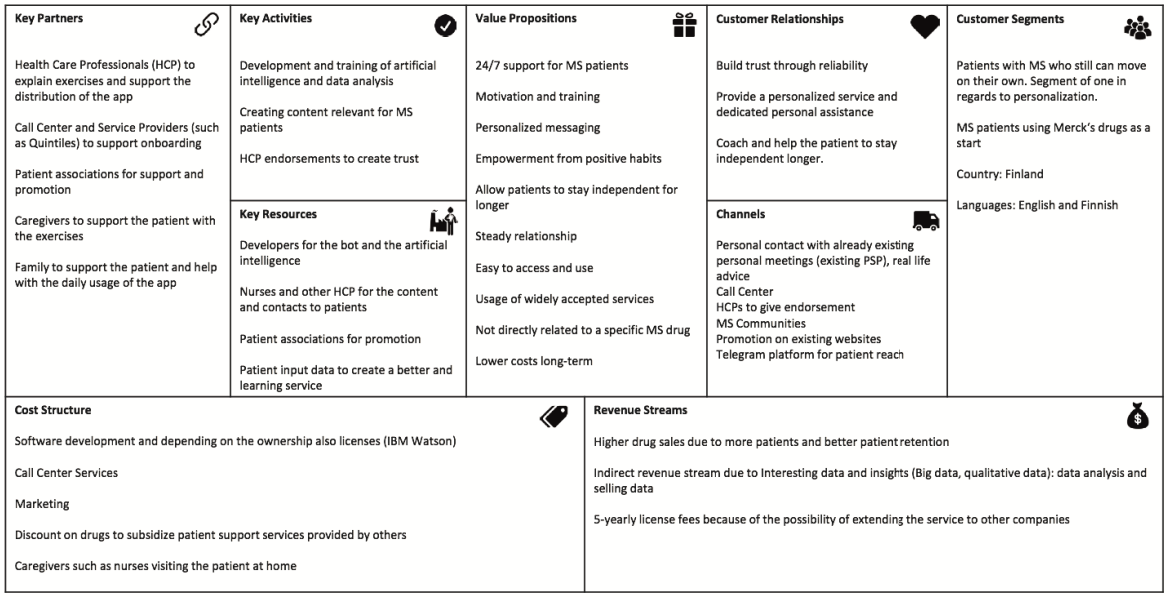



**Figure 33: Example of a System Vision in DT**
**(Source: University of St.Gallen, DT@HSG 2016)**

In Design Thinking the system vision is frequently phrased as a value proposition as a part of a (high-level) business model description (e.g. "24/7 support for MS patients through personalized motivation and training"). The Business Model Canvas as shown in the example is one of the most popular tools to describe a potentially successful operation of a business (Osterwalder and Pigneur 2010). It considers nine building blocks, including value proposition, revenue sources, customer base, key resources and activities, and details of financing.

### *5.3.3.25 Usage Model (DT and RE Artifact #25)*

*Description:* The 'usage model' illustrates the (black box) system behavior of the system vision (#24) from the user's point of view through an overview of use cases (Méndez Fernández and Penzenstadler 2014a, p. 37). A use case contains a set of interaction scenarios between the actors and the intended system (Cockburn 2000). It includes a task, an objective, and a causal relationship (pre- and post-conditions) and may be represented visually in UML. At least one functional scenario can be defined for each use case, in which actors participate who are triggered by certain events. A scenario stems from a requirement and is described by actor and system actions each processing data objects (Méndez Fernández and Penzenstadler 2014a, p. 37).

*Purpose:* The main purpose of the usage model is to provide an understanding about which system functions are performed for which actors (in their roles). The graphical overview depicts actors, their goals (represented as use cases), and any dependencies between those goals. In Requirements Engineering a 'usage model' typically consists of a use case overview diagram, a use case template, and a scenario diagram (Méndez Fernández and Penzenstadler 2014a, p. 37). In Design Thinking the usage model is partially covered through the specification of high-fidelity prototypes (#22) which describe concrete use cases. The purpose of the artifact types from both approaches is similar, however, most often they are executed on different abstraction levels.

---

*Example: Usage Model*

---

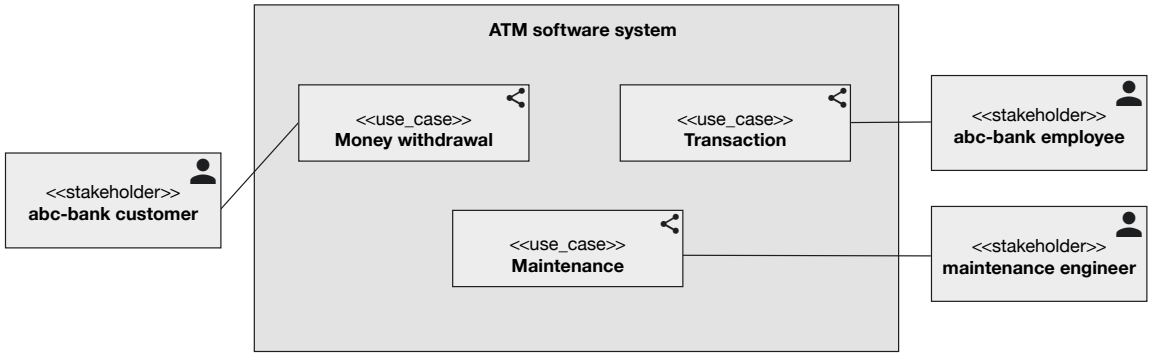



**Figure 34: Example of a Usage Model**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 34 illustrates a simplified usage model for using an ATM software system. The associations indicate that a stakeholder is involved with a use case. The following three use cases are visualized: (1) Stakeholder *abc-bank customer* withdraws money, (2) stakeholder *abc-bank employee* is involved in a transaction, (3) stakeholder *maintenance engineer* is maintaining the ATM software system.

---

*Interdependencies:* The usage model gains its input from the elaboration of personas (#13) and their customer journeys (#14) as well as from the specification in the system vision (#24) and service model (#26). The output of the usage model provides the basis for the functional hierarchy (#28), data model (#29), and quality requirements (#33).

*Notation:* The 'usage model' can be specified by structured text or UML activity diagrams.

#### *5.3.3.26 Service Model (DT and RE Artifact #26)*

*Description:* The ‘service model’ specifies the requirements and objectives of the intended services of the solution. Services are understood as a representation of user-visible functions through input/output-relations (Hummel and Thyssen 2009). The quality of the service is described by using service parameters that correspond to metrics from the quality requirements (#33) that inhabit particular service levels (Méndez Fernández and Penzenstadler 2014a, p. 37).

*Purpose:* The service model provides a comprehensive understanding of the services and their underlying resources and processes, whether seen or unseen by the user. For the project team it serves as a means to structure complex scenarios with multiple service-related offerings and often bridging cross-departmental efforts. In addition, the model can be used to integrate experts into the analysis and development process (to inform and gain feedback) and to orchestrate the implementation of the service (provide guidance on how different components should be used) (Bitner et al. 2008). The service model covers the same semantic meaning for both Design Thinking and Requirements Engineering, however, it can be expressed through different means as described below.

*Interdependencies:* The service model gains its input from the specification in the system vision (#24) and provides input for the usage model (#25), data model (#29), and quality requirements (#33). Depending on the project, the ‘service model’ itself can be composed by different artifacts, especially when using Design Thinking. For example, it can be expressed through a high-fidelity prototype (#22), customer journeys (#14), or dedicated tools like the service design blueprint (see example below). The service blueprint depicts the relationship between different service components – customers, objects (physical or digital evidence), and processes – that are directly tied to touchpoints in a specific customer journey (Bitner et al. 2008). These blueprints can also be used for problem analysis to discover weaknesses and opportunities for optimization or redesign.

*Notation:* The service model can be specified by structured text or graphs. Each step in the service experience can be enriched with drawings, pictures, or any other material that supports the understanding.

*Example: Service Model in Design Thinking*

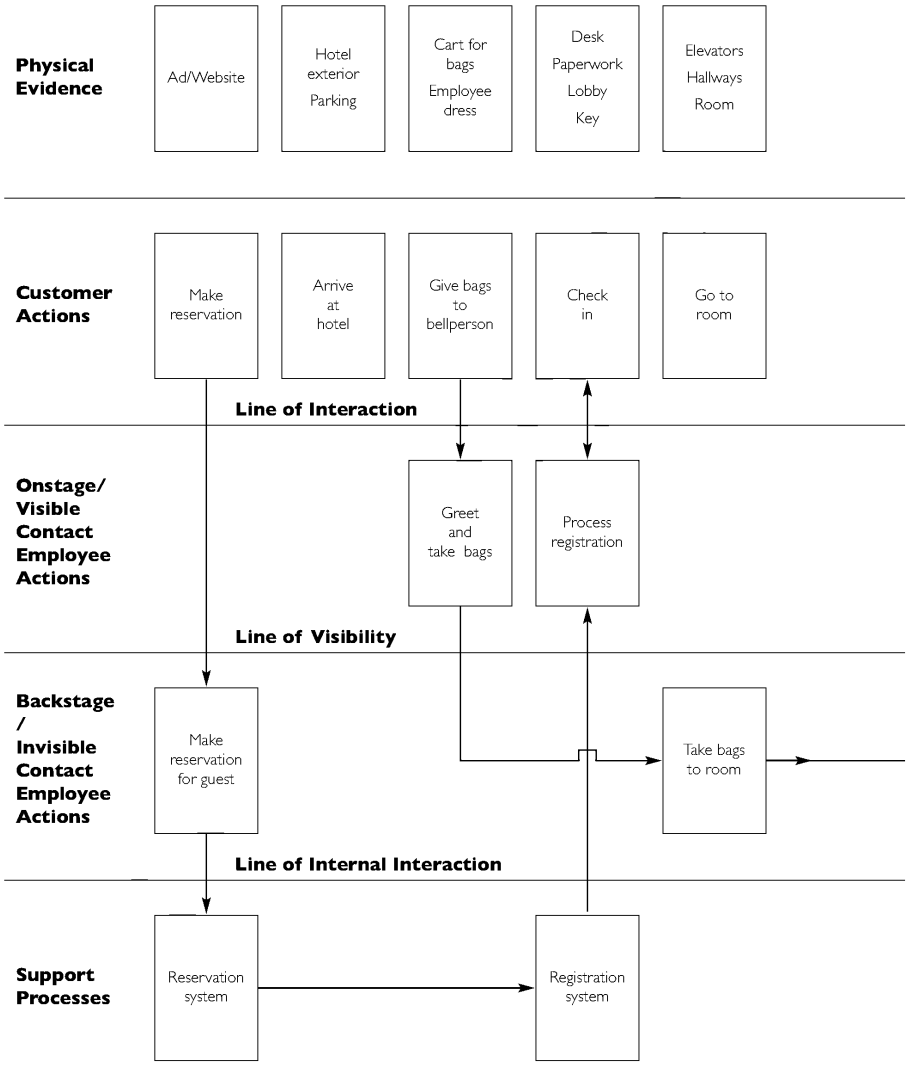



**Figure 35: Example of a Service Model in DT**
**(Source: Bitner et al. 2008, p. 76)**

The service design blueprint of a hotel stay is partially shown in Figure 35. It is structured according to a horizontal axis (actions performed by a customer and provider in chronological order) and a vertical axis (elaborates different areas of actions) (Bitner et al. 2008). The latter is divided into five rows such as *physical evidence* (touchpoints of the customer), *customer actions* (to reach a specific goal, e.g. make a reservation), *onstage actions* performed by the service provider (human-to-human or human-to-computer, e.g. check in), *backstage actions* (behind the scenes activities, e.g. take bags to room), and *support and managerial processes* on different layers (internal steps and interactions that support delivering the service, e.g. reservation process).

*Example: Service Model in Requirements Engineering*

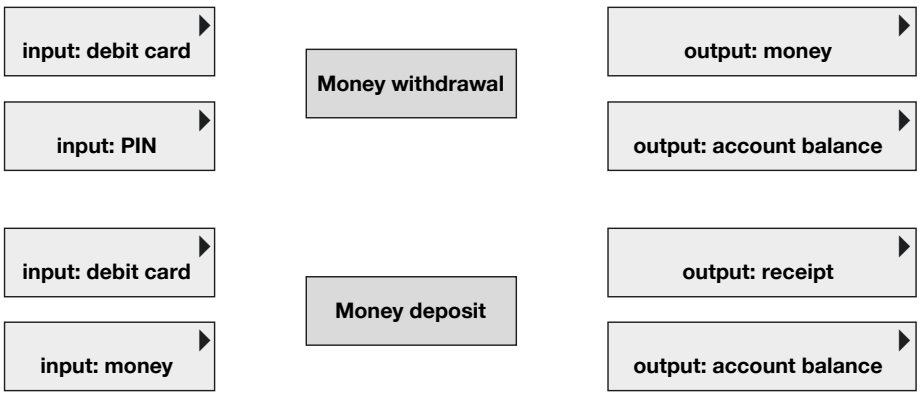



**Figure 36: Example of a Service Model in RE**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

The simplified service model of an ATM is specified by input-output relationships (**Fehler! Verweisquelle konnte nicht gefunden werden.**). For *money withdrawal*, for example, the defined input variables are "debit card" and "PIN" and the output variables are "money" and "account balance".

#### *5.3.3.27 Process Requirements (RE Artifact #27)*

*Description:* 'Process requirements' specify activities that should be performed by the developing organization. For example, they can determine the compliance to standards and process models that must be followed, the milestones of a project that must be kept, style-guides that must be followed, or the infrastructure that must be used (Méndez Fernández and Penzenstadler 2014a, p. 40).

*Purpose:* Process requirements provide the guidelines for a consistent design and implementation of the intended system.

*Interdependencies:* Process requirements are influenced by the specifications in the design challenge /project scope (#01).

*Notation:* Process requirements are usually specified by natural text.

*Example: Process Requirements*

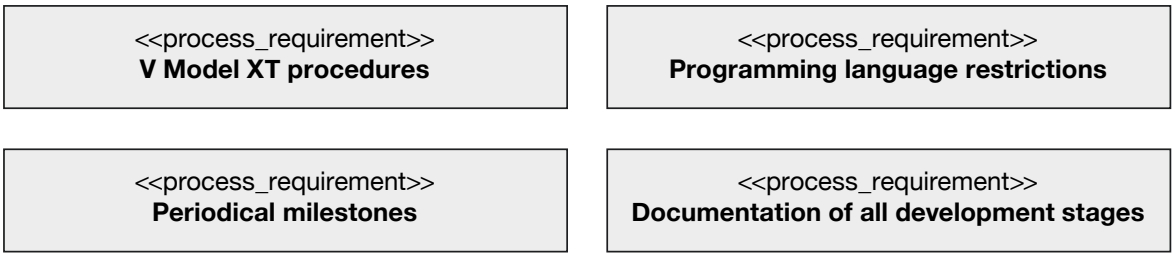



**Figure 37: Example of Process Requirements**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 37 lists a set of process requirements for a given project. It includes procedure requirements to be followed (V Modell XT procedures), the programming language restrictions to be obeyed, the timetable to be kept (periodical milestones), and the documentation that is required at all development stages. The process requirements are specified by natural text.

#### *5.3.3.28 Functional Hierarchy (RE Artifact #28)*

*Description:* The 'functional hierarchy' specifies functions and subfunctions and their respective relationships and dependencies (e.g. communication relation, task and subtask, timely orders). Functions are user-visible pieces of the system behavior. They correspond to services from the service model (#26) and realize the system actions from the usage model (#25) (Méndez Fernández and Penzenstadler 2014a, p. 40).

*Purpose:* The purpose of a functional hierarchy is to bridge the requirements and system specification as it provides a syntactic interface, determines logical components, and specifies behavior models. In a set of data flow diagrams, the functional hierarchy can be used as a guideline for obtaining and organizing system requirements.

*Interdependencies:* In the artifact model the functional hierarchy, together with the usage (#25) and service model (#26), provides the basis for the system specification, including a detailed data model (#29), function model (#36), component model (#38), and behavior model (#39).

*Notation:* The functional hierarchy is usually specified through graphs and input-output tables.

*Example: Functional Hierarchy*

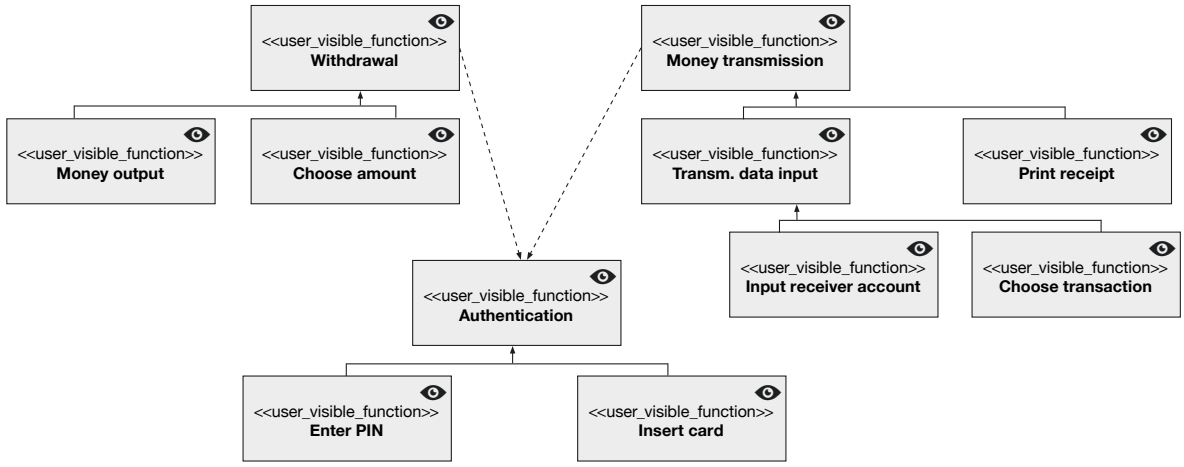



**Figure 38: Example of a Functional Hierarchy**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 38 shows an example of a functional hierarchy of user-visible functions for an ATM. User-visible functions include, for instance, "withdrawal" (= *function*) with its *subfunctions* "money output" and "choose amount". A relation can be defined between "withdrawal", and "authentication" (e.g. "a withdrawal *requires* authentication").

#### *5.3.3.29 Data Model (RE Artifact #29)*

*Description:* The 'data model' in the requirements layer summarizes all data objects and relations that are part of the system's functions and interaction scenarios (Méndez Fernández and Penzenstadler 2014a, p. 39).

*Purpose:* The data model supports the development of the intended system by providing the definition, format, and structure of the required data.

*Interdependencies*: The system functions for the data model are derived from the usage model (#25), service model (#26), and the functional hierarchy (#28). The elements of the data model in the requirements layer provide input for refining the data model (#37) in the system layer.

*Notation:* The data model is usually specified in a data modeling notation such as UML class diagrams.

*Example: Data Model*

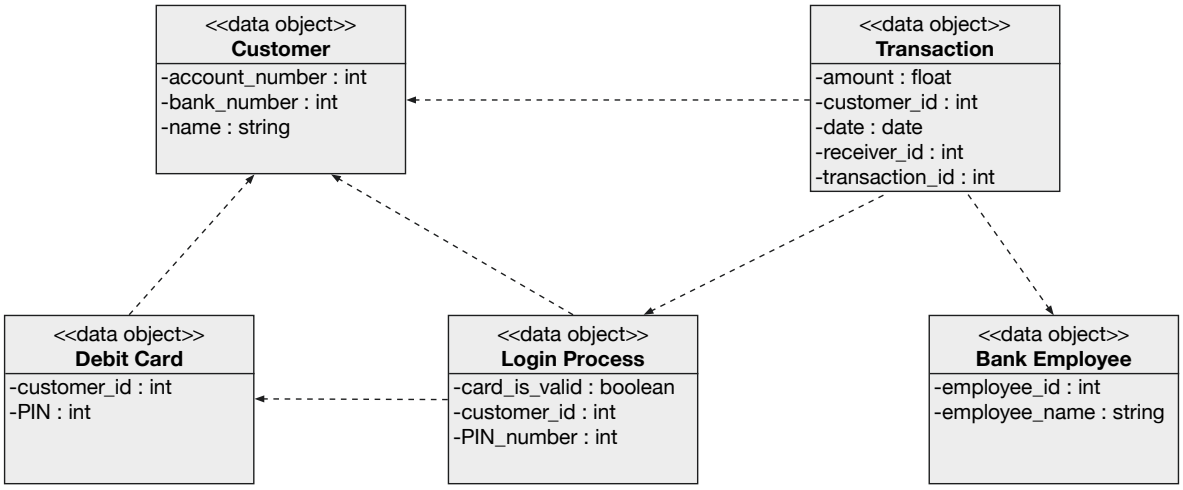



**Figure 39: Example of a Data Model in the Requirements Specification (Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 39 shows a data model excerpt for an ATM machine. It illustrates the data objects "Customer" class, "Transaction" class, and "Debit Card" class that are specified by their respective attributes. In this example, the "Customer" class is depicted via the attributes "account number", "bank number", and "name". The relationship among the classes are visualized through arrows. As the picture illustrates, the arrows indicate a unidimensional dependency between the classes (e.g. "Login Process" class depends on the "Customer" class).

