A Design Thinking Techniques Recommendation Tool: An Initial and On-going Proposal

Rafael Parizi
School of Technology – PUCRS
Porto Alegre, Brazil
rafael.parizi@edu.pucrs.br

Marina Moreira
School of Technology – PUCRS
Porto Alegre, Brazil
marina.moreira@acad.pucrs.br

Igor Couto
School of Technology – PUCRS
Porto Alegre, Brazil
igor.couto@edu.pucrs.br

Sabrina Marczak
School of Technology – PUCRS
Porto Alegre, Brazil
sabrina.marczak@pucrs.br

Tayana Conte
Federal Univ. of Amazonas - UFAM
Manaus, Brazil
tayana@icomp.ufam.edu.br

ABSTRACT

Putting the user at the center of the process has been considered key to achieve quality in modern software development. Discovering and framing the user needs are necessary activities to produce better quality software that fits the customer's needs and their satisfaction. User-centered Design approaches such as Design Thinking (DT) help to establish a creative mindset through convergent and divergent working spaces that allow for the team to better understand, to create empathy, to brainstorm ideas, to prototype, and to test innovative solutions in software development. However, navigating in this world is still challenging. Literature reports on the use of a large spam of techniques to support DT. Selecting an appropriate set of techniques considering the project context, the stakeholders' profiles, and other characteristics may be challenging, mainly to novice professionals. Thus, as part of an on-going long-term research project, we propose Helius, a collaborative recommender system for DT techniques in software development. Helius provides features supporting recommendations of DT Techniques based on the user previous experiences, techniques' evaluation ratings, and community feedback. This paper focuses on our preliminary results upon the tool proposal–on a comparison of Helius features with other related tools using the DESMET method. This comparison shows that Helius brings new features for the recommendation of DT techniques, thus, contributing to process quality improvement.

CCS CONCEPTS

• **Software and its engineering → Software development techniques**.

KEYWORDS

Recommendation System, Design Thinking Techniques, Techniques Evaluation, Tool Design, Quality Improvement

SBQS 2020, December 1–4, 2020, Virtual Conference

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Rafael Parizi, Marina Moreira, Igor Couto, Sabrina Marczak, and Tayana Conte. 2020. A Design Thinking Techniques Recommendation Tool: An Initial and On-going Proposal. In *Proceedings of SBQS 2020: Brazilian Symposium on Software Quality (SBQS 2020).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Software quality involves considering the characteristics of the product, the process, and the business context. Attributes such as usability, effectiveness, efficiency, satisfaction, and user experience (UX) are considered as fundamental for software quality [3].

Empathizing with users, understanding their needs, gathering different points of view, ideating to generate potential solutions, prototyping them in a suitable one, and testing are contemporaneous demands for software development [5]. Tanian [19] argues that development teams need to consider qualities such as reliability, fitness-for-use, and fitness-for-purpose to develop a successful software. Moving further on towards quality requires putting the user at the center of the process characterizing a User-Centered Design (UCD) approach [9]. Software development organizations are using UCD approaches in conjunction with Agile Methods, exploring quick feedback, frequent deliveries, and fostering an innovative and creative environment [2, 11].

Design Thinking (DT) has been adopted as a UCD alternative for software development [6]. Feasible for integration with software development activities [4], DT explores the problem in an iterative and multidisciplinary way, and it also fosters users' engagement [1].

Given that de Paula, Amancio and Flores [16] advocate that the key to a successful use of DT in software development is to have short DT sessions supported by the appropriated use of DT techniques aligned with customer's needs, our study takes the lens of Design Thinking as a set of techniques. We aim to contribute to advancing the state-of-the-art and the state-of-practice about the use of DT techniques in software development. Recent efforts [4, 17] have focused in investigating the use of DT techniques in software development, to preparing professionals, or focusing on helping industry professionals. However, there is still room for recommending techniques using collaborative and intelligent mechanisms as our tool proposes.

