Design Thinking and Non-Functional Requirements Elicitation: A Survey

Fábio Avigo de Castro Pinto[1[0000-0002-6002-1714]], Anarosa Alves Franco Brandão[1[0000-0001-8992-4768]], Fábio Levy Siqueira[1[0000-0002-7550-2000]]

[1] Escola Politécnica da Universidade de São Paulo, Brasil.
{fabio.avigo, anarosa.brandao, levy.siqueira}@usp.br

Abstract. Design Thinking (DT) is a human-centered approach used in innovative problem-solving. In software development projects, it can be usually applied as a toolbox, mindset or process. Moreover, DT has seen a large rise in popularity over the past decade, especially in its upfront use for eliciting requirements. Despite its many benefits and successful use cases in this scenario, some of the shortcomings related to its uses in Requirements Engineering (RE) have started to show up. One of them lies in the difficulty with eliciting other non-functional requirements (NFR) – besides usability –, possibly jeopardizing the subsequent software development life-cycle if these requirements are not properly addressed. This article presents a qualitative survey conducted via LinkedIn with 53 software developers that have had previous experiences with the use of DT as an upfront process in software projects. The focus of the research has been on identifying the existence of problems, how they became manifest, and whether the software community has tools for mitigating them. The results suggest the neglect of DT for requirements other than those of a functional or usability nature in this situation. Since this use of DT is focused on improving the users' experience, and identifying their needs, we argue that such needs are not limited only to the functions of a software or its usability. In fact, we also advocate that other aspects that are intrinsic to NFR, such as performance and reliability are part of such needs.

Keywords: Design Thinking, DT, Non-functional Requirements, NFR.

1 Introduction

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 have problems fitting the new products and service development requirements. In order to support them, companies should seek other possibilities using new approaches for creating value [1].

An approach that can help with this scenario is Design Thinking (DT), developed in Stanford, refined by IDEO and supported by Hasso Plattner, one of the founders of SAP [2]. It encompasses a team-based process for designing the solution of a given problem

[3]. Within DT, participants usually empathize with the final user, redefine the problem, match different perspectives, brainstorm solutions, and develop prototypes. Design Thinking has experienced a surge in popularity over the last decade, given its practical framework that leads to clear benefits to companies [2].

Moreover, DT represents a potential solution for some traditional challenges in Requirements Engineering, for eliciting user needs [4], being regarded by some authors, such as in [5], as a "modern form of RE". However, recent works present that there are still challenges to be overcome with the approach when applied for RE. A previous article by the authors with the scope of a systematic literature review with the objective of identifying problems that arise from DT in RE is a starting point for this study [6]. Among the several difficulties found by the authors, one must be highlighted: DT can neglect non-functional requirements, such as security or performance [7].

This paper seeks to deepen the understanding of DT applied for RE, focusing on its use as an upfront process in software projects. Our goal is to investigate how it handles the elicitation of non-functional requirements. To answer this, a qualitative survey was conducted with 53 software developers that have had previous experiences with DT in software projects. This work contributes to better understanding the limitations of DT in such a context.

This work is organized as follows: Section 2 presents a brief introduction to Design Thinking, while Section 3 presents an overview about Non-Functional Requirements. Following up, Section 4 depicts related work in the field and Section 5 presents the design of the survey. Section 6 presents the gathered results and Section 7 discusses them and presents the threats to validity. Finally, Section 8 concludes the article.

2 Design Thinking

Design Thinking is 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 [7–9].

A common approach to DT is the "Double Diamond", a process framework by Design Council [10]. A graphical representation for the process is presented on **Fig. 1**. The Double Diamond approach starts with a problem to be solved and has 4 phases, a divergent focused on empathizing with the user (Discover), followed by a convergent phase focused on defining the issue (Define), followed by another divergent phase focused on finding a solution (Develop) and 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 satisfactory result is achieved.

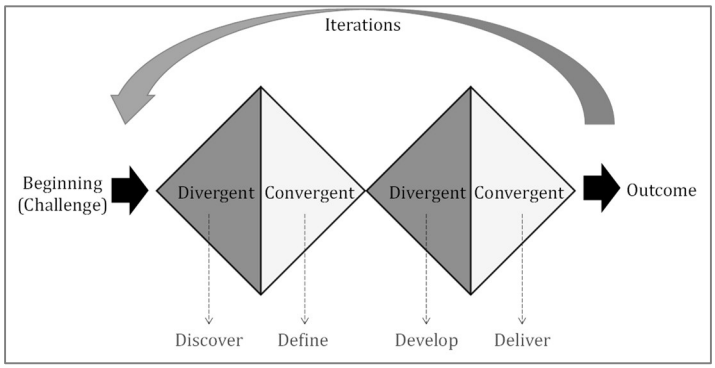

Fig. 1. The Double Diamond process. Adapted from [11].