#### *5.3.3.30 Deployment Requirements (RE Artifact #30)*

*Description:* 'Deployment requirements' describe the demands for making the software available for use, i.e. specifying the process of the deployment and the technical infrastructure during the initial release of the system or specific parts of it (Méndez Fernández and Penzenstadler 2014a, p. 40). They relate to the system's non-functional requirements.

*Purpose*: Deployment requirements contribute to the overall quality of the resulting system (Kupfer and Hadar 2008). They provide the basis for decision-making about the resulting system's non-functional properties and possible design trade-offs.

*Interdependencies:* Deployment requirements are influenced by the specifications in the design challenge/project scope (#01).

*Notation:* Deployment requirements are usually specified by natural text.

*Example: Deployment Requirements*

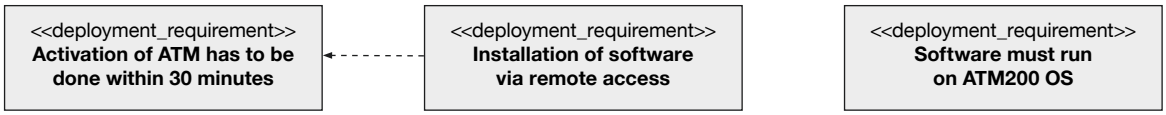



**Figure 40: Example of Deployment Requirements**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 40 shows three examples of deployment requirements for an ATM. They include, for instance, "the activation of the ATM has to be done within 30 minutes" or "the software must run on ATM200 OS". The requirements are phrased in natural text and can be put into a prioritized order. The specific customer requirements for deployment such as critical project deadlines, financial constraints, and other resource restrictions need to be taken into account when defining those deployment requirements.

#### *5.3.3.31 Risk List (RE Artifact #31)*

*Description:* The 'risk list' includes a description of all risks that stem from the various requirement types and potentially threaten the development or operation of a system. Risks are typically analyzed along stakeholder interests and estimated regarding their probability and potential damage (Glinz 2014, p. 19). They are used as a link to risk management.

*Purpose:* A comprehensive risk list helps to evaluate and judge the criticality of a system (Venkatesh Sharma and Kumar 2013, p. 250). It provides the foundation to introduce necessary countermeasures.

*Interdependencies:* The risk list is influenced by the specifications in the 'system vision' (#24).

*Notation:* The risk list is usually specified by natural text.

*Example: Risk List*

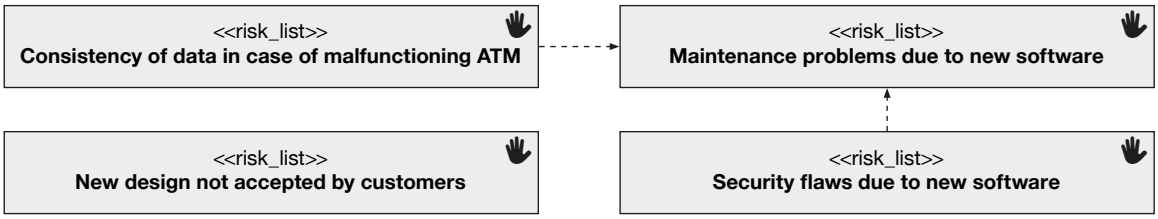



**Figure 41: Example of a Risk List**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 41 visualizes risks for an ATM. They include, for example, "consistency of data at malfunctioning ATM", "maintenance problems due to new software", or "new design not accepted by customers". The risks are phrased in natural text and a relationship between them is typically established. Each risk is caused by a risk factor. Risks have to be identified, assessed, and analyzed, so that effective actions can be introduced early on.

#### *5.3.3.32 System Constraints (RE Artifact #32)*

*Description:* 'System constraints' are logical and technical restrictions for the system architecture, its functionality, and quality (Méndez Fernández and Penzenstadler 2014a, p. 40). They do not necessarily result from a functional goal. System constraints are considered non-functional requirements.

*Purpose*: System constraints limit the solution space beyond what is necessary for satisfying the given functional and quality requirements (Glinz 2014, p. 11). They provide the boundaries for development and deployment.

*Interdependencies:* System constraint influence the specifications of the component model (#38).

*Notation:* System constraints are usually specified by natural text.

*Example: System Constraints*

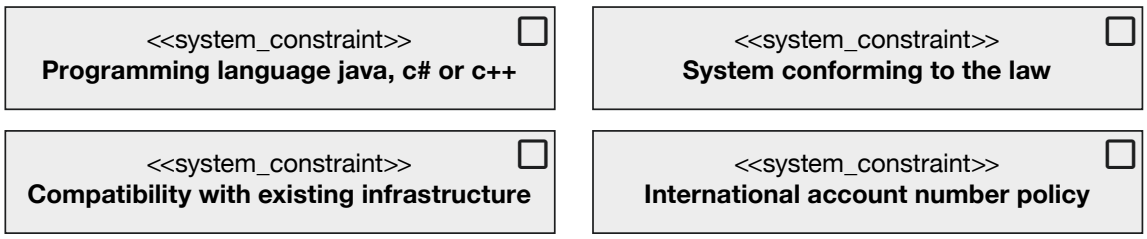



**Figure 42: Example of System Constraints**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 42 shows examples of system constraints that need to be taken into account for an ATM. They can include restrictions like "programming language java", "compatibility with existing infrastructure" or "international account number policy".

#### *5.3.3.33 Quality Requirements (RE Artifact #33)*

*Description*: 'Quality requirements' refer to the desired quality characteristics of a system beyond functionality and features (Glinz 2014, p. 16). Reliability, performance, security, usability, adaptability are examples for quality requirements. As they are often phrased in qualitative form (e.g. "the system shall be fast"), they require measurable and testable metrics for quantification and operationalization. Existing quality models like ISO/IEC 25010:2011can provide a guideline and help to quantify a specific requirement.

*Purpose*: Quality requirements help to validate the successful completion of an entire system or its respective functions and features.

*Interdependencies:* There are numerous input sources for quality requirements in the artifact model. For example, they can be elicited in field studies (#11), deduced from personas (#13), imposed by objectives and goals (05#), and drawn from the system vision (#24) or usage model (#25). Quality requirements provide input for the function model (#36) and the component model (#38).

*Notation:* Quality requirements are usually specified by natural text.

*Example: Quality Requirements*

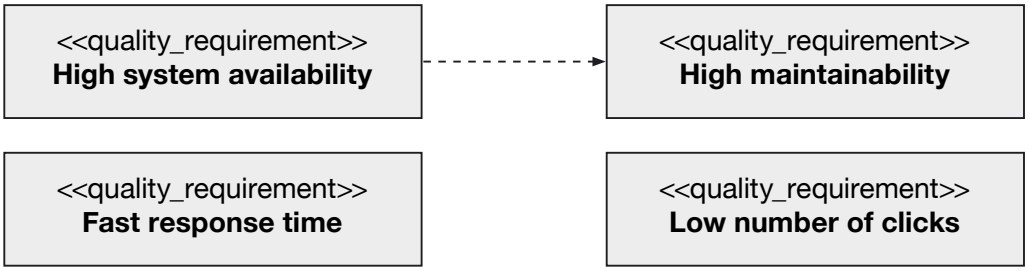



**Figure 43: Example of Quality Requirements**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 43 illustrates examples of high-level quality requirements for an ATM. They include for instance, "high system availability", "fast response time", or "low number of clicks". The detailed specification of a quality requirement (e.g. "usability: the system shall be easy to use") typically includes a *rationale* (e.g. "makes it more likely for users to accept a new system"), a *satisfaction criterion* (e.g. "90% of new users know how to carry out all features after 15 minutes of use"), a *measurement* (e.g. "test with 200 beta testers who receive a task and measure the time for completion"), and *risks* when not meeting the quality requirement (e.g. "if the system is not easy to use, it will be rejected by the users and will not conquer the market") (Penzenstadler 2017e, p. 18).

#### *5.3.3.34 Glossary (RE Artifact #34)*

*Description and interdependencies:* The 'glossary' in the requirements specification extends the glossary of context-relevant terms (#09) with requirements-specific terms. It will show up again in the system specification (#40) as more terms are added (Méndez Fernández and Penzenstadler 2014a, p. 37).

*Purpose:* For detailed reasoning see description of glossary (#09) in Section 5.3.3.9.

*Notation:* The glossary is usually specified by structured natural text.

*Example: Glossary*

**Glossary**

| Term | Abbreviation | Description | Synonym | Comment |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Figure 44: Example of a Glossary Structure**

**(Source: own illustration)**

See example description of glossary artifact (#09) in Section 5.3.3.9.

### *5.3.3.35 Architecture Overview (RE Artifact #35)*

*Description*: The 'architecture overview' includes the components and their interactions as depicted in the component overview (#38) and the major functions that encompass the functional hierarchy (#28) (Méndez Fernández and Penzenstadler 2014a, p. 41).

*Purpose:* The architecture overview provides a high-level understanding of the evolving system's architecture and, thus, functions as a guide for defining the more intricate functional and operational architecture (Mitra 2008, p. 2). The artifact is used to facilitate communication between different stakeholders and developers. Based on the architecture overview different architecture options for a particular solution idea can be explored and evaluated.

*Interdependencies:* The architecture overview influences the function model (#36) and component model (#38).

*Notation:* The architecture overview is usually specified by a component diagram.

*Example: Architecture Overview*

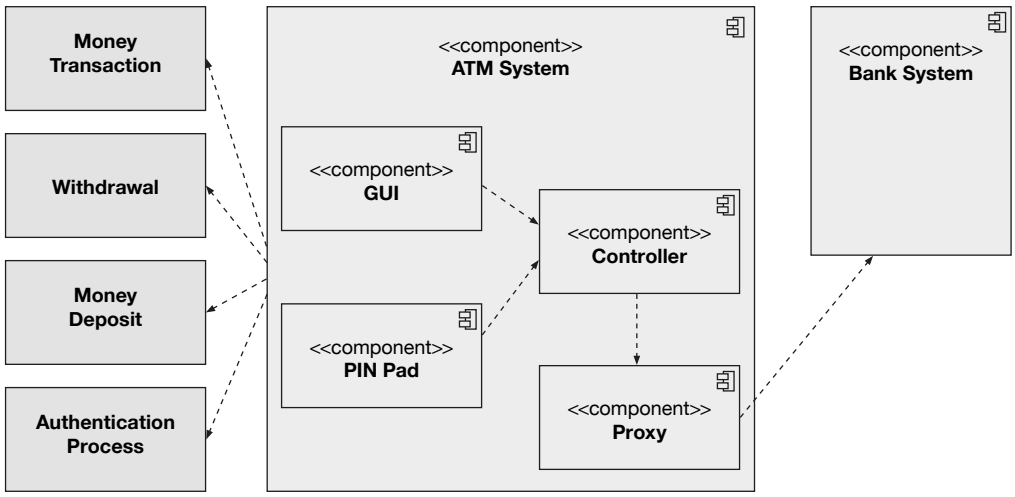



**Figure 45: Example of an Architecture Overview**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 45 presents a simplified architecture overview of an ATM. It depicts the components "ATM System" and "Bank System". The former includes sub-components like "GUI", "Controller", and "PIN Pad". "Money Transaction", "Withdrawal", "Money Deposit", and "Authentication Process" are use cases that influence the system design.

#### *5.3.3.36 Function Model (RE Artifact #36)*

*Description:* The 'function model' provides an overview diagram of the functions and their communication relationships (Braun et al. 2014, p. 28). It realizes the user-observable functions from the functional hierarchy (#28) and connects the ports specified in the component model (#38) (Méndez Fernández and Penzenstadler 2014a, p. 41).

*Purpose:* The function model ensures an overview of all functions and processes and, thus, assists in determining the scope for implementation and the product and service costs.

*Interdependencies:* The function model is composed of elements from the functional hierarchy (#28), quality requirements (#33), the architecture overview (#35), and the

component model (#38). The output of the function model feeds into the behavioral model (#39).

*Notation:* The function model is usually specified by graphs and tables.

---

*Example: Function Model*

---

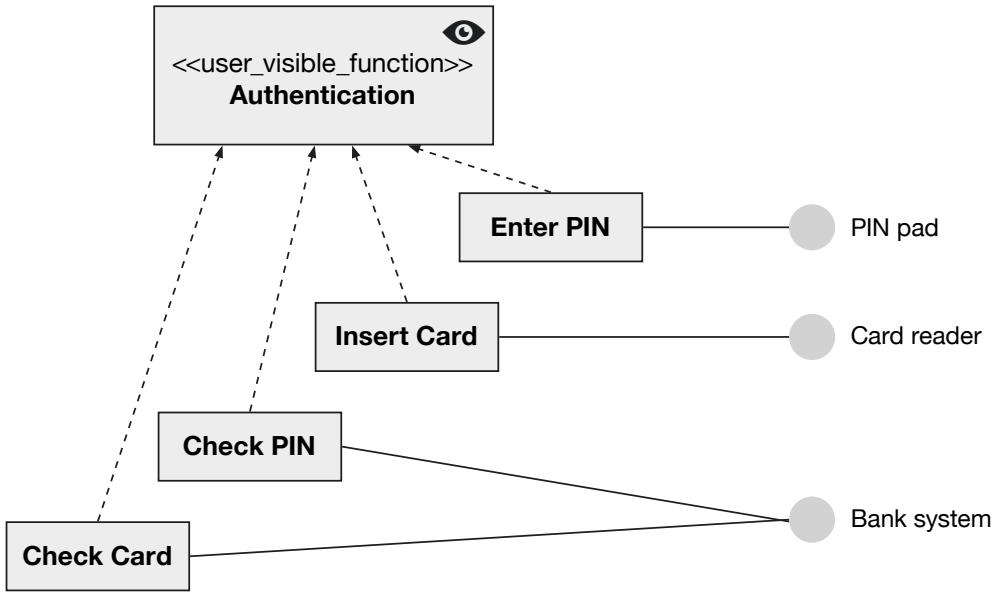



**Figure 46: Example of a Function Model**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 46 shows the function model of an ATM. It provides an overview of the activities that are to realize the user-visible function of "Authentication". These activities include, for example, "Enter PIN" (via the port "PIN pad"), "Check Pin", or "check card" (both via the port "Bank System").

---

#### *5.3.3.37 Data Model (RE Artifact #37)*

*Description:* The 'data model' captures the coarse-grained data objects and their relations that are required for the execution of the system's functions (Méndez Fernández and Penzenstadler 2014a, p. 42).

*Purpose:* The "data elements" of the data model refine the "data objects" from the data model (#29) in the requirements layer by using a particular data type. It is used to define the details of information to be stored for the generation of software code or it can aid in make-or-buy decisions of software. The data specification in the system layer is part

of a stepwise completion from moving the focus on defining user-visible functions towards specifying the design system with the component design (#38) and the behavior specification (#39).

*Interdependencies*: The data model (#29) of the requirements layer provides the input for the data model in the system layer. The functions are derived from the behavioral model (#39).

*Notation:* The data model is usually specified in a data modeling notation such as UML class diagrams.

---

*Example: Data Model*

---

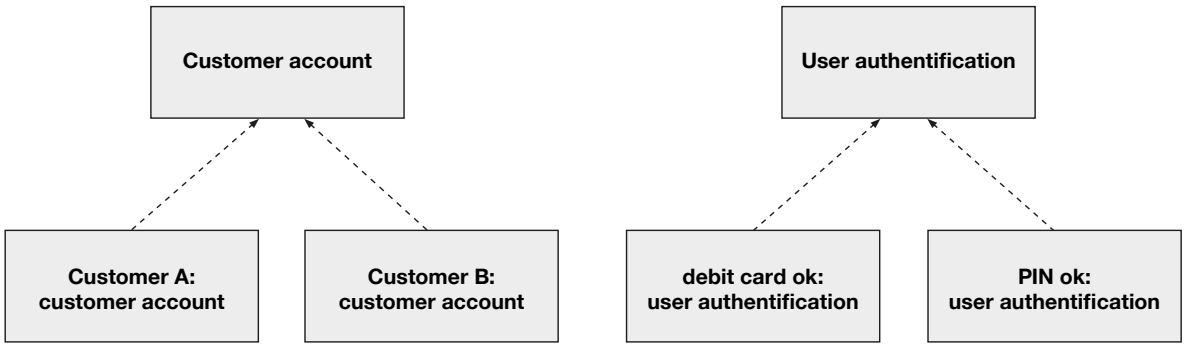



**Figure 47: Example of a Data Model in the System Specification (Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 47 illustrates the excerpt of a data model for an ATM system. The coarse-grained data objects in this example are "Customer account" and "User authentication". The "Customer account" gets input from the data of each individual customer's account (e.g. Customer A, Customer B). "User authentication" is achieved by checking the debit card and the PIN.

---

#### *5.3.3.38 Component Model (RE Artifact #38)*

*Description*: The 'component model' describes the components (i.e. building blocks) of a system's services and their respective channels and interfaces (Méndez Fernández and Penzenstadler 2014a, p. 42). Different types of components are for example application components, system software components, technical components, or hardware components. They can be broken down into sub-components. Components are identified

based on their contributions and responsibilities that form the system behavior collectively.

*Purpose*: The component model bridges the requirements layer with the system layer by laying out the main design principles and overall structure of the system. The model helps to decompose the complexity of a software system into smaller problem scopes and tasks. Each of those can then be handled and executed by specialists.

---

*Example: Component Model*

---

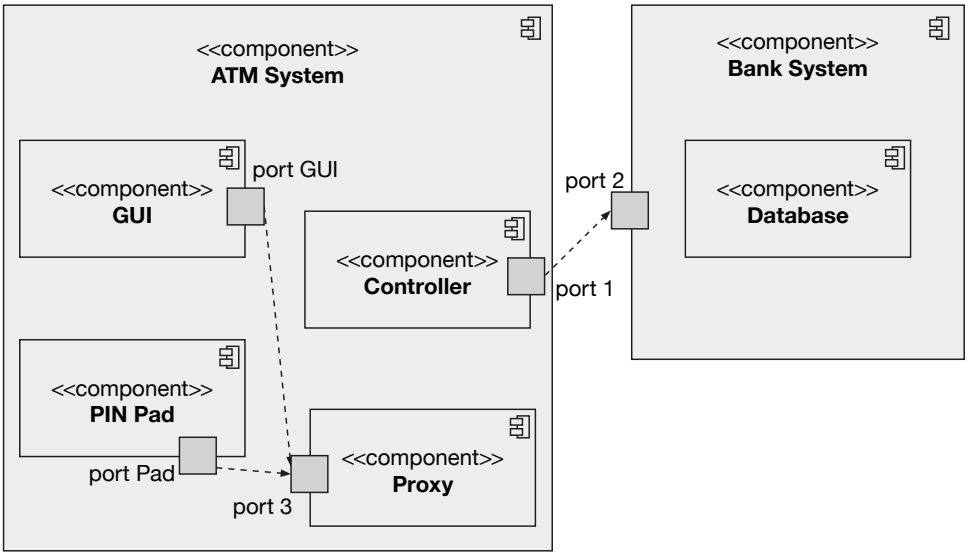



**Figure 48: Example of a Component Model**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 48 shows the component model of an ATM. In this example the components include the "Bank System" and the "ATM System" with its sub-components "GUI", "Proxy", "PIN Pad" and "Controller" which communicate with each other. Ports depict the respective communication end points.

---

*Interdependencies:* The component model gains input from the system constraints (#32), quality requirements (#33), and the architecture overview (#35). Its output feeds into the function model (#36) and behavior model (#39).

*Notation:* The 'component model' is usually specified by component diagrams.

