How has Design Thinking being Used and Integrated into Software Development Activities? A Systematic Mapping

Rafael Parizi[a,*], Matheus Prestes[a], Sabrina Marczak[a] and Tayana Conte[b]

[a]Pontifical Catholic University of Rio Grande do Sul - PUCRS
[b]Federal University of Amazonas - UFAM

ARTICLE INFO

Keywords:
Software Development
User-centered Design
Design Thinking
Literature Review
Systematic Mapping Study

ABSTRACT

Software companies have been using Design Thinking (DT) as a user-centered design approach, putting the user at the center of the software development process. In this article, we report a Systematic Mapping Study to investigate the use of DT in software development. We evaluated 127 papers from 2010 to 2021. We analyzed how DT is integrated in software development, what are the models and techniques, what are the criteria used for selecting DT techniques, and what are the key points that DT practitioners should be aware of when using DT. As a result, we identified 3 strategies to integrate DT in software development, 16 models, and 85 techniques. We also found that the selection of techniques is related to the models' working spaces being performed, and identified 7 criteria used for selecting DT techniques. Furthermore, we summarized 16 key points that DT practitioners should pay attention when using DT, and we proposed 4 takeaways for applying DT in software development. Thus, our study contributes to DT practitioners by providing information to be used either as a starting point, or to integrate it into activities already performed by teams, or as a strategy to foster changes in the entire organization's mindset.

1. Introduction

Software teams have been putting the user at the center of the development process (Péraire, 2019), using approaches denominated as part of the user-centered (UCD) umbrella. UCD aids a deep understanding of users and their goals, needs, and restrictions (González et al., 2010). It also foster empathy, team collaboration, and customer and development team interaction (Martins et al., 2019).

Design Thinking (DT) is a UCD approach that involves the user on the development of innovative software solutions (Pereira and Russo, 2018). DT seeks to foster creativity and to practice both convergent and divergent thinking (Alhazmi and Huang, 2020), helping teams to deal with wicked problems, i.e., ill-structured problems which have no clear definition and established solution (Buchanan, 1992; Senft et al., 2019; Sohaib et al., 2019). DT fosters the creation of multidisciplinary teams for exploring techniques and processes focused on satisfying users' expectations of the product/service developed (Vianna, 2012), and promotes the use of designers' empathy to address what is technologically suitable and feasible when proposing a solution.

Given its iterative approach to problem-solving, DT has been integrated with agile methods for boosting software development (Pereira and Russo, 2018; Magare et al., 2020). While DT fosters the understanding of the problem and the search for a solution that meets the user's needs, Agile methods focus on speed, simplicity, continuous and fast deliveries, frequent feedback, and quick reaction to changes (Gurusamy et al., 2016). Nevertheless, the nature of software development teams and structure associated with the lack of training on the design subject, and the number of DT models and techniques available, using DT becomes challenging. Therefore, it is important to investigate how DT has been used to support software development and what resources are available to meet user needs by delivering solutions that address the problem at hand.

Literature review studies such as the one by Souza et al. (2017) are research efforts for reporting the use of DT in software development aiming to help practitioners on how to use DT. The authors evaluated 22 papers and mapped 11 models and 55 techniques of DT. Results also show that DT is a dynamic approach that does not define an order to its working spaces, allowing adaptation according to the context of the problem. DT techniques support the development teams in the final product innovation process. Waidelich et al. (2018) performed a literature review and analyzed 35 documents, including journal papers, textbooks, and web documents in German and in English, to provide a broad overview of DT models used in practice, but not limited to software development. The authors conclude that there is no standardized model for use. They also pointed out that there is flexibility in the steps to be followed, indicating that the study of which techniques can be used in each working space is a research possibility to be carried out. Pereira and Russo (2018) present a literature review that evaluated 29 papers between 2008 and 2017 to identify how the DT approach and Agile methods are integrated into the development process, which strategies are used, and which models exist to carry out this integration. The authors figured out that the DT integrated to Agile seeks further to capture users' needs in the early stages and ensure the usability of the software.

*Corresponding author
rafael.parizi@edu.pucrs.br (R. Parizi); matheus.plautz@edu.pucrs.br (M. Prestes); sabrina.marczak@pucrs.br (S. Marczak); tayana@icomp.ufam.edu.br (T. Conte)
ORCID(s): 0000-0001-8550-1135 (R. Parizi); 0000-0002-4286-7011 (M. Prestes); 0000-0001-9631-8969 (S. Marczak); 0000-0001-6436-3773 (T. Conte)
[1]PUCRS: Ipiranga Avenue, 6681 - Partenon, RS, 90619-900, Brazil

On the other hand, literature also presents that DT can not be considered as a silver bullet. For instance, Pereira et al. (2021) investigated not only the benefits of using DT in software development, but also the challenges that IT professionals might face with the use of DT. The authors conducted a focus-group study with 39 professionals from distinct companies' roles, and pointed out that the use of DT requires attention to points such as the time pressure, the lack of participants' engagement, the resistance for applying DT, and the lack of empathy. De Paula et al. (2020) discuss points and counterpoints of applying DT in the software industry. They figured out that in addition to know the right problem and to identify the appropriate solutions, the use of DT might include some risks such as lack of participants commitment, or high effort to conduct DT activities that can be considered waste of time. Therefore, IT professionals should be aware of some attention points when using DT in software development in order to explore it effectively.

In this context, although the literature and the industry have been extensively studied, such studies are limited to showing the DT techniques, models, and tools used in software development. Therefore, given the importance that professionals have attributed to the DT (Levy and Huli, 2019), it is necessary to go further. This article aims to add to current knowledge by furthering the understanding of the use of DT in software development. In addition, our study aims to fill the gap of other literature reviews by identifying not only the DT models and techniques, but also what are the DT integration strategies in systems development activities, what criteria are considered by professionals in making decisions about which DT techniques to use, and what are the points that the DT practitioners have to pay attention when deciding to use DT. We performed a Systematic Mapping Study (SMS) following the guidelines defined by Petersen et al. (2008). Our study aims to answer the following Research Question (RQ): *How has Design Thinking been integrated into software development, what models and techniques are used, how are DT techniques selected, and what are the key points that DT practitioners should be aware of when using DT for software development?*".

Therefore, this study contributes with researchers and novice, or even experts DT practitioners by synthesizing and advancing what is known about the use of DT in software development. Also, we further previous studies and report 3 strategies to integrate DT in software development, 16 models that can guide this integration, 83 techniques that can be used for problem understanding and solutions proposal, 7 criteria that have been considered to select the techniques, 16 key attention points that DT practitioners should be aware of when using DT, and 4 takeaways for helping DT practitioners on the use of DT for software development. Our article supports on the use of DT in software development, whether as a pre-development work or to integrate DT in activities already performed by the teams, or even to encourage the changing in the organization's mindset.