Some authors, such as Vetterli et al. present that RE can make use of DT for eliciting customers' needs through the creation of fast and simple prototypes that converge into innovative solutions [4]. The resulting prototypes of DT help with the substantiation of different ideas, while focusing on specific and important needs within the design space. Brenner, Uebernickel and Abrell [2] suggested that DT can be applied as a mindset, toolbox, or process. A second definition by Hehn et al. deepened this understanding by defining three categories of possible applications: upfront, infused or continuous DT [11].

In this paper, we refer especially to the upfront approach, in which DT is conducted in the form of one or more workshops as a process previous to the activities pertaining to software development. In it, DT is used as a guiding process, and the proposed solution concept is linked to the identified customer needs. The outcomes comprise a clear solution vision in form of a mockup, and the DT artifacts are used as basis to perform further RE activities [11]. Some examples of this use can be found in [11–14].

3 Non-functional Requirements

Requirements are conditions or capabilities needed by users to solve a problem or achieve an objective [15]. 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).

While functional requirements are the ones that determine the actions that must be accomplished by a system [16], non-functional requirements lack a consensus in the RE community as for how they should be elicited, documented and validated [17]. Adding up, Chung and Leite 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 [18]. A summarization of the common understanding of NFR is an attribute of or a constraint on a system [17].

The categorization of NFR 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 [19].

The constraints that NFR bring are a key topic for this work. Landes and Studer have suggested that NFR establish the justifications for design decisions and thus restrict the way for the realization of the required functionality [20]. One example of this is defining when using or not external code libraries: this decision must be made at some point in time, and it can restrain services both ways. If used, the developing team will have to account for issues in versioning, compatibility, security, and performance. If unused, it can impact development times, code complexity and maintainability difficulty.

Another example lies in properly defining the user of the product, what is related to usability. Different classes of users can be familiar with different user interfaces. Some common examples are age differences, or systems built for a specific business context. This is a problem typically addressed with DT.

4 Related Work

This section presents some of the latest related works to this survey. There are plenty of studies to be found on the application of DT in use reports or case studies for software development, such as in [21–23]. However, it is observable that the term "Design Thinking" is still a buzzword. As Brenner et al. [2] put it, many of the projects labelled as DT are simply classical innovation, strategy or re-engineering, and do not represent the mastering or fully understanding of its principles. In this way, it is reasonable to presume that many companies and developers are still scratching DT's potential as an innovative tool and expanding its possibilities of use.

Presets et al. have reported in a related survey of DT on its use by the Brazilian Agile Software Development Community and cataloged 127 responses [24]. It is presented in the study the opinions of the respondents regarding the main benefits of using DT, such as keeping the user at the center of the process without neglecting business needs. Some of the difficulties for applying DT were also reported by participants. This survey was also conducted via the LinkedIn social network and followed a similar sampling approach with the use of a query to filter relevant profiles.

Schmiedgen et al. have also conducted a survey with industry practitioners focused on how organizations measure the impact of DT [25]. The authors combined quantitative and qualitative data with the use of a questionnaire distributed to managers and employees in organizations who applied DT. After it, 16 qualitative interviews were conducted to deepen the research. The authors present that many different practices are labeled DT, which creates an obstacle for its proper analysis. Additionally, the impact of DT is difficult to quantify, and very few people measure it.

Apart from recent surveys, some recent studies have also shown issues concerning NFR. For example, on comparing Design Thinking and Agile practices, Hehn and Uebernickel have commented that DT neglects NFR such as security or performance, albeit it enhances the priority of eliciting usability requirements [7].

5 Study Design

This section goes into detail regarding the survey that was conducted to investigate the elicitation of NFR within Design Thinking. We defined the survey protocol following the structure proposed by [26, 27]. All survey data, including the original questionnaire, complete answers and the analyses provided on the following sections is publicly available at https://doi.org/10.6084/m9.figshare.19354661.

5.1 Objective and Research Questions

The main objective of this study is to analyze whether DT faces challenges with the elicitation of NFR in software development projects. We considered the upfront use of DT, as described in Section 2. As these missing requirements may arise only after the workshops, the survey has investigated the perception of participants during the DT workshops as well as during the subsequent software development phases.