#### *5.3.3.39 Behavior Model (RE Artifact #39)*

*Description*: A 'behavior model' describes the internal behavior of a system with the goal to execute the defined functionalities. It visualizes the responses of the system to an external stimulus (Braun et al. 2014, p. 28). A stimulus can be incoming data that has to be processed or events that trigger system processing. The resulting state machines document the behavior and used data elements of the data model (#37) (Méndez Fernández and Penzenstadler 2014a, p. 42).

*Purpose:* The purpose of the behavior model is to depict a dynamic view of the system behavior and to illustrate how objects or system components interact to support use cases. This is relevant for the specification of the requirements in the system layer.

---

*Example: Behavior Model*

---

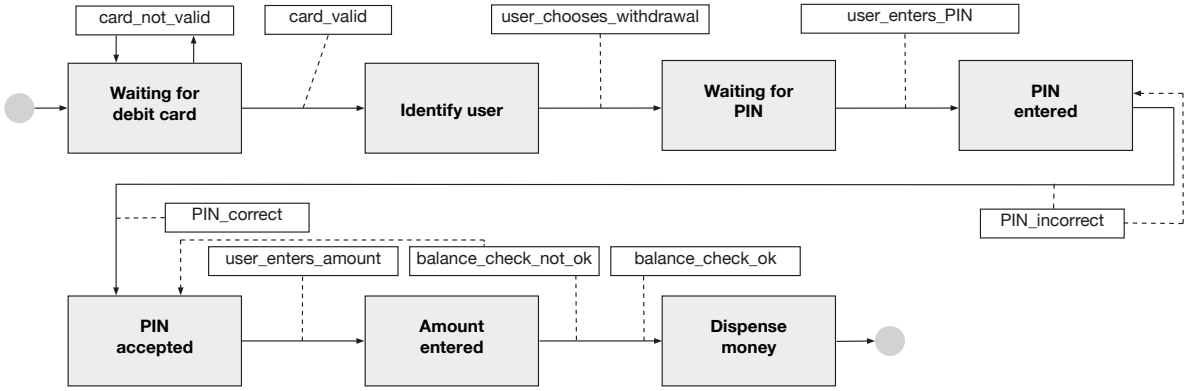



**Figure 49: Example of a Behavior Model**
**(Adapted from Méndez Fernández and Penzenstadler 2014b)**

Figure 49 presents a sample activity diagram that depicts the states of system behavior while using an ATM. The activity diagram puts into a sequence the activities, actions, start and end nodes, and decision points. Activities in this example include "Identify user", "PIN entered", "PIN accepted", "Amount entered". If the balance check is ok, money is dispensed; otherwise the system goes back to customer interaction.

---

*Interdependencies:* The behavior model is influenced by the functionalities in the usage model (#25), function model (#36), and component model (#38). It provides input for the data model (#37).

*Notation:* The behavior model is usually specified by interaction diagrams (communication and sequence) and behavioral state machines.

#### *5.3.3.40 Glossary (RE Artifact #40)*

*Description and interdependencies:* The 'glossary' in the system specification extends the previously defined glossary artifacts (#09, #34) with technical relevant terms.

*Purpose:* For detailed reasoning see description of glossary (#09) in Section 5.3.3.9.

*Notation:* The glossary is usually specified by structured natural text.

---

*Example: Glossary*

---

**Glossary**

| Term | Abbreviation | Description | Synonym | Comment |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Figure 50: Example of a Glossary Structure**
**(Source: own illustration)**

See example description of glossary artifact (#09) in Section 5.3.3.9.

---

## 5.4 Discussion

The combined artifact-based reference model was created to provide a solid foundation for comparing and combining artifacts from Design Thinking and Requirements Engineering. As artifact-orientation is independent from the surrounding development processes, this was considered appropriate due to the variability in different processes and methods in both approaches (Méndez Fernández and Penzenstadler 2014a; Uebernickel et al. 2015). The final artifact model offers a number of important insights and implications for using Design Thinking in the context of Requirements Engineering.

Various commonalities between Design Thinking and Requirements Engineering can be seen, if the latter is understood as an iterative approach. The differences should be seen as complementary activities. The artifact model distinguishes between problem and solution-oriented artifacts which addresses the principles of both Design Thinking and Requirements Engineering (Braun et al. 2014; Brown 2008). Problem-oriented artifacts contain information about the underlying problem context including the goals and needs

of stakeholders as well as specific system conditions or constraints. Solution-oriented artifacts contain information about the corresponding system vision and how to solve the problem stated in the project description (Braun et al. 2014).

The findings show that Design Thinking mainly contributes to early Requirements Engineering activities with up to 14 additional context artifacts for a comprehensive understanding of the problem domain. Accordingly, Design Thinking expands the toolbox for Requirements Engineering by emphasizing the creation of artifacts that describe the relevance of the system vision. Design Thinking could even be exclusively used as a way to perform these activities. A complementary approach of Design Thinking and Requirements Engineering, however, seems necessary for shaping the requirements layer. While both concepts produce overlapping artifacts (system vision, functional requirements, usage and service models), their realization might take different forms. Design Thinking uses mainly a high-fidelity prototype to describe the system vision and key functionalities. Requirements Engineering specifies the same mainly by using rich picture and class diagrams. In addition, other requirement types, such as quality or deployment requirements are predominantly specified with common Requirements Engineering techniques. Requirements Engineering is exclusively used to specify system artifacts and to provide the interface to system design activities. Hence, Requirements Engineering also expands the toolbox of Design Thinking.

Following the role model of Méndez Fernández and Penzenstadler (2014a) (see Figure 8), implications can be seen in expanding the knowledge of business analysts with Design Thinking skills and in equipping design thinkers with Requirements Engineering skills to gain appreciation for subsequent software design activities. A new role of a ‘human-centric requirements engineer’ who integrates both skills equally might be incorporated.

### 5.4.1 Implications for Theory

The theoretical contributions of this model are intended to serve more than one purpose. This study provides an evaluated artifact model of Design Thinking which has not been done so far. Based on empirical evidence and a literature review the main artifacts for Design Thinking have been distilled and put into an order to lay a terminological and conceptual foundation. While different classifications of Design Thinking methods, tools, and processes exist (e.g. Doorley et al. 2018; IDEO.org 2015; Uebernickel et al.

2015), none of them classify Design Thinking from an artifact-based view. This work contributes to existing method compendia with a new perspective on conducting Design Thinking activities for software-intensive systems.

This study lays a first fundamental, conceptual, and empirical basis for understanding the different artifacts of Design Thinking and Requirements Engineering. The combined artifact model offers a reference structure for all artifacts to be used or delivered as part of an innovative software-intensive development project that aims to combine Design Thinking and Requirements Engineering. This provides a basis for a systematic comparison and combination of both concepts that others can build on. On the one side, the combined model connects Design Thinking to the growing research area in artifact-based Requirements Engineering and, thus, strengthens the concepts of Design Thinking (e.g. Berenbach et al. 2009; Méndez Fernández and Penzenstadler 2014a). On the other side, Requirements Engineering connects to the upcoming research stream of applying Design Thinking for supporting new ways of working and contributing to agile software engineering practices (e.g. Przybilla et al. 2018; Kahan et al. 2019).

The artifact model can be seen as a starting point for the creation of a more comprehensive development process model, which can be defined as "a standardized organizational reference model that abstracts from the idealized execution of a development project, including a description of artifacts (deliverables) to be produced, activities to be performed, and roles to be assigned." (Gnatz 2005, p. 135) Building on this definition, a development process model can contain the following five sub-models: (1) The artifact model specifies *what* needs to be produced or exchanged; (2) the role model describes *who* should produce it and which particular responsibilities are needed; (3) the activity model describes *what* to do in order to create, modify, or use an artifact; (4) the process model denotes *when* the artifacts, roles, and activities should be produced or performed; and (5) standards and tools conceptualize *with what* all of the above mentioned activities are performed (Méndez Fernández and Penzenstadler 2014a).

### 5.4.2 Implications for Practice

Our proposed artifact-based reference model offers a number of practical contributions in the context of innovative software-intensive development projects.

For project managers, several major contributions can be seen. First, the model can be considered a support system to define and distinguish responsibilities in a project. Project roles can be directly coupled to the creation of artifacts, for which they have to take the responsibility. Second, project managers can assign completion levels and establish progress control for the creation of artifacts. Quality assurance metrics can help to objectively measure the degree of completeness of an artifact in the artifact-based reference model. Third, the model ensures flexibility for integrating processes and customizing the reference model at project level. The combined model allows for variations of the created artifacts in response to individual project characteristics. For example, by defining the content-focus of the project, the creation of either DT or RE artifacts might be of greater help as each approach emphasizes a different content type. For example, to better understand the user and business context, the creation of DT artifacts might be preferred. RE artifacts should be at the center of attention to better describe the technical perspective and answer feasibility questions. Teams may also jump back and forth between both approaches if new questions come up in one or the other area. Fourth, the model can act as basis for effective requirements management, whose objective is to administrate the outcome of Requirements Engineering activities (Fleischmann et al. 2004). This administration includes, for example, progress and traceability control, impact analyses, or risk mitigation (Jönsson and Lindvall 2005). A structured and consistent content specification is a prerequisite to perform such activities. Hence, the combined artifact model can enhance the effectiveness of requirements management activities due to its defined set of interdependencies and chosen artifacts.

For team members (i.e. requirements engineers, business analysts, or design thinkers) the model offers a blueprint for creating syntactically consistent and complete results with respect to the respective application domain. While not all artifacts from the model have to be considered in every project, the overview still serves as an orientation and connection to further design and development activities. The latter point is especially of interest for Design Thinking as this has been continuously criticized to be insufficiently linked to development processes (e.g. Häger et al. 2015).

For training providers, the artifact-based model can support the development of new trainings programs and learning formats about combining Design Thinking and Requirements Engineering. A new role with skills and talents in both approaches may

be fostered. Current training courses in Design Thinking or Requirements Engineering can be enhanced by integrating the respective other approach to gain understanding about the benefits and shortcomings of the two incorporated concepts.

### 5.4.3 Limitations and Outlook

As with any research, this work does have its limitations. First, the evaluation of the combined artifact model was performed with a limited number of experts. This issue was addressed by establishing high demands for the selection of experts in terms of their domain expertise and knowledge. In addition, the experts reviewed the model independently. Second, the generalizability of the findings of the combined artifact model can be questioned. Although this study aimed for providing as much generalizability as possible by choosing an established evaluated artifact model from Requirements Engineering as a basis for this research, the practical application of the model should be considered a future research opportunity. Therefore, the demand for further validation of the model through empirical studies and evaluation needs to be highlighted. Third, as the focus of this study was on defining an artifact-based reference model for innovative software-intensive systems, future work may also perform further domain-specific instantiations in different settings.

The findings presented in this study may also encourage future efforts to analyze the combined artifact model in the context of a broader development process model as suggested in the discussion in section 5.4. Looking at Design Thinking and Requirements Engineering from an even broader perspective provides great potential to ensure a sustainable combination of the two in the context of software-intensive development projects. Further research might also look into the development of tool support, activity, process, or designated role models. Addressing these topics would pave the way for leveraging the full potential of using Design Thinking to achieve a human-centered Requirements Engineering approach.

Finally, researchers and practitioners should feel encouraged to critically discuss the combined artifact approach and to empirically evaluate an artifact-based combination of Design Thinking and Requirements Engineering.

## 5.5 Conclusion of Chapter

This chapter provided a consolidated view on an artifact-based combination of Design Thinking and Requirements Engineering. Various commonalities between Design Thinking and Requirements Engineering can be seen, if the latter is understood as an iterative approach. The differences should be regarded as complementary activities. Both approaches aim at discovering goals and requirements. While both Design Thinking and Requirements Engineering are very distinct when it comes to the underlying philosophies, many artifacts are complementary or even overlapping. While Requirements Engineering success is often measured in documented requirements as a foundation for development and quality assurance, in Design Thinking the philosophy of domain understanding and the learning curve leading to it is predominant – regardless of the surrounding processes. Practitioners can apply the combined artifact model in their own contexts with the awareness of the benefits and shortcomings of the two incorporated concepts.

# 6 Operationalization Strategies

This chapter addresses the third research question of the dissertation. Building on the artifact model introduced in chapter 5, three operationalization strategies for combining Design Thinking and Requirements Engineering are investigated. This chapter presents the results of a study[5] that presented operationalization strategies with their respective pre-requisites, benefits, and challenges. The operationalization strategies are described as follows: Section 6.2 explains the upfront Design Thinking strategy, section 6.3 presents the infused Design Thinking strategy, and section 6.4 describes the continuous Design Thinking strategy. Section 6.5 discusses the results and their implications for research and practice. Section 6.6 concludes the chapter.

## 6.1 Customization

The combined artifact-based reference model enables a flexible creation of the introduced DT and RE artifacts. This means that the decision *when* and *which* artifacts should be produced can be customized according to specific project characteristics. To provide a guideline three operationalization strategies are proposed to integrate Design Thinking and Requirements Engineering in different ways. The strategies reflect existing research findings about integrating Design Thinking into software development practices (e.g. Dobrigkeit and de Paula 2019; Lindberg et al. 2012) as well as the empirical findings from industry cases (see chapters 3 and 7).

The following strategies are suggested: (1) Run Design Thinking prior to performing Requirements Engineering practices (*upfront Design Thinking*); (2) instill the existing Requirements Engineering process ad-hoc with selected Design Thinking tools and artifacts (*infused Design Thinking*); or (3) combine the previous two strategies and integrate Design Thinking into Requirements Engineering practices on an ongoing basis (*continuous Design Thinking*). The ratio between Design Thinking and Requirements Engineering differs within the three proposed operationalization strategies (see Figure 51). The better the original problem is understood, the more activities are biased towards straightforward design and implementation tasks (i.e. RE artifacts). The less it is understood, the more activities are directed towards context understanding and problem

[5] Parts of this chapter have been published as a preprint and have been accepted for publication in: Hehn, J., Mendez, D., Uebernickel, F., Brenner, W., Broy, M. 2020. "On Integrating Design Thinking for a Human-centered Requirements Engineering," IEEE Software, Special Issue Design Thinking.

exploration (i.e. DT artifacts). Thus, the defined project objective and context are the guiding parameters for the selection of an appropriate operationalization strategy. In the following each strategy is introduced along with their *objectives*, required *prerequisites*, performed *activities*, executing *roles*, and expected *outcome*. A short *case example* from practice concludes the description of each strategy.

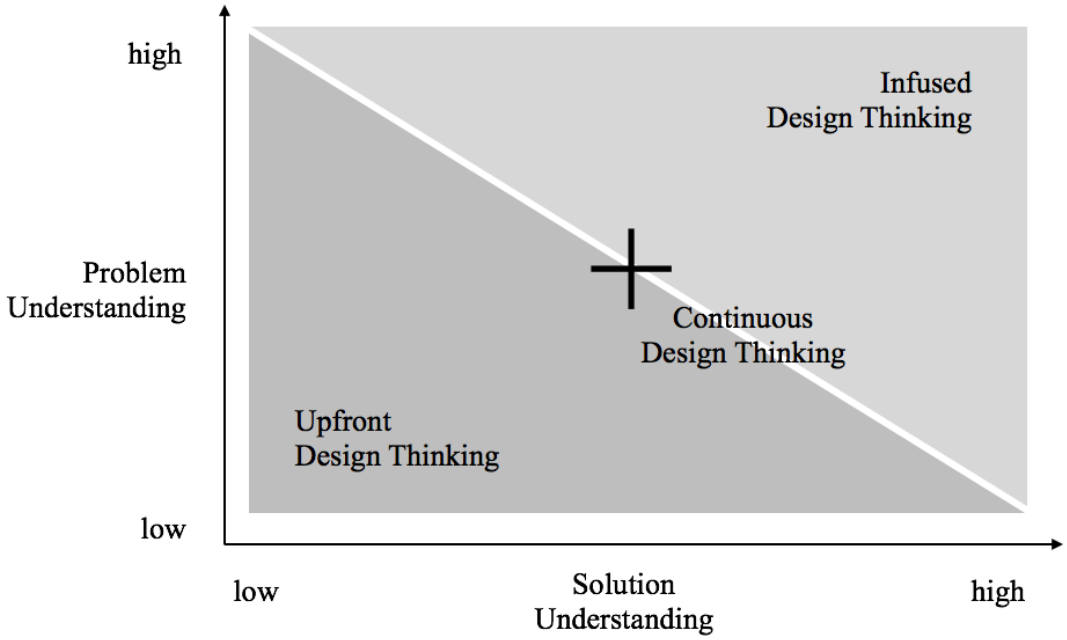



**Figure 51: Upfront, Infused, and Continuous Design Thinking Strategies (Source: own illustration)**

## 6.2 Upfront Design Thinking

*Objective:* Upfront Design Thinking is best applied when there is a high level of uncertainty about the problem (i.e. stakeholder and user needs) and the corresponding solution. Creating DT-related artifacts through applying Design Thinking helps to understand the problem in depth and to define the overall concept of an idea. It is typically used at an early project stage to provide clarity for unclear user needs and to define a solution vision.

*Prerequisites:* A problem statement should have been defined as a minimum starting point for applying upfront Design Thinking. Additional required conditions are access to potential users and other stakeholders as well as Design Thinking training for project members.

*Key activities:* Design Thinking activities are performed in form of a pre-project to identify relevant features that are worth implementing. The Design Thinking process model (define, needfinding, synthesis, ideation, prototyping, testing) guides through a cyclical creation of *context* and *requirements* artifacts (see Figure 15: Final Combined Artifact-based Reference Model). The outcome is used as a basis for performing further Requirements Engineering activities that complement DT artifacts with RE-specific ones.

*Roles:* Two roles during the upfront mode are required. First, the Design Thinking team is responsible for planning and executing the activities. This team consists of four to six people from different areas of expertise depending on what knowledge will be relevant for the project, e.g. subject matter experts, IT, marketing, sales, design (Häger et al. 2015). Second, a person or group of people, who has defined the initial design challenge and project scope, is the project sponsor. The person in this role typically provides continuous feedback to the team and connects it with others to enable synergistic effects and avoid duplicate efforts (Häger et al. 2015).

The following two roles are optional: First, an extended team of (internal) experts that provide further domain knowledge and expertise for the Design Thinking team. Second, a Design Thinking coach or coaches who support the project team with methodological guidance. They introduce Design Thinking techniques, facilitate team meetings, and ensure that the team is focused on delivering the tasks and artifacts. As such, the coach should have a profound understanding of Design Thinking to provide useful techniques and guidance at appropriate times (Häger et al. 2015).

*Outcome*: The main deliverable of the upfront strategy is a clear system vision as a basis for performing further Requirements Engineering activities. The system vision usually takes the form of a mockup (i.e. high-fidelity prototype). Along the way the team will create a comprehensive set of DT artifacts (see section 5.3.3), which should make it clear why each aspect of the prototype is intended in the way it is designed. High level user stories and a list of usability requirements based upon test results accompany the set of artifacts created by following the Design Thinking process.

*Case Example*

The international Alpha Insurance company wanted to develop a new service for their new target group of "young professionals". A project team stemming from five different business functions (marketing, IT, actuary, product manager, claims) spent 40% of their time to follow the Design Thinking process in an iterative manner for three months. The solution vision resulted in a tested medium-fidelity prototype for a digital on-demand insurance that could be activated and deactivated based on the user's preferences. The Design Thinking team handed over the prototype to the implementation team for further specification, testing, development, and market introduction. Transferred artifacts included a project documentation with 20 field studies, 2 personas, 5 opportunity areas, and 6 low-fidelity prototypes with learnings about failures. The final solution vision (in form of a mockup) specified key features and their usability. The implementation team performed tests to validate these features, their usability, and their service model.

## 6.3 Infused Design Thinking

*Objective:* The main goal of this strategy is to support existing Requirements Engineering activities with selected Design Thinking techniques. This includes, for example, activities to clarify fuzzy requirements, foster creativity, gain new ideas, or to better understand user needs.

*Prerequisites:* The prerequisites for applying this strategy depend on the specific problem to be addressed. The problem should have a clear scope. The prerequisites as described in section 6.2 still apply.

