UNIVERSIDADE DE SÃO PAULO

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Fábio Avigo De Castro Pinto

Design Thinking for Requirements Engineering: Problems and Opportunities on Non-Functional Requirements

São Paulo

2023

FÁBIO AVIGO DE CASTRO PINTO

Design Thinking for Requirements Engineering: Problems and Opportunities on Non-Functional Requirements

Versão Corrigida

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências

Área de concentração: Sistemas Digitais

Orientador: Prof. Dr. Fábio Levy Siqueira

São Paulo

2023

Catalogação-na-publicação

Pinto, Fábio
Design Thinking for Requirements Engineering: Problems and Opportunities on Non-Functional Requirements / F. Pinto -- versão corr. -- São Paulo, 2023.
105 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Engenharia de Requisitos 2.Design Thinking 3.Requisitos Não Funcionais I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

AGRADECIMENTOS

Agradeço à minha esposa, Juliana Jeronimides Avigo, por todo o apoio e paciência na condução deste trabalho.

Agradeço ao meu orientador, Professor Doutor Fábio Levy Siqueira, pelo apoio e pelas numerosas e valiosas conversas que tivemos para a estruturação e condução deste mestrado.

Agradeço ao meu pai, Doutor Isaias de Castro Pinto, e minha mãe, Rita de Cassia Avigo, por me trazerem ao mundo na USP e por me acompanharem nesta jornada onde, passados 31 anos, posso concluir hoje este trabalho.

Agradeço a todos os colegas e professores da Poli com quem tive o prazer de conviver no período vivido nesta incrível universidade.

RESUMO

PINTO, Fábio. **Design Thinking for Requirements Engineering: Problems and Opportunities on Non-Functional Requirements**. 2023. 105 p. Dissertação (Mestrado em Engenharia de Computação) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2023.

O *Design Thinking* (DT) tem sido usado na Engenharia de Requisitos (ER) para auxiliar o processo de descoberta das necessidades dos usuários, ao focar em princípios de prototipação e empatia com estes. Uma vez que o *Design Thinking* pode ser utilizado de diferentes formas, partiu-se de artigos recentes publicados sobre o uso do DT dentro da ER para identificar as suas diferentes alternativas no processo de desenvolvimento de software e traçar categorias de uso do DT na ER. Paralelamente, há restrições em todo o projeto de software que direcionam decisões de *design* e existem devido a uma série de fatores: o campo de aplicação, regulamentações, escolhas de *stakeholders*, relações de custo/eficiência, dentre outras. Parte significativa dessas restrições são manifestadas em um projeto na forma de requisitos não-funcionais (RNF), que devem ser endereçadas em algum momento. O objetivo deste projeto é analisar como se dá a consideração de RNF dentro do DT e como isto se reflete em restrições de *design*, considerando o uso dos resultados do DT para a ER. Para isso, foram realizadas três estudos, a saber: (1) uma revisão sistemática de literatura para identificar possíveis problemas gerais do DT na ER, (2) uma pesquisa de opinião qualitativa com desenvolvedores de software com experiência no DT buscando evidenciar a percepção de possíveis problemas sobre RNF no DT e (3) um estudo de caso composto de entrevistas semi-estruturadas com profissionais da indústria que reportaram a respeito de projetos de software em que se utilizou o DT como abordagem de elicitação de requisitos, com o intuito de explorar problemas relacionados a RNF, como estes impactam o software final e em que momento são descobertos. No uso do DT como abordagem de elicitação de requisitos em um projeto de software, os resultados apontam problemas relacionados aos RNF: quando não estão diretamente conectados ao contexto do projeto, ao histórico do time participante do DT ou não se trata de pontos comuns de interesse ao mercado em que o software ou a empresa se inserem, RNF tendem a ser ignorados durante o DT. Esses são, entretanto, descobertos mais tarde e exigem readequações significativas de projeto para serem acomodados. Tendo em vista a correlação entre os RNF e as restrições de *design* impostas, esta situação aponta para a necessidade de que RNF sejam descobertos o mais cedo possível em um projeto, sob risco de se perder o potencial de inovação do DT ou que a solução inicial idealizada se mostre inviável. Este trabalho mapeou problemas identificados na literatura associados ao tema e casos práticos reportados por profissionais, compilando projetos reais de uso do DT na ER e problemas atrelados a RNF descobertos durante o DT e após este. São também apresentadas lições aprendidas por profissionais, melhores técnicas sugeridas e boas práticas para a descoberta de RNF o mais cedo possível.

Palavras-chave: Design Thinking, Requisitos não-funcionais, Engenharia de Requisitos.

ABSTRACT

PINTO, Fábio. **Design Thinking for Requirements Engineering: Problems and Opportunities on Non-Functional Requirements**. 2023. 105 p. Dissertação (Mestrado em Engenharia de Computação) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2023.

Design Thinking (DT) has been used in Requirements Engineering (RE) to help with the process of discovery of user needs, by focusing on principles of prototyping and empathy with users. Since Design Thinking can be used in different formats, this work starts with recent published articles regarding the use of DT in RE to identify its different alternatives in the process of software development in order to establish use categories of DT in RE. In the meantime, there are restrictions in every software project that drive design decisions and exist due to a number of factors: the field of the application, regulations, stakeholders' choices, cost-efficiency relations, among others. A significant part of these restrictions are manifested in a project in the form of non-functional requirements (NFR), which have to be addressed at some point in time. The objective of this project is to analyze how NFR are considered in DT, and how this reflects on design restrictions, considering the use of DT results for RE. For that, three researches have been conducted: (1) a systematic review of literature to identify general possible problems of DT in RE, (2) a qualitative survey with software developers who had experience in DT to search for evidence of possible NFR problems in DT, and (3) a case study composed of semi-structured interviews administered with software developers with DT experience who reported on software projects in which DT was used as a requirements elicitation approach, with the goal of investigating problems related to NFR, how these impact the final software, and at which point they are discovered. In the use of DT as an approach of requirements elicitation in a software projects, the results point out to problems related to NFR: when these are not directly connected to the context of the project, to the background of the participating DT team, or they do not concern common points of interest to the market in which the software or the company pertain to, NFR tend to be overlooked during DT. These are, however, later discovered and demand significant project adjustments in order to make room for them. Considering the correlation between NFR and the imposed design restrictions, this situation denotes the need for NFR to be discovered as early as possible in a project, under jeopardy of losing the DT innovation potential, or that the initial ideated solution be unfeasible. This work has mapped problems identified in the literature associated with the theme and practical cases reported by professionals, compiling real use projects of DT in RE and problems linked to NFR discovered during and after DT. Lessons learned by professionals, as well as suggested best practices and techniques for the early discovery of NFR are also presented.

Keywords: Design Thinking, Non-Functional Requirements, Requirements Engineering

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

DT	Design Thinking
NFR	Non-Functional Requirements
RE	Requirements Engineering

SUMMARY

1 INTRODUCTION

The development of software is an increasingly challenging undertaking. In order to keep up with the requirements that arise from the many stakeholders and customers in different industries, products and services have a need to be in constant innovation, which also implies a shorter lifecycle. This situation leads to a complication with the tools and methods that exist and were used in the past, which may not be the best alternative to said new requirements.

Requirements Engineering (RE), as described by the ISO/IEC/IEEE:29148 (2018, p.5) is the “interdisciplinary function to arbitrate between the domains of the acquirer (customer) and supplier (service provider) in order to establish and maintain the requirements to be met by a given system, software, or service of interest”.

It is noteworthy that the nature of RE is inherently difficult, in behalf of requirements analysts currently starting with ill-defined and possibly conflicting ideas of what the system should do, as well as a result of RE being less constrained than other Software Engineering activities (CHENG; ATLEE, 2007).

In light of this scenario, RE has significantly changed over the last decade, and innovation tools have been incorporated into software projects. One of which is Design Thinking (DT). It is an innovative approach that can be used for discovering users’ needs. DT can be seen as a structured approach to tackle complex and wicked real-life problems with a set of principles focused on empathy with users, fast prototyping, tolerance for failure and iterative learning cycles (BROWN, 2008; HEHN; UEBERNICKEL, 2018; KOLKO, 2015).

In the meantime, there are restrictions in every software project that drive design decisions and exist due to a number of factors: the field of the application, regulations, stakeholders’ choices, cost-efficiency relations, among others. A significant part of these restrictions is manifested in a project in the form of non-functional requirements (NFR), which have to be addressed at some point in time. NFR are a type of requirements that typically influence design decisions and thus restrict the development of the required functionality (LANDES; STUDER, 1995).

In this way, a guiding question for the rationale behind this work is if there can be untapped innovation potential in Design Thinking due to missing NFR. That is, we start with NFR being key concepts into justifying design decisions and constraining the possible ways of materializing functionalities, and DT as an innovation approach that has an interface with the problem domain where RE is located. Is it possible, then, that some NFR are discovered later rather than early in the development cycle of a system/software/component and this situation, in turn, leads to suboptimal solutions that were initially prototyped during DT?

1.1 Research Goal

Design Thinking is an approach focused on building innovative solutions that has seen an increase in use in software projects for eliciting requirements. Additionally, it is still not a consensus in the community if it should concern itself with NFR, albeit there are known issues, such as the neglect of NFR (HEHN; UEBERNICKEL, 2018; MAHE et al., 2020; NEWMAN et al., 2015).

In this way, the goal of this work is to analyze how NFR are considered in Design Thinking and how they reflect on design restrictions, considering the use of the DT results for RE.

Secondary objectives are:

- Identifying underlying problems with the use of Design Thinking during the requirements phase in software projects.
- Identifying possible tools and best practices to improve the discovery of NFR during the use of Design Thinking.

1.2 Justification

As previously stated, NFR are a key factor in software development, due to their nature of imposing constraints on design decisions. These design decisions are of interest to DT, in the condition of its results being used for RE. Recent studies have suggested the existence of challenges that concern NFR in the use of DT for RE (HEHN; UEBERNICKEL, 2018; MAHE et al., 2020; NEWMAN et al., 2015)

An example for this situation is in performance issues within certain contexts, such as systems that deal with big data. These can be translated into specific NFR. It is not clear as of today at what moment they should be considered in a software project that uses DT results for RE. If brought into consideration only after the DT sessions take place, it is possible that the innovation potential brought by DT becomes unusable, as these new impositions may invalidate the solutions that were thought of. These NFR, however, may also be known beforehand, and can be used alternatively as an input to DT. But there are others that may arise during DT sessions, or even during the software development itself.

Alongside, DT’s main focus lies on innovation, and perhaps it should not concern itself with NFR directly. However, this issue can jeopardize the software development, especially if DT is used as a standalone approach for RE. In this way, the objective that was sought out to be answered was regarding the question of, in the scope of DT’s results being used for RE, should it deal with NFR?

1.3 Methods

This section presents the methods that were used for this work.

To answer the objective of this research, it is first necessary to set some baselines by exploring the concepts of NFR and the question of how DT can be integrated into Requirements Engineering. Since Design Thinking can be used in different situations, this work started with recent published articles regarding the use of DT in RE to identify its different alternatives in the process of software development in order to establish use categories of DT in RE.

After this, this work investigated the use of DT for RE. It focused not on the commonly reported benefits and potentials of DT, but rather on identifying if among case studies and experiment reports on using its results in RE there were also issues that arose with (or in spite of) it. For that, a systematic literature review (SLR) was chosen as the first study. It focused on identifying papers that fit a set of matching criteria and cataloging reported issues with this use. The SLR is reported in Chapter 4, and followed the suggested reference of Kitchenham and Charters (2007), regarding general guidelines for undertaking systematic reviews. The results pointed out the existence of problems and limitations on a number of categories. Among these challenges, the issue of NFR did come to light in DT’s difficulty in dealing with non-functional requirements (NFR) (HEHN; UEBERNICKEL, 2018).

The following part of this work focused on collecting primary data to help verify this issue. A qualitative survey was conducted with software developers that have had previous experiences with DT in software projects. It is presented on Chapter 5, and followed the reference proposed by Linåker et al. (2015) and the work presented by Kitchenham and Pfleeger (2002). It concentrated on bringing further evidence to this question, as well as to investigate how NFR become manifest in projects that used DT, and whether the software community has tools for helping in this regard. The results suggest the neglect of DT for requirements other than those of a functional or usability nature, and the possibility of untapped potential on DT when approaching NFR.

Following up on the survey results, an interview-based field study was then conducted in the form of semi-structured interviews administered with industry professionals who had experience with using DT for software projects. This had the goal of investigating how NFR relate to DT. It follows the design of the work of Runeson et al. (2012) for qualitative case studies, with a protocol elaborated based on the suggested framework by Brereton et al. (2008). It also refers to a similar recent work of Alegroth et al. (2022), in which semi-structured interviews were conducted in a qualitative approach, following the framework of a qualitative case study. The results brought further evidence to DT disregarding NFR when they are not associated to the experience of the DT participants, to the company context or related to topics that are common to the market in which the company operates. From the reports, it can be pointed out that NFR imposed vast design restrictions, and shifted the way in which all projects were constructed, which indicated the need for them to be discovered as early as possible.

1.4 Work Organization

This work is organized as follows.

Chapter 2 introduces the concept of Non-functional requirements and brings examples of design decisions that must be made.

Chapter 3 delves into the Design Thinking approach. It details it and then focuses on its how it can be integrated to Requirements Engineering.

Chapter 4 focuses on identifying possible issues with the use of DT results in the requirements phase of software projects. It describes the systematic literature review conducted on identified articles of case studies or experiment reports that have used DT for this end.

Chapter 5 undertakes the task of bringing further evidence to confirm the existence of issues regarding NFR in the use of DT for RE, by deepening the understanding of this relation via the gathering of primary data. A report is made on the survey that was conducted with software practitioners that have had previous experience with the use of the technique for RE.

Chapter 6 presents the case study focused on investigating how NFR relate to DT via semi-structured interviews conducted with industry professionals.

In Chapter 7 the conclusion of this work is presented along with a compiled list of suggestions, tools, and best practices for improving DT in regard to NFR that derived from the previous conducted studies. It also presents some suggestions for future works to be explored.

2 NON-FUNCTIONAL REQUIREMENTS

This chapter describes the common understanding of non-functional requirements. Firstly, it is important to define a requirement. According to the Standard Glossary of Software Engineering Terminology, a requirement is “(1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2)” (IEEE 1990, p.62).

That is, a requirement is a definition of something to be achieved by a product. Even daily mechanical products have requirements to be met. If we take a water hose for example, one could describe its requirements are: “it must be pluggable to a faucet”, “it must not leak”, “it must allow the transport of water from the faucet until further distances”, “it must be made of materials that do not contaminate the water”, “it must be durable”, and so on.

Missing requirements or poorly designed/developed requirements in a product directly relate to the user satisfaction with it and the product’s success/failure. A piece of software is in essence also a product – with requirements. Especially in software, requirements are necessary as a form of communication of what should be built.

Requirements are commonly distinguished as being of a functional or non-functional nature. A reference to this nomenclature is the difference between what the system shall do (functional), as opposed to how it should do it (non-functional).

The IEEE 830 (1998) proposes that functional requirements are the ones that determine the actions that must be accomplished by a system. They are usually represented by “shall” statements, as in “The system shall…” and include validity checks on the inputs, relationships of outputs to inputs, sequence of operations, among others.

Now regarding non-functional requirements, as Glinz (2007) puts it, there is no consensus in the requirements engineering community about what non-functional requirements (NFR) are and how they should be elicited, documented and validated. Many authors have described non-functional requirements building on the terms of property/characteristic, attribute, quality, constraint, and performance, but there is no consensus about the concepts denoted by said terms. Complementing this, Chung and Leite (2009) have suggested that the attention in the Software Engineering field has been centered on the functional characteristics of a system, and this practice led to needed quality characteristics being treated merely as technical issues related mostly to the detailed design or testing of an implemented system. Glinz (2007, p. 25) summarizes the common understanding of NFR, in that: “A non-functional requirement is an attribute of or a constraint on a system”.

The categorization of non-functional requirements that was chosen to be used in this work is the one of the ISO/IEC 25010 (2011), regarding product quality properties and excluding the one of functional suitability, due to it being too closely related to functional requirements. In this way, the possible categories of NFR as well as their definition are listed as follows:

- **Performance Efficiency**: performance relative to the amount of resources used under stated conditions.
- **Compatibility**: degree to which a system can exchange information with other systems and perform the required functions, while sharing the same environment.
- **Usability**: degree to which a system can be used to achieve specified goals with satisfaction in a context of use.
- **Reliability**: degree to which a system performs specified functions for a specified period of time.
- **Security**: degree to which a system protects information and data.
- **Maintainability**: degree of effectiveness with which a system can be modified by the maintainers.
- **Portability**: degree of effectiveness and efficiency with which a system can be transferred from one environment to another.

The constraints that NFR bring are a key topic for this work. Landes and Studer (1995) have suggested that NFR establish the justifications for design decisions and thus restrict the way for the realization of the required functionality.

Some examples of NFR that possibly restrict design decisions are cited below:

- **Big data applications**: A system to be built for big data brings several non-functional requirements in terms, for example, of storage and processing power (performance efficiency). These structural needs that derive from said requirements can be significantly different from common applications based on smaller datasets.
- **Data privacy**: Several systems deal with private data and must put measures in place to assume the security of this information. Alongside, several countries have also created regulations that define frameworks or establish procedures that a system must follow in order to be compliant to them. These restrictions can originate measures such as the need for creating anonymous databases, for restricting users' privileges, for securing public APIs, among countless others.
- **Using (or not) external code libraries**: The decision of using external code libraries must be made at some point in time, and it can restrain applications both ways. If used, the developing team will have to account for issues in versioning, compatibility, security and even performance. If unused, it can increase development times, code complexity and maintainability difficulty.
- **Different personas of users**: Different users/personas can be familiar with different user interfaces (such as lexicon and ease of use) according to differences in their profiles like their age, their background and education, or their culture. This is a problem typically addressed with Design Thinking.

2.1 Chapter Considerations

This chapter dove into the basic concepts behind requirements in software, and explored the rationale behind non-functional requirements, which are essential to the following steps of this work. Following up, we turn to the basics behind Design Thinking and its possible integrations into Requirements Engineering.

3 DESIGN THINKING IN REQUIREMENTS ENGINEERING

This chapter brings some definitions regarding the common uses of Design Thinking, and its integrations in Requirements Engineering. These are necessary to best explain the following chapters.

3.1 Design Thinking: An Innovation Approach

The market is currently influenced by a series of complex factors – emerging technologies, globalization processes, and a constant changing of requirements –, which makes it so that the life cycle of a service becomes shorter. This situation leads to a conflict among tools and methods that have been used in the past, which may no longer fit the new products and service development requirements (VETTERLI et al., 2013). In order to support them, companies should seek other possibilities using new approaches for creating value (VOLKOVA; JĀKOBSONE, 2016).

In this context, an approach that can help with this endeavor is *Design Thinking* (DT), developed in Stanford, refined by IDEO and supported by Hasso Plattner, one of the founders of SAP (BRENNER; UEBERNICKEL, 2016). This approach encompasses a team-based process for designing the solution of a given problem (PLATTNER; MEINEL; LEIFER, 2011). In the process, the participants usually empathize with the final user, define the problem, combine different perspectives, brainstorm solutions, and develop prototypes. It can be seen as a structured approach to tackle complex and wicked real-life problems with a set of principles focused on empathy with users, fast prototyping, tolerance for failure and iterative learning cycles (BROWN, 2008; HEHN; UEBERNICKEL, 2018; KOLKO, 2015).

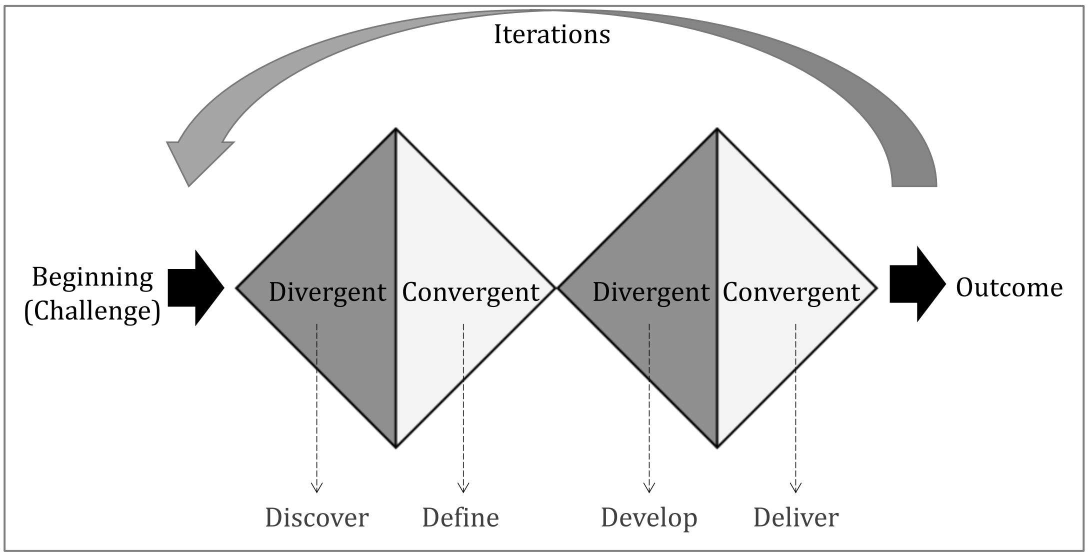
It is important to note, however, that Design Thinking is not a standardized process model with defined phases, and thus each author must explain exactly what is included in each individual phase and how they are related (WAIDELICH et al., 2018).