The remainder of this article is organized as follows: Section 2 introduces DT in a nutshell. Section 3 presents the method performed on this literature mapping. Section 4 shows the results, answering our research questions. Section 5 discusses the results of the SMS. Finally, Section 6 presents the final remarks of this article.

2. Design Thinking in a Nutshell

Brenner et al. (2016) present Design Thinking from 3 perspectives - DT as a mindset, as a process, or as a toolbox. This characterization has been widely accepted in the literature (Kuula et al., 2020; Levy and Huli, 2019; Hehn et al., 2020; Mahe et al., 2020). DT as a mindset considers that innovation is made by humans for humans, combines divergent and convergent thinking, promotes the philosophy of fail often and early, fosters the creation of prototypes that can be experienced, and tests early with customers (Dobrigkeit and de Paula, 2019). Dobrigkeit and de Paula (2019) also argue that DT as a mindset facilitates to work on teams that are composed by diverse professionals. Hehn et al. (2020) mention that the company's success includes changes in the development team's mindset. The authors argue that team members must be empathic to participate in co-creation activities since the value can be obtained only by understanding the customers' needs.

Design Thinking as a process is structured as a set of iterative working spaces, exploring both divergent and convergent thinking (Brenner et al., 2016). Literature reports a span of DT processes, also known as DT models. Each model defines DT as a set of working spaces to understand the problem and produce innovative solutions (Brown, 2008; Araújo et al., 2015). A set of techniques can be applied in each working space, configuring the third perspective of DT: as a toolbox (Hehn et al., 2020; Coutinho et al., 2016; Corral and Fronza, 2018). DT as a toolbox refers to the use of design methods and techniques from engineering, computer, and psychology to solve a problem (Kuula et al., 2020). Dobrigkeit and de Paula (2019) suggest that DT as a toolbox provides practitioners with multiple mechanisms to aid the creation of a solution in the design process. Also, the authors advocate that the use of appropriate methods is a core factor of success in the use of DT.

In software development, DT has been using as a problem-solving approach to support the understanding of the problem to be solved, to propose and to validate solutions that meet the users' needs (Alhazmi and Huang, 2020; Martins et al., 2019; Kuula et al., 2020), collaborating from early stages of software activities–from the elicitation of requirements (Hehn et al., 2020) to the creation of an innovative mindset in developers, engineers, and managers (Dobrigkeit and de Paula, 2019). DT also supports on a deep understanding of the user's needs, by increasing teams' collaboration, and by exploring innovation that fosters the development of user-centered software solutions (Dobrigkeit and de Paula, 2019; Hehn and Uebernickel, 2018; Vetterli et al., 2013).

Thus, our study aims to understand which strategies have been using to integrate DT into software development, which models and techniques are used, which criteria practitioners

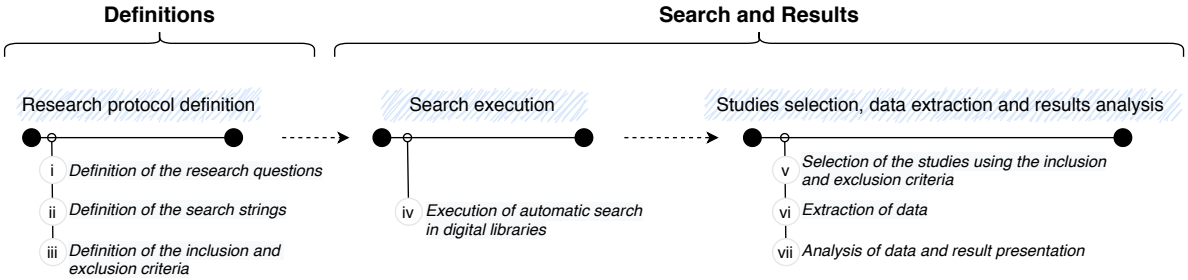

Figure 1: The Systematic Mapping Process and Respective Activities

have been using to select DT techniques, and what are the key points that DT practitioners should be aware of when using DT for software development.

3. Methodology

This section presents the systematic mapping study that we performed following the guidelines proposed by Petersen et al. (2008), aiming to identify papers that report the use of DT in software development. In addition, we performed a forward snowballing review as a way to supplement our literature mapping as reported by Felizardo et al. (2016). Our review aimed to answer the following RQ: "*How has DT been integrated into software development, what models and techniques are used, how are DT techniques selected, and what are the key points that DT practitioners should be aware of when using DT for software development?*".

3.1. Systematic Literature Mapping

Figure 1 illustrates the SMS process that we followed, which was composed of 7 activities (activities i to vii). We started by defining a research protocol for our mapping study. The protocol is composed of the research questions, the search approach, and the criteria to include or to exclude papers. Then, starting from the main RQ, we derived the research questions (activity i):

RQ1. What strategies for integrating Design Thinking in software development have been adopted?
RQ2. What DT models are used in software development?
RQ3. What Design Thinking techniques are used in software development?
RQ4. What is reported about the Design Thinking techniques selection in software development?
RQ5. What are the key points to be aware of when using DT in software development?

We formulated the search strings (activity ii) using the keywords strategy, as defined by Petersen et al. (2015). We used Software Development and Design Thinking as 2 keywords categories (see Table 1). The keywords in each category were combined with a Boolean operator "OR", and the categories were combined using a Boolean operator "AND". The search string defined was:

Table 1
Keywords used in this study

Category	Keywords
Software Development	software engineering software development software industry software construction software project software process software project management
Design Thinking	design thinking design session

Table 2
Inclusion and Exclusion Criteria

Type		Description
Inclusion	IC1	Papers related to DT in software development
Exclusion	EC1	Papers that do not attend the IC1, i.e., that do not discuss DT in software development
	EC2	Duplicated papers
	EC3	Papers not available for download
	EC4	Papers not written in English
	EC5	Papers that were not peer reviewed

(("software engineering") OR ("software development") OR ("software industry") OR ("software construction") OR ("software project") OR ("software process") OR ("Software project management")) AND (("design think*") OR ("design session")).

Table 2 shows the criteria that we used for selecting relevant papers. This study focuses on papers reporting DT usage in software development (Inclusion Criteria 1 – IC1). The exclusion criteria allow us to not include papers not available or duplicated, papers not written in English (EC2, EC3, and EC4), or papers not peer-reviewed (EC5).

Next, we performed automatic searches for papers in August 2020[2] on the following digital libraries (activity iv):

[2]We run the automatic searches on the digital libraries in August, 2020. Then, aiming to supplement the set of papers about the use of DT in software development, we performed a forward snowballing as suggested by Felizardo et al. (2016).