*Key activities:* An infused approach makes use of selected artifacts and leverages selected methods from the Design Thinking toolbox and integrates them into an existing Requirements Engineering process. In case of challenges encountered during the Requirements Engineering process, Design Thinking tools can be initiated; hence, their application is ad-hoc. The main activity of this strategy is the setup up of focused workshops with a selected Design Thinking tools (Dobrigkeit et al. 2108). These workshops can last three hours or several days depending on the objectives. For example, the goal of a workshop to generate new solution ideas could be formulated like this: "*Create ideas to optimize the user interface of our platform, making it look and feel*

*more emotional, and letting it appear less technical.*" This session used persona and customer journey artifacts to brainstorm new ideas.

*Roles*: In the infused setting, the people or person performing the Requirements Engineering activities are the addressees of receiving Design Thinking guidance in the form of workshops. Other workshop participants with different areas of expertise may be added, e.g. subject matter experts, IT, marketing, sales, design, depending on what knowledge will be relevant to achieve the workshop goal. A workshop typically consists of five to twenty participants. Similar to the upfront approach, a Design Thinking coach introduces the selected Design Thinking techniques and moderates the workshop and team discussions. The project sponsor can also be integrated to provide feedback and define the context for the general direction of the workshop.

*Outcome*: Due to the flexible approach of the infused strategy, the outcome is situation-dependent based on the previously defined objectives. The deliverables can be (new) features, user requirements, or test feedback – all following the Requirements Engineering process. In the context of the combined artifact model (Figure 15) this means that the creation of RE artifacts is enhanced by a selected set of DT artifacts.

---

*Case Example*

---

Beta Enterprises is an international electronics group that wanted to evaluate the possibilities of smartphone applications (e.g. emergency apps, task lists, maintenance procedures) for container ships in a marine context. The main goal was to define requirements from a user point of view and to foster creativity for solution finding. In a highly regulated environment, a Design Thinking infusion was chosen to support the ongoing Requirements Engineering activities with selected tools from needfinding and prototyping. Five Design Thinking infusion sessions (one to two days) were conducted within five months. Produced artifacts included field studies for precise user requirements (it was the first time the team had been in close contact with marine captains) and tested medium-fidelity prototypes to strengthen service and usage models. According to the workshop participants, having direct user contact raised the confidence level in the success of the intended solution. Initial concerns about not finding interview partners in a highly sensitive B2B setting turned out as unjustified.

---

## 6.4 Continuous Design Thinking

*Objective:* The main goal of this strategy is to integrate Design Thinking principles with Requirements Engineering activities on a continuous basis. Beyond the specific project context (which is the main focus of this dissertation), this can also become part of an organizational change program or corporate strategy.

*Pre-requisites:* Continuous Design Thinking is recommended when addressing complex ("wicked") problem settings, which require continuous user involvement along all software engineering activities. In addition to the prerequisites described for the previous two strategies (see sections 6.2 and 6.3), (selected) project members should possess both Design Thinking and Requirements Engineering knowledge.

*Key activities:* Continuous Design Thinking utilizes Design Thinking as guiding principle. On an operational level, this translates into a seamless combination of the upfront and infused strategy and the setup of a new project role for a human-centric requirements engineer. The activities comply with both Design Thinking and Requirements Engineering elements to establish an end-to-end view from exploring a user need to conceptualizing a solution vision and specifying a functional system. When starting a project, the upfront strategy can be used to provide clarity about the problem context and elicit (user) requirements in a structured yet creative manner. A high-resolution prototype can help to specify the functionalities of the system vision. When moving on to the more technical side of requirements specification, an ad-hoc usage of Design Thinking methods can still be initiated in case features are not defined well enough from a user point of view for example.

*Roles*: The instantiation of a new role incorporates Design Thinking expertise as well as Requirements Engineering expertise and mediating between both schools of thoughts. In this strategy it is of great importance that the new role can react quickly when choosing methods and artifacts. The role enables the team to work towards a final product in incremental steps. The responsibilities of the project team during this strategy are similar to the preceding ones as the continuous strategy combines the two other strategies. The team plans and executes the activities to define the final system. The project sponsor has similar responsibilities as described in the previous sections.

*Outcome*: The continuous strategy results in a comprehensive set of DT and RE artifacts as shown in Figure 15. The requirements specification and system design are based on and traceable to customer needs derived from the context specification.

---

*Case Example*

---

Gamma Energy is a large energy provider with subsidiaries worldwide. A diverse project team applied an upfront Design Thinking approach to explore the potential of platforms in the utility sector. The outcome was a solution vision for a digital home improvement platform to advance lead generation. To ensure a human-centered mindset throughout specification and development, a new role was established to use selected Design Thinking tools for enhancing the prototype and filling the backlog with new features. Produced Design Thinking artifacts included high-fidelity prototypes with usability- and feature-oriented test feedback and new solution ideas. Scrum became the guiding framework for development, which enabled the entire project team to work in sprints. During development Design Thinking prototypes were used as boundary objects to enhance communication with relevant internal stakeholders and to foster a human-centered mindset within the team (a detailed explanation and analysis of this case study is provided in chapter 7).

---

## 6.5 Discussion

This chapter aimed at understanding how Design Thinking can be used for Requirements Engineering in different ways in the context of software-intensive development projects. The three presented operationalization strategies reflect the ongoing discourse of describing Design Thinking at different levels in software engineering approaches (e.g. Brenner et al. 2014; Dobrigkeit and de Paula 2019). In line with other authors (e.g. Dobrigkeit et al. 2018), this study suggests that the way in which Design Thinking should be used depends on the specific context and objectives of a project. Accordingly, three different strategies with different Design Thinking formats (e.g. phases, workshops, single methods) were suggested which are similar to other proposed strategies in research in the context of (agile) software development. Depending on the situation each operationalization strategy offers different benefits but also challenges. Table 10 discusses both for each strategy.

Beside the project context, the existing maturity level of Design Thinking within an organization can be considered an influencing factor when choosing the 'right' strategy. While Requirements Engineering is usually an established practice in industry, Design Thinking is still relatively new. The decision to integrate the two approaches also depends on the level of courage, given time, and dedicated resources. As a rough guideline, the infusion strategy provides a reasonable starting point as it applies focused Design Thinking interventions within established practices. While the upfront strategy also keeps existing procedures, it requires more time and resources. Finally, the continuous strategy demands for a commitment from management to foster mindset change in an organization or department.

| Strategy | Benefits | Challenges |
|---|---|---|
| *Upfront Design Thinking* | - The full potential of Design Thinking is leveraged while changes to Requirements Engineering are not necessary<br>- Due to the focus on problem exploration deep context understanding is achieved<br>- The solution concept has traceable links to user needs | - Resource and time intense<br>- Lost (implicit) knowledge and potential starvation of results when handing over Design Thinking results<br>- Little attention is paid to further development critical artifacts such as quality requirements, system constraints, or data models |
| *Infused Design Thinking* | - Intervention character requires only minimal changes in existing Requirements Engineering practices<br>- Resource and time friendly due to ad-hoc usage of selected tools (especially compared to upfront approach)<br>- Low adoption hurdle for Design Thinking methods | - Risk of neglecting problem understanding (especially compared to the upfront approach)<br>- No embedding of Design Thinking mindset due to situational Design Thinking workshops<br>- Little attention is paid to further development critical artifacts such as quality requirements, system constraints, or data models |
| *Continuous Design Thinking* | - Seamless integration into existing Requirements Engineering practices including development critical artifacts<br>- High likelihood of infusing a human-centered mindset within the project team<br>- Precise and traceable (user) requirements through continuous identification of new requirements and testing | - Requires commitment, resources, and time to develop continuous integration of both approaches in an organization<br>- Continuous Design Thinking is highly dependent on the staffing of the project team<br>- Requires an organizational mind shift and support, potentially even an organizational restructuring |

**Table 10: Benefits and Challenges of each Operationalization Strategy**

**(Source: own illustration)**

### 6.5.1 Implications for Theory

From a theoretical perspective, the presented operationalization strategies offer new knowledge to emerging literature on how to combine the activities of Design Thinking and Requirements Engineering by identifying where and how Design Thinking adds value to the development of software-intensive systems. Specifically, the findings advance the knowledge of how Design Thinking is actually used within the industry not only in a pre-phase to software engineering (upfront) but also in later Requirements Engineering stages. This study builds on work of researchers who have proposed similar integration strategies in the context of software engineering. For example, Lindberg et al. (2012) investigated how Design Thinking can support software development in different ways. The upfront and infused strategies can be compared with the authors' separation of Design Thinking and software development (so called split project model) or the selection of single methods to overcome problems software developers cannot solve with common IT methods (so called toolbox model). In addition, this study proposes a continuous strategy to regularly integrate Design Thinking phases and elements into Requirements Engineering activities. The continuous Design Thinking strategy can be seen as an extension of the modes of Lindberg et al. (2012) by combining them with findings from creativity research (e.g. Carlgren et al. 2016; Wölbling et al. 2012). Accordingly, authors have shown that the continuous practice of Design Thinking can lead to the development of a Design Thinking mindset. The continuous Design Thinking strategy claims exactly that for the context of Requirements Engineering.

### 6.5.2 Implications for Practice

From an applied perspective, the findings can serve as a guide for practitioners on how to setup and implement Design Thinking activities into their Requirements Engineering practices. Practitioners can directly adopt the operationalization strategies in their own contexts with the awareness of the benefits and shortcomings of each incorporated strategy. Project managers can choose between the upfront, infused, and continuous strategy to steer their limited resources towards applying the most appropriate one given their individual project conditions. Beginners in the field of Design Thinking might gain a better understanding about different conceptualizations of Design Thinking in a software engineering context and start their learning journey guided by the application of the three strategies.

### 6.5.3 Limitations and Outlook

The proposed strategies are mainly derived from the authors' own practical experiences in applying Design Thinking in industry (see chapter 7) and findings from case studies in similar software engineering contexts (e.g. Lindberg et al. 2012). Future work will need to expand the findings and validate the operationalization strategies in more industry scenarios. Particularly, the following possibilities for further investigation should be considered.

First, there is great potential for further examining general project conditions to support decision-making in a more structured way – when to use which strategy. Exemplary questions could be: How can problems be efficiently classified at which stage of a project? What are typical project situations beyond the problem class which influence the choice of a strategy? How do these situations and the class of systems influence the choice of a strategy and/or single methods? How can these situations be characterized and assessed in early stages of a project (and with which confidence)?

Second, on an individual level, there is the opportunity to analyze and specify the proposed new role of a human-centric requirements engineer. Open issues to be explored are, for example, the definition of required soft and hard skills for such a role as well as their responsibilities in a project setting (in comparison to other roles).

Third, there is potential to quantify the effects of each operationalization strategy. A possible question could be: "How can different ways of implementing Design Thinking influence the team performance?" In addition, researching the influence of a person's Design Thinking knowledge and experience on the success of different operationalization strategies will provide insights about how to operationalize Design Thinking in the context of Requirements Engineering more effectively.

## 6.6 Conclusion of Chapter

The goal of this chapter was to understand how Design Thinking can support Requirements Engineering on an operational project level. In order to address this aim, research findings from literature and experience from industry projects were leveraged. As a result, three operationalization strategies were introduced that integrate Design Thinking into Requirements Engineering practices: (1) upfront Design Thinking, (2) infused Design Thinking, and (3) continuous Design Thinking. Upfront Design

Thinking leverages the Design Thinking process to produce DT artifacts with a system vision in form of a high-resolution prototype as an outcome. On this basis, technical requirements are specified with the help of Requirements Engineering practices. Infused Design Thinking leverages selected Design Thinking techniques for situational ad-hoc support for Requirements Engineering. Continuous Design Thinking seamlessly connects the generation of customer-centric innovative ideas and their translation into a system design.

# 7 Continuous Design Thinking: A Longitudinal Case Study

This chapter addresses the third research question of the dissertation and examines the continuous Design Thinking approach, which was introduced in chapter 6. It presents the results of a longitudinal case study that investigated usage patterns of Design Thinking for Requirements Engineering over a project lifecycle[6]. Section 7.2 provides the motivation for the case study. Section 7.2 explains the research methodology and offers a description of data collection and analysis. Section 7.3 reveals the results of the study. Section 7.4 discusses the implications for theory and practice and points at the study's limitations and an outlook for future research. Section 7.5 concludes the chapter.

## 7.1 Motivation

Despite the practical and theoretical advances in confirming the potential benefits of using Design Thinking for Requirements Engineering, little is yet known about how such an integration could be achieved. Particularly, knowledge about the specific role of Design Thinking in the context of Requirements Engineering practices remains still unclear. In this context, we have conducted a study to understand how Design Thinking is used for Requirements Engineering in a project setup for innovative software-intensive systems. We aim to answer the following research questions:

*How is Design Thinking used for Requirements Engineering in innovative software-intensive development projects?* The first research question examines the actual use of Design Thinking for Requirements Engineering from the very beginning starting with problem definition, the development of a proof-of-concept to market-ready implementation in an agile development setting. This first question will generate a comprehensive foundation for answering the second research question:

*How does Design Thinking address Requirements Engineering challenges?* This question examines the learning potential of Design Thinking for Requirements Engineering. It will generate findings, how it can complement current Requirements

[6] Parts of this chapter have been published in: Hehn, J., Uebernickel, F. 2018. "The Use of Design Thinking for Requirements Engineering – An Ongoing Case Study in the Field of Innovative Software-Intensive Systems," *Proceedings of the 26th IEEE International Requirements Engineering Conference (RE'18).* Banff, Canada: IEEE, and in: Hehn, J., Uebernickel, F. 2019. "The Use of Design Thinking for Requirements Engineering – An Ongoing Case Study in the Field of Innovative Software-Intensive Systems," *Lecture Notes: 49. Jahrestagung der Gesellschaft für Informatik*, Extended Abstract.

Engineering practices by addressing known challenges as encountered by the Requirements Engineering community of practitioners.

To investigate our research questions, we set up a longitudinal case study. We use an exploratory approach by analyzing the use of Design Thinking in the context of Requirements Engineering against the background of existing scientific findings of Design Thinking and challenges in (agile) Requirements Engineering.

## 7.2 Research Methodology

Our aim is to investigate how Design Thinking is used for Requirements Engineering in software-intensive development projects and what can be learned from it. We adopt a qualitative approach since we want to gain an in-depth understanding of Design Thinking for Requirements Engineering as a socio-technological activity (Runeson and Höst 2009). We believe that an exploratory case study (Yin 2011) is the best methodology for elaborating specific challenges encountered by practitioners of the Requirements Engineering community and theorizing potential concepts in the future. Thus, we follow a longitudinal case study approach to gain richer and more contextualized information of the usage of Design Thinking for Requirements Engineering over time in a real-life project setting (Runeson and Höst 2009; Yin 2014). In the following, we describe our case selection, data collection, and data analysis.

### 7.2.1 Case Selection

To identify an appropriate case, we got access from a Swiss-German consultancy to their project portfolio from the last five years. We evaluated their projects with regard to their suitability for our research objective and questions. In particular, we were looking for (1) the application of Design Thinking for Requirements Engineering practices, (2) the development of a new and innovative software-intensive system, and (3) an end-to-end setting, from the beginning of conceptualization until market-ready implementation. In 2016 we identified the project case at hand as particularly suitable. The project is based in the utility industry in Europe. The energy sector is currently under heavy pressure (in Europe) because of the need to transform from nuclear and coal power generation towards new and more sustainable ways of power generation such as solar power (photovoltaic) using water or wind energy. In addition, this industry is attacked by digital innovations and deals with changing customer behavior. Smart home

technologies like Nest or electric vehicles like Tesla demand digital solutions and innovation from utilities.

In this context, project Falcon (anonymized) was launched in 2016 to explore and exploit new options provided by digital platforms that could be used for the energy sector, without knowing what the final system would be or look like. This is why the project was set up to be highly agile, innovative, and human-centered from the beginning on. Based on these pre-requirements, the management team decided to apply Design Thinking and Scrum as leading methodologies for specifying requirements and developing the system vision.

### 7.2.2 Data Collection

From the beginning of the project in August 2016, the data were collected via multiple sources of evidence for the purpose of data triangulation, construct validity, and reliability (Runeson and Höst 2009). We compiled (1) data from contextual interviews, (2) participant-observations, (3) project documentations and presentations, and (4) physical artifacts. First, interviews were conducted with all project team members to gain a comprehensive understanding of each project phase and from all perspectives. Based on the aforementioned criteria and the relevance to our research question, the interview focus was put on the usage of Design Thinking as a mindset, process, and toolbox and its impact on Requirements Engineering. We mainly asked questions about the way of eliciting and capturing needs and requirements, investigating the methods as well as perceived challenges and benefits of Design Thinking for Requirements Engineering. Since we are still interviewing project members (as the project team is growing), we leverage information from interviews with the project lead, the user research and design team, the business model team, and the scrum master and software engineering team for the preliminary results of our study. All interviews lasted between 90 and 120 minutes. For confidentiality reasons four out of our ten interviews were not audio taped, while the rest was.

Second, our goal is to capture the actual Requirements Engineering activities in the project, beyond self-reported practices or official processes. This is why we not only conducted every interview in situ, but also collected participant-observations in form of meeting protocols and memos for a more objective picture. We expected to gain mainly process-oriented insights about applying Design Thinking for Requirements

Engineering. The researchers were on site every second or third week on average to interact with the project team on a continuous basis.

Third, project documentations included mainly pictures and presentations about intermediate results and project progress as well as access to the project's Jira and Confluence system (around 1200 user stories and 7 epics).

Fourth, we collected all physical and digital artifacts, i.e. prototypes ranging from low to high resolution (13 in total). In Design Thinking elicited requirements are expressed via (throwaway) prototypes which makes them a crucial part for our research evaluation.

### 7.2.3 Data Analysis

We relied on a team-based research approach and applied thematic coding as the study's analysis strategy (Runeson and Höst 2009). Each document was independently analyzed by two researchers to avoid subjective interpretation and enhance the validity of our study (Yin 2011). We applied descriptive, in vivo, as well as process coding to cover timing aspects of the material (Runeson and Höst 2009). We examined the raw data (interview transcripts, documentations, participant-observations, and artifacts) in reference to the dimensions of Table 2 (see page 19 of this dissertation). We always kept an open mind with regard to these challenges (Yin 2011). We constantly cross-examined the constructs in different interviews. Thus, we were able to understand multiple viewpoints as well as reconcile and integrate them. To demonstrate rigor, two researchers in an iterative approach discussed emerging patterns and revised them to produce themes. Furthermore, we were in constant contact with the project team to challenge our emerging concepts.

## 7.3 Results

### 7.3.1 How is Design Thinking used to support Requirements Engineering?

This project was initiated based on an idea for creating a platform around the topic Photovoltaic and Battery (PVB) to be used in private homes. Because the original problem statement was rather vague, the project management decided to apply Design Thinking to better understand the problem domain before drawing conclusions on a possible solution and IT architecture. The general project setup and interdisciplinary staffing was guided by 'working streams' according to the tripartite of desirability, viability, and feasibility (Brown 2008), which included domain experts, IT and

technology experts, user researchers, business specialists, and a project lead. The project was divided into three main phases: Exploration, alpha prototyping, and market launch. Table 11 provides an overview of the activities in each project phase.

| **Phase 1: Exploration, 8 FTE** | **Phase 2: Alpha prototyping, 15 FTE** | **Phase 3: Friendly User Test and Market launch, 22 FTE** |
|---|---|---|
| - 35 qualitative contextual interviews with potential users<br>- 55 insights, 4 personas<br>- 10 technology insights session with internal experts<br>- 24 idea concepts<br>- 4 value propositions and business model descriptions<br>- 5-10 customer journeys for each value proposition<br>- 12 contextual interviews with homeowners to test customer journeys<br>- focus group with 5 craftsmen to test customer journeys<br>- market and competitor analysis, definition of strategic partners<br>- identification of potential revenue models and first draft of business plan<br>- technology screening for IT architecture, cost estimation for IT<br>- 1 high-resolution mockup including 6 core functionalities | - 21 user tests of (iterations of) the mockup of phase 1<br>- MVP defined and product alpha produced (9 functions identified)<br>- business requirements based on advanced competitor analysis defined<br>- software and system requirements for the platform are defined<br>- quantitative study with 250 customers to validate user requirements<br>- development of an alpha-version prototype in Scrum sprints (12)<br>- epics (7), user stories (>1000), mockup, and flow charts are defined | - ongoing scrum sprints (7) and functionality testing with users<br>- implementation of a prioritized scope in software<br>- friendly user test with 25 participants including testing of functional journey, usability, navigation, bug identification |

**Table 11: Project Falcon: Timeline and Goals per Stage**

**(Source: own illustration)**

In the following we describe the usage of Design Thinking for Requirements Engineering with regard to (1) objective, (2) activities, (3) roles, (4) outcome, and (5) conclusion for each phase.