5.2 Research Questions

According to the defined objective of the study, 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?

5.3 Population Sampling

This survey was aimed at software developers that have had experiences 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 DT, rather than seeking confirmation to a certain hypothesis.

Qualitative research is used for the investigation of situations in which people are involved and different kinds of processes take place [28]. It aims to understand the reason and mechanisms explaining a phenomenon [29]. 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 [30].

A concept usually adopted in qualitative research 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 [31]. Although helpful in the conceptual level, it does not provide guidance for estimating actual sample sizes, prior to data collection [31]. As Boddy [32] 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 also comprised a list of sample sizes suggested by researchers, which vary in a range of 15 to 50 interviews [32]. Sandelowski suggests that a large sample size for qualitative studies is over 50 interviews [30]. Creswell and Poth 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 would be recommended [33].

Summing up, the sample size chosen for this survey was 50 interviews. 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.

5.4 Description and design of sampling method

The potential respondents for the survey were reached via the LinkedIn network, and 50 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 comprised 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 performing the searches. As the researcher who used LinkedIn for this purpose is from Brazil, most of the results were from this country. Therefore, the questionnaire has been applied in Portuguese to assure that the interviewees understood the meaning of each survey question precisely. The answers were then translated back to English to be presented here, in **Table 1**. The exact sequence of questions is the one presented there. This is not novel, and has been previously done in qualitative surveys, as in Wagner et al. [34].

The next step was then contacting industry professionals that matched the criteria. A message was provided with the context of the study and a friendly invitation to participate. Invitations have been daily sent until the desired amount of 50 answers were met.

The sampling was conducted over a period of two months (feb/21-apr/21). A total of 669 invites were sent, and a total of 53 responses were received and further processed. This adds up to a response rate of roughly 8%. The answers were recorded in Google Forms and then sent to a spreadsheet for analysis.

5.5 Questionnaire

As mentioned previously, a Design Thinking Workshop is how researchers and practitioners refer to the DT sessions/meetings.

Software projects that make use of DT can vary, as in the number of sessions, scope, number of participants and other techniques applied apart from DT. However, the chosen hypothesis was that each project made use of a DT Workshop for requirements elicitation (not necessarily as the single technique), and it was followed by some stage of software development, where the results of it could then be used to produce code. This simplified model was adopted to facilitate understanding the potential shortcomings of DT, for they can show up either during the workshop, or after it, during software development. And, in this manner, the software developer must also have participated in both phases of this project that was chosen as basis of report.

Table 1. Questionnaire

Question
1. City of residence / State of residence
2. What company do you currently work for?
3. What is your current organizational role or function?
4. What is your experience, in years, with software development?
5. What is your experience, in years, using Design Thinking?
6. How many projects have you participated in that used Design Thinking?
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?
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?
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? [19]
10. What support techniques were used for eliciting non-functional requirements during the Design Thinking Workshop (except usability)?
11. What difficulties or negative points regarding the elicitation of non-functional requirements (except usability) did you identify during the DT workshop?
12. In your opinion, what percentage of completion was reached regarding the implementation of the system/software component of the project thus far?
13. In your opinion, what percentage of completion was reached regarding the deployment of the system/software component of the project thus far?
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?
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?
16. If any non-functional requirements arose during the software development phase that were unforeseen in the DT Workshop, how did they become known?

The first part of the questionnaire (Q 1-6) presents profiling questions aimed at extracting data from different technological contexts, identifying validation threat biases from too many answers from the same company and categorizing answers. The second part (Q 7-11) is concerned with the experience during the DT Workshop, with questions that aimed at identifying if DT lacks in NFR elicitation, and what techniques were used that could help with this issue. The last part (Q 12-16) refers to the experience after the DT Workshop and during the development phase. It focused on identifying the potential

problems with NFR not properly treated during the workshop and that came up in a later stage of the project. No questions were mandatory, and since, aggregate results shown in the following section disregard blank answers.

6 Study Results

In this section, the survey results are presented.

6.1 Profile of Respondents (Q1 to Q6)

A total of 53 responses were collected. Regarding their current role, the vast majority consisted of software developers (57%), project owners (19%) and UX/UI Designers (9%). A total of 45 companies were listed as the current work from respondents. Most of the respondents (73%) were from 3 fields: IT Services & Consulting, Banking & Financial, and Health, Wellness, Fitness & Hospital.