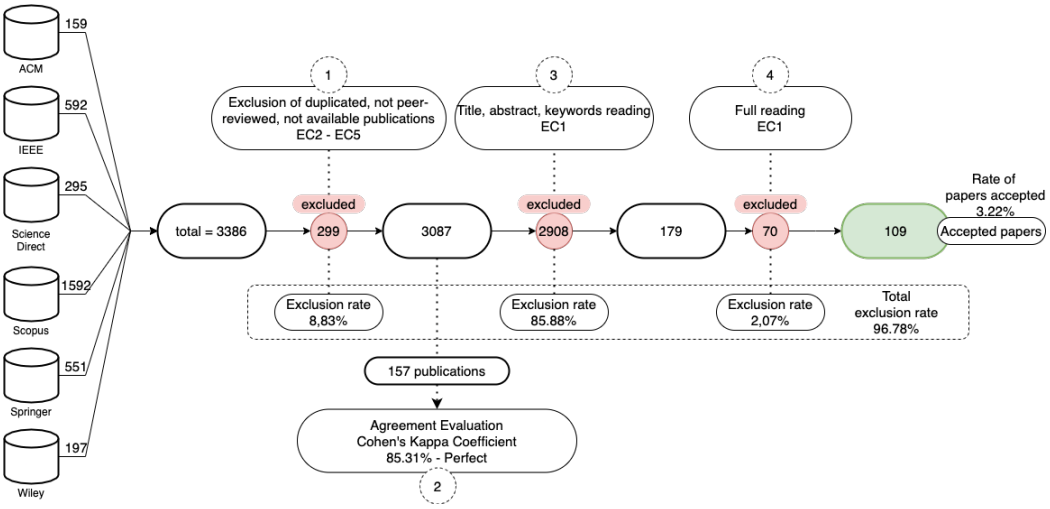

Figure 2: SMS - Papers selection process and results

Table 3
Papers retrieved from the digital libraries

Search Engine	Result (# of papers)
ACM	159
IEEE Xplore	592
Science Direct	295
Scopus	1592
Springer	551
Wiley	197
Total	**3386**

ACM Digital Library[3], IEEE Xplore[4], Science Direct[5], Scopus[6], Springer Database[7], and Wiley Online Library[8]. The total of retrieved papers was 3386 (see Table 3).

Figure 2 shows the papers' selection process (activity v). We performed the steps: 1) exclusion of duplicated, not available for download, not written in English, or not peer-reviewed papers, 2) evaluation of agreement using Cohen's Kappa coefficient (Landis and Koch, 1977), 3) reading the title, the abstract, and the keywords, and 4) reading the papers' full text.

1st step: Exclusion of duplicated, not available, not written in English, or not peer-reviewed papers

In the first step of the papers' selection process, we worked on removing duplicated papers by using the StArt tool[9], which provides technical support for the systematic research process. We also excluded papers not peer-reviewed, not available for downloading, and papers not written in English, according to criteria EC2 to EC5, respectively. In this first step, we removed 299 papers, remaining 3087 papers.

[3]https://dl.acm.org/
[4]https://ieeexplore.ieee.org
[5]https://www.sciencedirect.com/
[6]https://www.scopus.com
[7]https://link.springer.com/
[8]https://onlinelibrary.wiley.com/
[9]http://lapes.dc.ufscar.br/tools/start_tool

Table 4
KAPPA Coefficient Indexes Viera et al. (2005)

Kappa	Agreement description
< 0	Without agreement
0.01 – 0.20	Slight agreement
0.21 – 0.40	Fair agreement
0.41 – 0.60	Moderate agreement
0.61 – 0.80	Substantial agreement
0.81 – 0.99	Perfect agreement

Table 5
KAPPA Coefficient result

	Studies Accepted	Studies Rejected	KAPPA Coefficient
Author 1	33	124	0.8531
Author 2	37	120	

The exclusion rate of papers was 8.83%.

2nd step: Agreement evaluation using Kappa coefficient

In the second step, we performed an agreement evaluation using Cohen's Kappa coefficient (Landis and Koch, 1977). Kappa provides a coefficient for estimating the degree of agreement between two reviewers (Unnikrishnan and Hebert, 2005). Table 4 presents Kappa's agreement values.

In this step, we selected 5% of the papers (157 papers) to compute Cohen's Kappa agreement coefficient. The papers were selected randomly and 2 authors examined them. Through this step, we obtained an agreement's coefficient of 0.8531, considered perfect according to Kappa's agreement indexes. Table 5 illustrates the Kappa coefficient result. Author 1 accepted 33 papers and rejected 124 papers, while author 2 accepted 37 and rejected 120 papers, resulting in a perfect degree of agreement. Thus, we proceeded with papers selection considering the inclusion and exclusion criteria, based on Cohen's Kappa coefficient result.

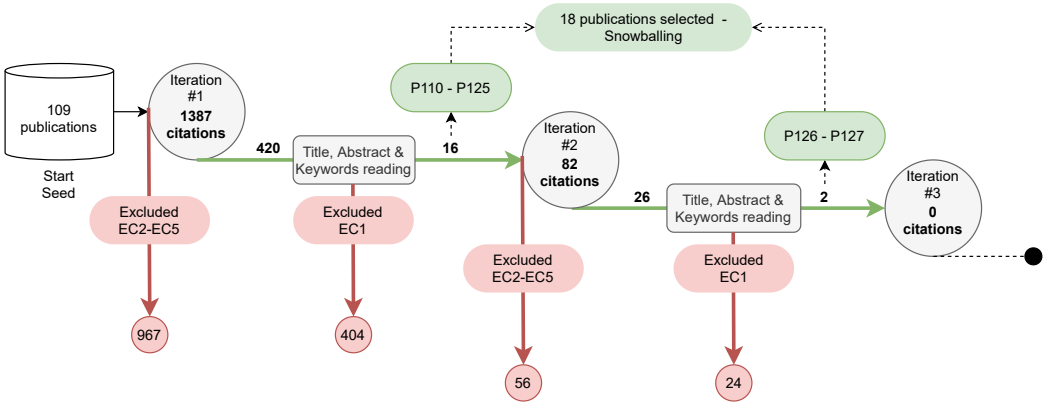

Figure 3: Forward snowballing iterations

3rd step: Reading the title, abstract and keywords

In this step, we performed the papers' selection by reading the title, keywords, and abstract. We read the 3087 papers and applied the IC1 or EC1 criteria. As a result, we excluded 2908 papers that did not satisfy the inclusion and exclusion criteria, remaining 179 papers of the total of retrieved papers from the automatic searches. The exclusion rate was 85.88%.

4th step: Reading the full text

In this step we read the full text of the 179 remaining papers. We excluded 70 papers that did not correspond to DT in software development, resulting in an exclusion rate of 2.07%. Thus, the papers' selection process resulted in 109 selected papers. The total exclusion rate was 96.78%, and the acceptance rate was 3.22%.

3.2. Forward Snowballing

Figure 3 shows the forward snowballing procedure we followed. Felizardo et al. (2016) argue that forward snowballing is an alternative procedure for updating literature review studies. We performed the forward snowballing procedure in 3 iterations. We used the set of 109 papers selected in the SMS as the start seed. So, we looked for citations from each of the 109 papers in the start seed. We used Google Scholar[10] to find citations for each paper.