Our preliminary investigation of literature [14] and of industry practice [15] has suggested that software development professionals still use a limited number of techniques either for not knowing about other options or for not realizing how to select new ones. Our preliminary validation of the idea of proposing a recommendation system to suggest DT techniques based on product and project contextual information (e.g., techniques' characteristics, stakeholders' involvement, and project domain) and on feedback of use by a community was well received by senior and novice DT users [12].

Thus, as part of our long-term research goal, we posed the research question to guide the development of a tool to recommend DT techniques: *"How can a recommendation system promote the use of appropriate DT techniques in software development?"*.

We thus proposed Helius, a tool-based recommendation system for DT techniques in software development. Helius intends to recommend techniques based on project characteristics, techniques similarities, team organization, and users' experiences on DT. Its core feature set is composed of the following: recommend techniques, filter techniques, evaluate techniques, show feedback for techniques used in conjunction and show techniques' information.

This paper focuses on presenting our first step towards validating the tool: we compare Helius proposed features with those of two other related tools by using the DESMET [8] as an strategy to demonstrate that Helius' scope indeed brings innovation. The results of the DESMET method confirm that Helius advances in the recommendation of DT techniques for considering the combined use of techniques, for allowing the evaluation of the techniques used and for considering the user experience in the recommendations.

The reminder of this paper is organized as follows. Section 2 briefly introduces DT techniques in software development. Section 3 presents the Helius features, potential users and screen overview. Section 4 shows our preliminary results comparing Helius with similar tools using the DESMET method. Section 5 presents our research agenda towards the tool validation, improvement, and pilot use. Section 6 concludes the paper with final remarks.

2 DESIGN THINKING TECHNIQUES IN SOFTWARE DEVELOPMENT

Literature reports some efforts to characterize how DT techniques are used in software development. Souza, Ferreira, and Conte [17], and Plautz [14] performed systematic literature studies summarizing the use of Design Thinking techniques in the area. These studies identified that techniques are used (i) to understand the users' needs, (ii) to generate a large number of ideas looking for the solution, and (iii) to prototype and validate the selected solution.

De Paula, Amancio and Flores [16] argument the key to making DT sessions with no quality loss is choosing the right techniques. They also pointed out that these techniques allow involving the stakeholders and gather different points of view, considering DT goal is promote creativity and innovation.

Hehn et al. [4] discuss about techniques and steps of DT for requirements engineering, a software development discipline that faces with the challenge of discovering the users needs. The authors propose a combined artifact model of DT and requirements engineering, mixing DT techniques and requirements engineering artifacts. Also, they present three strategies for integrating DT in software development– Upfront DT, Infused DT and Continuous DT. For each strategy they present a set of techniques to be used.

In a previous study [15], we conducted a survey to characterize how Brazilian professionals are using DT in their projects. We asked questions concerning the challenges and difficulties of choosing DT techniques. Between September and December of 2019, 171 professionals participated in our study. They pointed out how challenging it is to select the techniques considering the project context, the stakeholders' profiles, among others. In the survey, 59 techniques were cited as alternatives of DT in software development.

3 OUR PROPOSAL

Helius' solution idea resulted from a DT session we performed ourselves and of an early interview-based user validation with professionals aiming to preliminary validate the idea of a recommendation system and to gather a first round of needed features to do so. The professionals indicated that Helius has potential to facilitate the selection of DT techniques and to improve the quality of the whole software development process. Details of these co-creation and preliminary idea validation steps are reported in [12].

Based on this positive initial tool proposal user-based validation, we prototype our solution into high-fidelity prototypes and extracted the features for Helius. As it is a recommendation and a collaborative system, we designed a set of features aiming to assist software development professionals on the selection of DT techniques according to their needs. Thus, Helius is a collaborative recommendation tool that takes project context and previous experiences to recommend DT techniques in software development.

This section presents the tool target audience, the features, and a subset of screen overview to show how Helius works.

3.1 Helius in a Nutshell

The tool's target audience are professionals who work with software development (e.g., developers, requirements analysts, UX designers). Mainly, those who are beginners on the use of Design Thinking in the context of software development but not limited to this profile.