#### *7.3.1.1 Results of Phase 1: Exploration*

*Objective*: The goal of the exploration phase was to understand the problem and create a clear product vision.

*Activities*: The activities followed the Design Thinking process of empathize, define, ideate, prototype, test (Doorley et al. 2018). The team empathized by conducting contextual interviews with potential users of the platform, e.g. homeowners, installers, craftsmen, and tenants to elicit their needs and requirements. The interviews addressed their experience in daily situations and motivations behind improving or renovating the house. From this research the team derived needs and insights (define), e.g. "*Installers prefer to involve known craftsmen to build the PV system to avoid quality issues.*" For a second round of interviews, the team created ideas and stimuli for interview partners, mainly homeowners and craftsmen, to test initial value propositions for the platform (ideate, prototype, test). Here the questions focused more on potential features of the platform that could be further developed based on the testing results. Customer journeys in form of storyboards were developed for the most promising value propositions. They considered activity and task flows as well as contextual usage of platform features. The creation of personas for homeowners and installers guided the process of adjusting requirements and reflecting features when users were not available. In addition, market and competitor research helped to define and understand the requirements of the target market and a potential solution. Insight sessions with internal experts of the case company helped the team to gain knowledge about technological requirements. A sequence of framing and re-framing of the problem domain based on new learnings from interviews and testing sessions with customers resulted in a shift from the initial problem statement on PVB to a more generalized home improvement platform idea. Eventually, all tested requirements and core functionalities were aggregated into a larger more complex product scenario.

*Roles*: In this phase, regardless of the expertise and assigned work, each team member was involved in the same activities while undertaking the steps of the Design Thinking process to elicit needs and requirements from customers and gain the same level of empathy for the problem statement.

*Outcome*: The final deliverable was the specification of the elicited needs and requirements in form of (1) a mockup (InVision) that visualized six core functionalities of the intended digital software platform, (2) customer journeys specifying the experience and context of a future use of the platform, and (3) a video showing the defined personas interacting with the platform.

*Conclusion*: In phase 1 Design Thinking is the leading process for all activities and team members. The clear process framework guides a deconstruction of the complexity of the initial problem statement with an iterative step-by-step approach. The requirements elicitation techniques (e.g. interviews, focus groups, throwaway prototyping) are put into a logical sequence of team-based efforts, regardless of the specific expertise of the individual team member(s), resulting in a common understanding of the problem domain and a product vision expressing the elicited requirements.

#### *7.3.1.2 Phase 2: Alpha Prototyping*

*Objective*: The goal of phase 2 was to deepen the understanding of the problem and to develop a functional alpha version prototype based on the vision of phase 1.

*Activities*: Scrum was the guiding process framework that enabled the entire team to work in sprints. While sprint zero was used to build up the programming and operational infrastructure for the digital platform (three weeks), the user research and business model team refined the outcome of phase 1 based on additional qualitative and quantitative research. Starting with sprint 1, the sprint cycle time was reduced to two weeks. In each sprint, interviews were conducted to prioritize functions and test usability and user experience of the mockup. The team used several Design Thinking tools for enhancing communication and ideation with stakeholders with the aim to advance the product vision (e.g. mood boards, ideation workshops with other domains, feedback capture grids). Based on the feedback the team translated user requirements into epics and specified user stories, flow charts, and iterated versions of the mockup.

*Roles*: The team configuration was adapted to the objectives of phase 2. Scrum master and a development team were brought on board and additional people started to work on the business model. Four project members from the exploration phase stayed in the project as part of the user research team. In this project phase task distribution according to the dimensions of desirability, viability, and feasibility was intensified. However, user researchers and business developers were still acting as a unity – the product owner in a team-based effort.

*Outcome*: The goal of this phase was achieved, when the team showed the feasibility of the platform and confirmed real interest from users and customers.

*Conclusion*: As the focus shifts from understanding the problem to designing the solution, the guiding process model of Design Thinking makes way for the development focused approach of Scrum. Still, the toolbox of Design Thinking may easily integrate methods and tools into the flexible Scrum framework to elicit, refine, and specify requirements. Similar to phase 1, this task, i.e. the role of the product owner, is seen as a team-based approach to include various perspectives in an interdisciplinary setup.

#### *7.3.1.3 Phase 3: Friendly User Test and Market Launch*

*Objective*: The objective of phase 3 is the market launch of the platform.

*Activities*: In phase 3 every activity is focused towards implementation and market launch. Scrum is the guiding development framework as the development continues. Enhanced priority is giving to defining the go-to-market strategy and a suitable offering and partnering approach. A friendly user test with participants, who match the persona profiles, is conducted to test usability and eliminate bugs.

*Roles*: The software development team is extended and a clear split between the technical team, the business model stream, and the user research team is undertaken.

*Outcome*: We cannot provide the final outcome at this point in time because the project is still ongoing. A minimum viable product for a home improvement platform introduced on the market is to be expected.

*Conclusion*: Selected Design Thinking tools still provide methodological support (e.g., for testing and ideation), yet common agile development and Requirements Engineering practices dominate. However, we observe a manifestation of Design Thinking as the predominant human-centered mindset advancing the market launch.

### 7.3.2 How does Design Thinking address Requirements Engineering challenges?

To derive some overarching findings from applying the continuous Design Thinking approach, we analyzed the case from the perspective of challenges as outlined in Table 2 (page 19).

We found evidence that Design Thinking has the potential to solve some Requirements Engineering challenges in an agile setting. Regarding *problems with customers and users*, we found similar challenges when it comes to the availability of customers.

However, because of the structured process framework that Design Thinking provides, customer interviews are relatively easily planned in advance. This might help to overcome this challenge to a certain degree. We see a positive impact of Design Thinking on the challenge of *tacit requirements knowledge* because Design Thinking supports a team-based approach to Requirements Engineering. Knowledge is, thus, more evenly distributed. In addition, different viewpoints foster a more comprehensive elicitation, which may expose tacit knowledge of stakeholders and team members more likely.

Analyzing the following three challenges revealed a mixed picture. First, we found that Design Thinking also tends to *neglect non-functional requirements* such as security or performance requirements. Nevertheless, it strongly emphasizes the elicitation of usability requirements as part of non-functional requirements. Second, we found a high likelihood of *inappropriate architecture* by using Design Thinking because of short planning time. However, risk mitigation might be achieved through the thorough upfront Design Thinking approach, which explores the problem domain and typically also includes the evaluation of suitable technologies. Third, similar to the former challenge, Design Thinking may also struggle with *imprecise effort estimates*. Some risk mitigation might be achieved through the upfront approach, which helps to define a clear product vision and scope.

We did not find supporting evidence for addressing the following two challenges. First, *minimalistic documentation* is also a key characteristic (and challenge) of Design Thinking, which is known to have difficulties with traceability (e.g. Beyhl and Giese 2016). Elicited requirements are captured on post-its or in form of prototypes, mainly in an unstructured way to advance team collaboration and process speed. Second, Design Thinking also faces *difficulties in the prioritization of requirements*. Based on our findings, we cannot propose any solutions offered by Design Thinking for this challenge.

## 7.4 Discussion

Table 12 summarizes the main activities and results of the project phase exploration, alpha prototyping, and market launch. The aim of our study was to investigate the continuous Design Thinking approach for Requirements Engineering in software-

intensive projects. As this is a new area of investigation, which has not been addressed investigated yet, we set out to provide a first understanding by using empirical evidence.

| ***Phase*** Duration | ***Exploration*** 3 months, 8 FTE | ***Alpha Prototyping*** 7 months, 15 FTE | ***Market Launch*** 4 months, 22 FTE |
|---|---|---|---|
| Objective | Understand the problem and create a product vision | Develop a functional alpha prototype | Market launch of the platform |
| Activities | Design Thinking as guiding process: (1) empathize, (2) define, (3) ideate, (4) prototype, and (5) test | Scrum as guiding framework; Design Thinking tools to enhance communication and ideation with stakeholders | Scrum as guiding framework; enhanced priority on defining the go-to-market strategy; friendly user tests |
| Roles | Each team member is involved into all activities to elicit needs and requirements | Onboarding of development team; business model focus; Design Thinking team transitions into product owner role | Software development team is extended; split between technical, business model, and product owner role |
| Outcome | Mockup with core functio-nalities; customer journeys define context of use | Proof of Concept demonstrates feasibility and viability of the solution | Minimum Viable Product (MVP) is ready for market entry |
| Conclusion | Design Thinking as a guiding process; requirements elicitation is a sequence of team-based efforts | Design Thinking as a toolbox; Scrum is guiding framework; part of the Design Thinking team becomes product owner | Design Thinking as a mindset; selected Design Thinking tools provide support, yet agile development practices dominate |

**Table 12: Summary of Phases 1-3**

**(Source: Hehn and Uebernickel 2019)**

The findings suggest that Design Thinking has the potential to support current Requirements Engineering practices and vice versa. We now draw conclusions from our findings and discuss them in the context of the findings of the previously introduced combined artifact-based reference model (chapter 5) and the different operationalization strategies (chapter 6).

*Design Thinking supports a seamless integration of upfront and infused Design Thinking practices.* Our findings hint at a "morphing nature" of Design Thinking in software-intensive development projects, evolving from process-guidance, via toolbox support to the manifestation of a human-centered mindset of the project team. When approaching "wicked" problems, Design Thinking starts with a structured, upfront approach to define a clear product vision. Then, it transforms into a loose bundle of tools

and a mindset that link well to common agile practices. Figure 52 visualizes this evolution of Design Thinking.

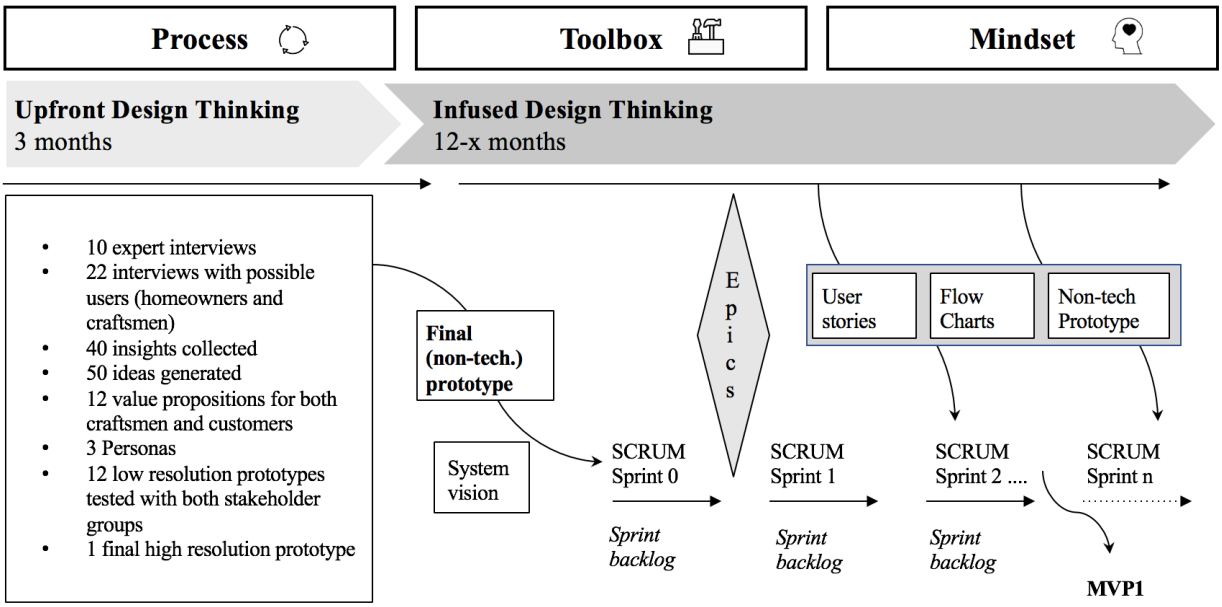



**Figure 52: Evolution from Process, via Toolbox, to Mindset**
**(Source: own illustration)**

*Design Thinking provides a structured process for requirements elicitation for "wicked" problems.* We found Design Thinking particularly supportive in approaching "wicked" problems and turning these problems into well-defined problems that can be addressed with established Requirements Engineering practices. Design Thinking provides a guideline to support the practical application of methods that are commonly used in requirements elicitation. In addition, it fosters creativity through continuous re-framing of the problem- and solution domain to find the best solution for the user. We see Design Thinking as an "extended arm" for Requirements Engineering to grasp "wicked" problems, while Requirements Engineering offers a strong framework for the integration of Design Thinking into the software development life cycle. A specific type of problem indicates to the type of approach that is to be taken. For well-defined problems, i.e. when the real-world problem is known and the solution is clear, the use of Design Thinking as a process might be ineffective and "over-engineered". Even so, we are convinced that an enhanced mindset for a more human-centered way of Design Thinking proves beneficial nevertheless and reduces the risk of deploying a system that does not meet its customers' needs.

*Design Thinking applies a team-based effort for requirements elicitation.* In other words, in Design Thinking the role of the product owner is performed by an interdisciplinary team. This results in a comprehensive requirements elicitation effort from different viewpoints to make better and more informed decisions at later stages in the development process. However, team-orientation is, like other agile approaches, highly dependent on the individual expertise of people, which makes it a critical success factor in the application of Design Thinking. Within the team low-fidelity prototypes ease communication, also with customers, which helps to expose tacit knowledge of the former and latter. However, this action-oriented working mode also results in minimalistic documentation efforts just like in common agile approaches.

*Design Thinking emphasizes the elicitation of user requirements with a special focus on usability*. Our findings show that Design Thinking puts priority on deriving a comprehensive user experience and, thus, on eliciting requirements in terms of usability, workflows tasks, and user interface. This leads to a symbiosis between Requirements Engineering and Design Thinking, as we expect higher completeness of requirements when both methodologies are applied complementarily.

### 7.4.1 Implications for Theory

By conducting a field investigation, this study discerned how Design Thinking can be combined with Requirements Engineering from the very beginning of a project until market implementation of the solution. The study contributes empirical evidence to the ongoing research on investigating the combination of Design Thinking and Requirements Engineering (e.g. Kahan et al. 2019; Martins et. al. 2019). The findings advance the understanding of how Design Thinking can support problem-solving in Requirements Engineering and answer the calls from various scholars to improve challenges in (agile) Requirements Engineering (e.g. Inayat et al. 2015; Heikkila et al. 2015).

Furthermore, this study uniquely addresses the different forms that Design Thinking can have on an operational level (*process*, *toolbox*, *mindset,* see also Figure 2, page 21), and investigates them together in one project context – a phenomenon that has been overlooked in current research. Current knowledge is largely limited to the application of one specific Design Thinking process or selected tools (e.g. Levy and Hadar 2018; Przybilla et al. 2018). By examining a real project, the importance was underlined to

investigate Design Thinking at different points in time over the duration of a project. The results of this study revealed that a continuous Design Thinking approach leads to the evolution of using Design Thinking from a guiding process, then using it as a toolbox and then implementing the principles in the mindset.

### 7.4.2 Implications for Practice

In addition to the theoretical contributions, the results of the case study present valuable insights for practitioners. The findings can be used as guidelines for companies proposing to adopt Design Thinking and for managers responsible for software-intensive development projects.

First, the findings inform practitioners about how Design Thinking can be used to support Requirements Engineering activities. An in-depth understanding of the different application possibilities of Design Thinking helps to bridge the gap between existing Requirements Engineering activities and Design Thinking. Acting upon these possibilities, project managers can use the results and plan the application of Design Thinking in software-intensive development projects more effectively.

Second, a valuable contribution for practitioners is made to gain improved appreciation for applying Design Thinking in the context of Requirements Engineering. Project members can use the different formats of Design Thinking to target and avoid some of the aforementioned challenges in agile Requirements Engineering settings.

### 7.4.3 Limitations and Outlook

The results and implications presented in this study should be regarded in light of its limitations. The empirical data from the single case study does not allow to make assumptions about the generalizability of the findings in this type of research (Myers 2009). The findings refer to the use of a continuous integration of Design Thinking and Requirements Engineering in the specific context of a utility company in Germany. Although project Falcon represents an innovative software-intensive development project in an agile setting, researchers should be careful about transferring the results to other organizational or project contexts. Further research should feel encouraged to investigate the generalizability of the findings in more detail. A positivist multiple case study with companies in different industries could be conducted to yield hypotheses for empirical evaluation.

The research focuses on the elicitation and specification of a system until market launch. A closer look at the next phase would be fruitful. Further research should focus on investigating Design Thinking for continuous software engineering (Bosch 2014). In this context, future research could build on findings from Johanssen et al. (2019), who investigated the continuous integration of user feedback to generate further ideas and features for the deployed software.

Scholars can also extend this research to development projects of safety-critical systems. Here, Design Thinking might help to understand the actual needs upfront, i.e. make the problem addressable, while Requirements Engineering then provides the methods to design the proper system architecture, safety certification requirements and so on.

## 7.5 Conclusion of Chapter

To empirically evaluate a continuous Design Thinking approach for Requirements Engineering a longitudinal case study was set up in an agile development setting. The study investigated a software-intensive development project from idea conceptualization to market-ready implementation in a large utility company in Europe. The results show that Design Thinking offers a prescriptive guideline to apply methods and create artifacts that are commonly used in Requirements Engineering to elicit stakeholder needs and requirements. The findings also indicate a seamless integration of upfront and infused Design Thinking practices for Requirements Engineering to foster a human-centered mindset. By doing so, Design Thinking has the ability to address some of the known challenges in agile Requirements Engineering.

# 8 Synthesis of Findings

The overarching objective of this dissertation was to study the usage patterns of Design Thinking for Requirements Engineering in innovative software-intensive development projects. Based on the findings presented in the dissertation, it is possible to draw a comprehensive picture of the benefits of eliciting requirements and the challenges of specifying them with Design Thinking (RQ1), understand the different artifacts produced in Design Thinking and Requirements Engineering (RQ2), and outline adequate operationalization strategies combining both (RQ3). The following sections integrate and discuss these findings.

## 8.1 Leveraging the Best of Both Worlds

With regard to the benefits and challenges of using Design Thinking for eliciting and specifying requirements (RQ1), the dissertation reveals two important insights. First, Design Thinking positively contributes to requirements elicitation by providing enriched quality, enhanced communication, and a structured discovery process. Similar to the findings of Martins et al. (2019), using Design Thinking for requirements elicitation means integrating customers continuously, putting more focus on the early phases of the process to determine the customer needs, requirements, and context, which affects the system vision and product features and functionalities. Second, Design Thinking faces output- and process-related challenges when specifying them, especially in terms of their comprehensibility and traceability. The former indicates a sole focus on user requirements while neglecting software and system requirements, which narrows the scope of using Design Thinking for requirements elicitation. The latter relates to the predominantly informal way of gathering information in Design Thinking. The identified needs and insights are often captured and analyzed in a mainly unstructured form, e.g. through notes, pictures, post-its, and low-fidelity prototypes, often due to time-saving reasons. This mainly unstructured and informal mode of specifying needs can be problematic as it might prevent the team to reflect on previously made findings and build on them for further activities. Also, a lot of knowledge of the team might stay implicit and gets lost when the team has to handover their results for implementation purposes, e.g. to software engineers (Häger et al. 2015). Both cases bear the risk of leading to a less desired outcome and, thus, to a solution that does not meet all relevant needs of its stakeholders.

In existing literature, the concrete benefits and challenges of using Design Thinking for Requirements Engineering are not well understood (Kahan et al. 2019). Taken together, the results from the multiple-case studies offer a number of interesting insights into the usage of Design Thinking in the context of Requirements Engineering. Thus, the first research question of the dissertation aimed to explore the benefits and challenges of using Design Thinking for eliciting and specifying requirements in software-intensive development projects. The findings to answer this research question were presented in chapters 3 and 4. Although Design Thinking has the means to go broad in eliciting requirements (chapter 3), specifying these requirements falls short (chapter 4). Design Thinking can positively impact requirements elicitation and can learn from Requirements Engineering to specify requirements in a more consistent and traceable manner. For creating a lasting impact of the system vision on the upcoming design and implementation activities, a balance should be found between the benefits of early experimentation as done in Design Thinking and the advantages of institutionalizing a proper structure and documentation for subsequent software engineering activities as achieved by Requirements Engineering.