We downloaded the citations and analyzed each one according to the inclusion and exclusion criteria previously established. We started the selection process excluding duplicated papers, not peer-reviewed, not written in English or not available for downloading. The remaining papers were analyzed by title, abstract, and keywords.

Iteration 1

In iteration 1, we found 1387 citations for the 109 papers of the start seed. Initially, we excluded 967 citations using the EC2-EC5 exclusion criteria, remaining 420 citations. Then, we read the title, abstract and keywords of the 420 citations. We excluded more 404 papers using EC1. Next, we did the full reading and accepted 16 papers in iteration 1.

Iteration 2

In iteration 2, considering as start seed the 16 papers that we selected in iteration 1, we found 82 citations. Initially, we excluded 58 citations using the EC2-EC5 exclusion criteria, remaining 26 citations. Then, we read the title, abstract and keywords of the 26 citations. We excluded more 24 papers using EC1. Next, we did the full reading and accepted 2 new papers in iteration 2.

Iteration 3 In iteration 3, considering as start seed the 2 papers that we selected in iteration 2, we did not find citations. So, we ended the forward snowballing iterations. Therefore, we have selected 127 papers in our mapping study. Section 4 presents the results we obtained performing the data extraction of the selected papers.

4. Results

This section presents the results of this Systematic Mapping Study. Initially, we show the results' metadata that we extracted from the selected papers. Next, we discuss each research question that we posed in Section 3.

Table 6 list the 109 accepted papers through the SMS, and Table 7 shows the 18 selected papers that we accepted through the forward snowballing procedure. Both tables include paper's ID, classification (Book chapter, Conference paper, or Journal article), title, and authors. We use the paper ID to identify the paper in the analysis of the results. We also classified the selected papers in empirical and non-empirical research (Cruzes and Dybå, 2010), as follows:

- Empirical research: papers with methodological procedures applied in a real context, such as:
  - Academic context: the paper was developed with students in an academic context;
  - Industry context: the paper was developed in an industry environment;

[10] https://scholar.google.com

Table 6
Selected Papers

ID	C	Title	Authors
001	○	Integrating the Design Thinking into the UCD's methodology	González et al. (2010)
002	●	A heuristic approach for supporting innovation in requirements engineering	El-Sharkawy and Schmid (2011)
003	○	Design thinking for search user interface design	Berger (2011)
004	○	Increasing Kenyan Open Data Consumption: A Design Thinking Approach	Mutuku and Colaco (2012)
005	●	From Palaces to Yurts: Why Requirements Engineering Needs Design Thinking	Vetterli et al. (2013)
006	○	Design thinking methodology for the design of interactive real-time applications	Sandino et al. (2013)
007	○	Fast train to DT: A practical guide to coach design thinking in software industry	Hiremath and Sathiyam (2013)
008	○	Reframed contexts: Design thinking for agile user experience design	Adikari et al. (2013)
009	○	CoDICE: Balancing software engineering and creativity in the co-design of digital encounters with cultural heritage	Díaz et al. (2014)
010	●	Design thinking: Expectations from a management perspective	Rhinow and Meinel (2014)
011	●	From product development to innovation	McMahon (2014)
012	○	Guiding novice database developers in database schema creation	Ahmad et al. (2014)
013	○	Lean UX - The next generation of user-centered Agile development	Liikkanen et al. (2014)
014	○	Design thinking for usability evaluation of cloud platform service	Kah-Hoe and Wang (2014)
015	○	The Role of Design Thinking and Physical Prototyping in Social Software Engineering	Newman et al. (2015)
016	○	Trends in the Use of Design Thinking for Embedded Systems	Araújo et al. (2015)
017	○	Eliciting Requirements Using Personas and Empathy Map to Enhance the User Experience	Ferreira et al. (2015)
018	○	Design thinking methods and tools for innovation	Chasanidou et al. (2015)
019	●	A Brief Introduction to Design Thinking	Luchs (2015)
020	○	Design Thinking Framework to Enhance Object Oriented Design and Problem Analysis Skill in Java Programming Laboratory: An Experience	Rajashekharaiah et al. (2016)
021	○	Can Metamodels Link Development to Design Intent?	Gamble (2016)
022	○	Aligning healthcare innovation and software requirements through design thinking	Carroll and Richardson (2016)
023	○	IBM design thinking software development framework	Lucena et al. (2016)
024	●	LODPRO: learning objects development process	Queiros et al. (2016)
025	●	Models as bridges from design thinking to engineering	Tellioğlu (2016)
026	○	OnTimeCargo: A smart transportation system development in logistics management by a design thinking approach	Azab et al. (2016)
027	○	Pet empires: Combining design thinking, lean startup and agile to learn from failure and develop a successful game in an undergraduate environment	de Paula and Araújo (2016)
028	○	The origins of design thinking and the relevance in software innovations	Jensen et al. (2016)
029	○	An Integrated Framework for Design Thinking and Agile Methods for Digital Transformation	Gurusamy et al. (2016)
030	●	Are We Ready for Disruptive Improvement?	Rösel (2016)
031	●	Communication Breakdowns in the Integration of User-Centred Design and Agile Development	Bordin and De Angeli (2016)
032	○	Increasing the Quality of Use Case Definition Through a Design Thinking Collaborative Method and an Alternative Hybrid Documentation Style	Matz and Germanakos (2016)
033	○	Developing High-Performing Teams: A Design Thinking Led Approach	Keighran and Adikari (2016)
034	○	Embedded Design Thinking in Co-Design for Rapid Innovation of Design Solutions	Adikari et al. (2016)
035	●	From the Real to the Virtual: Developing Improved Software Using Design Thinking	Malins and Maciver (2016)
036	○	Design Thinking Framework for Project Portfolio Management	Sarbazhosseini et al. (2016)
037	○	The Use of Design Thinking in Agile Software Requirements Survey: A Case Study	Canedo and Parente da Costa (2016)
038	●	Framing Design Thinking: The Concept in Idea and Enactment	Carlgren et al. (2016)
039	●	Applying design thinking methods to ecosystem management tools: Creating the Great Lakes Aquatic Habitat Explorer	Goodspeed et al. (2016)
040	○	Promoting the Elicitation of Usability and Accessibility Requirements in Design Thinking: Using a Designed Object as a Boundary Object	Levy (2017)
041	○	Question-answer analysis in design thinking at the conceptual stage of developing a system with a software	Sosnin (2017)
042	○	The Agile Manifesto, design thinking and systems engineering	Darrin and Devereux (2017)
043	○	Hackathons, semesterathons, and summerathons as vehicles to develop smart city local talent that via their innovations promote synergy between industry, academia, government and citizens	Avalos et al. (2017)
044	○	An entrepreneurial narrative media-model framework to knowledge building and open co-design for smart cities	Lee and Sohn (2019)
045	○	The Students' Perspectives on Applying Design Thinking for the Design of Mobile Applications	Valentim et al. (2017)
046	○	Infusing Design Thinking into a Software Engineering Capstone Course	Palacin-Silva et al. (2017)
047	○	Identifying Design Features Using Combination of Requirements Elicitation Techniques	Murugesan et al. (2017)
048	○	FATHOM: TEL Environment to Develop Divergent and Convergent Thinking Skills in Software Design	Reddy et al. (2017)
049	○	A2BP: A method for ambidextrous analysis of business process	Santos and Alves (2017)
050	○	Coupling design thinking, user experience design and agile: Towards cooperation framework	Nedeltcheva and Shoikova (2017)
051	○	Design thinking methods and techniques in design education	Kloeckner et al. (2017)
052	○	Introducing 'Human-Centered Agile Workflow' (HCAW) – An Agile Conception and Development Process Model	Glomann (2017)
053	○	The best of three worlds -The creation of innodev a software development approach that integrates design thinking, scrum and lean startup	Dobrigkeit et al. (2017)
054	○	Design Thinking and Agile Practices for Software Engineering: An Opportunity for Innovation	Corral and Fronza (2018)
055	○	The Importance of Empathy for Analyzing Privacy Requirements	Levy and Hadar (2018)
056	○	Policy Recommendations to Induce Behavioural Changes through Interactive Energy Visualisation	Murugesan et al. (2017)
057	○	A first implementation of a design thinking workshop during a mobile app development course project	Pham et al. (2018)