Considering the respondents' previous experience, **Table 2** presents the average, standard deviation, minimum, and maximum values of answers received in questions 4, 5, and 6. The average respondent has roughly 9 years of experience with software development and has practiced Design Thinking for 3 years.

Table 2. Respondents' experience with Software Development and Design Thinking

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

6.2 Design Thinking Workshop (Q7 to Q10)

Questions 7 to 10 focused on gathering the participants' thoughts on the workshop in eliciting NFR. **Table 3** 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). **Fig. 2** presents a visual representation of these responses: the scatter plot presents each individual answer (grouped for NFR, excluding usability), while vertical bars present the mean values obtained.

Table 3. Respondent's 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

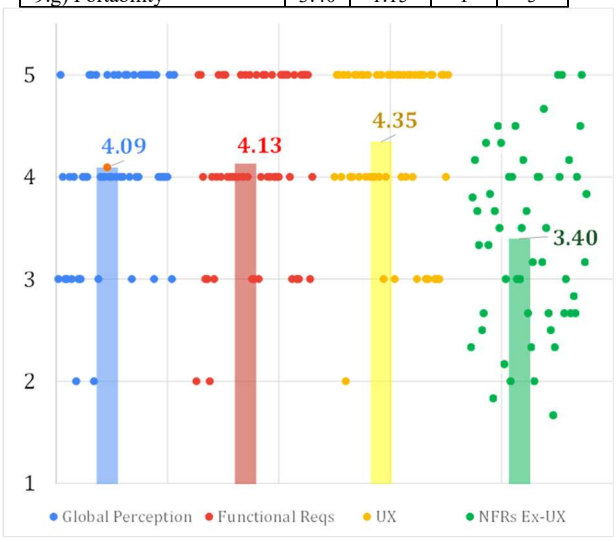

Fig. 2. Scatter plot and average values for the successfulness perception.

The average successfulness perception with eliciting requirements for Design Thinking has been 4.09. This value is close to the one of 4.13 that was found for the elicitation of functional requirements. The perception increased for usability requirements (4.35) but saw a sharp decline for all other NFR (averaged excluding usability: 3.40).

Most respondents have considered the successfulness perception of both the elicitation of functional requirements (81% of respondents), as well as the elicitation of usability requirements (87% of respondents) to be greater or equal to the averaged perception regarding the elicitation of the other NFR (excluding usability).

As per Question 10 about techniques used in the Workshop, a total of 44 different ones 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%).

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 into 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."
> "Scalability, capacity, safety and privacy were not discussed."
> "After DT, it is necessary that a dedicated technical team brings up the technical requirements and a list of technical non-negotiables concerning security, maintainability, versioning and portability."

6.3 Software Development post-Workshop (Q12 to Q16)

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.

When we turn to requirements not foreseen in the DT Workshop that became known during software development (questions 15 and 16), the most cited categories were of performance (49% of respondents), maintainability (49%) and security (47%). Despite DT's strength on usability requirements, 32% of respondents also adhered to the surfacing of new requirements. Most respondents claimed the rise from new NFR from testing (74% of respondents), developers' input (66%) and users' input (52%).

7 Discussion of Results

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

7.1 RQ1 – NFR Problems with DT

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 NFR during the DT Workshop. It is also remarkable the respondents' consensus on a lower successful perception of all NFR apart from usability in comparison to it, to functional requirements and to Design Thinking altogether. Several respondents have also commented on this issue directly.

A counterpoint to this was also present: two of the 53 respondents have given feedback in reminding that DT's focus is not on NFR, but on innovation. One can ponder, however, how relevant is the DT result, should it prove unfeasible due to unidentified NFR. The innovation effect 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 the DT results were based on, which proved to be unjustifiable when further analyzed. Another participant also pointed out that the ideas suggested during the workshop were too superficial without the correct guidance, and thus rendered useless. This echoes a case study by Mahe et al. [13], in which 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 the workshop are left aside.

7.2 RQ2 – NFR discovered after the DT Workshop

The second research question aimed at identifying whether NFR not identified during the DT workshop would become known only during the software development phase. It was addressed with questions 15 and 16. They asked participants to point out if new NFR have been elicited after the DT workshop (in categories), and how they became

known. Only 2 responses to question 16 were discarded (blanks), as it was not mandatory, and no answers were left blank for question 15 (also not mandatory). This brings some evidence to the fact that not all NFR become known during the DT Workshop, what would be expected as DT is not a requirements elicitation technique.