Table 1 presents a set of features and respective sub-features for our tool proposed based on the DT session [12]. We organized Helius into 6 main features: (F1) DT Techniques Recommendations, (F2) DT Techniques Filtering, (F3) DT Techniques Evaluation, (F4) DT Techniques Community Feedback, (F5) DT Techniques Information, and (F6) DT Techniques and Project Management.

The DT Techniques Recommendation feature (F1) is Helius' core feature. This will help users to receive recommendations of the techniques most suite to their needs according to the project characteristics and contextual information (e.g. project domain, stakeholders' commitment level, team expertise in Design Thinking). The recommendations are performed by Helius using the DT techniques information and using information from the user, making the tool as a collaborative system too. Therefore, once the user evaluates the techniques selected to be used in a project, Helius uses this data for recalculating the recommendation graph.

The DT Techniques Filtering feature (F2) allows the users to filter techniques based on the characteristics of techniques and context information of the project. This feature works as a catalog of techniques in which the user can filter them using a set of criteria.

Table 1: Helius Features

ID	Feature set	Sub-feature ID	Sub-feature description	Subfeature: level of importance	Feature set Importance Weighting
F1	DT Techniques Recommendations	F1-SF01 F1-SF02 F1-SF03	To consider the combined use of techniques in projects to recommend DT techniques To consider contextual information from the user's project to recommend DT techniques To consider feedback from other users to recommend DT techniques	Mandatory Mandatory Mandatory	0.3
F2	DT Techniques Filtering	F2-SF01 F2-SF02	To consider techniques characteristics to filtering similar DT techniques To consider project context information to filtering similar DT techniques	Highly Desirable Highly Desirable	0.2
F3	DT Techniques Evaluation	F3-SF01 F3-SF02	To allow evaluation of a single technique by the user To allow evaluation of the techniques used in conjunction in a DT project by the user	Highly Desirable Highly Desirable	0.2
F4	DT Techniques Community Feedback	F4-SF01 F4-SF02	To allow filtering feedback by technique To show related feedback for projects and techniques used in conjunction	Desirable Desirable	0.1
F5	DT Techniques Information	F5-SF01 F5-SF02	To show detailed information about each DT technique To show the related uses of a technique with other techniques	Nice to have Desirable	0.1
F6	DT Techniques and Project Management	F6-SF01 F6-SF02	To manage a project which uses DT techniques To share project data with other team members	Nice to have Nice to have	0.1

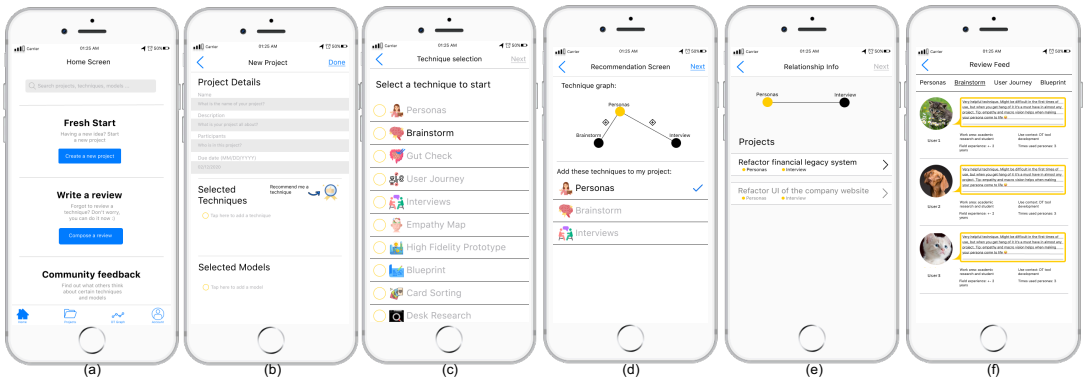

Figure 1: Screens overview: (a) Start; (b) Project; (c) Techniques; (d) Recommendation; (e) Previous usage; (f) Evaluation.

In this way, Helius, in addition to recommending the techniques (Feature F1), allows filtering DT techniques, helping the user to select the most appropriate techniques for her project. This feature supports professionals who already have experience in DT who, instead of requesting a technical recommendation, prefer to filter to learn about other techniques.