*Continue on the next page

Paper Category(C): ○ Conference proceedings | ● Book chapter | ● Journal article
Research Type (T): Empirical -{000 Academic | 000 Industry | 000 Innovation} | Non-empirical -000 Theoretical

Table 6
Selected Papers (continued from the previous page)

ID	C	Title	Authors
058	○	Adopting design thinking practices to satisfy customer expectations in agile practices: A case from Sri Lankan software development industry	Prasad et al. (2018)
059	○	CPM / PDD in the context of Design Thinking and Agile Development of Cyber-Physical Systems: Use cases and methodology	Luedeke et al. (2018)
060	○	Designing human-centric information systems: Towards an understanding of challenges in specifying requirements within design thinking projects	Hehn et al. (2018)
061	○	Educating for empathy in software engineering course	Levy (2018)
062	●	Juicing the game design process: towards a content centric framework for understanding and teaching game design in higher education	Larsen (2018)
063	○	Adding Scrum-style project management to an advanced Design Thinking class	Dobrigkeit et al. (2018)
064	○	Software 4.0: 'How' of building 'Next-Gen' systems	Pendse and Amre (2018)
065	○	The use of design thinking for requirements engineering: An ongoing case study in the field of innovative software-intensive systems	Hehn and Uebernickel (2018)
066	○	Visual analytics for cyber-physical systems development: Blending design thinking and systems thinking	Gürdür and Törngren (2018)
067	◌	Using Design Thinking for Requirements Engineering in the Context of Digitalization and Digital Transformation: A Motivation and an Experience Report	Carell et al. (2018)
068	●	Effective Design Methodologies	Asante (2018)
069	●	Integrating cell and molecular biology concepts: Comparing learning gains and self-efficacy in corresponding live and virtual undergraduate laboratory experiences	Goudsouzian et al. (2018)
070	●	Advanced agile approaches to improve engineering activities	Burchardt and Maisch (2018)
071	○	The Product Backlog	Sedano et al. (2019)
072	○	Dual-track agile in software engineering education	Péraire (2019)
073	○	Design thinking in practice: Understanding manifestations of design thinking in software engineering	Dobrigkeit and de Paula (2019)
074	○	Design Thinking and Acceptance Requirements for Designing Gamified Software	Piras et al. (2019)
075	○	Design Thinking in a Nutshell for Eliciting Requirements of a Business Process: A Case Study of a Design Thinking Workshop	Levy and Huli (2019)
076	○	A Step by Step Methodology for Software Design of a Learning Analytics Tool in Latin America: A Case Study in Ecuador	Ortiz-Rojas et al. (2019)
077	○	Definition of Indicators in the Execution of Educational Projects with Design Thinking Using the Systematic Literature Review	Almeida et al. (2019)
078	●	How digital transformation can influence business model, Case study for transport industry	Genzorova et al. (2019)
079	●	Creating an innovative digital project team: Levers to enable digital transformation	Guinan et al. (2019)
080	●	Implementing Experience Sampling Technology for Functional Analysis in Family Medicine – A Design Thinking Approach	Daniëls et al. (2019)
081	●	Design thinking: Challenges for software requirements elicitation	Martins et al. (2019)
082	○	Towards applying design-thinking for designing privacy-protecting information systems	Bargh and Choenni (2019)
083	○	Design Thinking's Resources for in-situ Co-Design of Mobile Games	Challiol et al. (2019)
084	○	A Lean Design Thinking Methodology (LDTM) for Machine Learning and Modern Data Projects	Ahmed et al. (2018)
085	○	Combining challenge-based learning and design thinking to teach mobile app development	Gama et al. (2018b)
086	○	A Hackathon Methodology for Undergraduate Course Projects	Gama et al. (2018a)
087	○	SMARTD Web-Based Monitoring and Evaluation System	Budiarto et al. (2018)
088	○	Design Thinking and Scrum in Software Requirements Elicitation: A Case Study	Braz et al. (2019)
089	○	Mobile application based on design thinking for teaching kinematics	Arbieto-Batallanos et al. (2019)
090	○	CALDET: A TRIZ-Driven Integrated Software Development Methodology	Brad et al. (2019)
091	●	Challenges in Requirement Engineering: Could Design Thinking Help?	Kahan et al. (2019)
092	○	Design Thinking Versus Design Sprint: A Comparative Study	Araújo et al. (2019)
093	●	A Value-Centered Approach for Unique and Novel Software Applications	Senft et al. (2019)
094	○	Innodeck: Card based innovation support - A modular human-centered approach to facilitate innovation workshops	Harriet et al. (2019)
095	●	Using Agile Approaches to Drive Software Process Improvement Initiatives	Nalepa et al. (2019)
096	●	Integrating design thinking into extreme programming	Sohaib et al. (2019)
097	○	Integrating Design Thinking into Scrum Framework in the Context of Requirements Engineering Management	Alhazmi and Huang (2020)
098	○	A LX (learner experience)-based evaluation method of the education and training programs for professional software engineers	Kawano et al. (2019)
099	○	Design Thinking Approach for Mobile Application Design of Disaster Mitigation Management	Suzianti et al. (2020)
100	●	Design Thinking in Industry	De Paula et al. (2020)
101	●	Three Phases of Transforming a Project-Based IT Company Into a Lean and Design-Led Digital Service Provider	Kuula et al. (2020)
102	●	On Integrating Design Thinking for Human-Centered Requirements Engineering	Hehn et al. (2020)
103	●	Migrating a Software Factory to Design Thinking: Paying Attention to People and Mind-Sets	Mahe et al. (2020)
104	●	When Does Design Help Thinking, and When Does Design Thinking Help?	Penzenstadler (2020)
105	●	Designing mangrove ecology self-learning application based on a micro-learning approach	Arayalert and Puttinaovarat (2020)
106	●	Embracing Quality with Design Thinking	Tannian (2020)
107	●	Operationalizing Design Thinking in Business Intelligence and Analytics Projects	Chongwatpol (2020)
108	●	"StoryWeb": A storytelling-based knowledge-sharing application among multiple stakeholders	Park et al. (2020)
109	◌	Inherent Mapping Analysis of Agile Development Methodology Through Design Thinking	Magare et al. (2020)