DT has experienced a surge in popularity over the last decade, given its practical framework that can lead to benefits to companies who use it (BRENNER; UEBERNICKEL, 2016). This popularity can be associated to DT's focus on better understanding consumers' needs and desires, and on the production of prototypes that converge on innovative solutions. These prototypes usually focus on the user's interaction and experience with the product being developed; hence DT's acceptance and increased usage on the UX/UI fields of expertise.

A common approach to Design Thinking is the "Double Diamond", a technique popularized by the British Design Council (2019). A graphical representation for the process is presented on Figure 1, and is comprised of 4 phases and an initially defined problem/challenge to be solved.

It starts with a divergent phase focused on empathizing with the user (*Discover*), followed by a convergent phase focused on defining the issue (*Define*), which is then followed by another divergent phase focused on finding a solution (*Develop*) and it is concluded by a convergent phase focused on delivering an outcome or prototype for it (*Deliver*). These phases can be repeated in an iterative manner until a satisfiable result is achieved.

Figure 1 – The Double Diamond process.



Source: Adapted from DESIGN COUNCIL (2019); HEHN et al. (2019).

The *Discover* phase is when the problem space is explored by empathizing with the end user. Some of the commonly used tools in this phase are interviews with users and the empathy map.

The *Define* phase is when the problem is properly defined. Commonly used tools are personas and journey maps. They focus on describing the user.

The *Develop* phase is when the solution space is explored. Commonly used tools are brainstorming, brainwriting, “now-how-wow” matrix, and point voting.

The *Deliver* phase is when the solution is defined. Commonly used tools are physical prototypes, storyboards, and role-playing.

At last, the outcome, which is comprised of prototypes, is properly assessed and tested. Prototypes are presented and feedback is collected. These results can then either be reused as the input for a new DT iteration, or considered final, and be the foundation for the upcoming product development.

3.2 Design Thinking and Requirements Engineering

This section introduces detail as to how DT is usually integrated into Requirements Engineering.

Brenner et al. (2016) presented DT as a future-oriented innovation method, and that three forms of its possible uses became important in the environment:

1. **Design Thinking as Mindset**: Application of a set of principles aligned with Design Thinking, such as failing often and early, building prototypes that can be experienced, testing early with customers and the combination of divergent and convergent thinking, among others.
2. **Design Thinking as Process**: Application of DT principles in a structured manner in a process. The authors suggest the use of a two-stepped process model.
3. **Design Thinking as Toolbox**: Application of numerous methods and techniques from various disciplines, such as: Empathy map, Persona, Storytelling.

More recently, Hehn et al. (HEHN et al., 2019) presented the possible strategies for integrating DT and RE. These strategies have a direct link with the previous possible applications defined by Brenner et al. The authors assert that the Requirements Engineering field faces the challenge of discovering and satisfying the confused needs and volatile requirements of the many stakeholders involved in the process, and claim that Design Thinking is a promising approach for addressing such challenge (HEHN et al., 2019). Their work presents three possible strategies for integrating Design Thinking to Requirements Engineering:

- **Upfront Design Thinking**: In this scenario, Design Thinking is used at an early project stage to identify relevant features to be developed. The result is used as grounds for performing the following Requirements Engineering activities. Closely related to DT as process.

- **Infused Design Thinking**: The Design Thinking technique is used as a set of tools for merging an existing Requirements Engineering process with chosen artifacts and methods. Closely related to DT as toolbox.

- **Continuous Design Thinking**: The Design Thinking activities and principles are continuously integrated to the Requirements Engineering practices and processes, while using DT as the guiding mindset. The DT prototypes are used as boundary objects between activities. Closely related to DT as mindset.

The first approach, regarding the Upfront DT, is better applied when there is some level of uncertainty in the beginning of a project regarding the customer needs and the corresponding solution. The delivery potential is leveraged with a framework that serves as a reference for continually fostering creativity. The core of the final solution has traceable connections with the customer’s needs, helping with the prioritization of relevant features. The prototype works as an artefact that supports communication among the different groups of stakeholders. However, the solution demands a significant amount of effort and resources to conduct a small project of its own. There is also a loss potential of the implicit knowledge and of the results generated in the process. On top of that, very little attention is given towards crucial artefacts to the following development activities, such as quality requirements, restrictions, or data models (HEHN et al., 2019).

The second approach, Infused DT, is particularly suitable when commonly used Requirements Engineering practices need specific interventions, as the need of new ideas. The interventionist character of Design Thinking requires small changes to the Requirements Engineering practices, with a small incremental expense of resources.

This strategy, however, has a limited creativity potential, if compared to the previous one. It also does not bear the sustainable impact of the Design Thinking method given its interventionist character and also does not pay attention to the subsequent details of artifacts that are critical to development, just as in the upfront approach (HEHN et al., 2019).

The third approach, Continuous DT, is recommendable for approaching complex problem settings, in which the customers must be continuously involved within a product development life cycle. Given the continuous interaction in this case, it must be considered the development of critical project artifacts, and there is an integration of a customer-centered mindset throughout the entire project, with requirements that are more accurate through the identification and prototyping phases. However, this approach is more resource intensive and demands more corporate support, as well as being more team-dependent for its proper usage (HEHN et al., 2019).

3.3 Applications of Design Thinking in Requirements Engineering

Moving up, we seek to deepen the understanding of how Hehn et al. (2019) approaches are commonly used in real software projects and bring evidence to each. Although the authors' chosen division is an useful framework for identifying and classifying the use of DT within RE, two points can be argued for improving it: (1) the need for crossing DT approach with requirements engineering processes to identify at which point each approach is used, and (2) recognizing that the upfront approach has more than one possibility of when to be used, as will be further explored.

For this, the ISO/IEC/IEEE 29148 was chosen as a starting point for the analysis. This standard presents the set of requirements engineering processes in a software project (ISO/IEC/IEEE, 2018), with a macroprocess from which requirements are narrowed down. Once defined, development can proceed to the following steps (Architecture Definition, Design, Development, Testing and Deployment).

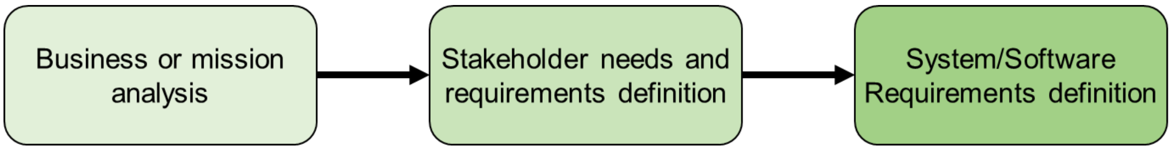
The three requirements engineering processes defined by the ISO/IEC/IEEE 29148 (ISO/IEC/IEEE, 2018) and displayed on Figure 2 are:

(1) The business or mission analysis, with a purpose to define the business or mission problem or opportunity, characterize the solution space, and determine potential solution class(es) that could address a problem or take advantage of an opportunity.

(2) The stakeholder needs and requirements definition, with a purpose of identifying stakeholders or stakeholder classes involved with the system throughout its life cycle, and their needs. These needs are analyzed and transformed in a set of stakeholder requirements.

(3) The system/software requirements definition, with a purpose to transform the stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user.

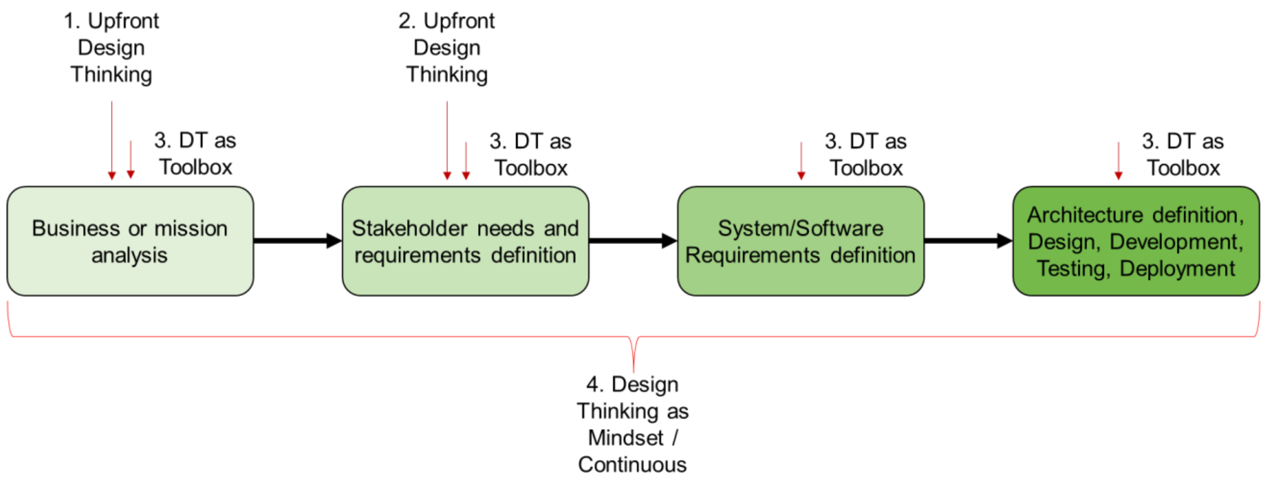
Figure 2 – Example of the sequence of requirements processes and specifications.


Business or mission analysis
Stakeholder needs and requirements definition
System/Software Requirements definition

Source: Adapted from (ISO/IEC/IEEE, 2018).

It is in this context that the common possible applications of Design Thinking in Requirements Engineering can be pointed out. These uses of DT, presented in Figure 3, are:

Figure 3 – Possible applications of DT in RE.


1. Upfront Design Thinking
2. Upfront Design Thinking
3. DT as Toolbox
3. DT as Toolbox
3. DT as Toolbox
3. DT as Toolbox
Business or mission analysis
Stakeholder needs and requirements definition
System/Software Requirements definition
Architecture definition, Design, Development, Testing, Deployment
4. Design Thinking as Mindset / Continuous

Source: author.

1. **Upfront Design Thinking in the Business or mission analysis**: DT is used in the problem domain, where the problem is not yet well-defined, or the business or mission analysis is not clearly structured. In this moment, it is not yet clear or necessary that the solution will be a software. The focus lies on using DT's innovation potential for studying the problem and defining preliminary solution classes. This practice can be found, for example in Newman et al. (2015), where Design Thinking was used in the context of Social Software Engineering for helping with designing a renewable energy forecast system developed in partnership with a remote Scottish Island Community.

2. **Upfront Design Thinking in the Stakeholder needs and requirements definition**: DT is used in the solution domain. The problem has already been discovered, but the solution lies to be explored and better defined. DT will be used to explore stakeholder needs to the given problem. It can be argued that this is perhaps the most common use of DT within RE. Some examples can be found in Carell et al. (2018) who present the use of DT within two case studies in a Technology Consulting Firm; in Canedo and Costa (2018) who present the use of DT for the modernization of a system; or in Schimmer and Meyer (2012) who report on the use of DT for creating a software for professional sailors. Additionally, in some articles, the term of a "Design Thinking Workshop" can be found, as used by researchers and practitioners implying the DT

sessions/meetings, and it usually presumes the adoption of this upfront approach. Some examples of this definition in use can be found in (DOBRIGKEIT; DE PAULA, 2019; HEHN et al., 2019; MAHE et al., 2020; ROZANTE DE PAULA; SANTANA AMANCIO; NONATO FLORES, 2020).

3. **Design Thinking as Toolbox / Infused Design Thinking**: Selected DT tools are applied on an existing RE process. Tools can vary in complexity, intensity, and the proper timing and step in the RE process for its use. This practice can be found, for example, in Ximenes et al. (2015), where a project management model combining Agile, Lean Startup and Design Thinking is described. In it, Design Thinking is used as a tool, and its techniques are applied on specific rounds of the project.
   Another example was reported by Dobrigkeit et al. (2021), in which the authors presented the use of five different tools from a previously developed Design Thinking toolbox with a selection of Agile software development teams and summarized findings contemplating the use cases, benefits, and challenges of given tools as experienced by the participants.

4. **Design Thinking as Mindset / Continuous**: DT is used as the guiding principle along the whole software development process. Some papers reference this practice, such as described by Mahé et al. (2020), in which Design Thinking is described as a “journey”, and its activities are carried out over 14 months accompanying a migration of a software factory to Design Thinking in a continuous manner.
   Another example of this use can be found in Hehn and Uebernickel (2018), in which a case study is presented combining DT and RE in an agile development setting from the beginning of the conceptualization until the product implementation.

Although Design Thinking can be used during different stages of a product’s life cycle, this classification of 4 categories helps with better identifying which form of DT one refers to.

In the subsequent chapters, when referring to the use of Design Thinking, it should be better understood as an upfront project during the stakeholder needs and requirements definition, that is, category #2.

3.4 Chapter Considerations

This chapter has explored the possible integrations of Design Thinking in Requirements Engineering by summarizing previous important work of Brenner et al. (2016) and Hehn et al. (2019), as well as expanding the definitions of the latter by crossing them with requirements engineering processes from the ISO/IEC/IEEE 29148 (2018). In that manner, 4 classes of possible integrations were derived and exemplified with recent published articles.

This classification helps with defining what use the term is applied to in each situation, as well as setting a baseline for the next chapters.

Next, Chapter 4 presents a systematic literature review that was conducted to identify recent papers that have used Design Thinking for RE.

4 DESIGN THINKING CHALLENGES ON REQUIREMENTS ENGINEERING

The Design Thinking approach represents a potential solution for some traditional challenges in Requirements Engineering, for eliciting user needs, instead of requirements (VETTERLI et al., 2013), being regarded as a "modern form of Requirements Engineering" (BEYHL; GIESE, 2016). However, even some Design Thinking enthusiasts, such as Uebernickel and Hehn (2018), present that there are still challenges to be overcome, like the limited capabilities of success due to the dependency of participants' individual experiences, or problems with non-functional requirements in terms of traceability, correctness, completeness, and consistency.

Vetterli et al. (2013) have justified why the Requirements Engineering field can make use of the Design Thinking approach for eliciting customers' needs: with the creation of fast and simple prototypes that converge into innovative solutions. According to the authors, given that the primary measure of an information system's success is the degree to which it meets its original purpose (that is, the stakeholders' goals), Requirements Engineering is a process of initial discovery and definition of purpose, which makes it inherently hard. On top of that, given that requirements analysts many times start with badly defined ideas (and often conflicting), the Requirements Engineering processes must be more iterative and involve a larger number of stakeholders than other Software Engineering activities (CHENG; ATLEE, 2007).

Thereby, the resulting prototypes of DT activities help with the substantiation of different ideas without the simplification of the environment, while focusing on specific and important needs within the design space. Although prototypes are also used within software development and in Requirements Engineering itself, the big advantage of DT is the identification and elimination of technical inconsistencies as soon as possible in the process (VETTERLI et al., 2013).

4.1 Problems of Design Thinking in Requirements Engineering

As presented, Design Thinking is an innovative tool that has seen a rise in popularity in software development and can be used for eliciting requirements and discovering innovative solutions. Given its simple conceptual framework, its current widespread status and also reports from experienced researchers on its known issues, it was chosen to investigate in this study whether there were problems that arose from (or in spite of) using DT in RE, and how the technique itself could be further improved.

This part of the work focused on investigating the problems that were reported by the scientific community with a systematic literature review (SLR) that is described as follows[1].

4.1.1 Method

This section summarizes the method applied to the conduction of the systematic literature research.

The following systematic literature review employed the framework outlined by Kitchenham and Charters (2007). Kitchenham and Charters present a systematic literature review as a form of secondary study, an effective tool for identifying, evaluating, and interpreting existing research related to a chosen phenomenon. In the report, the authors provide a comprehensive framework for researchers. The first stage involves the development of a research question, which helps to define the review's objective and scope. The question should be specific, measurable, achievable, realistic, and time-bound (SMART). It guides the review process and aids in establishing a relevant search strategy.

Next, researchers need to define their search strategy. The search strategy typically comprises a comprehensive list of databases and search engines to be used, keywords and phrases that capture the essential aspects of the research question, and specific inclusion and exclusion criteria. The search terms should be related to the review question, and the inclusion and exclusion criteria should filter out irrelevant studies.

[1] This systematic literature review was published in (PINTO; SIQUEIRA, 2020).

Once the search strategy has been executed, the next step is to select the studies to be included in the review. This process is typically done in two stages: first, by screening titles and abstracts, and then by reviewing full texts. Each study should be evaluated against the inclusion and exclusion criteria defined in the search strategy.

After the selection process, the selected studies need to be critically appraised. This means assessing the quality of the studies to determine whether their findings are valid and reliable. Quality assessment criteria may include: the clarity of the study's objective, the appropriateness of the study's design, the robustness of the data analysis, and the validity of the conclusions drawn.

Next, the data from the selected studies should be extracted and synthesized. This process involves gathering relevant information from each study, such as its purpose, methodology, findings, and conclusions. The method of synthesis will depend on the nature of the studies and the review question. It might involve a narrative or quantitative synthesis, or a combination of both.

Lastly, Kitchenham and Charters emphasize the importance of reporting the review in a clear, transparent manner. The report should include a detailed description of the methodology used in the review, the characteristics and findings of the included studies, and an interpretation of the overall findings. It should also discuss the limitations of the review and suggest areas for future research. In essence, the reader should be able to replicate the review based on the information provided in the report.

This review focuses on assessing the challenges encountered during the use of DT in software projects with the aim of identifying and addressing these issues.

The study starts by defining its objective and formulating research questions aimed at achieving this objective. The subsequent stage involves identifying pertinent literature reviews. This step is crucial in determining the need for the review, as finding a recent publication with similar objectives could be used as a foundation for the work or even render the study redundant. We present the objective, research questions and related reviews in sections 4.1.2, 4.1.3, and 4.1.4, respectively.

Section 4.1.5 presents the data collection and analysis procedure implemented. To procure primary data – recent papers that the researcher could analyze to answer the research questions – the subsequent step entailed identifying relevant research. This involved devising a search query to be executed across scientific databases, aimed at identifying papers that met the search criteria and were thus relevant to the review.

The query was developed directly from the research objective. Given its focus on identifying problems with DT usage in software projects, it was designed to locate papers reporting on this topic. The query was formulated, refined, and then slightly modified for each database. This search yielded numerous articles that constituted the core material for analysis.

Nevertheless, it is not uncommon for search results to yield a vast array of publications, many of which are false positives – results that fulfill the criteria but aren't pertinent to the research. To sift through these results, a title filter and an abstract filter were applied, effectively reducing the results to a manageable and relevant sample for review.

The selected papers were then analyzed in accordance with the established research questions. This examination allowed for the extraction of meaningful insights, addressing the challenges and potential solutions associated with the use of DT in software projects. The findings are presented in Section 4.1.6.

4.1.2 Objective

The systematic review has one main objective, to identify which problems arise from Design Thinking in Requirements Engineering, and one secondary objective, to compile researchers' suggestions into possible improvements for the DT approach.

4.1.3 Research Questions

Given the stated objective, two research questions were derived. The first and main one aims at identifying and compiling a base of Requirements Engineering problems with Design Thinking that were reported by researchers in software development projects. The second, which is a byproduct of the objective, comprises the secondary goal of identifying whether these same researchers also propose suggestions for improving DT itself in order to overcome related problems.

RQ1: *What Requirements Engineering problems were identified in software development projects that have used the Design Thinking approach?*

RQ2: *What are the authors' suggestions for improving Design thinking?*

4.1.4 Related Literature Reviews

Before starting the systematic literature review itself, related reviews have been searched for, and some recent reviews on Design Thinking have been found and are reported as follows.

Waidelich et al. (2018) present a systematic review upon 35 processual models in the Design Thinking Field aiming at comparing models regarding the number of process steps and specific terms used. The review is not exhaustive. The authors conclude that there is not a standardized process model with defined phases, and thus each author must explain exactly what is included in each individual phase and how they are related. The authors also state that the DT approach has been subsequently developed past 10 years since its basic research through the deepening of the scientific community of the subjacent principles acquired from research projects. There are a number of distinct models nowadays, and some researchers have worked with different models and reached similar results. On top of that, there is a recent tendency of using models with more phases.