The DT Techniques Evaluation feature (F3) allows the users to make evaluations considering the techniques they have used in their projects. Therefore, once the user selects a technique for evaluation, Helius shows a form composed by a field for evaluation rate, and 6 questions: *"How was your experience using the selected technique"*, *"What would you do differently"*, *"How many people participated in the use of this technique?"*, *''What techniques did you combine together?"*, *"What was the project context?"*, and *"Would you recommend this technique to others?"*.

The DT Techniques Community Feedback feature (F4) combines the evaluations of the users given information about others users' experiences using a technique. These feedback information allow the users to know each technique based on previous experiences.

Helius also provides the DT Techniques Information feature (F5). This feature organizes a collection of data about each technique, such as the technique definition, how to use it, and suggestions about when to use it. In addition, Helius includes the community feedback to the technique information.

The DT Techniques and Project Management feature (F6) allows the users to create a project, including data such as project name, project description, the participants profiles, project due date, project tasks, and team members. No information that could compromise the project's confidentiality is requested or exposed. Therefore, once the users have created a project, they can request a recommendation. Also, it is by using this feature that Helius provides the feature for techniques evaluations (ii). Users can evaluate the techniques they have used in their own projects.

3.2 Screens Overview

Figure 1 illustrates a subset of screen prototypes of Helius. Screen (a) shows the start user interface, which provides functionalities such as "create a project" (F6-SF01), "write a review" (F3-SF01, F3-SF02), and view the "community feedback" about the techniques and projects (F4-SF01, F4-SF02). Screen (b) illustrates the creation of a project to insert DT techniques. It is also possible to insert DT models. Additionally, on Screen (b), there is the functionality:

"Recommend me a technique" used to request DT techniques recommendations (F1-SF01, F1-SF02, F1-SF03). Once the user has selected recommendation feature, Helius shows Screen (c).

On Screen (c), the user is able to select one technique. After that, Screen (d) is opened, showing a graph of DT techniques and its relationship, i.e., techniques recommended according to their usage in previous projects (F1-SF01, F1-SF02, F1-SF03). For example, on Screen (d), Personas was the selected technique, and the other two techniques, Brainstorm and Interviews, were present in the graph. It means that Personas x Brainstorm and Personas x Interview were already used in conjunction before within a project (F5-SF01, F5-SF02). In addition, on Screen (d), the users may select which techniques want to insert in their project (we show the selection of Personas to be added to a project, marked with a blue 'v'). Once the user has clicked in the square over a graph edge, which creates a relation with two techniques, Helius presents Screen (e). On Screen (e), there is information related to the relationship of two techniques, such as the projects that they already were used in conjunction with (F5-SF01, F5-SF02). For example, on Screen (e), we show the relationship between Personas and Interview and the project where these two techniques were applied in conjunction. Helius also builds a graph from filters applied by users on the characteristics of the techniques and of the project (F2-SF01, F2-SF02).

Finally, on Screen (f), we show the feedback registered by the Helius tool users about the DT techniques they had used (F4-SF01, F4-SF02). This feedback helps other users describing previous experiences with the use of DT techniques.

4 PRELIMINARY RESULTS

Before kicking-off Helius prototype implementation, we set as a goal to validate the tool scope. Our first step towards this validation–focus of this paper, aims to demonstrate that the tool feature set indeed brings new contributions with regards to those tools already available, i.e., it innovates. Next in our research agenda is a validate study with end-users, as presented in Section 5.

This section presents the results of our feature analysis study comparing other related proposals to our project. The related tools we compare to our project were identified through a Systematic Literature Review (SRL) [14] and through a survey performed with IT professionals [15]. Thus, aiming to compare Helius with other tools, we performed feature analysis using DESMET [8]. In this study we compared the other tools with a prototyped version of Helius, previously validated by industry professionals [12]. The Helius' features were intentionally defined as a baseline for the comparison, following the procedures proposed in [8].