Paper Category (C): ○ Conference proceedings | ◌ Book chapter | ● Journal article
Research Type (T): Empirical -{000 Academic | 000 Industry | 000 Innovation} | Non-empirical -000 Theoretical

Table 7
Selected Publications through a forward snowballing

ID	C	Title	Authors
110	○	Experimenting with design thinking in requirements refinement for a learning management system	Freitas et al. (2013)
111	○	Promoting creativity and innovative thinking in software engineering teaching: a case study	González and Fleitas (2015)
112	●	An experimental study of the use of design thinking as a requirements elicitation approach for mobile learning environments	de Carvalho Souza and Silva (2015)
113	●	Systems thinking approach to implementing kanban: A case study	Senapathi and Drury-Grogan (2021)
114	○	Empowering Project Managers in Enterprises-A Design Thinking Approach to Manage Commercial Projects	Kongot and Pattanaik (2017)
115	●	Supporting the teaching of design thinking techniques for requirements elicitation through a recommendation tool	Souza et al. (2020)
116	●	Technique for representing requirements using personas: a controlled experiment	Ferreira et al. (2018)
117	●	A Brief Study on Enhancing Quality of Enterprise Applications using Design Thinking	De and Vijayakumaran (2019)
118	●	Innovation through Design Thinking, User Experience and Agile: Towards Cooperation Framework	Nedeltcheva and Shoikova (2018)
119	◐	DT@IT Toolbox: Design Thinking Tools to Support Everyday Software Development	Dobrigkeit et al. (2020a)
120	○	Design Thinking Use in Agile Software Projects: Software Developers' Perception	Canedo et al. (2020)
121	○	User Experience Design for Disaster Management Mobile Application using Design Thinking Approach	Diaz Intal et al. (2020)
122	◐	InnoDev: a software development methodology integrating design thinking, scrum and lean startup	Dobrigkeit et al. (2019)
123	○	InnoDev Workshop: A One Day Introduction to Combining Design Thinking, Lean Startup and Agile Software Development	Dobrigkeit et al. (2020b)
124	○	Requirement Engineering and the Role of Design Thinking	Husaria and Guerreiro (2020)
125	○	The design thinking of co-located vs. distributed software developers: distance strikes again!	Jolak et al. (2020)
126	●	Designing a Persuasive Application for Behaviour Change with Children	Puad et al. (2019)
127	○	A Novel Perspective to Threat Modelling using Design Thinking and Agile Principles	De (2020)

Publication Category(C): ○ Conference proceedings | ◐ Book chapter | ● Journal article
Research Type (T): Empirical -{000 Academic | 000 Industry | 000 Innovation} | Non-empirical -000 Theoretical

- Innovation context: the paper was developed for social innovation;

- Non-empirical research: papers discussing theoretical and philosophical aspects, not applied in a practical context.

  - Theoretical context: the paper was not applied to any context.

4.1. Results Metadata

Figure 4 shows the papers' frequency by year related to DT in software development, categorized as book chapters, conference papers, journal articles. Most of them were published in 2019 (29 papers), followed by 2016 and 2020 (20 papers), and 2018 (19 papers).

We also classified the papers following the classification proposed by Wieringa et al. (2006), such as:

- Evaluation research: papers that describe results from the investigation of a problem in practice, showing the use of a method in practice.

- Proposal of a solution: papers that propose a solution technique and argue about its relevance, without complete validation. The technique must be new, or at least an improvement on an existing technique.

- Validation research: papers that investigate the properties of a proposed solution that has not yet been implemented in practice. It also includes papers that involve experiments, simulation, prototype or mathematical analysis.

- Philosophical papers: papers that present discussions about ways to understand the phenomenon in a conceptual way.