Pereira and Russo (2018) present a systematic literature review regarding the integration between Design Thinking and agile methodologies for software development conducted upon 29 articles. Among the most relevant conclusions, it is presented that the most commonly used technique in this integration is Scrum; and in some cases, the software quality has been significantly improved and with acknowledged customers satisfaction.

Park and McKilligan (2018) present a review upon 72 papers published between 1972 and 2017 to answer the question of how does human-computer interaction (HCI) and the Design Thinking process overlap, differentiate and can improve each other. Among the conclusions, both approaches share similar and iterative steps: user understanding for identifying problems, ideation, prototyping and testing. However, they differ in the specific principles of each stage, determining the used tools and goals to be reached. HCI requires user comprehension for elaborating requirements, applying rules and principles for design, focusing overall on building the software. Design Thinking, on the other hand, focuses on building empathy for understanding users, projecting activities that generate ideas and solutions and encouraging that these be translated on the user’s life.

These identified related literature reviews do not, however, contemplate the objective of this work, and so a systematic literature review focused on identifying possible problems of DT was justifiable.

4.1.5 Strategy and procedures

In order to answer both research questions, it was first necessary to compile a list of articles that fit the criteria. The search was realized in February/2020 on the four main academic portals for Computer Engineering: Springer, ACM, IEEE, and Science Direct.

The main inclusion criteria for potential articles was to search for case studies or experience reports of the usage of Design Thinking on software development projects that have presented a direct relation with the Requirements Engineering field. In this manner, the search string has been defined as:

```
"design thinking" AND ("case study" OR "experience report") AND
("software" OR "development") AND ("requirements engineering" OR
"requirements elicitation" OR "requirements activities")
```

A total of 155 matching papers have been found on the 4 searched databases. After the initial selection, 2 sequential filters were applied, of Title and of Abstract, removing the false-positive articles that were not related to the theme. A date filter was intentionally not made on any moment of the review, nor a type of document filter. A language filter was also not applied, although after the abstract filter, every article was in either English or Portuguese and therefore, was kept. Table 1 presents the number of results found in each portal.

Table 1 – Paper results obtained from the search in academic portals.

Portal	Number of Results	Title filter	Abstract filter
Springer	57	32	6
ACM	18	7	4
IEEE	37	13	4
Science Direct	43	22	0
Total	**155**	**74**	**14**

Source: author.

4.1.6 Results

After the selection of the 14 articles, the review was conducted by the author through the reading and analysis of each article independently. In order to answer the research questions, a cataloging and categorization of the original statements from researchers was conducted. The author avoided making inferences on each experiment. Table 12 in the 0 presents the list of the 14 selected articles for the Systematic Literature Review.

A quick summary of each article is presented as follows:

- Carell, Lauenroth and Platz (2018) have as a context two case studies in a Technology Consulting firm (Adesso);
- Canedo and Costa (2018) address the use of DT for the modernization of a system;
- Braz et al. (2019) have used DT in an university project to design a resource allocation system;
- Schimmer and Meyer (2012) report the use of DT in a software for professional (competitive) sailors;
- Ximenes, Alves, and Araújo (2015) report on a case study regarding the development of an application for storing data in which a technique named

"Converge" was applied, in which DT is used in specific phases along the development cycle as an innovation tool;

- Gabrysiak, Giese, and Beyhl (2012) regard the creation of virtual prototypes and the usage experience of DT on a 3 year project;
- Dobrigkeit and de Paula (2019) describe a project regarding the understanding of the DT manifestations by different users in a software development company;
- Carroll and Richardson (2016) report the use of DT in a software for digital pharmacy (e-pharmacy);
- Newman et al. (2015) report the use of DT in a software for managing energy in a small community;
- Ferreira, Barbosa, and Conte (2018) report the use of DT in a software for an academic management system;
- Hehn and Uebernickel (2018) present a study regarding the continuous integration of DT with ER in the scope of an industrial utilities system;
- Levy and Huli (2019) report a workshop in a software development company;
- Piras et al. (2019) report a case study of DT and gamification in a requirements prioritization tool; and
- Mahé et al. (2020) discuss the migration of processes of a software development company to using DT in a continuous manner (MAHE et al., 2020).

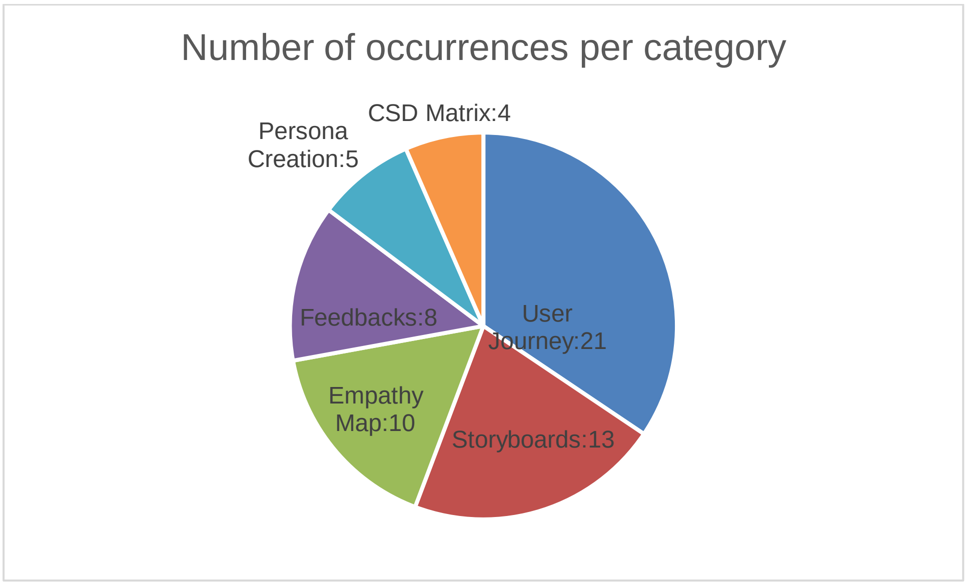
4.1.6.1 Identified problems (RQ1)

The first research question (RQ1) had the aim of identifying problems associated with Design Thinking in Requirements Engineering. Among the 14 articles, a total of 29 topics were cataloged as problems or challenges of the approach. In order to classify the results and to get some insights regarding the aggregated data, the items were categorized according to the category classification of Software Requirements in the SWEBOK (BOURQUE; FAIRLEY; IEEE COMPUTER SOCIETY, 2014). The categories with the result of RQ1 are presented categorized on Figure 4. The detailing of the problems found in each category is presented thereafter.

Requirements Processes: Within requirements processes, the following questions were found as problems/challenges:

- With the use of the approach, it isn't obvious to differentiate between "as-is" scenarios and "to-be" scenarios from the user perspective for composing the storymap (SCHIMMER; MEYER, 2012).
- The correct application of the approach and obtaining good results are directly dependent on the experience and knowledge of the DT activity participants, not only of the approach, but also regarding Software Engineering methods and processes, and also of the context itself (DOBRIGKEIT; DE PAULA, 2019; GABRYSIAK; GIESE; BEYHL, 2012; HEHN; UEBERNICKEL, 2018; XIMENES; ALVES; ARAÚJO, 2015).
- Design Thinking faces challenges regarding the availability of customers at the proper timing. While a feature is under construction, it is only natural that another question is being discussed, and results are not immediately relevant (DOBRIGKEIT; DE PAULA, 2019; HEHN; UEBERNICKEL, 2018).

Figure 4 - Number of occurrences of problems or challenges with the Design Thinking approach in the articles per category of the Software Requirements knowledge area according to the SWEBOK.



Source: author.

Requirements Elicitation: The requirements elicitation step was marked by the following issues:

- It is not always possible to identify user's need through questions, and they must somehow be tracked (CARELL; LAUENROTH; PLATZ, 2018).

- Design Thinking has information traceability difficulties. The elicited requirements are captured via post-its or prototypes, in an unstructured manner in order to speed up team collaboration and the process as a whole (HEHN; UEBERNICKEL, 2018).

Requirements Analysis: Regarding requirements analysis the following problems have been encountered:

- There are problems when the Design Thinking results conflict with management expectations or with the pre-defined focus of the product (requirements prioritization) (DOBRIGKEIT; DE PAULA, 2019; HEHN; UEBERNICKEL, 2018).
- The DT approach makes no expectations regarding which requirements will be addressed in the future (MAHE et al., 2020).

Requirements Specification: The following questions have been found in requirements specification:

- The interviewees have different needs and tasks to address. It is hard to maintain the construction with high standards of quality and usability that meet every user’s expectations, particularly given the time restriction (XIMENES; ALVES; ARAÚJO, 2015).
- The DT process neglects non-functional requirements such as security and performance (HEHN; UEBERNICKEL, 2018; MAHE et al., 2020; NEWMAN et al., 2015).

Requirements Validation: Within requirements validation there was one statement found:

- For complex multi-user systems, the creation of prototypes (which is fundamental to DT) that capture every underlying processes and dataflow is very costly, to the point of it becoming prohibitive (GABRYSIAK; GIESE; BEYHL, 2012).

Others: Under this category are listed the problems found that do not directly match any of the pre-defined SWEBOK categories:

- The sharing of results of the DT activities and the assurance of a good result in the final product are a barrier. Results must be shared with the whole team in a time-efficient manner, and they are only interesting for most only once the feature is already being developed (DOBRIGKEIT; DE PAULA, 2019).
- Due to the small planning time, DT can lead to an inappropriate architecture (HEHN; UEBERNICKEL, 2018).
- DT can be problematic regarding inaccurate effort estimates (HEHN; UEBERNICKEL, 2018).
- The initial steps of the DT approach appear to be slow and ineffective. In the following steps, there are problems regarding a moral fluctuation in the involved teams (MAHE et al., 2020).
- Given the nonconventional collaborative nature of the DT activities and the need to change customers' mindset, it is a challenge to prepare the customers in advance for the first meeting activities (MAHE et al., 2020).
- The high degree of interaction during the DT workshops points out that their logistics must be carefully prepared and in an advanced manner (MAHE et al., 2020).
- If a DT team is composed of people of too different hierarchies, it is possible that the global innovation effect is lost, since there will be a communication problem, and people tend to participate less (MAHE et al., 2020).
- The underlying problems found during DT activities are many times blurred due to the many suggested solutions (MAHE et al., 2020).
- DT's focus on the future helps to create unrealistic expectations. Customers may have the sensation that during development only the easier points will be addressed, and the promises of elements that were seen in the workshop are set aside (MAHE et al., 2020).

Discussion: The articles present that there are, effectively, problems with the DT approach for Requirements Engineering in every category of the SWEBOK.

The highest number of occurrences happened in the Requirements Processes step.

A common issue among many articles is that DT does not comprise a rigid process, and it can be applied in many formats and with different levels of experience. There are evidences to the existence of a learning curve that conducts to better results once the process is applied in a more evolved manner: Dobrigkeit and de Paula (2019) suggest for example that the DT implementation depends upon the experience and knowledge of its participants. In this way, there are signs that good results are directly linked to knowledge about the context of the information and about the DT method itself (DOBRIGKEIT; DE PAULA, 2019; GABRYSIAK; GIESE; BEYHL, 2012; HEHN; UEBERNICKEL, 2018; XIMENES; ALVES; ARAÚJO, 2015).

Some of the problems found empirically are already addressed directly by the researchers and can be mitigated with good planning practices in advance to the DT rounds. A common theme to Dobrigkeit, de Paula (2019) and Hehn, Uebernickel (2018) was, for example, the availability of customers, a problem linked to schedule planning.

Among the problems, some are of organizational nature of the company where the process is conducted: the presence of people of too distinct hierarchies in a same group inhibiting communication (MAHE et al., 2020), the prioritization of requirements and the potential management conflict (DOBRIGKEIT; DE PAULA, 2019; HEHN; UEBERNICKEL, 2018).

There is also evidence that points out to an ideal project size for which the Design Thinking approach presents better results, although the desirable range can be reasonably large. In too small activities or where the solution is already known, the technique isn't justifiable (DOBRIGKEIT; DE PAULA, 2019), and in too large or too complex projects, prototyping is costly and it becomes prohibitive (GABRYSIAK; GIESE; BEYHL, 2012).

4.1.6.2 Suggestions (RQ2)

The second research question (RQ2) sought out to identify the authors' proposals for improving Design Thinking. Ten suggestions were found and are presented as follows:

- Apart from the classic project management roles, a software project can benefit from having a person with the established responsibility of designing the software according to the users requirements, in a way to ensure the necessary focus on the design from the user point of view, and to avoid conflicts of interest as far as possible, given that an architect can, in the case of doubt, rule against the user in favor of a simpler technical solution (CARELL; LAUENROTH; PLATZ, 2018).

- The authors suggest the participation of a designer in order to facilitate the DT process, and point out to a direct correlation regarding the quality of the created prototypes (CANEDO; PARENTE DA COSTA, 2018).
- In the first divergent part of the DT process, the authors suggest the use of a “360 degrees research”, referring to the observation and to interviews with users and with secondary sources, such as analysts, thinkers and competitors in analogous and underlying domains (apart from direct user information, to search for benchmarks) (SCHIMMER; MEYER, 2012).
- User story maps can serve as a tool to fill the gap between empathy and insights acquired via DT practices and the backlog for building the solution (SCHIMMER; MEYER, 2012).
- The DT process can be approached in rounds (XIMENES; ALVES; ARAÚJO, 2015).
- In order to allow that every person involved in the project developed a common understanding, the authors utilized model prototypes combined with domain-specific interactive animations, so that the final users could then provide feedback about the model during the simulation iteratively (GABRYSIAK; GIESE; BEYHL, 2012).
- The authors point out that the use of physical artifacts can be efficient for representing an idea or an abstract concept, simplifying the objectives of the project for the participants (NEWMAN et al., 2015).
- Just as for changes in a company, the intense management support for DT is essential as a way to avoid the initial steps that may seem ineffective or take too long, and the moral fluctuation in teams that can appear sometimes (MAHE et al., 2020).
- In order to prepare the participants in advance, only sending an invite via e-mail is not enough, because it might lead to reactions of indifference or even refusals. It is especially important to clearly explain to management what will be

requested from the workers, as well as to actively involve the management in the organization of DT (MAHE et al., 2020).

- It is important that there is a dedicated notetaker, given that the underlying identified problems within the activity are obfuscated by the many suggested solutions. It would be necessary to record not only the words that are said, but even the non-verbal behavior of the participants, for example when deciding upon features priorities. On top of that, the richness of information produced in artefacts during the workshop must be carefully archived and indexed in order to support future developments (MAHE et al., 2020).

Discussion: The suggestions of improvements to the method were compiled from practical verifications by the authors for obtaining better results. It is not reasonable to presume that only the application of these should mitigate all of the compiled problems among the articles. It is also to be noted that their focus lies on DT itself, and not on requirements.

These could be, however, interesting points for avoiding some of the problems that arise from the occasional little experience of applicants of the method (especially those that are doing it for the first time), such as the need to structure DT in advance and to promote an organizational structure that allows the efficient application of the approach and aligned to the company long term goals.

The recommendations are, for the most part, turned towards the upfront application of DT, as presented in Section 3.3. They are also not formal recommendations and with proven effectiveness, what is justified by the format of the articles used for the review: their goal (for each one) is not to search for improvements, but rather to report their use within Software Engineering.

4.1.7 Threats to Validity

This section presents some of the possible threats to validity identified in this study.

A first threat to validity lies in the fact that there is a natural limitation to a literature review, in that it is not always possible to capture every intended piece of information to answer a research question only through published articles. This is acceptable, since the review is a secondary study, and, as such, it is limited in the text by what the authors of each article judged most relevant to be reported and also according to each article’s objective.

A second threat to validity is in the scope of this study. The review sought out to identify problems and improvements to Design Thinking related to Software Engineering projects that had a direct link with the Requirements Engineering (RE) field. Although the approach is especially manifested within RE, it is possible that there are potential impacts in other stages of the software life cycle development that are not captured by this review. In this way, it is acknowledged that this review is not exhaustive, given that the authors looked for studies that explicitly correlated to the Software Engineering field, but it is possible that other works regarding it have been ignored.

A third and relevant threat to validity that is also important to remark is that, although the recommendations of Kitchenham and Charters (2007) for conducting the systematic review with the reduction of researcher bias, the individual review of each extracted data of the articles did not go through a peer review activity.

4.2 Chapter Considerations

The review had the main goal of empirically identifying the existence of problems related to the DT approach for Requirements Engineering. The articles found within the selection filter presented in the research methodology have shown points of weakness of DT that are yet unsolved by its current state of approach and with the available tools within the Requirements Engineering field.

Several of the reported problems are linked to the presence of some external factors to the approach that are not directly dealt with by it. Among these, some can be cited: the corporate structure, the limitation of available resources, the experience of the process participants, the inadequacy of project size, and the lack of mechanisms for decision taking regarding requirements prioritization.

Among the identified problems we highlight the negligence of non-functional requirements (NFR), as shown especially by Uebernickel and Hehn (2018), Mahe et al. (2020) and Newman et al. (2015). One of the perceptions is that the emphasis exclusively on the client and on the problem to be solved focuses on some NFR, namely usability, taking for granted (and ignored) other NFR, such as security, performance, and maintainability.

Something to be also considered is the formal definition of the approach. As presented by (WAIDELICH et al., 2018), there is not a rigorous process for applying Design Thinking, and as such, it expected that new applicants of the approach can suffer from some of the reported problems due to inexperience.

Furthermore, it is also worth noting that the main question within this review is RQ1, regarding the related problems. In this way, the suggestions for improvement can be seen as a byproduct of it, and it is reasonable to consider further studies to answer RQ2 in a broader manner.

Next, in order to investigate if there are problems with the elicitation of non-functional requirements in DT, we turned to a survey conducted with software developers.

5 SURVEY: NON-FUNCTIONAL REQUIREMENTS IN DESIGN THINKING

Following up on the findings of the literature review, where some articles pointed out to the existence of issues with NFR in DT (HEHN; UEBERNICKEL, 2018; MAHE et al., 2020; NEWMAN et al., 2015), it was not enough evidence to confirm the issue of non-functional requirements in Design Thinking.

It is important to take note beforehand that Design Thinking is a technique focused on innovation, and it shouldn't be necessarily used as a standalone technique for Requirements Engineering. Some projects, however, make use of DT results for eliciting requirements. In such projects, it was uncertain whether this could lead to problems in missing requirements or inappropriate results, and if there were already known techniques for dealing with said question. To help in answering this, it was necessary to turn to primary data, and it was decided that a survey would be the best alternative.

In this way, this chapter goes into detail regarding the survey[2] that was conducted to investigate the elicitation of NFR with Design Thinking.

5.1 Method

This section elucidates the methodologies applied during the execution of the survey. The survey protocol was developed in alignment with the structure advocated by Linåker et al. (2015) and grounded in the guidelines from Kitchenham and Pfleeger (2002). The first, expanded upon the principles of the latter, provides a guideline for performing surveys within the context of software engineering, emphasizing the necessity for methodical preparation, deployment, and analysis of surveys.

The process begins with the planning stage. Similar to the systematic literature review, a survey also starts with formulating a research question. This is then used to identify

[2] This survey was published in (PINTO; BRANDÃO; SIQUEIRA, 2022).

the target population and to design the survey. The target population should be carefully selected to accurately reflect the research question's scope, while the survey should be designed to extract information that would lead to valid and reliable answers. In software engineering, a typical target population might include software developers, project managers, or users.

After designing the survey, a pilot test should be conducted. This helps to verify the clarity and appropriateness of the survey questions, identify potential issues, and ensure that the collected data can be correctly interpreted. Ideally, the pilot test should be performed with a small sample of the target population. Feedback from the pilot test should be used to improve the survey design before its final deployment.

Once the survey is ready for deployment, different methods can be used to distribute it to the target population, such as email, social media, or even physical mail. The chosen method should ideally be one that maximizes response rates while also being cost-effective and efficient. Linåker et al. further emphasize the importance of encouraging participation, which could be achieved by explaining the survey's purpose, ensuring respondent's confidentiality, and possibly offering incentives.

Upon collection of the survey data, it's time to analyze the responses. The data analysis should be based on the research question, and could include descriptive statistics, inferential statistics, or qualitative analysis. Furthermore, researchers should be aware of potential bias and error in the survey responses, and should make necessary adjustments in their analysis to account for these.

Lastly, the findings of the survey should be reported in a clear, transparent, and honest manner. The report should include the methodology used, the characteristics of the respondents, the main findings, and any limitations or potential sources of bias. Like with a systematic literature review, the reporting should be done in such a way that it allows for the study to be replicated.