DESMET is a method which provides an algorithm to compare tools when we intend to identify the best alternative for a specific domain [8]. For instance, Marshall [10] performed a study comparing tools for SLRs. Starting with the definition of the features, sub-features, and their respective degrees of importance, the authors conducted comparisons feature by feature, obtaining the most suitable tool for SRL.

Using the qualitative analysis provided by DESMET, we compare Helius (in its prototype-based version) in conjunction with two related tools: DTA4RE [18] and IDEO DT [7]. We start presenting an overview of similar tools. Next, we show the DESMET evaluation,

Table 2: Sub-features Importance levels

Importance level	Multiplier
Mandatory (M)	*4
Highly Desirable (HD)	*3
Desirable (D)	*2
Nice to have (N)	*1

Table 3: Feature Set Importance Weight (FIW)

ID	Feature Set	FIW
F1	DT Techniques Recommendations	0.3
F2	DT Techniques Filtering	0.2
F3	DT Techniques Evaluation	0.2
F4	DT Techniques Community Feedback	0.1
F5	DT Techniques Information	0.1
F6	DT Techniques and Project Management	0.1

extracting the percentage of agreement for the defined features expected in the recommender system of DT Techniques (Table 1).

DTA4RE is a tool that provides recommendations of DT techniques for software development. Through a form-based algorithm, the tool presents for the users a set of questions and, once they have answered, the tool exposes the applicable DT techniques. DTA4RE is a web system focused on DT techniques for requirements engineering, recommending techniques according to three working spaces of DT: inspiration, ideation, and prototyping.

Although providing information about the techniques, such as how to use a technique, when to use it, DTA4RE does not offer features related to DT techniques evaluations and also does not recommends techniques based on community feedback.

IDEO DT is a web-based tool that provides DT techniques suggestions according to filters that the user has applied. Once in the IDEO DT system, it is possible to select some need characteristics, such as what working space the user is in, and the scenario of use, among others. For each technique filtered, IDEO DT gives an overview, the steps to be done, the difficulty level for use it.

Similarly to DTA4RE, IDEO DT does not provide runtime evaluations of the recommended techniques nor consider the users' feedback to calculate the recommendations. Helius is a system that proposes to innovate using the users' collaborations to recommend the suitable DT Techniques, including the previous experiences from the users (community feedback-based) to support DT in software development projects.

4.1 DESMET Evaluation

DESMET is a evaluation method that requires the designation of scores for the features and sub-features, such as: sub-features importance levels, features weights, and judgment scales. Thus, following Marshall [10], each score was initially designated by the first author, and after they were analyzed by the fourth and fifth authors.

Using the DESMET method initially is needed to define the importance level for each sub-feature (Table 2). It designates a multiplier associated with each sub-feature representing its importance in the solution. Next, we evaluated the set of sub-features our proposal and attributed the corresponding importance level (Table 1 – column Sub-feature level of importance). For instance, we consider sub-feature F1-SF01 *"To consider the combined use of techniques in projects to recommend DT techniques"* as one of the most important

Table 4: Feature analysis results for three tools (Helius, DTA4RE, and IDEO DT)

				Helius				DTA4RE				IDEO DT			
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Feature Set	Subfeature	Subfeature importance level (M,HD,D,N)	Max Feature set Score	JI score Helius (JIS)	Weight Score	Feature set Score obtained	% Feature Set Score	JI score DTA4RE (JIs)	Weight Score	Feature set Score obtained	% Feature Set Score	JI score IDEO (JIS)	Weight Score	Feature set Score obtained	% Feature Set Score
F1	F1 - SF01	4	12	1	4	12	100%	1	4	8	66.67%	0	0	2	16.67%
	F1 - SF02	4		1	4			1	4			0.5	2		
	F1 - SF03	4		1	4			0	0			0	0		
F2	F2 - SF01	3	6	1	3	6	100%	1	3	6	100%	1	3	3	50.0%
	F2 - SF02	3		1	3			1	3			0	0		
F3	F3 - SF01	3	6	1	3	6	100%	0	0	0	0.0%	0	0	0	0.0%
	F3 - SF02	3		1	3			0	0			0	0		
F4	F4 - SF01	2	4	1	2	4	100%	0	0	0	0.0%	0	0	0	0.0%
	F4 - SF02	2		1	2			0	0			0	0		
F5	F5 - SF01	1	3	1	1	3	100%	1	1	1	33.3%	1	1	1	33.3%
	F5 - SF02	2		1	2			0	0			0	0		
F6	F6 - SF01	1	2	1	1	2	100%	0	0	0	0.0%	0	0	0	0.0%
	F6 - SF02	1		1	1			0	0			0	0		
				Overall score (OS)			100%	Overall score (OS)			43.33%	Overall score (OS)			20.83%