Despite DT's focus on usability, 32% of respondents claimed that new usability requirements also arose during software development. Also, the most problematic categories were performance, maintainability, and security. It is not possible to say whether these new NFR are expected to arise, or if it is a potential drawback of DT.

It is reasonable to assume 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 accomplish 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.

7.3 RQ3 – Tools for helping NFR with DT

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 the workshop as the chosen tools of the session, and not directly as means of improving NFR elicitation. Some of the most interesting that were cited, and it is left to be analyzed whether they could be used to help elicit NFR are: Developer Journey, Service Blueprint, and Event Storming.

7.4 Threats to validity

A first threat to validity is the one regarding additional RE techniques to be considered. Each project has its own decisions regarding approaches to best elicit requirements, and Design Thinking is not necessarily applied as a standalone approach, but rather as a complementary one aimed at innovation. In this way, from a purely innovative perspective, it cannot be said 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 into better understanding how to enhance it.

Another threat to validity regarding RQ2 is that some of the requirements to appear after a DT workshop are expected 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 to the upfront approach: the use of workshops followed by software development. As previously discussed, DT can also be used as a mindset or a toolbox, and also infused into other RE processes [2, 11]. While the workshop approach is less resource-intensive, it is to be expected that other manifestations of DT, such as the use of a higher number of iterations can help mitigate the aforementioned problems.

It is also important to notice that, albeit this work was elaborated with the help of several practitioners that provided valuable insights, no interviews were conducted.

8 Conclusion

This paper presented a survey conducted with software developers to analyze whether the Design Thinking (DT) approach 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 regarding NFR in software development projects especially if Design Thinking is deemed as a standalone approach 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 in the DT Workshop as a user need, and that is a complication. It is not probable, for example, that maintainability requirements will show up during a DT Workshop, 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 the DT sessions 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 could 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 during the software development stages, and so it becomes appropriate to make provisions and adjust estimates.

References

1. Volkova, T., Jākobsone, I.: Design Thinking as a Business Tool to Ensure Continuous Value Generation. Intellectual Economics. (2016). https://doi.org/10.1016/j.intele.2016.06.003.
2. Brenner, W., Uebernickel, F., Abrell, T.: Design Thinking as Mindset, Process, and Toolbox. In: Design Thinking for Innovation: Research and Practice. pp. 3–21. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-26100-3_1.
3. Plattner, H., Meinel, C., Leifer, L. eds: Design Thinking: Understand – Improve – Apply. Springer-Verlag, Berlin Heidelberg (2011). https://doi.org/10.1007/978-3-642-13757-0.
4. Vetterli, C., Brenner, W., Uebernickel, F., Petrie, C.: From Palaces to Yurts - Why Requirements Engineering Needs Design Thinking. IEEE Internet Computing. 17, (2013). https://doi.org/10.1109/MIC.2013.32.
5. Beyhl, T., Giese, H.: Connecting Designing and Engineering Activities III. In: Plattner, H., Meinel, C., and Leifer, L. (eds.) Design Thinking Research: Making Design Thinking Foundational. pp. 265–290. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-19641-1_16.