Throughout the survey process, it's crucial to adhere to ethical considerations. This includes ensuring informed consent from participants, maintaining confidentiality of responses, and being honest and transparent about the research process and intentions. Following these guidelines will help to ensure that the survey is both reliable and ethical.

The process began with the establishment of a clear goal and research questions, as detailed in Sections 5.2 and 5.3.

The subsequent phase focused on the definition of the population sampling, the details of which are elaborated in Section 5.4. The selection process aimed to capture a broad spectrum of experiences and perspectives relevant to the research focus. Once the target sample size and profile were defined, the specific sampling method was outlined in Section 5.5. This part of the process involved identifying potential respondents and inviting them to participate in the survey.

The questionnaire, detailed in Section 5.6, was the primary chosen tool in collecting data from the sample population. It was designed around the research questions, with the aim of profiling respondents and assessing their perceptions of Design Thinking and non-functional requirements during DT as well as during the software development per se. The questionnaire was refined after a pilot trial, and the final version is here presented, as was asked to all participants.

Following this, Section 5.7 provides a thorough compilation of the study results, interpreting the responses from the questionnaire to derive meaningful patterns, trends, and insights. Finally, Section 5.8 offers a comprehensive discussion of each research question in light of the findings.

5.2 Goal and Research Questions

The main goal of the survey was to analyze whether Design Thinking faces challenges with the elicitation of non-functional requirements in software development projects. As these missing requirements may manifest only during the software development life-cycle, the survey has investigated the perception of participants during DT as well as during the following software development phases, considering the use of Design Thinking in RE as an upfront approach in the Stakeholders and Requirements definition, as discussed in Section 3.3.

5.3 Research Questions

According to the defined goal of the survey, the following research questions were proposed:

RQ1: *Are there problems regarding non-functional requirements elicitation that can arise with the Design Thinking approach?*

RQ2: *Are there non-functional requirements that are not addressed during the DT phase and become known only during the software development phase?*

RQ3: *Does the software community have tools for mitigating issues regarding other non-functional requirements?*

5.4 Population Sampling

This survey was aimed at software developers that have had experience with Design Thinking in at least one software development project that reached some stage of development. This survey is qualitative in its nature, given its objectives of better understanding the questions concerning non-functional requirements elicitation with Design Thinking, rather than seeking confirmation to a certain hypothesis.

A qualitative research is used for the investigation of situations in which people are involved and different kinds of processes take place (DYBÅ et al., 2011). It aims to understand the reason and mechanisms explaining a phenomenon (FELDERER; TRAVASSOS, 2020). One major difference between qualitative and quantitative research approaches regards the population sampling. Quantitative approaches usually involve probability sampling, whereas qualitative approaches require purposeful sampling. That is, qualitative researchers value the deep understanding that comes from information-rich cases while quantitative researchers value the generalizations to larger populations given the random and statistically representative samples (SANDELOWSKI, 1995).

A concept usually adopted in qualitative researches is that of data saturation, which is the point at which no new information or themes are observed in the data from the completion of additional interviews or cases (GUEST; BUNCE; JOHNSON, 2006). Although helpful in the conceptual level, it does not provide guidance for estimating actual sample sizes, prior to data collection (GUEST; BUNCE; JOHNSON, 2006). As Boddy (2016) puts it, the issue of defining the appropriate sample size depends on the context and scientific paradigm of the research. And how to determine an adequate sample size for qualitative studies is not yet a consensus. Boddy (2016) comprised a list of sample sizes suggested by researchers, which vary in a range of 15 to 50 interviews. Sandelowski (1995) suggests that a large sample size for qualitative studies is of over 50 interviews. Creswell (2013) divided qualitative inquiries among five approaches, of which this study is better represented as a phenomenological study, and for such, a range of 5 to 25 interviews is recommended.

Summing up, the sample size chosen for this survey was of 50 respondents. The following section will go into details regarding how these cases will be obtained as well as how to assure their purposefulness for the study.

5.5 Description and Design of Sampling Method

This section describes the strategy that was followed for conducting the survey.

The potential interviewees for the survey were reached via the LinkedIn network. As described in the previous section regarding the sampling, 50 purposeful answers had to be obtained. For that, the first filter was to reach for software developers that have had experiences with the Design Thinking approach via the search filters. The chosen query for representing this population was:

```
“design thinking” and “developer” and “software”.
```

The results were comprised of people that have used all keywords in their profile. As of January-2021, there were 54,000 results to this search worldwide.

The LinkedIn search orders the results by relevance, according to the proximity degree between the people in the results and the one who is searching (degree of relationship, location, companies, field of interest, etc.). As the researcher who used LinkedIn is from Brazil, most of the results were from this country. Therefore, the questionnaire has been applied in Portuguese to assure that the interviewees understand the meaning of each survey question precisely. The answers were then translated back to English afterwards to be presented here. This is not novel, and has been previously done in qualitative surveys, as in Wagner et al. (2020).

A pilot for the survey was conducted prior to its release with a fellow Requirements Engineering researcher, who acted as a potential respondent and read each question individually to check if they could be understood as they should, proposing improvements that were incorporated in the survey.

The next step was then to contact the people with the matching criteria and ask them for help in answering the questionnaire. A message was provided with the context to the study and a friendly invitation to participate. Invitations have been sent daily until the desired amount of 50 answers were met.

The survey was entirely conducted with the use of the LinkedIn social network tool over a period of two months (feb/21-apr/21). A total of 669 invites were sent to professionals that fit the matching criteria, and a total of 53 responses were received and further processed. This adds up to a response rate of roughly 8%.

As of 2021, there were several difficulties in conducting surveys on LinkedIn. The first is the limitation of messaging people out of one’s direct network without adding them. A free account cannot do it at all, and a Premium account (over several plans) has a strong limit on the number of messages that can be sent periodically. Another difficulty is in the size of messages when adding someone: 300 characters. The original invitation letter sent to developers is available in 0.

Although not ideal, the main benefit of this approach is the possibility of reaching out to many industry professionals in a direct manner. It also helps with the reduction of several biases that could appear in different circumstances (especially due to unavailability), such as when interviewing professionals from the same company or professionals known to the interviewer. As will be shown in the next section, the 53 interviewees comprise a diverse group.

The answers were recorded in Google Forms and then sent to a spreadsheet where the data could be properly analyzed.

5.6 Questionnaire

This section presents the questionnaire that was answered in Table 2. The questions are presented in the exact order as seen by the respondents.

Software projects that make use of Design Thinking can differ in several aspects, such as the number of sessions, the scope, the number of participants and other techniques that were applied apart from DT. However, there was one underlying hypothesis in the survey, that the project made use of the DT upfront approach in the Stakeholders and Requirements definition (not necessarily as the single technique), and it was followed by some stage of software development, where the results of DT could then be used to produce code. The questionnaire used the term “Design Thinking Workshop” as a commonly referred term to represent this approach, as described in Section 3.3.

This simplified model was created to facilitate tackling the potential shortcomings of Design Thinking, for they can show up either during DT, or after it, during software development. And in this manner, the software developer must also have participated in both phases of said project.

The first part (questions 1-6) presents profiling questions aimed at extracting data from different technological contexts, identifying validation threat biases from too many answers from same company and categorizing answers. Among the profiling questions, two of them asked the interviewee for their experience in years with Software Development as well as with Design Thinking and one asked for the number of projects they participated that have used Design Thinking. Should the interviewee have no experience with either, their response would be discarded (considering the population described in Section 5.4).

The second part (questions 7-11) concerns the experience during DT, with questions that aimed at identifying if DT lacks in NFR elicitation, and what techniques that were used could help in mitigating the elicitation.

The last part (questions 12-16) refers to the experience after DT and during the development phase, and focused on identifying the potential problems with NFR that were not properly treated during DT, and thus came up in a later stage of the project, or not at all.

Table 2 – Questionnaire used in the survey, indicating the type of the question and the research question it relates to.

Question	Type	RQ
1. City of residence / State of residence	Open	-
2. What company do you currently work for?	Open	-
3. What is your current organizational role or function?	Closed	-
4. What is your experience, in years, with software development?	Number	-
5. What is your experience, in years, using Design Thinking?	Number	-
6. How many projects have you participated in that used Design Thinking?	Number	-
7. On a scale of 1 (extremely dissatisfied) to 5 (extremely satisfied), how was your satisfaction with the Design Thinking approach as a whole regarding requirements elicitation?	Scale	RQ1
8. On a scale of 1 (extremely unsuccessful) to 5 (extremely successful), how successful do you consider the Design Thinking workshop to be in eliciting functional requirements?	Scale	RQ1
9. On a scale of 1 (extremely unsuccessful) to 5 (extremely successful), how successful do you consider the Design Thinking workshop to be in eliciting the following non-functional requirements? [16]	Multiscale	RQ1
10. What support techniques were used for eliciting non-functional requirements during the Design Thinking Workshop (except usability)?	Open	RQ3
11. What difficulties or negative points regarding the elicitation of non-functional requirements (except usability) did you identify during the DT workshop?	Open	RQ1
12. In your opinion, what percentage of completion was reached regarding the implementation of the system thus far?	Closed	-
13. In your opinion, what percentage of completion was reached regarding the deployment of the system thus far?	Closed	-
14. The Design Thinking workshop usually delivers artifacts for the software development activities, such as low-fidelity prototypes or requirements captured via post-its. On a scale of 1 (useless) to 5 (extremely useful), how useful were these artifacts in capturing non-functional requirements (except usability) for the software development?	Scale	RQ1
15. If any non-functional requirements arose during the software development phase that were unforeseen in the DT Workshop, in what categories would these requirements better fit?	Closed	RQ2
16. If any non-functional requirements arose during the software development phase that were unforeseen in the DT Workshop, how did they become known?	Closed	RQ2, RQ3

Source: author.

5.7 Study Results

This section goes into details regarding the obtained results.

5.7.1 Profile of Respondents (Q1 to Q6)

A total of 53 responses were collected. This is a brief detailing of the respondents' profile. Regarding respondents' current role, Table 3 presents the distribution. The most common categories, adding up 85%, were software developers (57%), project owners (19%) and UX/UI Designers (9%). The other 15% comprise consultants, data scientists, researchers, and managing roles.

Table 3 – Respondents' current role.

Role	Number of Respondents	%
Software Developer	30	57%
Project Owner	10	19%
UX/UI Designer	5	9%
Other	8	15%

Source: author.

Table 4 presents the field of work from respondents. A total of 45 companies were listed as the current work from respondents, while 1 respondent was unemployed and 2 were self-employed. 3 fields comprise 73% of the respondents: IT Services & Consulting, Banking & Financial, and Health, Wellness, Fitness & Hospital. The other respondents work in Software Development, Credit Scoring, E-commerce, Educational, News/Media, Oil & Energy, Outsourcing, Real Estate, Research, and Telecommunications.

Table 4 – Field of respondents' work.

Field of Work	Number of respondents	%
IT Services & Consulting	22	42%
Banking & Financial	11	21%
Health, Wellness, Fitness & Hospital	5	10%
Other	14	27%

Source: author.

Most of the respondents were based in São Paulo (51%), but other 9 states have also shown respondents. Table 5 presents this regional distribution.

Considering respondents’ previous experience, Table 6 presents the average, standard deviation, minimum, and maximum values of answers in questions 4, 5, and 6. The average respondent has 9 years of experience with software development and has practiced Design Thinking for 3 years. The earliest adopter of Design Thinking per the answers was in 2013.

Table 5 – Regional distribution of respondents.

Brazilian State	Number of respondents	%
São Paulo	27	51%
Rio Grande do Sul	8	15%
Minas Gerais	7	13%
Paraná	3	6%
Pernambuco	2	4%
Rio de Janeiro	2	4%
Acre	1	2%
Distrito Federal	1	2%
Paraíba	1	2%
Santa Catarina	1	2%

Source: author.

Table 6 – Respondents’ experience.

Question	Avg	Stdev	Min	Max
4. Experience in years with software development	8.9	6.9	1.0	28.0
5. Experience in years with Design Thinking	2.9	1.7	0.5	8.0
6. Number of Previous Projects with Design Thinking	5.1	4.2	1.0	20.0

Source: author.

5.7.2 During Design Thinking (Q7 to Q10)

Table 7 presents the average, standard deviation, minimum and maximum values of answers regarding the perception of successfulness of respondents (in a range of 1 to 5, per questions 7-9).

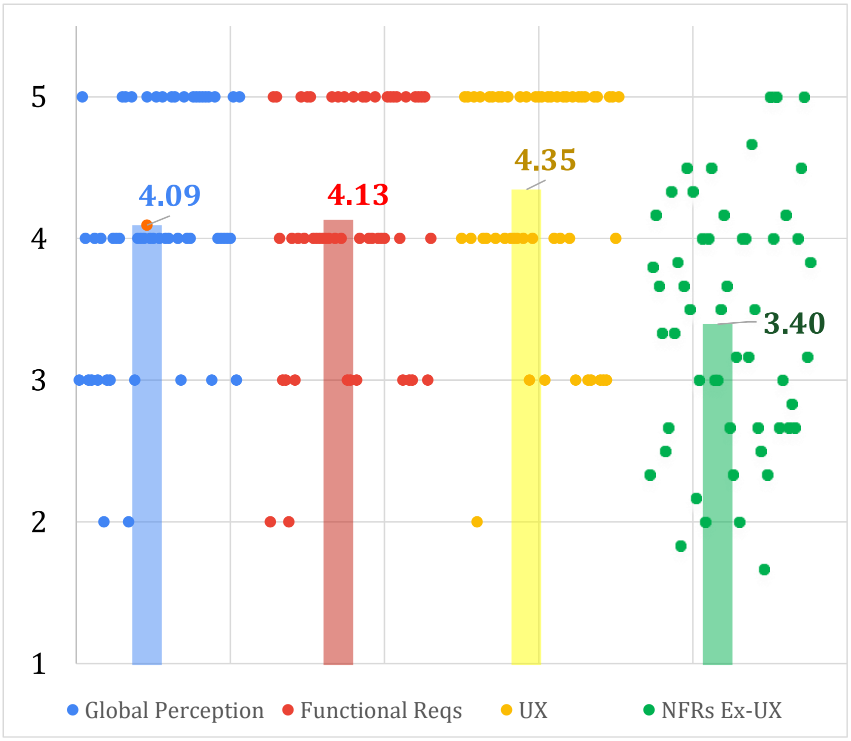
The average successfulness perception with eliciting requirements for Design Thinking has been of 4.09. Almost indistinguishable to the value of 4.13 that was found for the elicitation of functional requirements. This value increases for usability requirements (4.35) but sees a sharp decline for all other Non-functional requirements (averaged excluding usability: 3.40).

Table 7 – Respondents' perception of successfulness.

Successfulness Perception	Avg	Stdev	Min	Max
7. Design Thinking	4.09	0.83	2	5
8. Functional Reqs.	4.13	0.85	2	5
9.a) Usability	4.35	0.81	2	5
9.b) Performance	3.28	1.00	1	5
9.c) Compatibility	3.58	1.13	1	5
9.d) Reliability	3.68	1.13	1	5
9.e) Security	3.28	1.12	1	5
9.f) Maintainability	3.15	1.22	1	5
9.g) Portability	3.40	1.15	1	5

Source: author.

Figure 5 – Scatter plot and average values for the successfulness perception of respondents regarding the elicitation of requirements for DT (Global Perception), Functional Requirements, Usability (UX) Requirements and Non-Functional Requirements excluding Usability.



Source: author.

Most respondents have considered the successfulness perception of both the elicitation of functional requirements (81%) as well as the elicitation of usability requirements (87%) to be greater or equal to the averaged perception regarding the elicitation of the other non-functional requirements (excluding usability). Figure 5 presents the scatter plot for each of these responses, along with the averaged value.

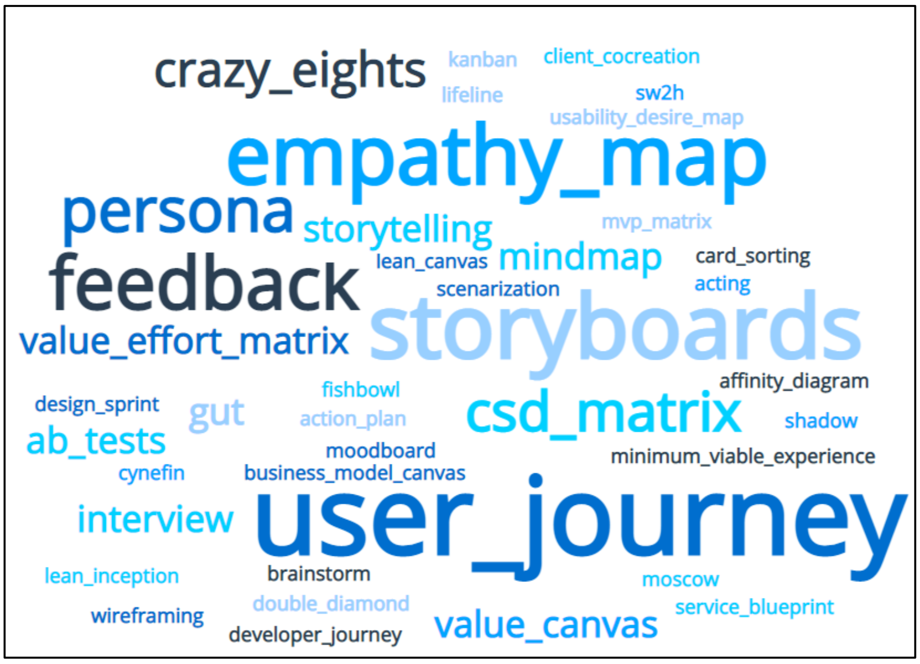
The lowest perception of successfulness in requirements elicitation was found in the categories of maintainability, security, and performance.

As per Question 10 about what techniques were used in the DT, a total of 44 different techniques cited were cataloged. The most popular techniques along with the percentage of respondents that cited them were: User Journey (40%), Storyboards (25%), Empathy Map (19%), Feedback Interviews (15%), Personas (9%), and CSD Matrix (8%). Most respondents have considered the successfulness perception of both the elicitation of functional requirements (81%) as well as the elicitation of usability requirements (87%) to be greater or equal to the averaged perception regarding the elicitation of the other non-functional requirements (excluding usability). Figure 5 presents the scatter plot for each of these responses, along with the averaged value.

The lowest perception of successfulness in requirements elicitation was found in the categories of maintainability, security, and performance.

Figure 6 presents a word cloud of all the suggested techniques by participants.

Figure 6 – Word cloud of Question 10, representing the suggested techniques.



Source: author.

Regarding the difficulties found with the elicitation of Non-Functional requirements with Design Thinking (Question 11), a total of 29 answers were given. Some of the highlights for problems with NFR that were given by respondents are presented as follows (translated to English):

“Some questions like data security or performance are neglected when they are not an evident problem.”

“Requirements like performance and security were hard to be represented.”

“Error tolerance was not foreseen in the process, and it led to a high degree of (software) maintenance.”

“Discussions on non-functional requirements were too fast and superficial.”

“The focus/objective of Design Thinking isn’t in validating NFR.”

“NFR answers aren’t expected, but rather user stories, usability and performance.”

“Scalability, capacity, safety and privacy were not discussed.”

“After DT it is necessary that a dedicated technical team bring up the technical requirements and a list of technical non-negotiables concerning security, maintainability, versioning and portability.”

Some respondents have also commented on topics regarding not only NFR, but rather inherent to the technique and to Requirements Engineering itself. Some of the answers were transcribed as follows:

“It is difficult to preserve the focus of participants and making sure they keep actively contributing.”

“Developers in particular have a hard time transforming the discussions that happened during DT sessions in a real product.”

“The cost-benefit analysis of a presented proposal at the end of the DT sessions was not clear: it was based on fragile hypotheses regarding the users, which proved themselves wrong.”

5.7.3 Software Development post-DT (Q12 to Q16)

This section details the answers regarding the software development that was reached after Design Thinking.

Concerning questions 12 and 13, the average (not median) respondent claimed that the development of the last project they participated in reached a percentage of conclusion of 80%, whereas the deployment reached 71%.

Regarding the perception of usefulness of DT's artifacts (question 14), respondents averaged a value of 4.04 (in a range of 1-5), with a standard deviation of 0.94.

Table 8 – Respondents' perception of classification of unforeseen NFR in DT that arose during Software Development.

Requirements Categories	# Responses	Percentage
15.a) Usability	17	32%
15.b) Performance	26	49%
15.c) Compatibility	15	28%
15.d) Reliability	14	26%
15.e) Security	25	47%
15.f) Maintainability	26	49%
15.g) Portability	12	23%

Source: author.

Table 9 – Respondents' perception to how unforeseen NFR in DT arose during software development.