## 8.2 Complementary Artifacts for a Comprehensive Blueprint

Chapter 5 provides a better understanding of the different artifacts produced in Design Thinking and Requirements Engineering (RQ2). Building on an established artifact-based reference model from Requirements Engineering (Méndez Fernández and Penzenstadler 2014a) and enriching it with DT-specific artifacts, this dissertation contributes an evaluated artifact-based reference model for Design Thinking and Requirements Engineering that can be tailored to specific project situations. The model is descriptive and prescriptive at the same time. It depicts the current nature of common DT and RE artifacts used in software-intensive development projects. It is descriptive as it provides a general understanding of the artifact landscape from both approaches. The model can also be seen as a blueprint for designing new innovative systems, which makes it also prescriptive as it provides a guideline and orientation for generating the artifacts in development projects.

The artifact-based reference model addresses calls of related literature for enhancing the understanding of how an integration of Design Thinking and Requirements Engineering could be achieved (Kahan et al. 2019; Vetterli et al. 2013; Beyhl and Giese 2016).

Comparing the artifacts shows the potential of complementary synergies when combining both approaches. Design Thinking expands the toolbox for Requirements Engineering by emphasizing artifacts for defining the relevance of the system vision. It fosters a holistic exploration of the problem context and defines precise user requirements. A prototype shapes the vision of the system. These artifacts complement the more technical-oriented artifacts from Requirements Engineering with a human-centered perspective. In addition, Requirements Engineering expands the toolbox of Design Thinking by connecting DT artifacts to later-staged software development processes. In this sense, DT-related artifacts are transformed into functionalities for technical realization. What counts in the end in Requirements Engineering is the set of elaborated requirements, while in Design Thinking, not only the prototype is the ultimate outcome, but the learning curve leading to it.

## 8.3 Three Ways to Operationalize a Combination

The first two research questions provide evidence that Design Thinking and Requirements Engineering should be used complementarily to achieve innovative yet feasible solutions. While the unified artifact model in chapter 5 described relevant DT and RE artifacts to be produced, chapters 6 and 7 suggest how to actually operationalize the combination of the two in software-intensive development projects (RQ3).

Operation modes that integrate Design Thinking into (agile) software development approaches have been proposed before (e.g. Lindberg et al. 2012; Häger et al. 2015; Dobrigkeit et al. 2018). Building on these findings and triangulating them with empirical data from industry, three operationalization strategies are proposed in which Requirements Engineering can profit from Design Thinking: (1) Run it upfront to Requirements Engineering practices (*upfront Design Thinking*), (2) infuse the Requirements Engineering process with selected Design Thinking tools (*infused Design Thinking*), or (3) apply Design Thinking principles continuously by combining the first two strategies in a flexible manner (*continuous Design Thinking*). The decision which strategy to follow depends on the project context and objective. The first strategy is recommended when the problem and solution space is unclear (e.g. *"How does the future patient support program for multiple sclerosis patients look like?"*). Following the Design Thinking process provides a guiding structure for requirements elicitation and the specification of a solution vision. The second strategy offers requirements

engineers a way to make use of selected Design Thinking methods when they feel it is necessary. Typically, these are situations in which project members face difficulties in an ongoing Requirements Engineering process that might be addressed by Design Thinking methods (e.g. *"Create ideas to optimize the user interface of our platform, making it look and feel more emotional and letting it appear less technical."*). The third strategy supports a continuous yet flexible application of the Design Thinking process and ad-hoc tools. By conducting a longitudinal case study (see chapter 7), the importance of looking at Design Thinking at different points in time was underlined. The results indicate that the continuous approach entails the evolution from using Design Thinking as a guiding process to applying it as a toolbox for adaptive support up to implementing Design Thinking principles in the mindset of project members. This strategy should be chosen when (1) a sustainable integration of both Design Thinking and Requirements Engineering is intended and (2) the project requires a continuous integration of users into the development project (e.g. *"How might we create the next generation home improvement platform?"*).

Another important finding is the need for a new role incorporating skills from both disciplines: the human-centric requirements engineer. Business analysts may leverage Design Thinking to deeply explore the system context while design thinkers may equip themselves with Requirements Engineering knowledge to better connect their results to subsequent software design. The findings of the dissertation address the call from Lauenroth (2018a) to form a new profession of what he refers to as 'digital designers.' In a manifesto Lauenroth (2018b) specifies this role in the following way: "A digital designer is someone who thinks about the future, someone who is capable of creating a vision for digital products, processes, services, business models, or even entire systems, free from technical or organizational obstacles as well as apparent reservations (outside-in thinking). Digital designers are also capable of ultimately turning this vision into reality. They transfer (technological) possibilities into (new) product/process/ service/business model/system design. To do all of this, digital designers must be skilled in design and the available technologies and be capable of interacting with all stakeholders." (p. 8) In line with this quote, the findings of this dissertation emphasize the great potential of combining Design Thinking and Requirements Engineering knowledge to create and realize innovative systems in a more creative and human-centered manner.

# 9 Theoretical Contributions

The dissertation offers theoretical contributions for research in Requirements Engineering and Design Thinking. The following sections discuss three main contributions.

## 9.1 Contributing to Artifact-based Requirements Engineering

The paradigm of artifact-orientation in Requirements Engineering research has gained increasing attention during the last years (Méndez Fernández et al. 2019). The findings of this dissertation contribute to the body of knowledge of artifact-based Requirements Engineering in general and artifact-based reference models in particular, as shown in the following.

With regard to research on artifact-based Requirements Engineering, this dissertation provides a first conceptual and empirical basis for understanding the different artifacts originating from Design Thinking, a concept, which has received considerable interest from Requirements Engineering scholars over the past few years (e.g. Martins et al. 2019). Researchers have based their work on experience reports (e.g. Caroll et al. 2018), experiments (e.g. Levy and Hadar 2018), or lessons learned from workshops (e.g. Kahan et al. 2019) and have therefore taken an activity-oriented view to analyze and merge practices from both Design Thinking and Requirements Engineering. The results of this dissertation provide a complementary perspective, i.e. an artifact-oriented view, on the topic by comparing artifacts of the two approaches. The corresponding results of each Design Thinking and Requirements Engineering are highlighted without having to take into account the variability of their processes or the compatibility of methods. The findings indicate that Design Thinking can contribute 16 concrete artifacts in addition to the Requirements Engineering approach when designing innovative software-intensive systems from a human-centered point of view.

With regard to research around artifact-based reference models, this dissertation offers a new reference model that combines DT and RE artifacts. This model joins in a line of research to address the problems of neglecting artifacts and their dependencies by using artifact-based reference models (e.g. Berenbach et al. 2009; Méndez Fernández et al. 2019; Silva et al. 2009). In this dissertation an established domain-independent artifact model for Requirements Engineering (Méndez Fernández and Penzenstadler 2014a) is

used as a frame of reference to integrate DT artifacts. The resulting model provides a result-driven view on Design Thinking and Requirements Engineering and acts as a framework for the design of innovative software-intensive systems. The novelty of this contribution is given by offering original insights into a new and, so far, undertheorized phenomenon on how to harmonize Design Thinking and Requirements Engineering (e.g. Kahan et al. 2019; Carell et al. 2018). The scientific purpose is to develop a theoretical model that integrates the human-oriented Design Thinking approach within the more technically oriented Requirements Engineering discipline.

## 9.2 Providing Prescriptive Knowledge for Problem Solving

While initial research efforts have already indicated the potential benefits of using Design Thinking in the context of Requirements Engineering (e.g. Levy and Hadar 2018), research still lacks evidence on how to actually operationalize such an integration (Kahan et al. 2019). To this end, Vetterli et al. (2013) argue that it is crucial to understand how an integration can be achieved. The dissertation addresses this gap in three ways.

First, this dissertation provides prescriptive knowledge in the form of three operationalization strategies based on industry experience and related work in software engineering (e.g. Lindberg et al. 2012; Häger et al. 2015; Przybilla et al. 2018). Specifically, the findings advance the body of knowledge of how Design Thinking is used within industry not only in a pre-phase to software engineering (cf. Martins et al. 2019) but also in later Requirements Engineering stages. As an extension of the two main existing strategies that have been used to connect Design Thinking to software development (Lindberg et al. 2012), this dissertation proposes a third strategy, which is based on findings from creativity research and is called continuous Design Thinking in this dissertation (e.g. Carlgren et al. 2016; Wölbling et al. 2012). This strategy integrates Design Thinking phases and elements throughout Requirements Engineering activities which should eventually establish a human-centered mindset.

Second, while the distinction between different concepts of Design Thinking (*process*, *toolbox*, *mindset*) is not new (e.g. Brenner et al. 2014), this dissertation advances these perspectives by analyzing how these concepts can change over time in a project context over a longer time period – a phenomenon that has been overlooked in current research. The results of this dissertation show that Design Thinking evolves from a guiding process to toolbox to a mindset. This goes beyond existing research that looks at the

conceptualization of Design Thinking in Requirements Engineering rather statically (e.g. Levy and Hadar 2018; Przybilla et al. 2018) and underlines the importance of analyzing Design Thinking in different points in time over the duration of a project.

Third, the results of this dissertation address the calls from various scholars to improve challenges faced in Requirements Engineering (e.g. Inayat et al. 2015; Heikkila et al. 2015). The concrete benefits of using Design Thinking for Requirements Engineering are indicated but not well understood in existing literature (Kahan et al. 2019; Vetterli et al. 2013). So far, his topic was addressed mainly by case studies in single domains (Levy and Hadar 2018) or by literature reviews (Martins et al. 2019). Based on cross-case insights from multiple-case studies the results of this dissertation provide an extended understanding of how Design Thinking can address common challenges in Requirements Engineering. Empirical evidence suggests a positive contribution of Design Thinking to Requirements Engineering by enhancing quality, fostering stakeholder communication, and guiding the discovery process. Going beyond existing research (e.g. Beyhl et al. 2016; Häger et al. 2015) the results of this dissertation also provide a comprehensive view on challenges when using Design Thinking, especially in terms of documentation, traceability, and scope.

## 9.3 Linking Design Thinking to IS and Software Engineering

While research considers Design Thinking to have "a lack of theoretical integration" (Badke-Schaub et al. 2010, p. 40), scholars are showing increasing interest in Design Thinking and its multi-facetted opportunities for IS research. For example, in a commentary of the MISQ, Youngjin Yoo (2017) affirms that Design Thinking can create new digital opportunities to "shape human experiences in a digital world" (p. v). The findings presented in this dissertation contribute to research on Design Thinking in IS in two main respects.

First, the understanding of how Design Thinking can support problem-solving in IS is advanced by providing empirical evidence in the context of innovative software-intensive development projects. Yoo (2017) encouraged IS scholars to "help Design Thinking practice by inventing new constructs, models, methods, and instantiations" (p.v) A contribution to increase the knowledge of problem-solving by using Design Thinking in the context of IS is made by means of an evaluated model of DT artifacts (Figure 11). While this model is an initial work product in the process of designing the

combined artifact model for Design Thinking and Requirements Engineering, it already contributes to Design Thinking research on its own. Artifact-oriented Design Thinking adds a new dimension to the existing body of knowledge around classifying Design Thinking methods, tools, and processes (e.g. Doorley et al. 2018; IDEO.org 2015; Uebernickel et al. 2015).

Second, this dissertation connects Design Thinking to the more mature research topic of Requirements Engineering. By doing so, Requirements Engineering can provide established tools and methods to further promote the conceptualization of explorative approaches like Design Thinking. The artifact-based reference model provides a common language and a connection to later-staged software design and development activities. This has been consistently criticized to be insufficiently linked to the development processes (e.g. Häger et al. 2015; Przybilla et al. 2018).

# 10 Practical Implications

This chapter discusses three major implications for practitioners highlighting the practical use of the findings of the dissertation.

## 10.1 Avoiding Common Challenges in Requirements Elicitation

Despite significant progress in the body of research, requirements elicitation is still considered an inherently difficult and complex task with multiple challenges. Design Thinking provides an approach for eliciting needs and requirements by applying qualitative research methods and producing fast and simple prototypes that converge on innovative solutions. The results of this dissertation provide a number of recommendations on how to address these challenges.

First, practitioners can leverage the prescriptive process framework of Design Thinking to support the practical application of elicitation methods which has been criticized to be insufficiently guided. Second, to avoid communication challenges, low-resolution prototypes should be considered an effective way to ease collaboration with stakeholders and expose their tacit knowledge. Fostering a continuous interaction with stakeholders helps in dealing with changing requirements and decreases invented requirements that are not based on user needs. Third, practitioners should install a team-based effort for requirements elicitation to achieve comprehensive requirements elicitation through various viewpoints. Finally, practitioners should consider the benefits of Design Thinking with a differentiated view which is also indicated by the findings of this dissertation. When implementing Design Thinking activities into their work routines, they should also be aware of the shortcomings of Design Thinking. For example, Design Thinking emphasizes the elicitation of user requirements with a special focus on usability, while neglecting other requirement classes.

## 10.2 “Doing the Right Things Right”

Practitioners often face difficulties when it comes to grasping the potential that arises from Design Thinking. By showing what it can do for Requirements Engineering and how it can be used, professionals gain a better understanding of the approach. Three main implications can be identified.

First, the results of this dissertation provide guidance when to use Design Thinking best, when not to use it all, and what should be done complementarily to design innovative software-intensive systems. The combined artifact model offers professionals a complete and consistent set of Design Thinking and Requirements Engineering artifacts that should be produced or at least taken into consideration when designing systems. Although not all of them have to be produced in each project, the overview still serves as an orientation and connection to further design and development activities. The findings show that Design Thinking is most appropriate for defining the problem context, i.e. to elicit needs and requirements. It could even be used exclusively for this. A complementary approach of both Design Thinking and Requirements Engineering is recommended when specifying the requirements layer. While some requirement types are defined by both, other ones, such as quality or deployment requirements are predominantly specified with common Requirements Engineering techniques. Requirements Engineering should be exclusively used to specify system artifacts and to provide the interface to system design activities. To summarize: Design Thinking should design the visible (from a user point of view), Requirements Engineering should specify the invisible.

Second, practitioners can also use the model to foster reflections and project discussions. This might help to circumvent a Requirements Engineering process that is often criticized to be driven solely by the subjective preconceptions of one single requirements engineer or business analyst (Méndez Fernández and Wagner 2014). However, one risk should be taken into account: Similar to other commonly used models or templates, the analyst might feel the need to "tick all boxes", which means, to force the generation of all artifacts, regardless of whether they make sense or not in a specific project context. This might reduce efficiency which is why the analyst should be sensitive and attentive to this fact.

Third, training providers can use the results of this dissertation to develop new training programs that combine Design Thinking and Requirements Engineering skills. Existing training courses for design thinkers can be enhanced by the structured technical side of Requirements Engineering and trainings for requirements engineers can be enriched by the experimental human-centered ways of Design Thinking.

## 10.3 Improving the Effectiveness of Projects

As the analysis of this dissertation shows, Design Thinking and Requirements Engineering are not mutually exclusive but rather reinforce and complement each other. Based on the presented findings, practitioners receive recommendations on *how* to use Design Thinking for Requirements Engineering in innovative software-intensive projects. Managers can evaluate their Requirements Engineering process and, thereby, improve its effectiveness and create solutions in a more human-centered fashion.

Project managers in the quest of making optimal decisions can take away the following: First, the findings provide guidance to balance resources, time, and activities for conducting more effective software-intensive development projects. Project managers can use the differentiation between the upfront, infused, and continuous strategy to steer their limited resources towards applying the most appropriate one depending on the individual project conditions. The advantages to learn quickly in the early phases with the rapid working style of Design Thinking should be weighed wisely with the demand for more structure and documentation in the later stages connecting to system design and implementation. Second, with help of the artifact model, project managers can assign completion levels and establish progress control for the creation of artifacts. Quality assurance metrics can help to objectively measure the degree of completeness of the reference model. Third, project managers can use the model as a support for defining and distinguishing responsibilities in a project. Specific project members can be directly coupled to the creation of artifacts, for which they have to take responsibility.

Project members gain an in-depth understanding of the different application possibilities of Design Thinking for Requirements Engineering. They can directly adopt the operationalization strategies in their own contexts with the awareness of the respective benefits and shortcomings. Beginners and professionals in the field of Design Thinking can use the strategies combined with the artifact model as guidelines to apply Design Thinking most effectively. Although experts are expected to intuitively do the right things based on their profound experience, the proposed artifact model and operationalization strategies can still help them to gain a more differentiated view on the various layers of software-intensive systems and to deal with complex problems more confidently. For beginners they provide a pragmatic reference to approach unknown situations in a more self-assured way.

# 11 Limitations and Avenues for Future Research

Several limitations to this dissertation need to be acknowledged and this is why the findings must be approached with appropriate caution. The following sections discuss these limitations and outline how future research may address them or extend the dissertation's results.

## 11.1 Focus on Project Level

The dissertation focuses on schemes to use Design Thinking in Requirements Engineering on a project level. As also emphasized by related research (e.g. Dobrigkeit et al. 2018; Kahan et al. 2019), the project level is an important aspect of understanding how organizations may benefit from combining Design Thinking and Requirements Engineering. Nevertheless, there is great potential for future research to extend the findings in the following directions: Enhance the current results on a *project* level by undertaking research and investigate the topic on an *individual* or at an *organizational* level.

On a project level, the created artifact-based reference model hides the variability of different processes and methods that come with each Design Thinking and Requirements Engineering. This deemed to be an appropriate foundation to compare and combine both approaches on a similar level. Accordingly, the model provides a common language for researchers and practitioners to further investigate, discuss, critique, and reflect upon the integration of Design Thinking and Requirements Engineering. However, artifact-orientation is only one important aspect when defining a comprehensive development process model, which integrates the artifacts with *activities* to be performed, *roles* to be taken, *processes* to be followed, and *tools* to support execution (Gnatz 2005; Méndez Fernández and Penzenstadler 2014a). Examining these variables and relating them to the artifact model should be addressed by further research.

On an individual level, future attention should be paid to examining individual Design Thinking needs of different team members/roles and how they should be trained with regard to those needs. In addition, and more importantly, the individual attitude towards adopting Design Thinking should be examined in the context of joining it with Requirements Engineering. According to research with regard to adoption processes

(Rogers 2003), the potential adopter weighs the expected benefits of adopting against the presumed loss of abandoning existing procedures. The successful adoption of a combined approach depends therefore on the acceptance by requirements engineers They are the ones, who are supposed to complement their activities with Design Thinking and could reject or underuse Design Thinking. In a similar context, previous authors have distinguished between Design Thinking process, toolbox, and mindset and shown that their different operationalizations influence the willingness of team mebers to use Design Thinking in combination with agile development techniques (Dobrigkeit et al. 2018; Dobrigkeit and de Paula 2019). A similar exploration with the specifics of Requirements Engineering will provide additional insights and can assist in integrating Design Thinking into Requirements Engineering frameworks more easily. In this context, investigating the characteristics of opinion leaders or gatekeepers who would probably promote using Design Thinking for Requirements Engineering could also be of interest.

On an organizational level, there have been multiple calls from scholars in the fields of IS and management research in recent years to investigate how organizations adopt and use Design Thinking (e.g. Engberts and Borgmann 2018; Lindberg et al. 2012). Based on the findings presented in this dissertation, examining the topic of organizational acceptance of a combined approach of Design Thinking and Requirements Engineering would be of great interest. Researchers of different disciplines have identified a wide range of organizational characteristics that they consider conducive to organizational adoption. Contextual variables (e.g. organizational size, age, structure) and climate variables (e.g. openness to change, external orientation, support and collaboration) have been consistently found to influence organizational adoption (Nystrom et al. 2002). They can also be regarded influential when adopting a combined approach of Design Thinking and Requirements Engineering. To test variables with this objective could be a task for future research.

## 11.2 Boundary Conditions of Innovative Software-intensive Systems

This dissertation examines Design Thinking and Requirements Engineering to design innovative software-intensive systems, for which Design Thinking can leverage its greatest potential. This makes it possible to study in detail how Design Thinking can be leveraged for the purpose of Requirements Engineering. While these conditions are in

many ways ideal for studying this phenomenon, the findings are still limited to the context of innovative software-intensive systems and should be regarded in light of the characteristics and boundary conditions of this particular type of system. In other contexts beyond innovative software-intensive systems, different findings may emerge and different operationalization strategies might be necessary depending on the specific project scope and conditions.