sub-feature of Helius. Therefore we defined it as a Mandatory (M) sub-feature. We marked it as M.

In the sequence, in DESMET are defined importance weights for each feature set. It means the importance of each set of features for a tool. For our study, we defined the importance weight, as shown in Table 3. For example, we defined F1 as the most important feature set, with an importance weight of 0.3, which represents 30% of importance in this context. All importance weights can be seen in Table 1 – column Feature set importance weighting).

Another definition that needs to be done using DESMET is the Judgement scale and its Interpretation (JI). JI means to consider if each sub-feature is in the tool evaluated. We defined JI following Marshall [10] definitions: *Yes* – score 1, if a sub-feature was fully identified in the tool; *Partly* – score 0.5, if a sub-feature is not fully present or was implemented differently, and; *No* – score 0, when the sub-feature can not be found in a tool.

Table 4 – columns E, I, and M show the JIs we attributed for each sub-feature of Helius, DTA4RE, and IDEO DT tools, respectively. For example, we fully identify the sub-feature F1-SF01 in Helius (score 1 on column E); F1-SF01 was also fully identified in DTA4RE (score 1 on column I); but F1-SF01 was not identified in IDEO DT (score 0 on column M). It is important to mention 2 points: (i) as described by Kitchenham [8] this JIs evaluation is according to the evaluator's perception, and; (ii) all sub-features have JI with score 1 for Helius because we used the prototype-based version of the tool, considering all features we intend to implement on it.

Consequently, based on the previous definitions, we performed the DESMET evaluation. Table 4 shows the complete data of this evaluation. Table 4-(D) shows the maximum possible feature score for a feature set. It is the sum of the importance levels of the sub-features from each feature set. For example, 12 represents the max feature set score reachable in the feature set F1, 6 for F2, and so on.

Next, based on the Judgment scores and its Interpretations (JIs), shown in Table 4 – columns E (Helius), I (DTA4RE), and M (IDEO DT), a weight score for each feature of each tool is calculated (Table 4 – F (Helius), J (DTA4RE), N (IDEO DT)). The weight score is the multiplication of the JI with the respective sub-feature Importance level (Table 4 – (C)). For example, for F1-SF01, the weight score for Helius is 4 (Table 4 – (F)), obtained with Table 4 – (E)(JIS) * Table 4 – (C)(sub-feature importance level).

Next, is calculated a Feature Set Score Obtained (Table 4 – (G (Helius), K (DTA4RE), O (IDEO DT))). This feature set score obtained is the sum of the weight scores for a feature set. For example, in Table 4 – (K) we show the feature score obtained for the F1 by DTA4RE, resulting in 8 (4+4+0).

In the sequence, it is calculated the percentage of the feature set score reached (Table 4 – (H (Helius), L (DTA4RE), P (IDEO DT))). The values are calculated dividing the feature score obtained (Table 4 – (G (Helius), K (DTA4RE), O (IDEO DT))) by the max possible feature score for a each Feature set (Table 4 – (D)). For instance, for F1 DTA4RE reached to 66.67% of the max possible feature set score (Table 4 – (L)); DTA4RE reached, and IDEO DT reached to 16.67% (Table 4 – (P)).

Finally, the overall score for each tool is calculated with the sum of each % feature set score, considering the feature set importance weight (Table 3). For example, considering the importance weight of F1 as 0,3, the % feature set score for F1 for DTA4RE (Table 4 – (L)) is 66,67% * 0,3 = 20%. For F2, the importance weight is 0,2, then it is calculated using 100% * 0,2 = 20%, and so on. At the end, all these values are summed and the overall score is obtained.