How NFR arose	# Responses	Percentage
16.a) Testing	37	74%
16.b) Company Management	10	20%
16.c) User Input	26	52%
16.d) Developers Input	33	66%
16.e) Others	3	6%

Source: author.

Table 8 and Table 9 present the answers to questions 15 and 16 about requirements that were not foreseen in DT and became known during the software development. The most cited categories in which they fit were performance (49% of respondents), maintainability (49%) and security (47%). Despite DT's focus on usability requirements, 32% of respondents also affirmed the surfacing of new requirements during this phase.

Most respondents claimed the arise from NFR during testing (74% of respondents), developers input (66%) and user input (52%). No answers were left blank for question 15 and only one was for question 16.

5.8 Discussion

This section brings a discussion of the obtained results from the survey.

5.8.1 RQ1

The first proposed research question aimed at identifying the existence of problems regarding the elicitation of NFR that can arise with the Design Thinking approach.

The main identifiable problem lies in the neglect of non-functional requirements during DT, which was also presented as an issue in the Systematic Review that was presented in Chapter 4. It is also to be remarked the respondents' consensus on a lower successfulness perception of all non-functional requirements excluding usability (Table 7) in comparison to usability, functional requirements and Design Thinking altogether. Several respondents have also commented on this issue directly.

A counterpoint to this was also present: 2 of the 53 respondents have given feedback in reminding that DT's focus is not on NFR, and so this sort of behavior should be expected. One can ponder, however, how relevant are the proposed innovations in Design Thinking expected to be, should they prove unfeasible due to some unidentified NFR.

Another facet of this problem is that the innovation effect with Design Thinking might be somewhat unrealized without the proper guidance that NFR can provide to help with defining what are the design constraints. One respondent commented on the fragile hypotheses that DT results were based on, which proved to be unjustifiable when further analyzed. Another participant also pointed out that the created ideas were too superficial without the correct guidance, and thus rendered useless. This echoes to a case study by Mahé et al. (2020), where the authors reported that DT's focus in the future facilitates the creation of unrealistic expectations and the promises of elements that are seen in DT are left aside.

Question 11 regards the difficulties/negative points with NFR in DT. Some respondents have also focused on issues not only concerning NFR, but also the technique and its results. It is possible to evidence three categories of problems:

- Those of a “natural occurrence”, that are not explicitly linked to DT or Requirements Engineering (RE), but are self-explanatory and can sometimes be addressed with best practices, such as communications problems due to hierarchy differences, or preserving the focus of participants.
- Those inherent to RE and that can be present in DT sessions as well as in other RE techniques and are related to known issues within the RE community, such as the difficulties of developers to transform discussions during DT sessions in real products.
- Those intrinsic to DT, such as that weak hypotheses can lead to wrong conclusions, that user journeys can be too focused on “happy paths”, disregarding the correct guidance of business goals, as well as the previous reported issues on NFR. Problems in this category are the ones where it should be more justifiable to act for improving DT.

5.8.2 RQ2

The second research question aimed at identifying whether NFR not identified during DT would become known only during the software development phase.

This question was addressed mainly with questions 15 and 16, which asked about non-functional requirements that arose during the software development phase and were unforeseen in DT. Firstly, it is important to notice that only 2 of the 53 respondents, when asked on how NFR arose after DT chose to not answer question 16, as it was not mandatory. One respondent asserted that it was out of scope and one left it blank. The other 51 claimed to have found unforeseen NFR that arose during software development. However, no answers were left blank for question 15 (also optional), which brings evidence for confirming RQ2.

The majority of “new” NFR were found via testing (74%) and from developers’ input (66%). This can partly be justified by the majority of respondents’ current role being of software developers (57%). The categories with the least respondents were portability and reliability.

Despite DT’s focus on usability, 32% of respondents claimed that new usability requirements also arose during software development. At this moment and with the given feedbacks, it is not possible to say whether these additional requirements are expected and correspond to natural deviations from prototypes or if it is a potential drawback of DT.

The most problematic categories of unforeseen NFR according to respondents were performance, maintainability, and security.

It is reasonable to incur that DT is strongly concentrated on identifying users’ direct needs and mostly in a functional manner, but short-sighted as to defining the means of how best to fulfill them. Moreover, it offers no clear benchmarks for assessing if said results are adequate or if additional requirements will be needed for improving the overall quality of the underlying software project. Not even for paramount requirements that may ruin the project or incur great costs if not deliberately addressed by the team, which is an existing hindrance.

5.8.3 RQ3

The third research question aimed at identifying whether the software community has tools for mitigating the issue presented and confirmed in the previous research questions. When asked about techniques that were used for eliciting non-functional requirements, a total of 44 support techniques were cited by respondents (Question 10). However, it is conceivable that most of the techniques were simply used in DT as the chosen tools of the session, and not directly as means of improving NFR elicitation.

5.9 Threats to Validity

This section presents some of the threats to validity of the survey.

A first threat to validity is the one regarding additional Requirements Engineering techniques to be considered. Each project has its own decisions regarding approaches to best elicit requirements, and Design Thinking should not be applied as a standalone technique, but rather as a complementary technique aimed at innovation. In this way, from a purely innovative perspective, one could wonder if DT needs any structural changes. However, it's argued that this potential for innovation in DT might not be fully tapped, and NFR can be a key to better understanding how to enhance it.

Another threat to validity regarding RQ2 is that some of the requirements to appear after DT are natural due to agile software development, and that does not necessarily implicate a shortcoming of DT. Also in this manner, it is important to notice that this study was approached by simplifying the possible uses of integrating DT into RE (as per Section 3.3) to the upfront approach in the Stakeholder and Requirements definition. As discussed in Section 3.3, DT can also be used as a mindset or a toolbox and also infused into other RE processes (HEHN et al., 2019). While the upfront approach represents a less resource-intensive approach, it is to be expected that other approaches, such as the use of a higher number of iterations or the infused approach could help mitigate the aforementioned problems.

The sampling method also poses a threat to the validity. The respondents were found using a query to filter relevant profiles in the LinkedIn social network, similarly to (PRESTES et al., 2020). This can be considered a convenience sampling. Although not ideal, other reliable sources for the population could not be identified.

It is also important to notice that, although this work was elaborated with the help of several practitioners that have provided valuable insights, no interviews were conducted that could help deepen the understanding of the reported issues.

5.10 Chapter Considerations

This chapter presented a survey conducted with software developers to analyze whether the Design Thinking (DT) technique faces challenges with the elicitation of Non-Functional Requirements in software development projects. We sent 669 invites using LinkedIn, obtaining 53 responses. We conclude that there can be issues with software development projects especially if Design Thinking is deemed as a standalone technique for eliciting requirements.

In its current state and given DT's focus on functional requirements and usability, it is to be expected that any other NFR be neglected unless it is known beforehand or if explicitly referenced in DT as a user need, and that is a complication. It is not probable, for example, that maintainability requirements will show up during DT, and that can mean a missed opportunity at designing innovative solutions that properly consider said requirement. In this way, the following RE activities can distort what has been discussed and agreed upon during DT simply because they are unachievable, leaving unrealized innovation potential. Similar scenarios can be played out for each of the other NFR categories.

Some users are aware of this shortcoming and turn to complementary techniques, by using DT only to its extent of innovation and creativity. Notwithstanding, one should ponder about the order of such techniques for better results, or even if DT's single results are relevant without the needed additions, in the event of the presence of a NFR that was not identified by DT and that can change the direction of the project entirely.

It is also noteworthy that several NFR can be expected to appear during the software development stages, and so it becomes appropriate to make provisions and adjust estimates due to it.

The work conducted and presented thus far has shown the existence of issues in the current use of DT for eliciting NFR. In this manner, a question still left to be answered and that motivates the conclusion of this research is whether DT should concern itself with it at all.

6 INTERVIEW-BASED FIELD STUDY

Following up on the objective of this work, to analyze how NFR are considered in Design Thinking, regarding its use for Requirements Engineering and based on the results that arose regarding the issues of DT with NFR from the systematic literature review presented on Chapter 4 and the survey presented on Chapter 5. This study aimed at collecting data by interviewing software developers who had experiences with Design Thinking.

6.1 Method

This section presents the method used to execute this study. It was based on the work of Alegroth et al. (2022), in which semi-structured interviews were conducted in a qualitative approach, following the framework of a qualitative case study.

The design of this study is grounded in the work by Runeson et al. (2012). The protocol was crafted following the suggested framework by Brereton et al. (2008). The authors provide detailed guidelines on how to conduct qualitative case study research in software engineering, which was tailored to an interview-based field study.

Furthermore, to ensure the ethical integrity of the research, the study has been submitted for approval to the Brazilian Ethic Committee under Plataforma Brasil approval and has been approved under process number CAAE:65388922.4.0000.0138.

The process commences with the formulation of a research question that defines the problem and purpose of the study, and it should guide the selection of cases and data collection techniques. The research question should be clear, focused, and relevant to the field of software engineering.

Once the research question has been established, researchers should select appropriate case or cases for study. The selected cases should offer insights into the research question and should be a representative sample of the phenomenon under study. The choice between a single-case and multiple-case study design will depend on the nature of the research question and the resources available.

Preparation for data collection is the next critical step. The development of an interview protocol, as proposed by Brereton et al. (2008), is a significant aspect of this stage. An interview protocol is a structured guide that lists the questions or topics that need to be covered during the interview. The protocol helps to ensure that the same basic lines of inquiry are pursued with each person interviewed, thus improving the reliability of the study.

In the data collection stage, researchers conduct interviews based on the established protocol. Interviews can be face-to-face, over the phone, or even via video conferencing. It is essential to create an environment where participants feel comfortable and are willing to share their experiences. Interviewers should be trained to conduct the interview impartially, without influencing the participant's responses.

After data collection, the gathered information is analyzed. The analysis of qualitative data often involves the coding of data into themes or categories and then interpreting these themes to answer the research question. The analysis should be systematic, transparent, and reproducible, ensuring that the conclusions drawn are reliable and valid.

Finally, researchers should report the findings of the study in a transparent and comprehensive manner. The report should include the research question, case selection, data collection and analysis procedures, findings, and limitations. This reporting should be done in such a way that it allows others to replicate the study, enhancing the credibility and reliability of the research. Throughout the entire process, ethical considerations should be maintained, ensuring informed consent, privacy, and confidentiality of the participants.

An elucidation of the context of the study is offered in Section 6.2, which provides essential background information and spells out the research objective and questions. Moving ahead, the study’s design and the rationale behind the number of interviews conducted are encapsulated in Section 6.3.

The selection criteria come into focus in Section 6.4, where the process of identifying suitable interviewees is explained. Ensuing this, Section 6.5 outlines the data collection procedures, offering insights into the systematic approach used for data acquisition.

The procedures for data analysis are then explained in Section 6.6, detailing how the collected data was processed to extract meaningful insights. The research design's robustness, detailed in the study plan's validity, is thoroughly evaluated in Section 6.7.

In Sections 6.8 and 6.9, a closer look at the collected data and its subsequent analysis is provided, Here, the raw interview data for each of the five participants is presented, followed by an exploration of the data analysis process.

Section 6.10 discusses the findings, offering a comprehensive interpretation of the data.

6.2 Background

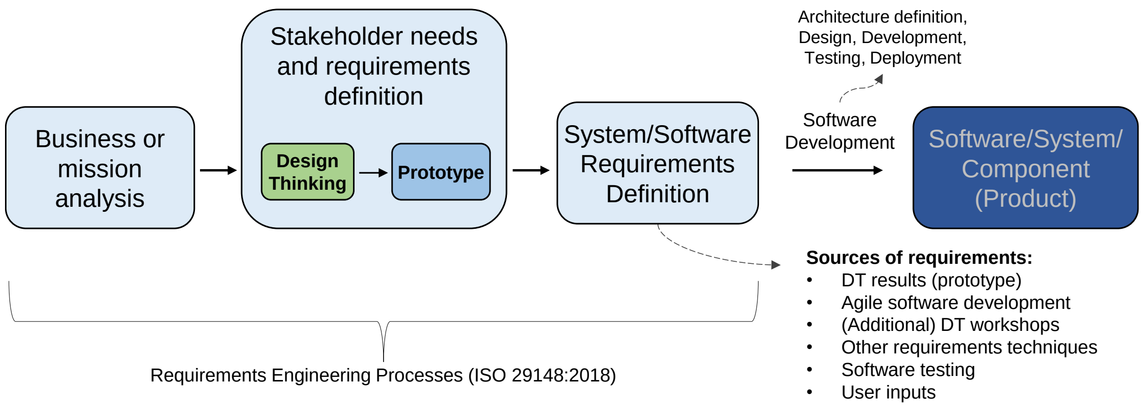
There are several restrictions in a software project regarding its design. They bear an impact on the end product and can be manifested due to several factors: the field of application (context), local regulations, industry standards, stakeholders' choices, cost-efficiency, market desirability, among others.

It is beneficial for the company involved in the project that these restrictions become known as soon as possible, in terms of sparing development resources and in assessing that the desired end-product is feasible.

Some of these design constraints are directly related to non-functional requirements (NFR), among performance efficiency, compatibility, usability, reliability, security, maintainability, and portability (ISO/IEC, 2011). Landes and Studer (1995) have suggested that the NFR establish the justifications for design decisions, and thus restrict the way for the realization of the required functionality.

According to the requirements processes defined by the ISO 29148 (ISO/IEC/IEEE, 2018) and the possible integrations of DT into RE, as explored in Section 3.3, this case study considered the use of DT as an upfront project during the stakeholder needs and requirements definition. That is, DT should deliver one or more prototypes that contemplate the design restrictions that were considered and deemed most important. These prototypes are some of the artifacts that connect to the system/software requirements definition, and precede the software development per se. This relation is depicted in Figure 7. Other important sources of requirements include agile software development, other requirements techniques, software testing, new user inputs, as well as additional rounds of DT.

Figure 7 – Graphical Representation of the context where DT is used as an upfront process.


Business or mission analysis
Stakeholder needs and requirements definition
Design Thinking
Prototype
System/Software Requirements Definition
Software Development
Architecture definition, Design, Development, Testing, Deployment
Software/System/ Component (Product)
Sources of requirements:
DT results (prototype)
Agile software development
(Additional) DT workshops
Other requirements techniques
Software testing
User inputs
Requirements Engineering Processes (ISO 29148:2018)

Source: author.

The primary objective of this case study is to investigate how non-functional requirements are considered in Design Thinking, and how they reflect on design restrictions. The chosen research questions are as follows:

- RQ1: How are NFR discovered during DT?
- RQ2: What used techniques in DT were the best for eliciting NFR?
- RQ3: What could be done for NFR to be discovered during DT rather than after it?

In this manner, this work will explore how DT users address the eliciting of NFR, so as to mitigate problems in the prototype that may reflect on problems to the final product (the software). RQ1 focuses on discovering interviewees' biases and approaches to this issue.

RQ2 focuses on discovering what commonly used techniques in DT are the best ones for discovering new NFR.

Should the desired NFR become known as soon as possible, this would lead to an economy of time and resources. Yet, it is still unclear what prevents NFR from being discovered earlier (during DT). In this way, RQ3 focuses on investigating what were the circumstances that lead to NFR being discovered later in time.

6.3 Design

The object of this study is the use of DT results for RE. It happens primarily in the form of prototypes that capture the design restrictions discovered to be of relevance. These prototypes become an important artifact for the development of the software. In this way, this case study is proposed as a tool for better understanding how this consideration of non-functional requirements during DT can impact the software.

This work is based on qualitative case study design, but it does not comprise an individual case. Rather, it gathers data from several cases – the projects in which each interviewee has participated in the past. Individual projects could be deeply explored, but this study would benefit more from having a few cases available instead, since its objective focused on the relationship between NFR and DT, and a set of cases allowed similarities to be sought for.

A desired number of respondents would be the one to reach a saturation limit, as was previously done in the survey and is described in Section 5.4. Recent studies in the field of software engineering using interviews have used a number of interviewees in the range of 10-20 (ALEGROTH et al., 2022; BRANDT; ZAIDMAN, 2022; FOUNDJEM et al., 2022; RAHMAN; KHOMH; CASTELLUCCIO, 2022).

Creswell (2013) divided qualitative inquiries among five approaches. This study fits between two possible options: (1) a phenomenology study, focused on understanding the essence of an experience lived by a number of people (for which 5 to 25 interviews are recommended), and (2) as a case study, focused on developing an in-depth description and analysis of a case (for which a number of 1 to 4 cases are recommended).

Due to the aforementioned references and the available time for conducting this research, considering its iterative nature, as described by Runeson et al. (2012), a total of 5 interviews were sought out.

Data collection and analysis were conducted between February/2023 and March/2023. Reporting of results were conducted in March/2023.

6.4 Selection

Interviewees were selected based on availability and on experience with Design Thinking and Software projects. The interview itself was conducted in Portuguese, and the results have been translated to English for reporting.

The minimum criteria for participating in the interview was of interviewees to have used Design Thinking for eliciting requirements in a software project in the upfront approach during the Stakeholders and Requirements Definition (as described in Section 3.3), and that have had issues concerning non-functional requirements, be they during DT or after it. This project must have had some feature delivered (if agile) or be in architecture definition (if waterfall), so that some time has passed since the use of DT and the results are perceived and can be reported on.

This study is voluntary, and interviewees could choose to withdraw from it at any time.

The selection was made according to availability via the LinkedIn network, following data from respondents to a previous survey.

6.5 Data Collection Procedures

The data collection phase is comprised of data from semi-structured interviews conducted with developers who have had experiences with Design Thinking in software projects. The interview guide is presented on 0.

Data was collected through online videocalls with each interviewee, and the interviews were then transcribed. Participants were then asked to correct or complement the answers, before they could be analyzed.

This comprised the main data of analysis for this part of the study. Data was stored on on spreadsheets, and is available on Figshare[3].

[3] Available at: <https://doi.org/10.6084/m9.figshare.22269073>.

6.6 Data Analysis Procedures

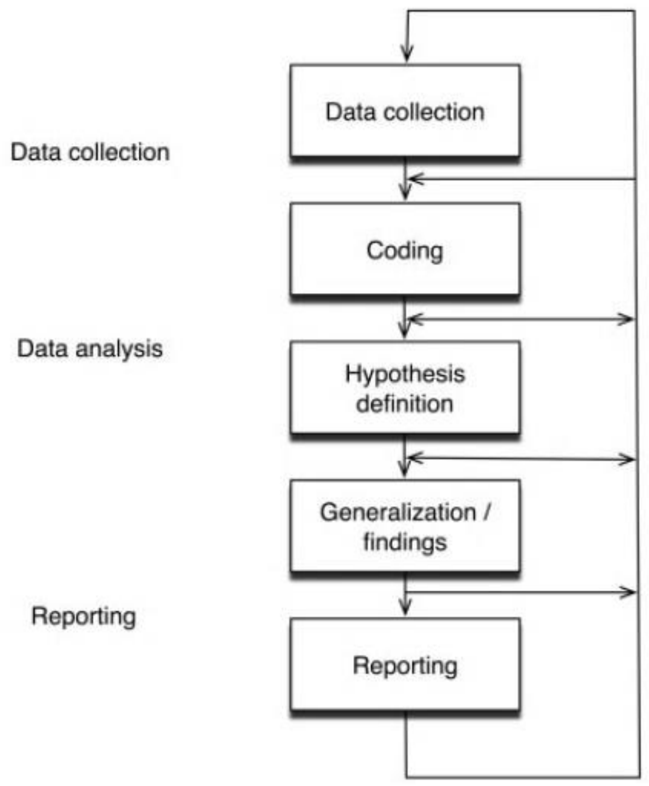
Data was analyzed based on the work of Runeson et al. (2012) regarding the analysis of data in flexible research. It refers to the works of Robson and McCartan (2002).

The data collected from interviews (transcripts) were coded (parts of the text are given a code representing a theme/area/construct). Then the author worked through the material (codes and transcripts) to identify a first set of hypotheses. These hypotheses were then used alongside the previous data available from the project to formulate a small set of generalizations, which can be seen as the final result of the research. Figure 8 depicts the process of data analysis starting from the data collection.

These steps can be conducted iteratively, and as interviews are conducted, and the data is analyzed, the data collection also may be improved and worked on.

All transcripts were recorded and documented, alongside all links between data, codes, and memos. The approach corresponded to an editing approach (ROBSON; MCCARTAN, 2002 apud RUNESON et al., 2012), as it included a few a priori codes, that is, codes defined based on findings of the researcher during the analysis. This work is of an exploratory purpose, and could generate a few hypotheses that are described in the discussion on Section 6.10.

Figure 8 – Main steps of data analysis.



Source: Runeson et al. (2012, p.63).

6.7 Plan Validity

The plan has been checked against Höst and Runeson’s checklist items for the design and the data collection, and the answers are available on 0.