Thus, further research is needed to draw a more comprehensive picture of Design Thinking and Requirements Engineering beyond innovative software-intensive systems. Based on the findings presented in this dissertation, there are three major avenues that would be particularly interesting for future research: (1) How may Design Thinking be used for the development and design of safety critical systems (incl. proper system architecture and safety certification requirements)? (2) How will use patterns of Design Thinking look like for conducting continuous software engineering activities? (3) How and to what extent will findings change when investigating projects that require remote Requirements Engineering activities? Addressing these questions may help provide a more general understanding of using Design Thinking to perform Requirements Engineering activities.

## 11.3 Generalizability of Findings and Quantification

While the qualitative research design of this dissertation has the strength of providing rich contextual insights, some of the findings lack generalizability. Although the multiple-case studies have been conducted in a setup close to real life (real problem statements provided by real corporate partners), it can be argued that the sources of evidence are closely aligned with the specific project setup at the university. Due to the nature of these academic-industry collaborations, specific organizational difficulties (like internal politics or stakeholder management issues) might not be as apparent as might one expect in other settings. In addition, the Design Thinking approach, as investigated in this dissertation, is closely aligned with the specific Design Thinking approach of the University of St.Gallen, including its particular assessment policies and Design Thinking philosophy, which neglects the myriad of different Design Thinking models in practice. Supplementary studies seem necessary to validate the existing results and increase the reliability of the outcome. Therefore, scholars should build on the findings and further investigate an integration of Design Thinking and Requirements

Engineering with different Design Thinking approaches and in actual organizational settings. Interesting organizational factors to take into account could be, for example, the maturity of Design Thinking implementation, resource availability, role distribution, or organizational stability. Building on the qualitative findings of this dissertation, further attention should also be paid to quantifying their effects. Future work could derive concrete propositions from the presented operationalization strategies to validate and measure their (positive) effects on outcome and team performance.

# 12 Conclusion

It is certainly too early to consider Design Thinking a lasting contribution to Requirements Engineering but, as the results of this dissertation indicate, it indeed has left first traces. The human-centered approach of Design Thinking can be considered a powerful way to complement Requirements Engineering activities when designing innovative systems. While in recent years research has already made great strides to indicate potential benefits of bringing Design Thinking and Requirements Engineering together, it has remained mostly unclear how to actually realize a combination of the two. Hence, the objective of this dissertation was to identify usage schemes of Design Thinking in order to achieve a (more) human-centered Requirements Engineering approach. To accomplish this goal, the dissertation was organized into three research questions. Based on multiple-case studies the first research question aimed to understand the value of using Design Thinking for two main activities of Requirements Engineering, i.e. requirements elicitation and specification. The second research question combined and compared Design Thinking and Requirements Engineering artifacts for designing innovative software-intensive systems. Based on an exploratory case study the third research question derived concrete operationalization strategies on how to use Design Thinking for Requirements Engineering depending on the specific project context. Overall, this dissertation provides a number of important theoretical contributions. First, it offers empirically sound evidence beyond current research findings on how to address common challenges in Requirements Engineering with Design Thinking. Second, it contributes to ongoing research on artifact-oriented Requirements Engineering with an evaluated reference model for the design of innovative and human-centered software-intensive systems. Third, it offers prescriptive knowledge on how to use Design Thinking for Requirements Engineering. In this context the differentiated view on Design Thinking as process, toolbox, and mindset was substantiated and expanded. For practitioners the dissertation offers recommendations on how to improve the effectiveness of Requirements Engineering with the help of Design Thinking depending on the specific project context and objective. Overall, the results of this dissertation provide a solid foundation for many researchers and practitioners because they give a better understanding of *what* Design Thinking represents in the context of Requirements Engineering and *how* both approaches can be combined in innovative software-intensive development projects.

## References


Açar, A.E., Rother, D.S. 2011. "Design Thinking in Engineering Education and its Adoption in Technology-driven Startups," *Proceedings of the 8th Global Conference on Sustainable Manufacturing*, Berlin, pp. 57-62.

Ahlemann, F. and Gastl, H. 2007. "Process Model for an Empirically Grounded Reference Model Construction," *Reference Modeling for Business Systems Analysis*, P. Fettke and P. Loos (eds), Hershey, PA and London: Idea Group Publishing, pp. 1-20.

Ambler, S. W. 2018. The Initial Domain Model (http://agiledata.org/essays/agileDataModeling.html#InitialDomainModel, accessed 17 November 2019).

Aurum, A. and Wohlin, C. (eds) 2005. Engineering and Managing Software Requirements. Berlin: Springer-Verlag.

Babich, N. 2017. Prototyping 101: The Difference between Low-Fidelity and High-Fidelity Prototypes and When to Use Each. (https://theblog.adobe.com/prototyping-difference-low-fidelity-high-fidelity-prototypes-use/, accessed 7 November 2019).

Badke-Schaub, P., Roozenburg, N., Cardoso, C. 2011. "Design Thinking: A Paradigm on its Way from Dilution to Meaninglessness?," in *Interpreting Design Thinking, Special Issue of Design Studies*, S. Stewart (ed.) (32:6), pp. 39-49.

Bahill, A.T. and Madni, A.M. 2017. "Discovering System Requirements," in *Tradeoff Decisions in System Design*, A.T. Bahill and A.M. Madni (eds.), Cham: Springer, pp. 373-457.

Baxter, G. and Sommerville, I. 2011. "Socio-Technical Systems: From Design Methods to Systems Engineering," *Interacting with Computers* (23:1), pp. 4-17.

Beaudouin-Lafon, M. and Mackay, W. 2003. *Prototyping Tools and Techniques, The Human-Computer Interaction Handbook*. Hillsdale, NJ: L. Erlbaum Associates Inc., pp. 1006-1031.

Becker, J., Delfmann, P., Knackstedt, R. 2007. "Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models," *Reference Modeling*, J. Becker and P. Delfmann (eds), Heidelberg: Physica, pp. 27-58.

Berenbach, B., Paulish, D. J., Kazmeier, J. and Rudorfer, A. 2009. *Software and Systems Requirements Engineering: In Practice*. McGrawHill: New York.

Berger, W. 2012. The Secret Phrase Top Innovators Use. *Harvard Business Review*, September (https://hbr.org/2012/09/the-secret-phrase-top-innovato, accessed 13 August 2019).

Beyer, H. and Holtzblatt, K. 1999. "Contextual design," *Interactions* (6:1), pp. 32-42.

Beyhl, T., Berg, G., Giese, H. 2014. "Connecting Designing and Engineering Activities," in *Design Thinking Research, Understanding Innovation*, H. Plattner, C. Meinel, and L. Leifer (eds), Cham: Springer-Verlag, pp. 153-182.

Beyhl, T. and Giese, H. 2015. “Connecting Designing and Engineering Activities II,” in *Design Thinking Research, Understanding Innovation*, H. Plattner, C. Meinel, and L. Leifer (eds), Cham: Springer-Verlag, pp. 211-239.

Beyhl, T. and Giese, H. 2016. “Connecting Designing and Engineering Activities III,” in *Design Thinking Research, Understanding Innovation*, H. Plattner, C. Meinel, and L. Leifer (eds), Cham: Springer-Verlag, pp. 265-290.

Bitner, M. J., Ostrom, A. L., Morgan, F. N. 2008. “Service Blueprinting: A Practical Technique for Service Innovation,” *California Management Review* (50:3), pp. 66-94.

Blomberg, J.L. 1988. “The Variable Impact of Computer Technologies on the Organization of Work Activities,” in *Computer-Supported Cooperative Work: A Book of Readings*. San Francisco, CA: Morgan Kaufmann Publishers Inc., pp. 771-789.

Boland, R.J. and Collopy, F. 2004. “Design Matters for Management,” Managing as Designing, R.J. Boland and F. Collopy (eds.), Stanford, pp. 3-18.

Bosch, J. 2014. *Continuous Software Engineering: An Introduction*. Cham: Springer International Publisihing Switzerland.

Bostrom, R.P. and Heinen, J.S. 1977. “MIS Problems and Failures: A Socio-Technical Perspective. Part I: The Causes,” *MIS Quarterly*, pp. 17-32.

Bowen, G.A. 2009. “Document Analysis as a Qualitative Research Method,” *Qualitative Research Journal* (9:2), pp. 27-40.

Braun, P., Broy, M., Houdek, F., Kirchmayr, M., Müller, M., Penzenstadler, B., Pohl, K. and Weyer, T. 2014. “Guiding Requirements Engineering for Software-Intensive Embedded Systems in the Automotive Industry,” *Computer Science - Research and Development* (29), pp. 21-43.

Brenner, W., Uebernickel, F., and Abrell, T. 2016. “Design Thinking as Mindset, Process, and Toolbox,” in *Design Thinking for Innovation: Research and Practice*, W. Brenner and F. Uebernickel (eds), Cham: Springer International Publishing Switzerland, pp. 3-21.

Brown, T. 2008. “Design Thinking,” *Harvard Business Review* (86:6), pp. 84-92.

Brown, T. 2009. *Change by Design, How Design Thinking Transforms Organizations and Inspires Innovation*, New York: HarperBusiness.

Browne, G. J. and Rogich, M. B., 2001. “An Empirical Investigation of User Requirements Elicitation: Comparing the Effectiveness of Prompting Techniques,” *Journal of Management Information Systems* (17:4), pp. 223-249.

Broy, M. 2006. “Requirements Engineering as a Key to Holistic Software Quality,” in *Proceedings of the 21th International Symposium on Computer and Information Sciences*, pp. 24-34.

Broy, M. 2015. “Rethinking Nonfunctional Software Requirements,” in *Computer* (48:5), pp. 96-99.

Buchanan, R. 1992. “Wicked Problems in Design thinking,” *Design Issues* (8:2), pp. 5-21.

Carell, A., Lauenroth, K, and Platz, D. 2018. “Using Design Thinking for Requirements Engineering in the Context of Digitalization and Digital Transformation: A Motivation and an Experience Report,” in The Essence of Software Engineering, V. Gruhn and R. Striemer (eds), Cham: Springer, pp. 107-120.

Carlgren, L., Rauth, I., and Elmquist, M. 2016. “Framing Design Thinking: The Concept in Idea and Enactment: Creativity and Innovation Management,” *Creativity and Innovation Management* (25:1), pp. 38-57.

Carrizo, D., Dieste, O., and Juristo, N. 2014. “Systematizing Requirements Elicitation Technique Selection,” *Information and Software Technology* (56:6), pp. 644-669.

Carroll, N. and Richardson, I. 2016. “Aligning Healthcare Innovation and Software Requirements through Design Thinking,” *Proceedings of the International Workshop on Software Engineering in Healthcare Systems*, Austin, Texas, pp. 1–7

Chung, L. and do Prado Leite, J. C. S. 2009. “On Non-Functional Requirements in Software Engineering,” in Conceptual Modeling: Foundations and Applications. *Lecture Notes in Computer Science*, A.T. Borgida, V.K. Chaudhri, P. Giorgini, E.S. Yu (eds), Heidelberg: Springer, pp. 363-379.

Cockburn, A. 2000. *Writing Effective Use Cases.* Addison-Wesley Longman Publishing Co., Inc.

Collins, H. 2010. *The Theory and Practice of Research for the Creative Industries*. Switzerland: AVA Publishing SA.

Cooper, R. and Foster, M. 1971. “Sociotechnical Systems,” *American Psychologist* (26), pp. 467-474.

Cooper, A., Reimann, R., Cronin, D., and Noessel, C. 2014. *About Face: The Essentials of Interaction Design.* Indianapolis, IN: Wiley Publishing Inc.

Corral, L. and Fronza, I. 2018. “Design Thinking and Agile Practices for Software Engineering: An Opportunity for Innovation,” *Proceedings of the 19th Annual SIG Conference on Information Technology Education*, Fort Lauderdale, Florida, USA, pp. 26-31.

Crai, L. 2003. “Right-Rapid-Rough. The Ideo Way”, *The NASA Academy of Program and Project Leadership*, pp. 12-16.

CS Software Engineering Note 2: Software Requirements. http://www.inf.ed.ac.uk/teaching/courses/cs2/LectureNotes/CS2Ah/SoftEng/se02.pdf (Accessed: 23.09.2017)

Dam, R. and Siang, T. 2017. “Learn How to Use the Best Ideation Methods: Challenge Assumptions,” Interaction Design Foundation (https://www.interaction-design.org/literature/article/learn-how-to-use-the-best-ideation-methods-challenge-assumptions, accessed 15 November 2019).

Davey, B. and Parker, K. R. 2015. "Requirements Elicitation Problems: A Literature Analysis," *Issues in Informing Science and Information Technology* (12), pp. 71-82.

de Carvalho Souza, C.L. and Silva, C. 2015. "An Experimental Study of the Use of Design Thinking as a Requirements Elicitation Approach for Mobile Learning Environments," *CLEI Electronic Journal* (18:1), pp.1-18.

Dobrigkeit, F., de Paula, D., and Uflacker, M. 2018. "InnoDev - A Software Development Methodology Integrating Design Thinking, Scrum and Lean Startup," in *Design Thinking – Research Looking Further: Design Thinking Beyond Solution-Fixation*, H. Plattner, C. Meinel, L. Leifer (eds), Cham: Springer-Verlag, pp. 199-228.

Dobrigkeit F. and de Paula, D. 2019. "Design Thinking in Practice: Understanding Manifestations of Design Thinking in Software Engineering," in *Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19),* Tallinn, Estonia: ACM, pp. 1059-1069.

Doorley, S., Holcomb, S., Klebahn, P., Segovia, K., and Utley, J. 2018. *Design Thinking Bootleg*, d.school at Stanford University, CA.

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

Ebert, C. and Wieringa. R. 2005. "Requirements Engineering: Solutions and Trends," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin (eds), Berlin: Springer-Verlag, pp. 453-476.

Eisenhardt, K. M. 1989. "Building Theories from Case Study Research," *Academy of Management Review* (14:4), pp. 532-550.

Emam, K. E., and Koru, A. G. 2008. "A Replicated Survey of IT Software Project Failures," *IEEE Software* (25:5), pp. 84-90.

Emery, F.E. and E. Trist. 1960. "Socio-technical Systems," *Management Sciences Models and Techniques*, C.W. Churchman and M. Verhulst (eds), Vol. 2, London: Pergamon Press, pp. 83-97.

Engberts, F. and Borgman, H. 2018. "Application of Design Thinking for Service Innovation: Current Practices, Expectations and Adoption Barriers," *Proceedings of the Hawaii International Conference on System Sciences (HICCS)*, Hawaii: AIS, pp. 1611-1620.

Fettke, P. and Loos, P. 2004. "Reference Modelling Research," Long version of a paper (Rep. No. 16). ISYM: *Information Systems & Management*. Mainz: Universität Mainz.

Fettke, P. and Loos, P. 2007. "Perspectives on Reference Modeling," *Reference Modeling for Business Systems Analysis*, P. Fettke and P. Loos (eds), Hershey, PA and London: Idea Group Publishing, pp. 1-20.

Fleischmann, A., Geisberger, E., and Pister, M. 2004. "Herausforderungen für das Requirements Engineering eingebetteter Systeme," *Technical Report TUM-I4014*, Technische Universität München.

Forrester 2018. "The Total Economic Impact™ Of IBM's Design Thinking Practice. How IBM Drives Client Value and Measurable Outcomes with its Design Thinking Framework" *A Forrester Total Economic Impact™ Study,* commissioned by IBM.

Fraser, H. 2011. "Business Design: Becoming a Bilateral Thinker," *Rotman Magazine*, Winter, pp. 70-76.

Friedrich, J., Hammerschall, U., Kuhrmann, M. and Sihling, M. 2008 "Das VModell XT," *Informatik im Fokus*. Berlin: Springer-Verlag.

Geisberger, E., Broy, M., Berenbach, B., Kazmeier, J., Paulish, D. and Rudorfer, A. 2006. Requirements Engineering Reference Model (REM). *Technical Report TUM-I0618*, Technische Universität München.

Glinz, M. 2014. "A Glossary of Requirements Engineering Terminology," *International Requirements Engineering Board (IREB)*, Version 1.6.

Glinz, M. 2007. "On Non-Functional Requirements," *Proceedings of the 15th IEEE International Conference on Requirements Engineering (RE' 07)*, IEEE Computer Society, pp. 21-26.

Glinz, M. and Wieringa, R. J. 2007. "Stakeholders in Requirements Engineering," *IEEE Software* (28:1), pp. 18-20.

Gnatz, M. 2005. *Vom Vorgehensmodell zum Projektplan*. PhD thesis, Technische Universität München.

Gutzwiller, T. 1994. *Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen*. Heidelberg: Physica.

Greenbaum, J. and Kyng, M. 1991. *Design at Work: Cooperative Design of Computer Systems*. LEA, Hillsdale, NJ.

Häger, F., Kowark, T., Krüger, J., Vetterli, C., Übernickel, F., Uflacker, M. 2015. "DT@Scrum: Integrating Design Thinking with Software Development Processes," in *Design Thinking Research: Understanding Innovation*, H. Plattner, C. Meinel, L. Leifer (eds), Cham: Springer-Verlag, pp. 263-289.

Hadar, I., Soffer, P., Kenzi, K. 2014. "The Role of Domain Knowledge in Requirements Elicitation via Interviews: An Exploratory Study," *Requirements Engineering* (19:2), pp. 143-159.

Hammerschall, U. 2008. "Flexible Methodenintegration in anpassbare Vorgehensmodelle," PhD thesis, Technische Universität München.

Hansen, S., Berente, N., Lyytinen, K. 2009. "Requirements in the 21st Century: Current Practice and Emerging Trends," in *Design Requirements Engineering: A Ten-Year Perspective*, K. Lyytinen, P. Loucopoulos, J. Mylopoulos, B. Robinson (eds.), Heidelberg: Springer, pp. 44-87.

Harte, R., Glynn, L., Rodríguez-Molinero, A., Baker, P. M., Scharf, T., Quinlan, L. R., and ÓLaighin, G. 2017. "A Human-Centered Design Methodology to Enhance the Usability, Human Factors, and User Experience of Connected Health Systems," *JMIR Human Factors* (4:1), e8.

Hartson, R., Pyla, P.S. 2012. *The UX Book*. Morgan Kaufmann.

Heath, C., Luff, P. 1991. "Collaboration and Control: Crisis Management and Multimedia Technology in London Underground Line Control Rooms. *Computer Supported Cooperative Work* (1), pp. 69-94.

Hehn, J. and Uebernickel, F. 2018a. "Towards an understanding of the Role of Design Thinking for Requirements Elicitation – Findings from a Multiple-Case Study," in *Proceedings of the 24th Americas Conference on Information Systems (AMCIS 2018)*, New Orleans, USA: AIS

Hehn, J. and Uebernickel, F. 2018b. "The Use of Design Thinking for Requirements Engineering – An Ongoing Case Study in the Field of Innovative Software-Intensive Systems," in *Proceedings of the 26th IEEE International Requirements Engineering Conference (RE'18),* Banff, Canada: IEEE.

Hehn, J. and Uebernickel, F. 2019. "The Use of Design Thinking for Requirements Engineering – An Ongoing Case Study in the Field of Innovative Software-Intensive Systems," in *Lecture Notes: 49*, *Jahrestagung der Gesellschaft für Informatik*, Extended Abstract.

Hehn, J., Uebernickel, F., Stöckli, E., and Brenner, W. 2018. "Towards Designing Human-Centered Information Systems: Challenges in Specifying Requirements in Design Thinking Projects," in *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2018)*, Lüneburg, Germany: AIS.

Hehn, J., Uebernickel, F., and Herterich, M. 2018. "Design Thinking Methods for Service Innovation – A Delphi Study," in *Proceedings of the 22nd Pacific Asia Conference on Information Systems (PACIS 2018)*, Yokohama, Japan: AIS.

Hehn, J., Mendez, D., Uebernickel, F., Brenner, W., and Broy, M. 2020 (forthcoming). "On Integrating Design Thinking for a Human-centered Requirements Engineering," *IEEE Software, Special Issue Design Thinking*.

Heikkila, V.T., Damian, D., Lassenius, C., and Paasivaara, M. 2015. "A Mapping Study on Requirements Engineering in Agile Software Development," in *41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA'15)*, pp. 199-207.