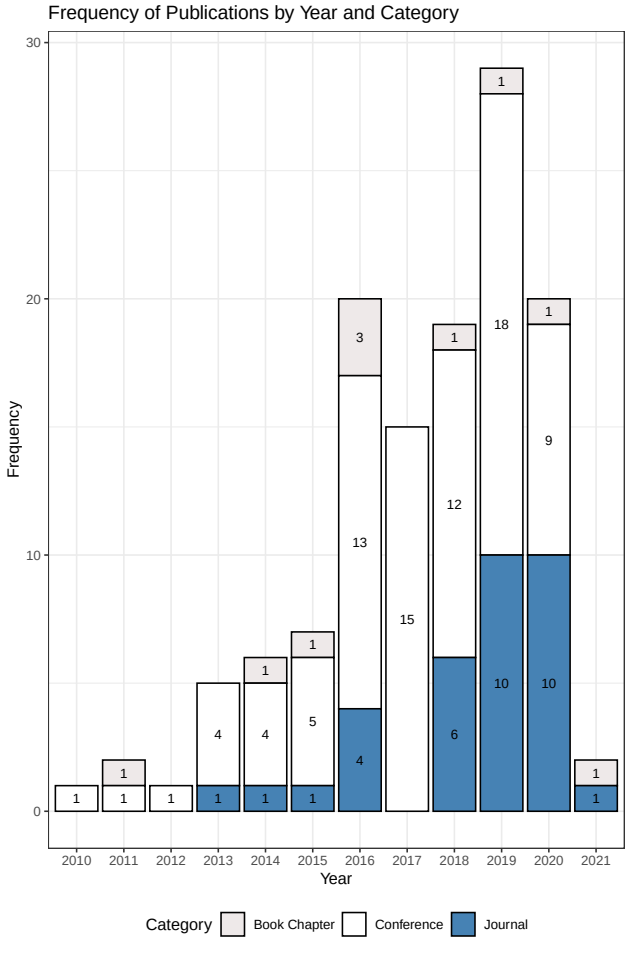

Figure 4: Frequency x Year of Each Papers Category

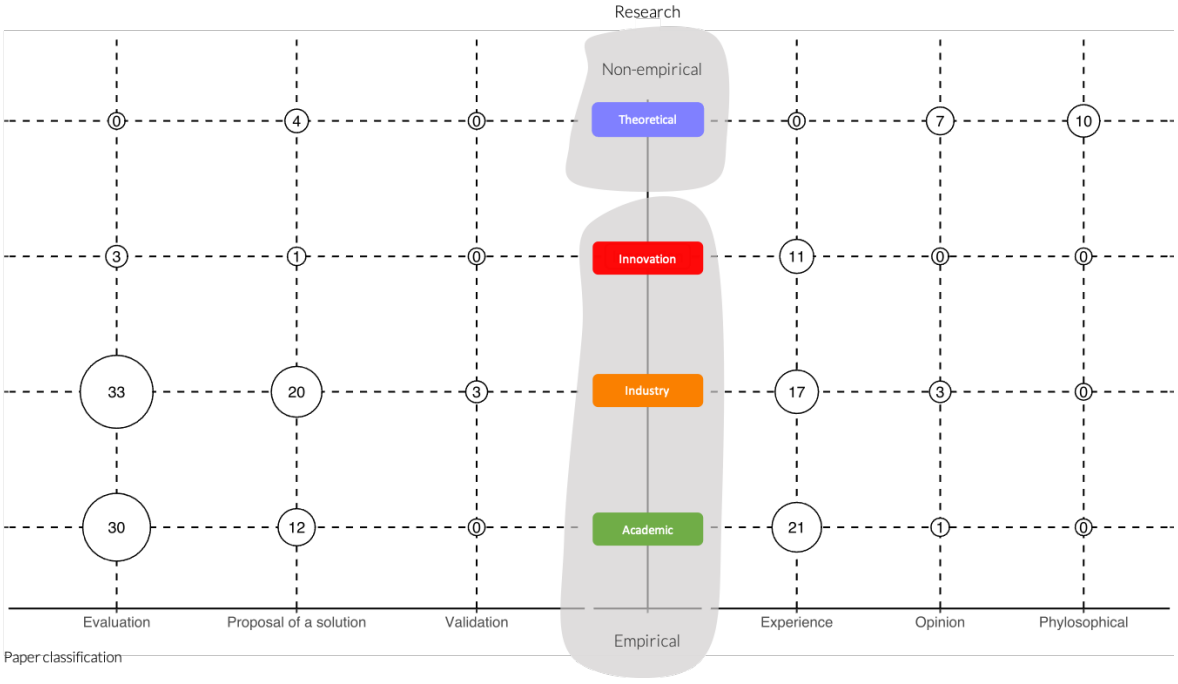

Figure 5: Papers' Classification x Research type

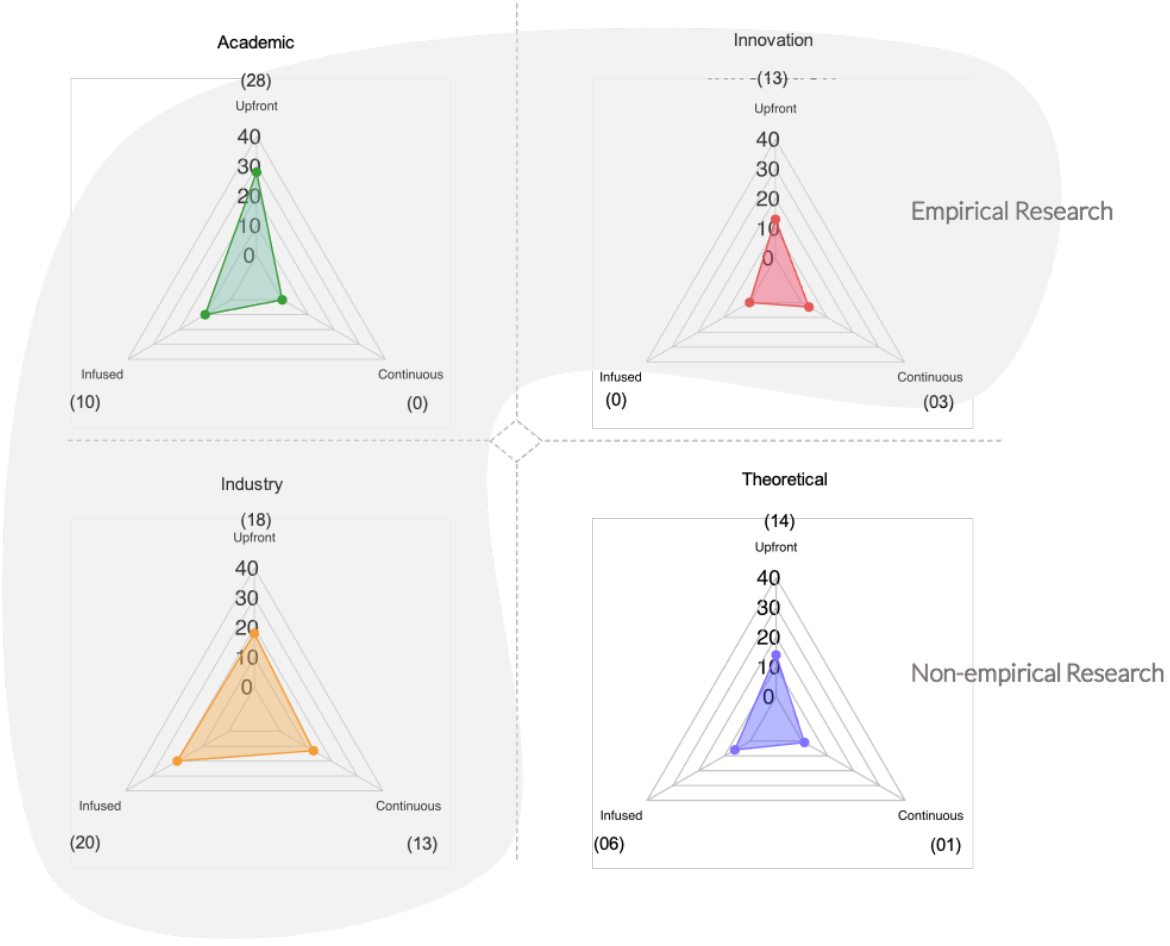

Figure 6: Strategies for Integrating DT in Software Development

- Opinion papers: papers that have the author's opinion about what is wrong or right about a given topic.
- Personal experience papers: papers that report experiences from applying some method or technique, including a set of lessons learned.

Wieringa et al. (2006) mention that the same paper can be classified in more than one class. Following this classification, we correlated each paper according to the 2 research types, including empirical research on the academic, industry, or innovation contexts, and non-empirical research in theoretical context. Figure 5 shows that most of studies were performed as empirical research in the industry context for evaluation purposes (33 papers), followed by experience reports in academic context (21 papers) and proposal of a solution in industry (20 papers).

4.2. Research Questions Results

RQ1: What strategies for integrating DT in software development have been adopted?