A pilot was conducted with a senior professional in design thinking for assessing its quality. That is, if questions were well formulated and could be understood by interviewees, as well as if other questions were necessary to fully explore the objective of this research. A simulation was run, and this professional answered as an interviewee would, and the results were studied to improve the questions, but were discarded of the comprising data for this study.

The following steps were proposed and followed for each activity in order to attain higher levels of quality assurance, validity and reliability. This derived from the suggestions of Runeson et al. (2012) and have been similarly applied by Alegroth et al. (2022).

- During the case study design, the draft case study was peer reviewed.
- During the interviews, independent observers and peers were invited to observe the conducted interviews.
- After the interview, the interviewers were asked to review and comment on the transcription of the interviews, in order to ensure that the transcripts matched their opinions on the subject.

Threats to validity have also been identified and are explored on Section 6.11.

6.8 Interviews Data

In total, 5 industry professionals were interviewed. They all had previous experiences with Design Thinking and used it as an approach on multiple software projects mainly for eliciting requirements and creating product prototypes. These prototypes are then commonly used as an artifact for developing an MVP: a minimum viable product.

All interviewees were contacted via Linkedin for participating in the project and accepted voluntarily. In total, 12 requests were sent to potential respondents who had listed Design Thinking in their professional page and that had a potential profile to answer. Some of them were respondents to the survey conducted on Chapter 5 who left their contact information and availability to answer. No interview was discarded, since they all matched the previously chosen filter criteria: having at least 1 year working with DT and presenting a previous project in which DT was used in the upfront approach for eliciting requirements.

For confidentiality, the sensible data, including their names, places of work and further details on each reported project were intentionally left out of this text. The full answer transcripts (with filtered sensitive information) are publicly available on Figshare[4].

[4] Available at: <https://doi.org/10.6084/m9.figshare.22269073>.

6.8.1 Participant 1

The first participant was an UX Coordinator in a bank with 6 years of experience with software development, being 3 years with Design Thinking and eliciting requirements. He reported on a project aimed at credit analysis, in which the response for each client should be standardized and automated. The result was a tool that was created to automate the process for the credit analysts.

He worked as a mediator during DT and was in charge of bringing areas together to be involved with the project. Regarding NFR that arose during DT, he reported on data reliability and problems regarding specific data sources. Also, security was a very important topic for the company context, as well as data protection regulations. Among the best techniques for eliciting NFR, he suggested the use of the CSD matrix and prototypes. New NFR did appear especially in the form of usability (more specifically operability or ease of learning): the initial prototype was unfeasible due to it being too ambitious and complex and had to be properly toned down and suited to the company expectations.

6.8.2 Participant 2

The second participant was a Project Manager in a software development company with 7 years of experience in software development, being 2 years with Design Thinking and eliciting requirements. She reported on a project that arose from an internal audit in which an operational risk was detected, and several spreadsheets used in a process that was manual were to be automated into a dashboard. She participated in DT as a Project Manager, and DT was explicitly used in the upfront manner in order to understand each data flow to be automated and prototyped, so that it could then be properly developed.

Regarding NFR that arose during DT, she reported on security and portability requirements due to company restrictive policies. There are hard cybersecurity regulations in place, and company dictates the need for offline tools and restrictions for mobile access. Other NFR were maintainability requirements, due to the way the company operates: the product is sold to the client along with forecasted development hours, which represents developers to be allocated to the project for support and development, without an end in sight. In this way, provisions are made for this needed maintenance. Regarding the best tools, she reported specifically on prototyping for this project, as it helps her to validate with the clients for the adequate solutions.

New NFR did also arise in previously mapped NFR due to new cybersecurity regulations that were put into place. There were also performance-related issues due to the chosen architecture, and the initial infrastructure was no longer suitable once the product was operational. New usability issues also appear due to changes in requirements by the clients along the development of the product, but that should be expected.

6.8.3 Participant 3

The third participant was a Design Lead in a bank (not related to the first participant), and had 4 years of experience in software development, being 2 years with Design Thinking and eliciting requirements. She reported on a project focused on creating a management dashboard for HR to help with the supervision of company personnel. In this project, she worked as a designer during the discovery phase, prototype, and handover to the development. She conducted CSD matrixes, stakeholders map, quantitative and qualitative researches, desk researches, benchmarks, empathy maps, personas creation and ideations with people involved and identified via the stakeholders map.

Regarding NFR problems during DT, she commented specifically on usability, security, due to the context in which the company operates and recent frauds that happened in the sector, as well as performance. The issues were relevant to the context and history of the participants. Since they dealt with sensitive data, data regulations also were relevant to the project. With reference to new NFR, she commented on previously unmapped information that came by and directly relates to security: some data couldn’t be accessed due to company policies, and led to extra development times. She also commented directly on how the learning curve of the participants directly impacts the project, and people with lesser experience in a complex project have little maturity of the processes and approaches, and lead to poorer results.

6.8.4 Participant 4

The fourth participant was a Product Owner in an energy company, and had 6 years of experience in software development, being 3 years with Design Thinking and eliciting requirements. She reported on a project focused on building an international energy e-commerce. She worked as a Scrum Master in the underlying project, and conducted the DT process while running the tools and helping the Product Owners with inputs and capturing needs.

DT was conducted explicitly in the upfront manner for the period of 3 months in which the stakeholders were fully committed to it. Regarding NFR issues during DT, she commented on security issues that were relevant at the time due to a recent hacking attempt at the company. Regarding other topics, little was discussed, apart from reliability: the product was relevant in terms of revenue, and it should be reliable from the start. As for the best techniques in discovering these NFR, some reported on included stakeholders map, personas creation, journey mapping and brainstorming.

As for NFR that arose after DT, she commented specifically on usability. Missing personas led to missing requirements, and this, in turn, led to a huge need for further working. Among the cited learned lessons, this was perhaps the most cited: DT should always involve the end user, and not assume that they are fully known. There can be a range of end users with different profiles, and if misrepresented, the user experience can be imparted.

6.8.5 Participant 5

The fifth participant was a Product Director in a gaming company, and had 11 years with software development, being 9 years with eliciting requirements and Design Thinking. He reported on a project from a previous brokerage firm in which an app was to be developed for automating the execution of trade orders of options. During DT, he conducted the discovery process for assessing whether the business idea was viable and that there was demand for it. A month was spent in DT along with the allocated team. Among the NFR that arose during DT, the most important was reliability, due to the potential of causing losses to the company in the case of a systemic failure that didn’t represent the market conditions at the time.

Among the suggested tools, he reported on desk research, blueprint, user journey and on specific company policies that helped with mapping the needed NFR to mitigate the operational risks associated with the product.

New NFR that arose after DT included portability: the developed app had to be embedded into the brokerage superapp. Also, new UX/UI requirements arose due to mismatching visual identity to the company policies. NFR also became manifest regarding performance issues.

6.9 Data Analysis Results

In this section the data analysis derived from the interviews is presented. Table 10 and Table 11 summarize the answers to the research questions that were chosen on Section 6.1

All interviewees have used Design Thinking as an approach for eliciting requirements in the upfront approach during the stakeholder needs and requirements definition, and had projects to report on.

Table 10 – Summary of participants’ responses to questions pertaining to RQ1.

	RQ1: How are NFR discovered during DT?		
Participant / Question	**8. Were there problems during DT regarding NFR?**	**9. Regarding NFR, how were they discovered during DT?**	**11. Can you recall any restrictions to functionalities in the prototype that arose from the presence of these NFR?**
Participant 1	Data availability, performance, reliability, security, data privacy, portability.	Data privacy regulations.	Data privacy limited the possible solutions. Performance was contemplated taking into account the deadlines, resource management and complexity. Portability oriented the solutions towards the customer common usage: the need to be responsive, mobile-first and offline-first.
Participant 2	Portability, maintainability.	It was already in the project scope.	Restricted code libraries, restricted programming languages, restricted online access. Cloud-specific restrictions and lowcode development.
Participant 3	Usability, security, performance.	Participants' history, Data privacy regulations.	Some parts had to be made invisible and user roles had to be assigned in order to adhere to data privacy policies.
Participant 4	Security, reliability.	Security: Company checklist; reliability: revenue relevance of the product.	The projected software architecture revolved around the reliability requirement.
Participant 5	Reliability.	Company policy for managing operational risk.	Several features had to be developed to contemplate the reliability requirement, such as disabling buttons, or limiting allowed trading amounts based on the user profile.

Source: author.

The cited NFR that arose during DT included: security, reliability, maintainability, portability, and usability.

The cited NFR that arose after DT included: security, portability, performance, and usability.

As per the interviews, it was possible to acknowledge that performance was a topic not usually discussed during DT, being regarded as a technical issue to be addressed by the development team. Security issues on the other hand, were the most mentioned. However, performance did seem to be an item of relevance once the development phase took place.

Table 11 – Summary of participants' responses to questions pertaining to RQ2 and RQ3.

Participant / Question	**RQ2: What used techniques in DT were the best for eliciting NFR?** **10. What techniques were used that brought the best results in discovering new NFR?**	**RQ3: What could be done for NFR to be discovered during DT rather than after it?** **12. Do you recall new restrictions that arose after DT in the form of NFR? How did they translate into the product? (RQ3)**	**13. In your opinion, how could these NFR be discovered earlier, during DT?**	**14. What could have been done differently in the project regarding NFR?**
Participant 1	In depth interviews, CSD Matrix, prototyping and co-creation with the IT team.	Usability: the initial prototype was unfeasible due to it being too ambitious and complex and had to be properly toned down and suited to the company expectations.	A list/framework of NFR could be used for brainstorming and considering whether the class of NFR and whether their most significant stakeholders are present.	No further comments.
Participant 2	Prototyping.	New cybersecurity regulations and company policies. Performance-related issues due to the chosen architecture. New usability due to changes in requirements by the clients.	Trying to foresight the future of the product for addressing possible issues and budgeting reserves.	More access to documenting and information. Lacking knowledge regarding performance of tools. Should spend more time during discovery.
Participant 3	Quantitative and qualitative researches. CSD Matrix.	Security: some previously mapped needed data was not available for use, and new back-end processes had to be created.	The project grew fastly in complexity, and the team lacked experience. Following similar projects were much simpler due to the familiarity of the team.	Discovery demanded more time than was initially planned for. The learning curve of participants has a great impact on its success.
Participant 4	Stakeholders mapping, personas, journey mapping, brainstorming.	Usability. Missing personas led to missing requirements.	The final user has to be directly involved during DT, and all classes of users should be contemplated in the personas.	No further comments.
Participant 5	Desk research, blueprint, user journey and on specific company policies that helped with mapping the needed NFR.	Portability, usability (UX/UI changes to adapt to the company visual identity), code maintainability, performance (new servers had to be created), security (electronic signing of transactions as well as tokens)	More time could had been spent with the discovery process. Specifically, could have investigated the work of other product squads in the company.	No further comments.

Source: author.

Some of the NFR that arose after DT seem to be unrelated to the projects: new management decisions, such as deciding to embed an app. Some are directly linked to regulations: new regulations led to new requirements.

Some of the NFR, however, did seem related to the project, in that they could had been discovered earlier on if properly addressed, including usability (in case the right personas were considered, with missing personas leading to missing requirements), or where the DT prototype was too unfeasible, and had to be refactored.

Interviewees reported on used tools for discovering requirements, and they not necessarily follow any framework. The DT outcome is directly linked to the experience of the participants and in the project leaders choosing the right tools.

Interviewees all agreed on NFR being drivers of the main restrictions behind each project.

6.10 Discussion

This section presents key findings identified from the data analysis, related to the chosen research questions.

6.10.1 RQ1: How are NFR discovered during DT?

The first research question concerned how NFR are discovered during DT: what are the main drivers behind NFR arising during DT. That is, what makes NFR become relevant to the discussion during DT so that they come to light.

The main discovered NFR in each reported project during DT can be enumerated in the following categories:

- **They are known beforehand**: some of the NFR are already known by the team before DT starts. One example is reliability: if a project is deemed too important and it may impact on loss of revenue from system outages, reliability will be discussed and prioritized from the start.
- **They derive from previous experience of DT participants**: participants share stories and significant NFR depending on their backgrounds, previous works/projects they have participated in, their experience with DT, with software in general and with the project scope, as well as the department/field they work in.
- **The company context**: financial forecasts, internal risk assessments and guidelines, management decisions and previous experience of managers, client metrics and indicators (NPS, churn rate, app feedback, among others).

- **Topics common to the market in which the company operates / in which the software is developed**: regulations (such as data protection), historical events (like hacking incidents in the company or in competitors derive the need for security to be discussed),

Some important NFR discovered during DT and that were cited included: reliability, portability, security and usability.

These NFR directly relate to the main restrictions that will arise in the software and impact its design decisions. Some examples cited by the participants are in the form of restricted libraries and programming languages to be used (participant 2); the architecture to ensure reliability of the product (participants 4 and 5); features that must be added to address operational risks (participant 5); visibility of certain data and creation of user roles (participant 3); the need for mobile-first and offline-first solutions (participant 1).

6.10.2 RQ2: What used techniques in DT were the best for eliciting NFR?

The cited techniques by the participants included prototyping, CSD matrix, quantitative and qualitative researches, stakeholder mapping, personas, journey mapping, brainstorming, desk research, blueprints and user journeys.

Design Thinking, as an approach that fosters innovation and has no rigid framework to be followed, can be used freely with a large number of techniques and tools. It is still unclear, however, which are the best for eliciting requirements, and that could provide the best results.

Future research could focus on crossing the use of tools during a Design Thinking session with the results in terms of requirements completeness, functional suitability of the result, as well as indirect indicators of success, such as the user satisfaction with the end result.

6.10.3 RQ3: What could be done for NFR to be discovered during DT rather than after it?

All of the projects reported on had important NFR that were only discovered after Design Thinking sessions took place, and these new NFR added constraints that meaningfully shifted the project to accommodate for them. Additionally, all interviewees agreed that the initial mapped NFR were instrumental in driving the main restrictions behind each project. These conclusions highlight the importance of trying to discover the meaningful NFR as early as possible in a software project, in order to avoid developing suboptimal solutions that may be fully discarded and remade once these new NFR come to light, incurring in needless costs. Additionally, when asked how one could try to anticipate the discovery of these NFR, all interviewees agreed on the need for more time spent on DT exploring user needs. This suggests that companies could take advantage of investing more in the discovery phase of a product before beginning its development.

No matter how exhaustive the elicitation phase, it is however important to allocate resources for unknown requirements, that are yet to be discovered in a project. In spite of the efforts to try and discover as many NFR as possible early on, the DT prototype should not necessarily be seen as a complete solution.

Some interviewees suggested on best practices for trying to discover more NFR early on, and they include:

- **The use of personas**. Personas are fictional characters used to represent the different user types that will use the product in a similar way. It is a tool for helping to empathize with the end user and trying to understand what their motivations are and how they will use it. It is a common approach during Design Thinking, but the team must make sure that the chosen personas truly represent the end user. Among the reported learned lessons, some interviewees commented on missing personas leading to missing requirements during DT. That is, some users were not fully/appropriately considered during DT, and the solution was unsatisfactory in catering to them.
- **Having stakeholders present that can respond for multiple areas involved in the product**. During DT, the result will inevitably be biased by the

stakeholders who are present, and who will report on the questions of most relevance to them. An incomplete team may lead to an incomplete set of requirements, and the team should make an effort to bring significant stakeholders to the sessions.

- **Ideating on a template of NFR** to identify requirements that may not yet be represented and/or may not have present stakeholders that would report on their significance. Some requirements may not be too evident at first, and one suggested good practice was having a list of some basic NFR (or classes of NFR) that the participants could then be asked on whether each is relevant to the problem and if someone else should be included (which could also help with having a complete team of stakeholders). This might be a significant step to take during DT sessions, since there was no agreement in regard to the best used techniques (in DT) for discovering NFR, as they seem to be known beforehand (from the context or history of the participants), or they are not effectively discovered during DT.

6.11 Threats to Validity

Threats to validity have been identified and are treated as follows:

- Interviewees using Design Thinking in a different manner than expected (as a mindset/toolbox as opposed to as a process): this will be filtered out by asking during the profiling how DT was used. If it doesn't match the use as an upfront project, the interview would be discarded.
- Interviewees not familiar with Design Thinking / interviewees not familiar with software development and/or NFR: interviewees will be selected based on their LinkedIn profile reporting experience with Design Thinking as well as with software development, and the profiling questions will help with filtering interviewees that might not be suitable for answering.
- Interviewees not familiar with the project reported on: interviewees will be asked to report on a project that they have personally participated in.
- Hidden biases: interviewees might be willing to lie (albeit possibly even unaware of it), in order to guard for their reputation and/or the project. Also, results and

mistakes might be over or underestimated. To minimize this threat, all data was anonymized.

- The project might not have relevant NFR problems. In this situation, the interview would be discarded.

6.12 Chapter Considerations

This chapter presented a case study conducted with industry professionals who had experience with using Design Thinking for software projects as an upfront approach in the Stakeholder Needs and Requirements definition. It presented the design behind it and the achieved results.

The case study represented the final part of this work, and aimed at exploring how NFR relate to DT. It delved into real software projects to identify the conditions that make certain NFR be relevant during DT, and what are some possible causes for it to appear at a later time.

It also helped establishing the relationship between the presence of significant NFR and how these shifted the design decisions of each project. This brings further evidence to the relevance of discovering NFR as early as possible, so as to improve the quality of the DT results and make the upcoming software activities more effective.

The next chapter presents the conclusion to this work and summarizes the most relevant findings that were brought to light.

7 CONCLUSION

This work helped in exploring the use of Design Thinking (DT) in software projects for eliciting requirements, identified some common issues that arise in it, and dove deeper into the concerns regarding non-functional requirements (NFR), to properly analyze how they are considered into DT and how they reflect on design restrictions. It also brought a collection of lessons, suggested tools and best practices from industry professionals regarding the use of DT and how to improve the elicitation of NFR in it.

A starting point for this work was in bringing some definitions regarding the common uses of Design Thinking, and its integrations in Requirements Engineering. Previous authors identified the possible integrations of Design Thinking as a Mindset, Toolbox or Process (BRENNER; UEBERNICKEL; ABRELL, 2016), or as an Upfront, Infused, or Continuous approach (HEHN et al., 2019). These definitions were expanded to contemplate the underlying software processes, as a way to better identify the use of Design Thinking in which situation. The result was a total of 4 categories of possible DT integrations into RE: (1) DT as an upfront approach in the business or mission analysis, (2) DT as an upfront approach in the stakeholder needs and requirements definition, (3) DT as Toolbox/Infused, (4) DT as Mindset/Continuous.

Design Thinking is traditionally used in the first situation: as an upfront approach in the business or mission analysis. That is, in the problem domain, where the problem itself is not yet well-defined, or the business or mission analysis is not clearly structured. In this moment, it is not yet clear or necessary that the solution will be a software. The focus lies on using DT’s innovation potential for studying the problem and defining preliminary solution classes.

This work chose, however, the second identified class, the use of Design Thinking as an upfront approach in the stakeholder needs and requirements definition, as common ground for the following studies regarding DT in RE.

Starting off, a systematic literature review (SLR) was conducted to identify possible problems that the use of DT for RE might face, and it was presented on Chapter 4. Among its results, the possible neglect of non-functional requirements (NFR) within DT can be highlighted (HEHN; UEBERNICKEL, 2018). The study brought evidence of a possible disregard of requirements during DT for those not of a functional or usability nature.

An article detailing the SLR has been published, and its original quote follows:

> Pinto, Fábio Avigo de Castro, and Siqueira, Fábio Levy. "Problemas do Design Thinking para a Engenharia de Requisitos: uma Revisão Sistemática da Literatura." WER (2020).

Following up, a qualitative survey was then conducted with software developers to analyze whether the Design Thinking (DT) technique faces challenges with the elicitation of NFR in software development projects. The survey was presented on Chapter 5. This study confirmed that there can indeed be issues with the use of DT for software development projects especially if Design Thinking is deemed as a standalone technique for eliciting requirements. In its current state and given DT's focus on functional requirements and usability, it is to be expected that any other NFR be neglected unless explicitly referenced during DT as an explicit user need.

The survey collected responses from 53 industry professionals among software developers, project owners, UX/UI designers, and other respondent profiles. When polled for the perception of successfulness in discovering requirements with DT, respondents gave DT a good perception of success. When asked for independent NFR classes, however, the results fell sharply for all except usability. Additionally, all respondents pointed out new NFR that arose only after DT. The most relevant NFR in terms of frequency were those of a security, performance, and maintainability nature.

The survey helped in pointing out that DT has shown itself to be strongly concentrated on identifying users' direct needs and mostly in a functional manner, but short-sighted as to defining the means of how best to fulfill them. Moreover, DT offers no clear benchmarks for assessing if said results are adequate or if additional requirements will be needed for improving the overall quality of the underlying software project. Not even for paramount requirements that may ruin the project or incur great costs if not deliberately addressed by the team, which is a clear existing hindrance.