Hickey, A. M. and Davis, A. M. 2004. "A Unified Model of Requirements Elicitation," *Journal of Management Information Systems* (20:4), pp. 65-84.

Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Essay in Information Design Science systems," *MIS Quarterly* (28:1), pp. 75-105.

Hummel, B. and Thyssen, J. 2009. "Behavioural Specification of Reactive Systems Using Stream-Based I/O Tables," *Proceedings of the 7th IEEE International*

*Conference on Software Engineering and Formal Methods (SEFM 09)*, IEEE Computer Society, pp 137-146.

Hull, E., Jackson, K. and Dick, J. 2011. *Requirements Engineering*. London: Springer-Verlag.

Kröper, M., Lindberg, T., Meinel, C. 2010. "Interrelations between Motivation, Creativity and Emotions in Design Thinking Processes – An Empirical Study Based on Regulatory Focus Theory", *Proceedings of the 1st International Conference on Design Creativity*, Kobe, pp. 97-104.

Kuhrmann, M. and Hammerschall, U. 2008. "Anpassung des V-Modell XT – Leitfaden zur organisationsspezifischen Anpassung des V-Modell XT," *Technical Report TUM-I0831*, Technische Universität München.

IBM Corporation 2018: "Enterprise Design Thinking by IBM. Toolkit: Assumptions and Questions" (https://www.ibm.com/design/thinking/page/toolkit/activity/assumptions-and-questions, accessed 13 October 2019).

IDEO LLC. 2012. "Design Thinking for Educators," 2nd Edition. (http://designthinkingforeducators.com/, accessed 14 September 2019).

IDEO.org. 2015. Field Guide to Human Centered Design. (http://www.designkit.org/resources/1, accessed 3 January 2019).

IEEE. 1998. "IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998," *Technical Standard IEEE Std 830-1998*, The Institute of Electrical and Electronics Engineers, Inc.

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., and Shamshirband, S. 2015. "A Systematic Literature Review on Agile Requirements Engineering Practices and Challenges," *Computers in Human Behavior* (51), pp. 915-929.

ISO, International Standards Organisation 2010. *Ergonomics of Human–System Interaction – Part 210: Human-centered Design for Interactive Systems.* ISO, Geneva, Switzerland.

ITMP. 2015. "Project Documentation," *Internal Report*, unpublished.

ITMP. 2017. "Project Documentation," *Internal Report*, unpublished.

ITMP. 2018. "Project Documentation," *Internal Report*, unpublished.

Jarke, M., Loucopoulos, P., Lyytinen, K., Mylopoulos, J., Robinson, W. 2011. "The Brave New World of Design Requirements," *Information Systems* (36:7), pp. 992-1008.

Jönsson, P. and Lindvall, M. 2005. "Impact Analysis," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin (eds), Springer-Verlag Berlin Heidelberg 2005, pp. 117-142.

Johanssen, J.O., Kleebaum, A., Bruegge, B., and Paech, B. 2019. "How do Practitioners Capture and Utilize User Feedback during Continuous Software Engineering?" in *Proceedings of the 27th IEEE International Requirements Engineering Conference (RE'19)*, Jeju Island, South Korea: IEEE.

Kahan, E., Genero, M. and Oliveros, A. 2019. “Challenges in Requirement Engineering: Could Design Thinking Help?,” *Quality of Information and Communications Technology. Communications in Computer and Information Science*, Vol. 1010, Cham: Springer, pp. 79-86.

Karlsson, L., Dahlstedt, Å., Dag, J. N., Regnell, B., and Persson, A. 2002. “Challenges in Market-Driven Requirements Engineering,” *Proceedings of the Eighth International Workshop on Requirements Engineering*, pp. 37-49.

Kolko, J. 2015. “Design Thinking Comes of Age,” *Harvard Business Review* (93:9), pp. 67-71.

Kupfer, M. and Hadar, I. 2008. “Understanding and Representing Deployment Requirements for Achieving Non-Functional System Properties,” *The 1st International Workshop on Non-Functional System Properties in Domain-Specific Modeling Languages*, Toulouse, France.

Lauenroth, K. 2018a. “Softwareentwicklung braucht mehr Gestaltungkompetenz: Digital Design als neues Rollenideal im Software Engineering,” in *Software Engineering and Software Management*, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, and J.-P. Steghöfer (eds), pp. 69-71.

Lauenroth, K. 2018b. “Digital Design Manifesto: A Self-confident Design Profession is the Key to Successful and Sustainable Digitalization, Berlin: Bitkom, (https://www.digitaldesign.org/content/1-home/digital-design-manifesto.pdf, accessed 8 November 2019).

Leavitt, H.J. 1965. “Applied Organizational Change in Industry: Structural, Technological and Humanistic Approaches,” *Handbook of Organizations*, pp. 1144-1170.

Leavy, B. 2010. “Design Thinking – A New Mental Model of Value Innovation,” *Strategy and Leadership*, (38:3), pp. 5-14.

Levy, M. and Hadar, I. 2018. “The Importance of Empathy for Analyzing Privacy Requirements,” *Proceedings of the IEEE 5th International Workshop on Evolving Security & Privacy Requirements Engineering (ESPRE),* Banff, Canada, pp. 9-13.

Leonard, D. and Rayport, J.F. 1997. “Spark Innovation Through Empathic Design,” *Harvard Business Review* (75:6), pp. 102-113.

Liedtka, J. and Ogilvie, T. 2011. *Designing for Growth: A Design Thinking Tool Kit for Managers.* Columbia University Press.

Lincoln, Y.S. and Guba, E.G. 1985. *Naturalistic Inquiry*. Beverly Hills, CA: Sage.

Lindberg, T., Meinel, C., and Wagner, R. 2011. “Design Thinking: A Fruitful Concept for IT Development?” *Design Thinking: Understand – Improve – Apply*, H. Plattner, C. Meinel, L. Leifer (eds), Cham: Springer-Verlag, pp. 3-18.

Lindberg, T., Köppen, E., Rauth, I., and Meinel, C. 2012. “On the Perception, Adoption and Implementation of Design Thinking in the IT Industry,” *Design Thinking*

*Research*, H. Plattner, C. Meinel, L. Leifer (eds), Cham: Springer-Verlag, pp. 229-240.

Lyytinen, K., Newman, M. 2008. "Explaining Information Systems Change: A Punctuated Sociotechnical Change Model," *European Journal for Information Systems* (17), pp. 589-613.

Martin, R. 2009. *The Design of Business. Why Design Thinking is the Next Competitive Advantage*, Boston: Harvard Business Review Press.

Maguire, M. and Bevan, N. 2002. "User Requirements Analysis," in: Usability, J. Hammond, T. Gross, J. Wesson, (eds), Springer, Boston, MA, pp. 133–148.

Maiden, N., Gizikis, A., and Robertson, S. 2004. "Provoking Creativity: Imagine What Your Requirements Could Be like," *IEEE Software* (21:5), pp. 68-75.

Martins, H.F, de Oliveira Junior, A.C., Canedo, E.D., Dias Kosloski, R.A., Paldês, R.A., and Oliveira, E.C. 2019. "Design Thinking: Challenges for Software Requirements Elicitation," in *Information* (10), pp. 1-27.

ME310. 2010. ME310 Design Innovation at Stanford University. Micro Cycle. https://web.stanford.edu/group/me310/me310_2016/, accessed January 13, 2019.

Méndez Fernández, D. and Penzenstadler, Birgit. 2014a. "Artefact-based Requirements Engineering: The AMDiRE Approach," Requirements Engineering (20:4), pp. 405-434.

Méndez Fernández, D. and Penzenstadler, Birgit. 2014b. Supplementary Material from "Artefact-based Requirements Engineering: The AMDiRE Approach," *Requirements Engineering* (20:4), pp. 405-434 (http://www.mendezfe.org/publications/, accessed 6 October 2019).

Méndez Fernández, D. and Wagner, S. 2014. "Naming the Pain in Requirements Engineering: A Design for a Global Family of Surveys and First Results from Germany," *Information and Software Technology* (57), pp. 616-643.

Méndez Fernández, D., Böhm, W., Vogelsang, A., Mund, J., Broy, M. Kuhrmann, M., Weyer, T. 2019. "Artifacts in Software Engineering: A Fundamental Positioning," *International Journal on Software and Systems Modeling* (18:5), pp. 2777-2786.

Mich, L., Anesi, C., and Berry, D. M. 2005. "Applying a Pragmatics-Based Creativity-Fostering Technique to Requirements Elicitation," *Requirements Engineering* (10:4), pp. 262-275.

Mitra, T. 2008. "Develop the Architecture Overview. Three Complementary Views Lay the Foundation," Documenting Software Architecture, Part 3 (https://www.ibm.com/developerworks/library/ar-archdoc3/ar-archdoc3-pdf.pdf, accessed 12 October 2019).

Monk, A. and Howard, S. 1998. "Methods and Tools: The Rich Picture: A Tool for Reasoning About Work Context," *Interactions Magazine* (5:2), pp. 21-30.

Mumford, E. 2006. "The Story of Socio-Technical Design: Reflections in its Successes, Failures and Potential," *Information Systems Journal* (16), pp. 317-342.

Myers, M. D. 2009. *Qualitative Research in Business and Management*, Thousand Oaks, CA: Sage Publications.

Newman, P., Ferrario, M.A., Simm, W., Forshawz, S., Friday, A., Whittle, J. 2015. "The Role of Design Thinking and Physical Prototyping in Social Software Engineering," *Proceedings of the 37th International Conference on Software Engineering*, Florence, Italy, (2), pp. 487-496.

Nuseibeh, B. and Easterbrook, S. 2000. "Requirements Engineering: A Roadmap," *Proceedings of the International Conference of Software Engineering*, pp. 35-46.

Nystrom, P.C., Ramamurthy, K., and Wilson, A.L. 2002. "Organizational Context, Climate and Innovativeness: Adoption of Imaging Technology," *Journal of Engineering and Technology Management* (19:3), pp. 221-247.

Okoli, C. and Pawlowski, S. D. 2004. "The Delphi Method as a Research Tool: An Example, Design Considerations and Applications," *Information and Management* (42:1), pp. 15-29.

Orlikowski, W.J. 2000. "Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations," *Organizational Science* (11), pp. 404-428.

Osterwalder, A. and Pigneur, Y. 2010. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*, Hoboken, New Jersey: John Wiley and Sons.

Paetsch, F., Eberlein, A., and Maurer, F. 2003. "Requirements Engineering and Agile Software Development," *Proceedings of the 12th IEEE International Workshop Enabling Technologies: Infrastructure for Collaborative*, pp. 308-313.

Paré, G., Cameron, A.-F., Poba-Nzaou, P., and Templier, M. 2013. "A Systematic Assessment of Rigor in Information Systems Ranking-Type Delphi Studies," *Information and Management* (50:5), pp. 207-217.

Peffers, K., Tuunanen, T., Rothenberger, M.A., and Chatterjee, S. 2007. "A Design Science Methodology for Information Systems Research," *Journal of Management for Information Systems* (24:3), pp. 45-77.

Penzenstadler, B. 2017a. Lecture: Requirements Engineering: Business Case Analysis, Requirements Engineering (CECS 590), California State University Long Beach, delivered 2017.

Penzenstadler, B. 2017b. Lecture: Requirements Engineering: Goals and Constraints, Requirements Engineering (CECS 590), California State University Long Beach, delivered 2017.

Penzenstadler, B. 2017c. Lecture: Requirements Engineering: Domain Model, Requirements Engineering (CECS 590), California State University Long Beach, delivered 2017.

Penzenstadler, B. 2017d. Lecture: Requirements Engineering: Non-functional Requirements, Requirements Engineering (CECS 590), California State University Long Beach, delivered 2017.

Przybilla, L., Schreieck, M., Klinker, K., Pflügler, C., Wiesche, M., and Krcmar, H., 2018. “Combining Design Thinking and Agile Development to Master Highly Innovative IT Projects,” in *Projektmanagement und Vorgehensmodelle 2018 – Der Einfluss der Digitalisierung auf Projektmanagementmethoden und Entwicklungsprozesse*, M. Mikuzs, A. Volland, M. Engstler, E. Hanser, and O. Linssen, (eds), Bonn: Gesellschaft für Informatik, pp. 113-124.

Roberston, S. and Roberston, J. 2013. *Mastering the Requirements Process: Getting Requirements Right*, Pearson Education Inc.

Robertson J. and Robertson, S. 2018. Volere Requirements Specification Templates - Edition 18. (https://www.volere.org/templates/volere-requirements-specification-template/, accessed 15 October 2019).

Rogers, E.M. 2003. *Diffusion of Innovations*, 5th ed., New York: Free Press.

Rolland, C. and Salinesi, C. 2005. “Modeling Goals and Reasoning with Them,” in Engineering and Managing Software Requirements, A. Aurum and C. Wohlin (eds), Springer-Verlag Berlin Heidelberg 2005, pp. 189-217.

Rouse, W. B. 2007. *People and Organizations: Explorations of Human-Centred Design*, Hoboken: Wiley.

Runeson P. and Höst, M. 2009. “Guidelines for Conducting and Reporting Case Study Research,” *Software Engineering: Empirical Software Engineering* (14: 2), pp.131-164.

Sandino, D., Matey, L.M., Vélez, G. 2013. “Design Thinking Methodology for the Design of Interactive Real-time Applications,” *DUXU 2013*. LNCS, vol. 8012, A. Marcus (ed.), pp. 583–592. Heidelberg: Springer

Schmiedgen, J., Rhinow, H., Köppen, E., and Meinel, C. 2015. “Parts Without a Whole? – The Current State of Design Thinking Practice in Organizations,” *Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam*, Study Report No. 97.

Schlagheck, B. 2000. *Object-oriented Reference Models for Process and Project Controlling. Foundation—construction—fields of application.* Wiesbaden: Deutscher Universitäts-Verlag.

Schön, D.A. 1984. *The Reflective Practitioner: How Professionals Think in Action*, New York: Basic Books.

Schön, E.M.; Thomaschewski, J.; Escalona, M.J. 2017. “Agile Requirements Engineering: A Systematic Literature Review,” *Computer Standards & Interfaces* (49), pp. 79-91.

Schütte, R. 1998. *Grundsätze ordnungsmäßiger Referenzmodellierung, Konstruktion konfigurations- und anpassungsorientierter Modelle*. Wiesbaden: Springer.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., and Elmqvist, N. 2018. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Harlow: Pearson.

Silva, M., Oliveira, T., and Bastos, R. 2009. “Software Artifact Meta-model: An Approach to Software Artifact Authoring,” *Proceedings of the XXIII Brazilian Symposium on Software Engineering*, pp. 306-316.

Simon, H. A. 1969. The Sciences of the Artificial, Cambridge, MA: MIT Press.

Soledade, M.P.; Freitas, R., Peres, S.M., Fantinato, M., Steinbeck, R., Araújo, U. 2013. “Experimenting with Design Thinking in Requirements Refinement for a Learning Management System,” in: *Anais do Simpósio Brasileiro de Sistemas de Informação*, pp. 1-13.

Sommerville, I. and Dewsbury, G. 2007. “Dependable Domestic System Design: A Sociotechnical Approach,” *Interacting with Computers* (19), pp. 438-456.

Stake, R. E. 2005. *Multiple Case Study Analysis*, New York: Guilford Press.

Stickdorn, M. and Schneider, J. 2012. *This Is Service Design Thinking: Basics, Tools, Cases*, Amsterdam: BIS Publishers.

Sun, Y. and Kantor, P.B. 2006. “Cross-Evaluation: A New Model for Information System Evaluation,” *Journal of the American Society for Information Science and Technology* (57:5), pp. 614-628.

Taylor, F.W. 1911. *Principles of Scientific Management*. NY: Harper & Row, New York.

Uebernickel, F., Brenner, W., Naef, T., Pukall, B., and Schindlholzer, B. 2015. *Design Thinking: Das Handbuch*, Frankfurt: Frankfurter Allgemeine Buch.

University of St.Gallen, DT@HSG. 2012. “Final Project Documentation,” *Internal Report*, unpublished.

University of St.Gallen, DT@HSG. 2014. “Final Project Documentation,” *Internal Report*, unpublished.

University of St.Gallen, DT@HSG. 2016. “Final Project Documentation,” *Internal Report*, unpublished.

Van Lamsweerde, A. 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications.* West Sussex, England: John Wiley and Sons Ltd.

Venkatesh Sharma, K. and Kumar, P.V. 2013. “A Method to Risk Analysis in Requirement Engineering Using Tropos Goal Model with Optimized Candidate Solutions,” *International Journal of Computer Science Issues* (10:6), pp. 250-259.

Vetterli, C., Brenner, W., Uebernickel, F., Petrie, C. 2013. “From Palaces to Yurts: Why Requirements Engineering Needs Design Thinking,” *IEEE Internet Computing* (17:2), pp. 91-94.

Walker, M., Takayama, L., and Landay, J.A. 2002. “High-Fidelity or Low-Fidelity, Paper or Computer? Choosing Attributes when Testing Web Prototypes,” *Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting*, pp. 661-665.

Webster, J. and Watson, R.T. 2002. "Analyzing the Past to Prepare for the Future: Writing a Literature Review," *MIS Quarterly* (26:2), pp. xiii-xxiii.

Whetton, S. 2005. *Health Informatics: A Socio-technical Perspective*. South Melbourne, Australia: Oxford University Press.

Wiegers, K. 2003. *Software Requirements*. Redmond,WA, USA: Microsoft Press.

Winter, R. and Schelp, J. 2006. "Reference Modeling and Method Construction: A Design Science Perspective," *Proceedings of the 21st ACM Symposium on Applied Computing (SAC 06)*, New York: ACM, pp. 1561-1562.

Wölbling, A., Krämer, K., Buss, C.N., Dribbisch, K., LoBue, P., and Taherivand, A. 2012. "Design Thinking: An Innovative Concept for Developing User-Centered Software," in *Software for People: Fundamentals, Trends and Best Practices*, A. Maedche, A. Botzenhardt, and L. Neer (eds)., Berlin Heidelberg: Springer, pp. 121-136.

Yilmaztürk, N. 2005. "Good Quality" Requirements in Unified Process," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin (eds), Springer-Verlag Berlin Heidelberg 2005, pp. 373-403.

Yin, R. K. 2011. *Applications of Case Study Research*, Los Angeles: SAGE.

Yin, R. K. 2014. *Case Study Research: Design and Methods*, Los Angeles: SAGE.

Yoo, Y. 2017. "Design Thinking for IS Research," *MIS Quarterly* (4:1), iii-xviii.

Zhang, Z. 2007. "Effective Requirements Development-A Comparison of Requirements Elicitation Techniques," in *Software Quality Management XV: Software Quality in the Knowledge Society*, E. Berki, J. Nummenmaa, I. Sunley, M. Ross, and G. Staples (eds), pp. 225-240.

Zowghi, D. and Coulin, C. 2005. "Requirements Elicitation: A Survey of Techniques, Approaches, and Tools," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin (eds), Berlin, Heidelberg: Springer-Verlag, pp. 19-46.

## Curriculum Vitae

## Personal Data

| | |
|---|---|
| Name: | Jennifer Hehn |
| Date of Birth: | 26.06.1984 |
| Nationality: | German |

## Practical Experience

| | |
|---|---|
| Since 2017 | Research Associate<br>Institute of Information Management, University of St.Gallen |
| Since 2014 | Senior Manager<br>IT Management Partner St.Gallen AG, St.Gallen |
| 2014-2017 | Executive Director Design Thinking<br>Institute of Information Management, University of St.Gallen |
| 2012-2014 | Teaching Team Member for Design Thinking<br>Institute of Information Management, University of St.Gallen |
| 2004-2012 | Various Internships<br>e.g. Dr. Ing. h.c. Porsche AG; Galerie Schlichtenmaier; Werbe-Design Tauber; Career Service University of Würzburg |

## Education

| | |
|---|---|
| 2017-2020 | Ph.D. in Business Innovation<br>University of St.Gallen (HSG) |
| 2005-2012 | Diploma in Business Administration (Dipl.-Kffr.)<br>Julius-Maximilians-University of Würzburg, Germany |
| 2004-2010 | Magistra Artium in Art History, European Ethnology, Business Administration<br>Julius-Maximilians-University of Würzburg, Germany |
| 1994-2003 | Abitur<br>Matthias-Grünewald-Gymnasium, Tauberbischofsheim |