6. Pinto, F., Siqueira, F.L.: Problemas do Design Thinking para a Engenharia de Requisitos: uma Revisão Sistemática da Literatura. WER 2020. 14 (2020).
7. Hehn, J., Uebernickel, F.: 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). pp. 400–405 (2018). https://doi.org/10.1109/RE.2018.00-18.
8. Brown, T.: Design Thinking. Harvard Business Review. 86, 84–92 (2008).
9. Kolko, J.: Design Thinking Comes of Age. Harvard Business Review. Organizational Behavior, 7 (2015).
10. Design Council: What is the framework for innovation? Design Council's evolved Double Diamond, https://www.designcouncil.org.uk/news-opinion/what-framework-innovation-design-councils-evolved-double-diamond.
11. Hehn, J., Mendez, D., Uebernickel, F., Brenner, W., Broy, M.: On Integrating Design Thinking for a Human-centered Requirements Engineering, http://arxiv.org/abs/1908.07223, (2019). https://doi.org/10.48550/arXiv.1908.07223.
12. Dobrigkeit, F., de Paula, D.: 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). pp. 1059–1069. , New York, NY, US (2019). https://doi.org/10.1145/3338906.3340451.
13. Mahe, N., Adams, B., Marsan, J., Templier, M., Bissonnette, S.: Migrating a Software Factory to Design Thinking: Paying Attention to People and Mind-Sets. IEEE Software. 37, 32–40 (2020). https://doi.org/10.1109/MS.2019.2958646.
14. Rozante de Paula, T., Santana Amancio, T., Nonato Flores, J.A.: Design Thinking in Industry. IEEE Softw. 37, 49–51 (2020). https://doi.org/10.1109/MS.2019.2959473.
15. IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990. 1–84 (1990). https://doi.org/10.1109/IEEESTD.1990.101064.
16. IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998. 1–40 (1998). https://doi.org/10.1109/IEEESTD.1998.88286.
17. Glinz, M.: On Non-Functional Requirements. In: 15th IEEE International Requirements Engineering Conference (RE 2007). pp. 21–26. IEEE, Delhi (2007). https://doi.org/10.1109/RE.2007.45.
18. Chung, L., do Prado Leite, J.C.S.: On Non-Functional Requirements in Software Engineering. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., and Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos. pp. 363–379. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02463-4_19.
19. ISO/IEC JTC 1/SC 7 Software and systems engineering: ISO/IEC 25010:2011, https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/03/57/35733.html, last accessed 2020/09/30.
20. Landes, D., Studer, R.: The treatment of non-functional requirements in MIKE. In: Schäfer, W. and Botella, P. (eds.) Software Engineering — ESEC '95. pp. 294–306. Springer, Berlin, Heidelberg (1995). https://doi.org/10.1007/3-540-60406-5_21.
21. Carell, A., Lauenroth, K., Platz, D.: Using Design Thinking for Requirements Engineering in the Context of Digitalization and Digital Transformation: A Motivation and an Experience Report. In: Gruhn, V. and Striemer, R. (eds.) The Essence of Software Engineering.

pp. 107–120. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-73897-0_7.
22. Carroll, N., Richardson, I.: Aligning healthcare innovation and software requirements through design thinking. In: Proceedings of the International Workshop on Software Engineering in Healthcare Systems. pp. 1–7. Association for Computing Machinery, Austin, Texas (2016). https://doi.org/10.1145/2897683.2897687.
23. Hildenbrand, T., Meyer, J.: Intertwining Lean and Design Thinking: Software Product Development from Empathy to Shipment. In: Maedche, A., Botzenhardt, A., and Neer, L. (eds.) Software for People: Fundamentals, Trends and Best Practices. pp. 217–237. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31371-4_13.
24. Prestes, M., Parizi, R., Marczak, S., Conte, T.: On the Use of Design Thinking: A Survey of the Brazilian Agile Software Development Community. In: Stray, V., Hoda, R., Paasivaara, M., and Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming. pp. 73–86. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-49392-9_5.
25. Schmiedgen, J., Spille, L., Köppen, E., Rhinow, H., Meinel, C.: Measuring the Impact of Design Thinking. In: Plattner, H., Meinel, C., and Leifer, L. (eds.) Design Thinking Research: Making Design Thinking Foundational. pp. 157–170. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-19641-1_11.
26. Linåker, J., Sulaman, S.M., Maiani de Mello, R., Höst, M.: Guidelines for Conducting Surveys in Software Engineering. (2015).
27. Kitchenham, B.A., Pfleeger, S.L.: Principles of survey research part 2: designing a survey. SIGSOFT Softw. Eng. Notes. 27, 18–20 (2002). https://doi.org/10.1145/566493.566495.
28. Dybå, T., Prikladnicki, R., Rönkkö, K., Seaman, C., Sillito, J.: Qualitative research in software engineering. Empirical Software Engineering. 16, 425–429 (2011). https://doi.org/10.1007/s10664-011-9163-y.
29. Felderer, M., Travassos, G.H.: The Evolution of Empirical Methods in Software Engineering. In: Felderer, M. and Travassos, G.H. (eds.) Contemporary Empirical Methods in Software Engineering. pp. 1–24. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-32489-6_1.
30. Sandelowski, M.: Sample size in qualitative research. Research in Nursing & Health. 18, 179–183 (1995). https://doi.org/10.1002/nur.4770180211.
31. Guest, G., Bunce, A., Johnson, L.: How Many Interviews Are Enough? Field Methods - FIELD METHOD. 18, 59–82 (2006). https://doi.org/10.1177/1525822X05279903.
32. Boddy, C.R.: Sample size for qualitative research. Qualitative Market Research: An International Journal. 19, 426–432 (2016). https://doi.org/10.1108/QMR-06-2016-0053.
33. Creswell, J.W., Poth, C.N.: Qualitative Inquiry and Research Design: Choosing Among Five Approaches. SAGE Publications (2016).
34. 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. Springer, Cham (2020).