This study also helped to show three categories of problems regarding NFR in DT:

- Those of a natural occurrence that are not explicitly linked to DT or RE but are self-explanatory and can sometimes be addressed with best practices, such as communications problems due to hierarchy differences, or preserving the focus of participants.
- Those inherent to RE and that can be present in DT sessions as well as in other RE techniques and are related to known issues within the RE community, such as the difficulties of developers to transform discussions during DT sessions in real products.
- Those intrinsic to DT, such as that weak hypotheses or missing NFR can lead to wrong conclusions and wrong design constraints.

An article detailing the survey has been published, and its original quote follows:

> Pinto, Fábio Avigo de Castro; Brandão, Anarosa Alves Franco; Siqueira, Fábio Levy. “Design Thinking and Non-Functional Requirements Elicitation: A Survey.” Workshop em Engenharia de Requisitos (2022).

A case study was then conducted with industry professionals who had experiences with Design Thinking projects for Requirements Engineering, aiming at exploring problems within software projects regarding NFR in practical uses of DT. Its objective was to properly analyze how NFR are considered in DT and how they reflect on design restrictions. Semi-structured interviews were administered with a total of 5 respondents. The study has been submitted to the “Plataforma Brasil” for approval, and has been approved under process number CAAE:65388922.4.0000.0138. The study was presented on Chapter 6.

The results of the case study brought further evidence to DT disregarding NFR when they are not associated to the experience of the participants, to the company context or related to topics that are common to the market in which the company operates. From the reports, it can be pointed out that NFR imposed vast design restrictions, and shifted the way in which all projects were constructed, which indicated the need for them to be discovered as early as possible.

In this way, one can still argue that DT might not be suited as the best approach for directly eliciting NFR. However, once it is chosen as the main guiding process/framework during the discovery phase of a software project, these conditions regarding NFR must be observed, at risk of jeopardizing the end result or incurring greater costs later on.

Among the results collected and presented in Chapters 4, 5 and 6, some of the most notable best practices that can be used for improving the results of DT are put together:

- NFR impose design restrictions on a software and should be discovered as early as possible for improving a solution in an efficient manner. However, new requirements (including NFR) are expected to arise after the discovery process has been conducted, and provisions should be made for such. Some of them derive from external factors however, and usually can’t be foreseen: management decisions and spin-off projects, new regulations, clients’ new desires, among others. It’s usually beneficial to invest more time in discovering requirements as early as possible, though.
- All industry professionals interviewed agreed on the need for more time having been spent on the discovery phase of a software project before the development effectively began.
- Some NFR can derive not only from tools or the experience from participants, but also from company policies directed at identifying and mitigating risks. Some participants have reported on similar situations being used as a means for discovering NFR pertaining to reliability (from possible revenue losses due to system outages), performance (from company experience and legacy systems), and security (due to company operational risks).
- Design Thinking faces challenges if used as a standalone technique regarding the completeness of requirements and the presence of non-obvious NFR. Some survey participants and interviewees commented on the need for a technical team being present and helping assess the result of the DT solution as a way of validating it and possibly identifying the needed NFR. This shouldn’t be seen as an exhaustive solution, though, since the NFR might not be evident from the prototype, nor have been even discussed during DT.
- Initial literature findings suggested an adequate size of projects for which DT could be properly used: in projects too small where the solution is already known beforehand, it is not necessary, and in projects where the problem to be

explored is too complex, the DT results might not fully contemplate the facets of it, being inadequate. However, DT can still be used as a mindset or as a guiding framework (in its infused form) in larger projects (DOBRIGKEIT; DE PAULA, 2019; GABRYSIAK; GIESE; BEYHL, 2012).

- Reports both from literature and from survey and the case study participants suggest the strong presence of a learning curve regarding the use of DT. The quality of the DT results is directly dependent on the experience of the participants (DOBRIGKEIT; DE PAULA, 2019; GABRYSIAK; GIESE; BEYHL, 2012; HEHN; UEBERNICKEL, 2018; XIMENES; ALVES; ARAÚJO, 2015).
- Among the available tools for eliciting NFR within DT, some can be highlighted, as indicated by industry professionals. These include prototyping, CSD matrix, quantitative and qualitative researches, stakeholder mapping, personas, journey mapping, brainstorming, desk research, blueprints and user journeys.
- Pointing out classes of NFR from a list (such as the one from the ISO/IEC 25010 (2011): Usability, Performance, Compatibility, Reliability, Security, Maintainability, and Portability) and asking DT participants whether the topic has been considered and if stakeholders that could have interests on it are represented is a simple approach that can help with identifying further NFR and assessing the completeness of the set of elicited requirements.

7.1 Future Work

One possible future work is expanding the case study. A total of 5 people were interviewed, which could be viewed as its first iteration. It could be beneficial to bring further experience from different professionals until a saturation point is reached.

A point that has not been directly explored by this research involves the effectiveness of each suggested technique within DT for eliciting requirements (especially NFR). Future research could focus on trying to measure and assess which techniques do bring the best results in terms of requirements completeness and users' satisfaction. One study that could be conducted in this direction would be an experiment with sample groups comparing different sets of techniques.

Other issues that arose from the conducted literature review left with the use of DT for RE are still left to be better explored and have been listed on section 4.1.6.1. Some examples are:

- **Prioritizing requirements during Design Thinking**: The interviewees have different needs and tasks to address. It is hard to maintain the construction with high standards of quality and usability that meet every user's expectations, particularly given the time restriction (XIMENES; ALVES; ARAÚJO, 2015).
- **The use of prototypes**: For complex multi-user systems, the creation of prototypes (which is fundamental to DT) that capture every underlying processes and dataflow is very costly, to the point of it becoming prohibitive (GABRYSIAK; GIESE; BEYHL, 2012). Future works could focus on addressing the use of prototypes. Prototypes have also come up as important tools during the survey and during the case study. Especially in the use of DT as an upfront approach, be it during the business or mission analysis or as during the stakeholder needs and requirements definition, prototypes can help with conveying the message with a visual aid.
- **Proper assessment of time/timing during DT**: The sharing of results of the DT activities and the assurance of a good result in the final product are a barrier. Results must be shared with the whole team in a time-efficient manner, and they are only interesting for most only once the feature is already being developed (DOBRIGKEIT; DE PAULA, 2019). Due to the small planning time, DT can lead to an inappropriate architecture (HEHN; UEBERNICKEL, 2018). DT can be problematic regarding inaccurate effort estimates (HEHN; UEBERNICKEL, 2018).

REFERENCES

ALEGROTH, Emil; KARL, Kristian; ROSSHAGEN, Helena; HELMFRIDSSON, Tomas; OLSSON, Nils. Practitioners’ best practices to Adopt, Use or Abandon Model-based Testing with Graphical models for Software-intensive Systems. **Empirical Software Engineering**, *[S. l.]*, v. 27, 2022. DOI: 10.1007/s10664-022-10145-2.

BEYHL, Thomas; GIESE, Holger. Connecting Designing and Engineering Activities III. *In*: PLATTNER, Hasso; MEINEL, Christoph; LEIFER, Larry (eds.). **Design Thinking Research: Making Design Thinking Foundational**. Understanding InnovationCham: Springer International Publishing, 2016. p. 265–290. DOI: 10.1007/978-3-319-19641-1_16. Available at: https://doi.org/10.1007/978-3-319-19641-1_16. Accessed on: 16 mar. 2020.

BODDY, Clive Roland. Sample size for qualitative research. **Qualitative Market Research: An International Journal**, *[S. l.]*, v. 19, n. 4, p. 426–432, 2016. DOI: 10.1108/QMR-06-2016-0053.

BOURQUE, Pierre; FAIRLEY, R. E.; IEEE COMPUTER SOCIETY. **Guide to the software engineering body of knowledge**. [s.l: s.n.].

BRANDT, Carolin; ZAIDMAN, Andy. Developer-centric test amplification: The interplay between automatic generation human exploration. **Empirical Software Engineering**, *[S. l.]*, v. 27, 2022. DOI: 10.1007/s10664-021-10094-2.

BRAZ, Rafael dos Santos; MERLIN, José Reinaldo; FREITAS GUILHERMINO TRINDADE, Daniela; EDUARDO RIBEIRO, Carlos; SGARBI, Ederson Marcos; JUNIOR, Fabio de Sordi. Design Thinking and Scrum in Software Requirements Elicitation: A Case Study. *In*: (Aaron Marcus, Wentao Wang, Eds.)DESIGN, USER EXPERIENCE, AND USABILITY. DESIGN PHILOSOPHY AND THEORY 2019, Cham. **Proceedings** ... Cham: Springer International Publishing, 2019. p. 179–194. DOI: 10.1007/978-3-030-23570-3_14.

BRENNER, Walter; UEBERNICKEL, Falk (EDS.). **Design Thinking for Innovation: Research and Practice**. [s.l.] : Springer International Publishing, 2016. DOI: 10.1007/978-3-319-26100-3. Available at: https://www.springer.com/gp/book/9783319260983. Accessed on: 27 feb. 2021.

BRENNER, Walter; UEBERNICKEL, Falk; ABRELL, Thomas. Design Thinking as Mindset, Process, and Toolbox. *In*: **Design Thinking for Innovation: Research and Practice**. Cham: Springer International Publishing, 2016. p. 3–21. DOI: 10.1007/978-3-319-26100-3_1.

BRERETON, Pearl; KITCHENHAM, Barbara; BUDGEN, David; LI, Zhi. Using a protocol template for case study planning. **Proceedings of EASE**, *[S. l.]*, v. 2008, 2008.

BROWN, Tim. Design Thinking. **Harvard Business Review**, *[S. l.]*, p. 84–92, 2008.

CANEDO, Edna Dias; PARENTE DA COSTA, Ruyther. The Use of Design Thinking in Agile Software Requirements Survey: A Case Study. *In*: DESIGN, USER EXPERIENCE, AND USABILITY: THEORY AND PRACTICE: 7TH INTERNATIONAL CONFERENCE, DUXU 2018, HELD AS PART OF HCI INTERNATIONAL 2018, LAS VEGAS, NV, USA, JULY 15-20, 2018, PROCEEDINGS, PART I 2018, Berlin, Heidelberg. **Proceedings** ... Berlin, Heidelberg: Springer-Verlag, 2018. p. 642–657. DOI: 10.1007/978-3-319-91797-9_45. Available at: https://doi.org/10.1007/978-3-319-91797-9_45.

CARELL, Angela; LAUENROTH, Kim; PLATZ, Dirk. Using Design Thinking for Requirements Engineering in the Context of Digitalization and Digital Transformation: A Motivation and an Experience Report. *In*: GRUHN, Volker; STRIEMER, Rüdiger (eds.). **The Essence of Software Engineering**. Cham: Springer International Publishing, 2018. p. 107–120. DOI: 10.1007/978-3-319-73897-0_7. Available at: https://doi.org/10.1007/978-3-319-73897-0_7. Accessed on: 16 mar. 2020.

CARROLL, Noel; RICHARDSON, Ita. Aligning healthcare innovation and software requirements through design thinking. *In*: PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING IN HEALTHCARE SYSTEMS 2016, Austin, Texas. **Proceedings** ... Austin, Texas: Association for Computing Machinery, 2016. p. 1–7. DOI: 10.1145/2897683.2897687. Available at: https://doi.org/10.1145/2897683.2897687. Accessed on: 16 mar. 2020.

CHENG, Betty H. C.; ATLEE, Joanne M. Research Directions in Requirements Engineering. *In*: 2007 FUTURE OF SOFTWARE ENGINEERING 2007, USA. **Proceedings** ... USA: IEEE Computer Society, 2007. p. 285–303. DOI: 10.1109/FOSE.2007.17. Available at: https://doi.org/10.1109/FOSE.2007.17. Accessed on: 8 mar. 2021.

CHUNG, Lawrence; DO PRADO LEITE, Julio Cesar Sampaio. On Non-Functional Requirements in Software Engineering. *In*: BORGIDA, Alexander T.; CHAUDHRI, Vinay K.; GIORGINI, Paolo; YU, Eric S. (eds.). **Conceptual Modeling: Foundations and Applications**. Lecture Notes in Computer ScienceBerlin, Heidelberg: Springer Berlin Heidelberg, 2009. v. 5600p. 363–379. DOI: 10.1007/978-3-642-02463-4_19. Available at: http://link.springer.com/10.1007/978-3-642-02463-4_19. Accessed on: 6 apr. 2021.

CRESWELL, John. Qualitative Inquiry & Research Design: Choosing Among Five Approaches. **SAGE Publications**, *[S. l.]*, v. 11, 2013.

DESIGN COUNCIL. **What is the framework for innovation? Design Council's evolved Double Diamond**. 2019. Available at: https://www.designcouncil.org.uk/news-opinion/what-framework-innovation-design-councils-evolved-double-diamond.

DOBRIGKEIT, Franziska; DE PAULA, Danielly. Design thinking in practice: understanding manifestations of design thinking in software engineering. *In*: PROCEEDINGS OF THE 2019 27TH ACM JOINT MEETING ON EUROPEAN SOFTWARE ENGINEERING CONFERENCE AND SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERING (ESEC/FSE 2019) 2019, New York, NY, US. **Proceedings** ... . *In*: ASSOCIATION FOR COMPUTING MACHINERY. New York, NY, US p. 1059–1069. DOI: 10.1145/3338906.3340451.

DOBRIGKEIT, Franziska; MATTHIES, Christoph; PAJAK, Philipp; TEUSNER, Ralf. Cherry Picking - Agile Software Development Teams Applying Design Thinking Tools. *In*: (Peggy Gregory, Philippe Kruchten, Eds.)AGILE PROCESSES IN SOFTWARE ENGINEERING AND EXTREME PROGRAMMING – WORKSHOPS 2021, Cham. **Proceedings** ... Cham: Springer International Publishing, 2021. p. 201–206.

DYBÅ, Tore; PRIKLADNICKI, Rafael; RÖNKKÖ, Kari; SEAMAN, Carolyn; SILLITO, Jonathan. Qualitative research in software engineering. **Empirical Software Engineering**, *[S. l.]*, v. 16, p. 425–429, 2011. DOI: 10.1007/s10664-011-9163-y.

FELDERER, Michael; TRAVASSOS, Guilherme Horta (EDS.). **Contemporary Empirical Methods in Software Engineering**. [s.l.] : Springer International Publishing, 2020. DOI: 10.1007/978-3-030-32489-6. Available at: https://www.springer.com/gp/book/9783030324889. Accessed on: 25 jan. 2021.

FERREIRA, Bruna; BARBOSA, Simone; CONTE, Tayana. Creating Personas focused on Representing Potential Requirements to Support the Design of Applications. *In*: 2018, **Proceedings** ... [s.l: s.n.] p. 1–9. DOI: 10.1145/3274192.3274207.

FOUNDJEM, Armstrong; CONSTANTINOU, Eleni; MENS, Tom; ADAMS, Bram. A mixed-methods analysis of micro-collaborative coding practices in OpenStack. **Empirical Software Engineering**, *[S. l.]*, v. 27, 2022. DOI: 10.1007/s10664-022-10167-w.

GABRYSIAK, Gregor; GIESE, Holger; BEYHL, Thomas. Virtual Multi-User Software Prototypes III. *In*: PLATTNER, Hasso; MEINEL, Christoph; LEIFER, Larry (eds.). **Design Thinking Research: Measuring Performance in Context**. Understanding InnovationBerlin, Heidelberg: Springer, 2012. p. 263–284. DOI: 10.1007/978-3-642-31991-4_15. Available at: https://doi.org/10.1007/978-3-642-31991-4_15. Accessed on: 16 mar. 2020.

GLINZ, M. On Non-Functional Requirements. *In*: 15TH IEEE INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE 2007) 2007, Delhi. **Proceedings** ... . *In*: 2007 IEEE INTERNATIONAL CONFERENCE ON REQUIREMENTS ENGINEERING. Delhi: IEEE, 2007. p. 21–26. DOI: 10.1109/RE.2007.45. Available at: http://ieeexplore.ieee.org/document/4384163/. Accessed on: 6 apr. 2021.

GUEST, Greg; BUNCE, Arwen; JOHNSON, Laura. How Many Interviews Are Enough? **Field Methods - FIELD METHOD**, *[S. l.]*, v. 18, p. 59–82, 2006. DOI: 10.1177/1525822X05279903.

HEHN, Jennifer; MENDEZ, Daniel; UEBERNICKEL, Falk; BRENNER, Walter; BROY, Manfred. **On Integrating Design Thinking for a Human-centered Requirements Engineering**. arXiv, , 2019. DOI: 10.48550/arXiv.1908.07223. Available at: http://arxiv.org/abs/1908.07223. Accessed on: 19 jun. 2022.

HEHN, Jennifer; UEBERNICKEL, Falk. The Use of Design Thinking for Requirements Engineering: An Ongoing Case Study in the Field of Innovative Software-Intensive Systems. *In*: 2018 IEEE 26TH INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE) 2018, **Proceedings** ... . *In*: 2018 IEEE 26TH INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE). [s.l: s.n.] p. 400–405. DOI: 10.1109/RE.2018.00-18.

IEEE Recommended Practice for Software Requirements Specifications. **IEEE Std 830-1998**, *[S. l.]*, p. 1–40, 1998. DOI: 10.1109/IEEESTD.1998.88286.

IEEE Standard Glossary of Software Engineering Terminology. **IEEE Std 610.12-1990**, *[S. l.]*, p. 1–84, 1990. DOI: 10.1109/IEEESTD.1990.101064.

ISO/IEC. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. ISO/IEC 25010:2011. 2011. Available at: https://www.iso.org/standard/35733.html. Accessed on: 9 mar. 2023.

ISO/IEC/IEEE. Systems and software engineering — Life cycle processes — Requirements engineering. ISO/IEC/IEEE 29148:2018. 2018. Available at: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:29148:ed-2:v1:en. Accessed on: 7 mar. 2023.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**. [s.l.] : Technical report, EBSE Technical Report EBSE-2007-01, 2007. Available at: https://www.cs.auckland.ac.nz/~norsaremah/2007%20Guidelines%20for%20performing%20SLR%20in%20SE%20v2.3.pdf. Accessed on: 8 aug. 2014.

KITCHENHAM, Barbara A.; PFLEEGER, Shari Lawrence. Principles of survey research part 2: designing a survey. **ACM SIGSOFT Software Engineering Notes**, *[S. l.]*, v. 27, n. 1, p. 18–20, 2002. DOI: 10.1145/566493.566495.

KOLKO, Jon. Design Thinking Comes of Age. **Harvard Business Review**, *[S. l.]*, v. Organizational Behavior, n. R1509D-PDF-ENG, p. 7, 2015.

LANDES, Dieter; STUDER, Rudi. The treatment of non-functional requirements in MIKE. *In*: (Wilhelm Schäfer, Pere Botella, Eds.)SOFTWARE ENGINEERING — ESEC ’95 1995, Berlin, Heidelberg. **Proceedings** ... Berlin, Heidelberg: Springer, 1995. p. 294–306. DOI: 10.1007/3-540-60406-5_21.

LEVY, M.; HULI, C. Design Thinking in a Nutshell for Eliciting Requirements of a Business Process: A Case Study of a Design Thinking Workshop. *In*: 2019 IEEE 27TH INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE) 2019, **Proceedings** ... . *In*: 2019 IEEE 27TH INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE). [s.l: s.n.] p. 351–356. DOI: 10.1109/RE.2019.00044.

LINÅKER, Johan; SULAMAN, S. M.; MAIANI DE MELLO, R.; HÖST, M. Guidelines for Conducting Surveys in Software Engineering. *[S. l.]*, 2015.

MAHE, Nolwen; ADAMS, Bram; MARSAN, Josianne; TEMPLIER, Mathieu; BISSONNETTE, Sylvie. Migrating a Software Factory to Design Thinking: Paying Attention to People and Mind-Sets. **IEEE Software**, *[S. l.]*, v. 37, n. 2, p. 32–40, 2020. DOI: 10.1109/MS.2019.2958646.

NEWMAN, Peter; FERRARIO, Maria; SIMM, Will; FORSHAW, Stephen; FRIDAY, Adrian; WHITTLE, Jon. The Role of Design Thinking and Physical Prototyping in Social Software Engineering. *In*: 2015, **Proceedings** ... [s.l: s.n.] p. 487–496. DOI: 10.1109/ICSE.2015.181.

PARK, Hye; MCKILLIGAN, Seda. A Systematic Literature Review for Human-Computer Interaction and Design Thinking Process Integration. *In*: (Aaron Marcus, Wentao Wang, Eds.)DESIGN, USER EXPERIENCE, AND USABILITY: THEORY AND PRACTICE 2018, Cham. **Proceedings** ... Cham: Springer International Publishing, 2018. p. 725–740. DOI: 10.1007/978-3-319-91797-9_50.