The overall score for DTA4RE was 43,33% and for IDEO DT was 20,83%. These results show Helius as an appropriate recommender and collaborative system for the features we defined (Table 1), aiming to support the use of DT techniques in software development.

5 RESEARCH AGENDA

This section presents our next planned studies to conclude the Helius tool idea validation, to improve the recommendation feature,

and to pilot the tool use with software engineering students, and to next explore it with industry professionals.

- **Prototype-based user validation** - In this activity our goal is to perform a prior tool validation with professionals to validate the features and the prototypes. We intend to demonstrate the features, sub-features, and prototypes that we created for Helius to collect feedback from potential users.
- **SLR on Recommendation Systems** - In this activity we intend to explore the literature through a systematic review to obtain the state-of-the-art about recommendation algorithm to be used on Helius, to recommend the appropriate DT techniques according to the user's context. We will update the tool prototype with the findings from this study.
- **Tool pilot-validation (SE students)** - Next, we aim to perform an evaluation with Software Engineering students before testing Helius performance and validity with industry professionals. Thus, we aim to gather the students impressions about Helius, and to validate the operation of the tool.

Apart from these activities, we will also carry out additional studies with the participation of industry professionals to further evaluate our DT techniques recommendation tool but rather focusing in whether the tool proposal indeed facilitates the selection of DT techniques. For instance, we plan an industry-based case study to identify the perception of how the tool recommendations help with improving the adoption of DT in practice and, as a consequence, improves the overall software development process experience and. Our full research agenda is detailed in [13].

6 FINAL REMARKS AND CONTRIBUTIONS

Design Thinking is applied to software development based on a set of techniques that explore creativity and innovation. DT provides techniques that allow to meet the user's needs supporting activities for the development of software with quality considering the user's co-creation activities to achieve quality attributes such as reliability, fitness for use, security, privacy, usability, maintainability However, recognizing which techniques are appropriate for this dynamic universe is a challenge to overcome.

In this paper, we proposed Helius for helping software development professionals to select suitable techniques to apply in their projects. Characterized as a recommender and collaborative system, Helius is composed of 6 features: (F1) DT Techniques Recommendations, (F2) DT Techniques Filtering, (F3) DT Techniques Evaluation, (F4) DT Techniques Community Feedback, (F5) DT Techniques Information, and (F6) DT Techniques and Project Management.

Our preliminary results obtained comparing Helius with related tools using DESMET methodology showed the potential of Helius to recommend DT techniques for software development.

With this in mind, Helius can be pointed out as an appropriate tool for DT techniques recommendation for helping the software development professional to apply DT in their projects and for producing a solution that fits the users' needs.

Our future work aims to validate Helius with professionals from industry for gathering relevant feedback and improving the features of our tool. Also, we aim to study in depth the recommender systems and its challenges. Moreover, we plan to perform studies with professionals working on real projects to evaluate the potential of improvement in DT activities integrated to software process and perceived effects on software quality.

ACKNOWLEDGMENTS

We thank PUCRS BPA 2019 and 2020 projects (Programa de Bolsas Pesquisa Alunos da PUCRS/Chamada Geral 1/2019 e 1/2020) and CNPq PIBIC 2019/2020 project (Programa Institucional de Bolsas de Iniciação Científica do CNPq) for the undergraduate research assistant scholarships. This study is also partially financed by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil) – Código de Financiamento 001.