PEREIRA, Julio Cesar; RUSSO, Rosaria de F. S. M. Design Thinking Integrated in Agile Software Development: A Systematic Literature Review. **Procedia Computer Science**, CENTERIS 2018 - International Conference on ENTERprise Information Systems / ProjMAN 2018 - International Conference on Project MANagement / HCist 2018 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2018. *[S. l.]*, v. 138, CENTERIS 2018 - International Conference on ENTERprise Information Systems / ProjMAN 2018 - International Conference on Project MANagement / HCist 2018 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2018, p. 775–782, 2018. DOI: 10.1016/j.procs.2018.10.101.

PINTO, Fábio Avigo de Castro; BRANDÃO, Anarosa; SIQUEIRA, Fábio Levy. Design Thinking and Non-Functional Requirements Elicitation: A Survey. **WER 2022**, *[S. l.]*, p. 14, 2022.

PINTO, Fábio Avigo de Castro; SIQUEIRA, Fábio Levy. Problemas do Design Thinking para a Engenharia de Requisitos: uma Revisão Sistemática da Literatura. **WER 2020**, *[S. l.]*, p. 14, 2020.

PIRAS, Luca; DELLAGIACOMA, Daniele; PERINI, Anna; SUSI, Angelo; GIORGINI, Paolo; MYLOPOULOS, John. Design Thinking and Acceptance Requirements for Designing Gamified Software. *In*: 2019, **Proceedings** ... [s.l: s.n.] p. 1–12. DOI: 10.1109/RCIS.2019.8876973.

PLATTNER, Hasso; MEINEL, Christoph; LEIFER, Larry (EDS.). **Design Thinking: Understand – Improve – Apply**. Berlin Heidelberg: Springer-Verlag, 2011. DOI: 10.1007/978-3-642-13757-0. Available at: https://www.springer.com/gp/book/9783642137563. Accessed on: 16 mar. 2020.

PRESTES, Matheus; PARIZI, Rafael; MARCZAK, Sabrina; CONTE, Tayana. On the Use of Design Thinking: A Survey of the Brazilian Agile Software Development Community. *In*: (Viktoria Stray, Rashina Hoda, Maria Paasivaara, Philippe Kruchten, Eds.)AGILE PROCESSES IN SOFTWARE ENGINEERING AND EXTREME

PROGRAMMING 2020, Cham. **Proceedings** ... Cham: Springer International Publishing, 2020. p. 73–86. DOI: 10.1007/978-3-030-49392-9_5.

RAHMAN, Mohammad; KHOMH, Foutse; CASTELLUCCIO, Marco. Works for Me! Cannot Reproduce – A Large Scale Empirical Study of Non-reproducible Bugs. **Empirical Software Engineering**, *[S. l.]*, v. 27, 2022. DOI: 10.1007/s10664-022-10153-2.

ROBSON, Colin; MCCARTAN, Kieran. **Real World Research, 2nd Edition**. [s.l.] : John Wiley & Sons, 2002.

ROZANTE DE PAULA, Talita; SANTANA AMANCIO, Tatiane; NONATO FLORES, Jose Adan. Design Thinking in Industry. **IEEE Software**, *[S. l.]*, v. 37, n. 2, p. 49–51, 2020. DOI: 10.1109/MS.2019.2959473.

RUNESON, Per; HÖST, Martin; RAINER, Austen; REGNELL, Björn. **Case Study Research in Software Engineering -- Guidelines and Examples**. [s.l: s.n.]. DOI: 10.1002/9781118181034.

SANDELOWSKI, Margarete. Sample size in qualitative research. **Research in Nursing & Health**, *[S. l.]*, v. 18, n. 2, p. 179–183, 1995. DOI: https://doi.org/10.1002/nur.4770180211.

SCHIMMER, Tobias; MEYER, Johannes. Intertwining Lean and Design Thinking: Software Product Development from Empathy to Shipment. *In*: [s.l: s.n.]. p. 217–237. DOI: 10.1007/978-3-642-31371-4_13.

UEBERNICKEL, Falk; HEHN, Jennifer. Towards an Understanding of the Role of Design Thinking for Requirements Elicitation - Findings from a Multiple-Case Study. *In*: 2018, **Proceedings** ... [s.l: s.n.]

VETTERLI, Christophe; BRENNER, Walter; UEBERNICKEL, Falk; PETRIE, Charles. From Palaces to Yurts - Why Requirements Engineering Needs Design Thinking. **IEEE Internet Computing**, *[S. l.]*, v. 17, 2013. DOI: 10.1109/MIC.2013.32.

VOLKOVA, Tatjana; JĀKOBSONE, Inga. Design Thinking as a Business Tool to Ensure Continuous Value Generation. **Intellectual Economics**, *[S. l.]*, 2016. DOI: 10.1016/j.intele.2016.06.003.

WAGNER, S., MENDEZ, D., FELDERER, M., GRAZIOTIN, D., KALINOWSKI, M. Challenges in Survey Research. *In*: **Felderer, M., Travassos, G. (eds) Contemporary Empirical Methods in Software Engineering.** Cham: Springer, 2020. Available at: https://link.springer.com/chapter/10.1007/978-3-030-32489-6_4. Accessed on: 19 jun. 2022.

WAIDELICH, Lukas; RICHTER, Alexander; KOELMEL, Bernhard; BULANDER, Rebecca. Design Thinking Process Model Review. *In*: 2018, **Proceedings** ... [s.l: s.n.] p. 1–9. DOI: 10.1109/ICE.2018.8436281.

XIMENES, Bianca H.; ALVES, Isadora N.; ARAÚJO, Cristiano C. Software Project Management Combining Agile, Lean Startup and Design Thinking. *In*: (Aaron Marcus, Ed.)DESIGN, USER EXPERIENCE, AND USABILITY: DESIGN DISCOURSE 2015, Cham. **Proceedings** ... Cham: Springer International Publishing, 2015. p. 356–367. DOI: 10.1007/978-3-319-20886-2_34.

APPENDIX A - PAPERS IDENTIFIED IN THE SYSTEMATIC LITERATURE REVIEW

This appendix presents the list of articles identified and used during the Systematic Literature Review. They are presented on Table 12.

Table 12 – Selected articles for the Systematic Literature Review.

#	Title	Reference
1	Using Design Thinking for Requirements Engineering in the Context of Digitalization and Digital Transformation: A Motivation and an Experience Report	(CARELL; LAUENROTH; PLATZ, 2018)
2	The Use of Design Thinking in Agile Software Requirements Survey: A Case Study	(CANEDO; PARENTE DA COSTA, 2018)
3	Design Thinking and Scrum in Software Requirements Elicitation: A Case Study	(BRAZ et al., 2019)
4	Intertwining Lean and Design Thinking: Software Product Development from Empathy to Shipment	(SCHIMMER; MEYER, 2012)
5	Software Project Management Combining Agile, Lean Startup and Design Thinking	(XIMENES; ALVES; ARAÚJO, 2015)
6	Virtual Multi-User Software Prototypes III	(GABRYSIAK; GIESE; BEYHL, 2012)
7	Design Thinking in Practice: Understanding Manifestations of Design Thinking in Software Engineering	(DOBRIGKEIT; DE PAULA, 2019)
8	Aligning Healthcare Innovation and Software Requirements through Design Thinking	(CARROLL; RICHARDSON, 2016)
9	The Role of Design Thinking and Physical Prototyping in Social Software Engineering	(NEWMAN et al., 2015)
10	Creating Personas Focused on Representing Potential Requirements to Support the Design of Applications	(FERREIRA; BARBOSA; CONTE, 2018)
11	The Use of Design Thinking for Requirements Engineering: An Ongoing Case Study in the Field of Innovative Software-Intensive Systems	(HEHN; UEBERNICKEL, 2018)
12	Design Thinking in a Nutshell for Eliciting Requirements of a Business Process: A Case Study of a Design Thinking Workshop	(LEVY; HULI, 2019)
13	Design Thinking and Acceptance Requirements for Designing Gamified Software	(PIRAS et al., 2019)
14	Migrating a Software Factory to Design Thinking - Paying attention to people and mindsets	(MAHE et al., 2020)

Source: author.

APPENDIX B - SURVEY DATA

This section presents the survey data that was presented on Chapter 5, including the original questionnaire, the invitation and the list of suggested techniques.

Table 13 – Original questionnaire (in Brazilian Portuguese).

Pergunta
1. Concordo em participar desde estudo de pesquisa. Entendo o propósito e a natureza do estudo e estou participando voluntariamente. Entendo que posso me retirar do estudo a qualquer momento, sem qualquer penalidade ou consequência.
2. Qual é a sua cidade e estado de residência?
3. Para qual empresa você atualmente trabalha?
4. Qual é o seu papel ou função organizacional?
5. Qual é a sua experiência, em anos, com desenvolvimento de software?
6. Qual é a sua experiência, em anos, usando Design Thinking?
7. Você participou de quantos projetos que utilizaram Design Thinking?
8. Em uma escala de 1 a 5, qual foi a sua satisfação com a abordagem de Design Thinking como um todo no que se refere à elicitação de requisitos?
9. Em uma escala de 1 a 5, quão bem-sucedido você consideraria o Workshop de Design Thinking para a elicitação de requisitos funcionais?
10. Em uma escala de 1 (extremamente malsucedido) a 5 (extremamente bem-sucedido), quão bem-sucedido você consideraria o Workshop de Design Thinking para a elicitação dos seguintes requisitos não-funcionais? [Usabilidade]
11. Em uma escala de 1 (extremamente malsucedido) a 5 (extremamente bem-sucedido), quão bem-sucedido você consideraria o Workshop de Design Thinking para a elicitação dos seguintes requisitos não-funcionais? [Desempenho]
12. Em uma escala de 1 (extremamente malsucedido) a 5 (extremamente bem-sucedido), quão bem-sucedido você consideraria o Workshop de Design Thinking para a elicitação dos seguintes requisitos não-funcionais? [Compatibilidade]
13. Em uma escala de 1 (extremamente malsucedido) a 5 (extremamente bem-sucedido), quão bem-sucedido você consideraria o Workshop de Design Thinking para a elicitação dos seguintes requisitos não-funcionais? [Confiabilidade]
14. Em uma escala de 1 (extremamente malsucedido) a 5 (extremamente bem-sucedido), quão bem-sucedido você consideraria o Workshop de Design Thinking para a elicitação dos seguintes requisitos não-funcionais? [Segurança]
15. Em uma escala de 1 (extremamente malsucedido) a 5 (extremamente bem-sucedido), quão bem-sucedido você consideraria o Workshop de Design Thinking para a elicitação dos seguintes requisitos não-funcionais? [Manutenibilidade]
16. Em uma escala de 1 (extremamente malsucedido) a 5 (extremamente bem-sucedido), quão bem-sucedido você consideraria o Workshop de Design Thinking para a elicitação dos seguintes requisitos não-funcionais? [Portabilidade]
17. Quais técnicas de apoio foram utilizadas para elicitar requisitos não-funcionais (exceto usabilidade) durante o Workshop de Design Thinking?
18. Quais dificuldades ou pontos negativos na elicitação de requisitos não-funcionais (exceto usabilidade) durante o Workshop de Design Thinking você identificou? Caso nenhuma tenha sido identificada, deixe em branco.
19. Qual percentual (aprox.) de completude (0-100%) foi alcançado no que se refere à implementação (desenvolvimento) do software até o momento?
20. Qual percentual (aprox.) de completude (0-100%) foi alcançado no que se refere à implantação (*deployment*) do software até o momento?
21. O Workshop de Design Thinking usualmente produz artefatos para as atividades de desenvolvimento de software, como protótipos de baixa fidelidade ou requisitos capturados através de post-its. Em uma escala de 1 a 5, quão úteis foram estes artefatos em capturar requisitos não-funcionais (exceto usabilidade) para o desenvolvimento do software?

Continues on the following page.

Conclusion.

Pergunta
22. Se um ou mais requisitos não-funcionais surgiram durante a etapa de desenvolvimento de software e não foram previstos no Workshop de Design Thinking, em quais categorias estes requisitos melhor se encaixariam?
23. Se um ou mais requisitos não-funcionais surgiram durante a etapa de desenvolvimento de software e não foram previstos no Workshop de Design Thinking, como estes requisitos adicionais se tornaram conhecidos?
24. Gostaria de deixar algum feedback, comentários ou sugestões para melhorar esta pesquisa?
25. Se você gostaria de ser informado sobre os resultados desta pesquisa, por favor deixe um e-mail de contato, e será um prazer compartilhar os avanços.
26. Agradecemos imensamente a sua participação! Você permite ser contatado futuramente para tirarmos dúvidas sobre suas respostas no questionário?

Source: author.

The following text is the original invitation (in Brazilian Portuguese) that was sent to software developers via LinkedIn when asked to participate in the survey (available in the forms short link).

> “Olá! Muito prazer, sou pesquisador em Eng de Software pela USP. Estou realizando uma pesquisa sobre Design Thinking e Requisitos não-funcionais. Envio este convite p/ participar. A pesquisa é anônima e toma 10 min. Se possível, segue o link: https://forms.gle/5v7B7gz62iSRPiyt9.
>
> Agradeço imensamente!”

Translation:

> “Hello! Nice to meet you, I am a researcher in Software Engineering by the USP. I’m performing a survey on Design Thinking and Non-Functional Requirements. I’m sending you this invitation to participate. The survey is anonymous and takes 10 min. If interested, here’s the link: https://forms.gle/5v7B7gz62iSRPiyt9.
>
> Thank you!”

Table 14 presents the list of all suggested techniques by survey participants, along with the number of times each technique was cited.

Table 14 – Suggested techiques by survey participants.

Suggested Techniques	**Number of times cited**
User Journey	21
Storyboards	13
Empathy Map	10
Feedbacks	8
Persona Creation	5
CSD Matrix	4
Prototyping	3
Crazy Eights	3
Value Canvas / Lean Canvas / Business Model Canvas	3
GUT	2
Userflow	2
A/B Tests	2
Value-Effort Matrix	2
Others*	1 each

*Others: Brainstorming, Roleplaying, Shadow, Affinity Diagram, Conceptual Map, Cocreation with clients, software architecture flow applied to the business model, developer journey, action plan, kanban, lean inception, 5 whys, mind map, life line, double diamond, moodboard, event storming, mvp matrix, Moscow, usability testing, usability x desirability map, sw2h, service blueprint, design sprint, as-is x to-be scenarios, cynefin, minimum viable experience, card sorting, worst possible idea, wireframing, fishbowl.

Source: author.

APPENDIX C - INTERVIEW GUIDE

This interview guide was based on the work of Alegroth et al. (2022), and follows a similar structure.

Before interview: Suitable candidates were identified via the LinkedIn network by availability and with profiles with experience on Design Thinking and software projects. They were contacted regarding their interest and availability to be interviewed. The topic of the interview was presented and asked if they thought they would be a suitable and willing candidate to answer. The interview was then scheduled.

Explanation of study's purpose: The purpose of the study is to investigate the relationship between Design Thinking and Non-Functional requirements. This topic was explored and a few of the previous results from this work were shared with each interviewee.

Statement of anonymity and approval to record: Interviewees were informed that all answers would be anonymized and no answer could be traced back to the interviewee. All data transcripts bore no names neither references to any person or affiliated companies. Interviewees were asked on their approval to record the interview. The recordings were confidential and not shared, nor any information provided to any colleagues or managers.

Statement of possibility to review: Interviewees were also told that they would be given the possibility to review all materials before they were analyzed. That is, transcripts of interviews were shared with interviewees for the opportunity to correct or complement items before the data was further analyzed.

Clarifications of study scope: This was a small summary presented to interviewees regarding the study scope. We understand Design Thinking as a human-centered approach that can be used to tackle complex and wicked real-life problems with a set of principles focused on empathy with users, fast prototyping, tolerance for failure and iterative learning cycles. We specifically turn to its use in software development, during the requirements elicitation.

We understand requirements as conditions or capabilities needed by users to solve a problem or achieve an objective. They are commonly distinguished as being of a functional or non-functional nature, which represents the difference between what a system shall do (functional), as opposed to how it should do it (non-functional). Non-Functional requirements are an attribute of or a constraint on a system. Usually they are divided in classes, according to the ISO/IEC 25010 (2011): Usability, performance, compatibility, reliability, security, maintainability.

Requested data: This was a small summary presented to interviewees regarding the requested data. We primarily seek empirical experiences and knowledge from actual use of DT for eliciting requirements in a software project. Do you have a project to report on that matches these criteria and that you have personally participated on? The following questions will be based on it.

Questionnaire: Table 15 presents the leading questions asked to interviewees, along with the associated research question.

Table 15 – Questionnaire of case study.

Background & Previous Project (Profiling Questions)
1. What is your main role today?
2. How many years of industrial experience do you have in the software development industry?
3. How many years of industrial experience do you have with eliciting requirements?
4. How many years of industrial experience do you have of DT?
5. Can you describe what you have/had been working with in this chosen project?
6. As an user of DT, what was/is your role?
7. How was DT used in this project? (Filter question)
DT x NFR
8. Were there problems during DT regarding NFR? (RQ1)
9. Regarding NFR, how were they discovered during DT? (They came up naturally, they came through a specific technique…) (RQ1)
10. What techniques were used that brought the best results in discovering new NFR? (RQ2)
11. Can you recall any restrictions to functionalities in the prototype that arose from the presence of these NFR? (RQ1)
Prototype x Software Development
12. Do you recall new restrictions that arose after DT in the form of NFR? How did they translate into the product? (RQ3)
13. In your opinion, how could these NFR be discovered earlier, during DT? (RQ3)
Other
14. What could have been done differently in the project regarding NFR? (RQ2)
15. Do you have any additional comments about DT or NFR that you think we have not covered in the interview?

Source: author.

APPENDIX D - ANSWERS TO THE CHECKLIST FOR READING AND REVIEWING CASE STUDIES

This section details the questions and answers to the checklist for reviewing case studies, as described by Runeson et al. (2012).

1. What is the case and its units of analysis?: The case comprises software developers with experience with using DT for eliciting requirements. The units of analysis are individual interviews.
2. Are clear objectives, preliminary research questions, hypotheses (if any) defined in advance?: Yes, that DT has issues with NFR, and that can hinder software projects.
3. Is the theoretical basis—relation to existing literature or other cases—defined?: Yes, as presented in the SLR available on Chapter 4.
4. Are the authors' intentions with the research made clear?: Yes, as described in the Objective on Section 6.3.
5. Is the case adequately defined (size, domain, process, subjects?)?: Yes, as described in Section 6.1.
6. Is a cause–effect relation under study? If yes, is it possible to distinguish the cause from other factors using the proposed design?: No, the study is of an exploratory nature.
7. Does the design involve data from multiple sources (data triangulation), using multiple methods (method triangulation)?: Not exactly, it involves multiple interviews. It does not involve other methods, since it starts from a solid basis of literature research and a previous survey presented on Chapters 4 and 5.
8. Is there a rationale behind the selection of subjects, roles, artifacts, viewpoints, and so on?: N/A.
9. Is the specified case relevant to validly address the research questions (construct validity)?: Yes, the case study was suggested by peers, and was considered the best approach for addressing the research objective, as explained on Section 6.1.
10. Is the integrity of individuals/organizations taken into account?: No.

11. Is a case study protocol for data collection and analysis derived (what, why, how, when)? Are procedures for its update defined?: Yes, the protocol was defined and its design is presented on Section 6.1.
12. Are multiple data sources and collection methods planned (triangulation)?: No, albeit the case study starts from previous conducted studies.
13. Are measurement instruments and procedures well defined (measurement definitions, interview questions)?: Questions are well defined and adhere to the objective, but measurements were not put in place, as the study is of an exploratory nature (qualitative).
14. Are the planned methods and measurements sufficient to fulfill the objective of the study?: Yes.
15. Is the study design approved by a review board, and has informed consent obtained from individuals and organizations?: Yes, it was approved by "Plataforma Brasil" under process number CAAE:65388922.4.0000.0138.
16. Is data collected according to the case study protocol?: Yes, as described on Section 6.5.
17. Is the observed phenomenon correctly implemented (e.g., to what extent is a design method under study actually used)?: To be defined in each individual interview via a filter question. Should the question be inappropriate, the answers would be disregarded.
18. Is data recorded to enable further analysis?: Data from the interviews would be transcribed and reviewed by each interviewee. Transcriptions were made available on 10.6084/m9.figshare.22269073.
19. Are sensitive results identified (for individuals, the organization or the project)?: No sensitive results were identified.
20. Are the data collection procedures well traceable?: Yes, according to the published data, all answers derived from specified interviewee and question.
21. Does the collected data provide ability to address the research question?: Yes, the collected data allowed the research questions to be answered.