REFERENCES

[1] Tim Brown and Barry Katz. 2011. Change by Design. *Journal of Product Innovation Management* 3 (2011), 381–383.
[2] Luis Corral and Ilenia Fronza. 2018. Design Thinking and Agile Practices for Software Engineering: An Opportunity for Innovation. In *Proceedings of the SIG Conference on Information Technology Education*. ACM, Fort Lauderdale, USA, 26–31.
[3] Aline de Oliveira Sousa and Natasha Malveira Costa Valentim. 2019. Prototyping Usability and User Experience: A Simple Technique to Agile Teams. In *Proceedings of the Brazilian Symposium on Software Quality*. ACM, Fortaleza, Brazil, 222–227.
[4] J. Hehn, D. Mendez, F. Uebernickel, W. Brenner, and M. Broy. 2020. On Integrating Design Thinking for Human-Centered Requirements Engineering. *IEEE Software* 37, 2 (2020), 25–31.
[5] Jennifer Hehn, Falk Uebernickel, and Daniel Mendez Fernandez. 2018. DT4RE: Design Thinking for Requirements Engineering: A Tutorial on Human-Centered and Structured Requirements Elicitation. In *Proceedings of International Requirements Engineering Conference*. IEEE, Alberta, Canada, 504–505.
[6] Muktha Hiremath and Visvapriya Sathiyam. 2013. Fast Train to DT: a practical guide to coach design thinking in software industry. In *Proceedings of Conference on Human-Computer Interaction*. Springer, Las Vegas, USA, 780–787.
[7] IDEO. Accessed Mar 20, 2020. *IDEO Design Kit.* https://designkit.org/methods
[8] Barbara Ann Kitchenham. 1996. Evaluating software engineering methods and tool part 1. *ACM SIGSOFT Software Engineering Notes* 21, 1 (jan 1996), 11–14.
[9] Tilmann Lindberg, Christoph Meinel, and Ralf Wagner. 2011. *Design Thinking: A Fruitful Concept for IT Development?* Springer, Heidelberg, Germany, 3–18.
[10] Christopher Marshall, Pearl Brereton, and Barbara Kitchenham. 2014. Tools to support systematic reviews in software engineering. In *Proc. of the Intl' Conf. on Evaluation and Assessment in Soft. Eng.* ACM, New York, USA, 1–10.
[11] Galia Novakova Nedeltcheva and Elena Shoikova. 2017. Coupling Design Thinking, User Experience Design and Agile: Towards Cooperation Framework. In *Proceedings of the International Conference on Big Data and IOT*. ACM, London, UK, 225–229.
[12] Rafael Parizi, Marina da Silva, Igor Couto, Kendra Trindade, Matheus Plautz, Sabrina Marczak, Tayana Conte, and Heloisa Candello. 2020. Design Thinking in Software Requirements: What Techniques to Use? A Proposal for a Recommendation Tool. In *Proc. of the Ibero-American Conf on Soft. Eng.* Curitiba, Brazil.
[13] Rafael Parizi and Sabrina Marczak. 2020. A Context-based Recommendation Model for Design Thinking: Techniques Selection in Software Development. In *Proceedings of the Brazilium Symposium on Software Engineering*. Natal, Brazil.
[14] Matheus Prestes. 2020. *Estudo Exploratório sobre Design Thinking no Desenvolvimento de Software.* Master's thesis. Escola Politécnica, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brazil.
[15] Matheus Prestes, Rafael Parizi, Sabrina Marczak, and Tayana Conte. 2020. On the Use of Design Thinking: A Survey of the Brazilian Agile Software Development Community. In *Proceedings of the International Conference on Agile Software Development*. Springer, Copenhagen, Denmark, 73–86.
[16] T. Rozante de Paula, T. Santana Amancio, and J. A. Nonato Flores. 2020. Design Thinking in Industry. *IEEE Software* 37, 2 (2020), 49–51.
[17] Anderson Souza, Bruna Ferreira, and Tayana Conte. 2017. Aplicando Design Thinking em Engenharia de Software: Um Mapeamento Sistemático. In *Proceedings of the Ibero-American Conference on Software Engineering*. Curran Associates, Buenos Aires, Argentina, 719–732.
[18] Anderson Felipe Barros de Souza. 2019. *DTA4RE: um assistente de apoio ao design thinking para elicitação de requisitos.* Master's thesis. Universidade Federal do Amazonas, Manaus, Brazil.
[19] Mark F. Tannian. 2020. *Embracing Quality with Design Thinking.* Springer, Cham, Switzerland, 161–174.