This research question classifies the use of DT integrated to software development following the integration strategies defined in (Hehn et al., 2020):

1. Upfront DT: in this strategy, DT is considered as a start activity of a software project. DT is used to understand the customer and to identify useful features to be implemented in the software.
2. Infused DT: in this strategy, DT is considered as a toolbox for supporting existing RE activities.
3. Continuous DT: in this strategy, DT is seen as a whole integrated approach in software development, involving the customers along the full development cycle.

Figure 6 illustrates the 3 integration strategies of DT in software development. We mapped the integration strategies for each research context: empirical research in the Academic, Industry and Innovation context, and non-empirical research in Theoretical papers. It is worth mentioning that we do not include in the plot the paper which introduced the strategies (e.g, (Hehn et al., 2020)).

The results show that Upfront DT is the most cited strategy for integrating DT into software development. It means that DT supports the teams to discover the users' needs, validating the candidate solution proposals at the beginning of the requirement engineering activities in a software development process. The studies using DT in an Upfront strategy consider it to understand and define the problem, ideate it into several candidate solution ideas, prototype the candidate solutions, and get feedback from the stakeholders. For instance, Dobrigkeit and de Paula (2019) performed an interview-based study and collected the team's perceptions about DT in software development. The interviewees reported that they use DT as a pre-development phase for understanding the problem, for gathering users' needs proposing solutions and validating prototypes before starting the development process itself.

Péraire (2019) used the Infused DT strategy for integrating DT in a Dual-track Agile approach, combining Human-Computer Interaction and Requirements Engineering in a course with students for software development. The authors used practices such as ideation and experimentation for generating alternatives of viable interaction design concepts that can satisfy stakeholders' needs.

Mahe et al. (2020) have used the Continuous DT integration strategy. The authors report the use of DT to change a company's mindset. They applied DT to a company aiming to modify the traditional software development methodology that had been used and that had been resulting in difficulties such as the reduction of employees' motivation, lack of communication between consultants and developers, and poor requirements elicitation.

Thus, taking DT as a continuous strategy, the company achieved an improvement on the requirements elicitation processes and it helped consultants and analysts to identify the most relevant features, to improve the relationship between customers and managers and company consultants, and to foster team engagement, to promote employee creativity, to encourage a strong collaboration, and to lead to a "fail fast, learn fast" philosophy.

RQ2: What DT models are used in software development?

In this question, we show the DT models that we found in the literature. A DT model represents a set of DT working spaces, encompassing activities that provide a way to start from the problem space and to evolve to the solution space (Dobrigkeit and de Paula, 2019). Brown and Wyatt (2010) advocate that the DT process is best thought of as a system of overlapping spaces rather than a sequence of orderly steps.

Table 8 shows the DT models that we identified in our Systematic Mapping Study, considering the number of working spaces themselves, the set of working spaces, and the studies that mentioned them.

 # DT model with 2 working spaces

Divergent-Convergent Inquiry-based Design Thinking model (DCIDT) is a DT model that organizes DT exploring divergent and convergent spaces associated with 2 activities: inquiry and questioning. Adikari et al. (2013) proposed DCIDT integrated with a method called A2BP to systematize the phase of analysis of the business process modeling. This method allows investigating how to explore internal and external business process opportunities.

 # DT models with 3 working spaces

Brown's DT model (Brown, 2008) was the second most cited DT model in literature. Brown's DT model has 3 working spaces: Inspiration, Ideation, and Implementation. Inspiration is a working space that helps to understand the problem, the needs, and the challenges of end-users. Ideation is a working space that allows analyzing the data collected previously to turn it into ideas to prototype the solution. Implementation is a working space performed to test

Table 8
Design Thinking Models, Working Spaces and Papers

# of Working Spaces	Model	Working Spaces	Papers
2	DCIDT	Divergent - Convergent	008
3	Brown	Inspiration - Ideation - Implementation	045 046 085 086 089 051 002 078 073 103 115 008
	Souza e Silva	Immersion - Ideation - Prototyping	017 112 077 088
	Daniëls et. al	Understand - Explore - Materialize	080 087
	Codice	Resources - Ideas - Design Products	009
4	Dunne and Martin	Generate Ideas (abduction) - Predict Consequences (deduction) - Test - Generalize(Induction)	008
	IBM DT	Understand - Explore - Prototype - Evaluate	043 023 050 118
	HCAW	Research - Ideation - Prototyping - Evaluation	052
	Double Diamond	Discover - Define - Develop - Deliver	061 083 113 102 106 055
	Luchs	Discover - Define - Create - Evaluate	019
5	d.school	Empathy - Define - Ideate - Prototype - Test	011 014 020 054 057 076 104 099 107 116 004 028 036 038 049 050 058 065 066 067 073 075 091 096 098 101 109 124 018 022 039 105 120 126 016 035 041 042 068 082 097 106 117 127
	Meinel and Leifer	- (Re) define - Needfinding and Synthesis - Ideate - Prototype - Test	060 033 034
6	Driving board	Approach - Develop - Present and Provoke - Explore - Reflect - Escape	015
	Hasso Plattner Institute (HPI)	Understand - Observe Point of view - Ideate - Prototype - Test	063 052 059 070 073 114 123 030 122 003 029 092 053
	Nordstrom (de Paula & Araújo)	Define - Observe - Form Insights - Frame Opportunities - Brainstorm - Experiment	027
7	Hiremath & Sathyiam	Scoping - Research - Synthesis - Ideation - Prototyping - Validation - Implementation	007
	Sandino	Define - Explore - Ideate - Prototype - Select - Implement - Review	006 037

Research type: Empirical{000 Academic 000 Industry 000 Innovation} | Non-empirical - 000 Theoretical

the generated prototypes and collect feedback from users.

Souza and Silva DT model (Souza and Silva, 2014) contains 3 working spaces: Immersion, for capturing and processing users' information, Ideation, working on the ideas most referenced in the information provided by users, enabling the identification of requirements and characteristics of the proposed solution, and Prototyping, focused on transforming the requirements and characteristics into a real solution, and on verifying with the user whether requirements match the users' needs. Ferreira et al. (2015) used this set of working spaces to present Pathy, a technique for exploring empathy, and Braz et al. (2019) applies DT with Immersion, Ideation, and Prototyping working spaces integrated to Scrum, focusing on requirements elicitation activities.

Daniëls DT model (Daniëls et al., 2019) represents DT into the following working spaces: Understand, performed to empathize with users and define the problem, Explore, to ideate solutions and prototype them, and Materialize, to test and evaluate the solution created. The model was used to redesign an e-health application. Budiarto et al. (2018) applied this model to develop an application for the Indonesian government's agriculture sector.

CoDICE (Codesigning DIgital Cultural Encounters) DT model (Díaz et al., 2014) organizes Design Thinking into 3 working spaces to co-design smart objects for enhanced encounters with cultural heritage to contribute to a joint European project. CoDICE's DT model working spaces are: Resources, or Situated Resources Gathering, working space where co-designers can collect useful or inspirational material while visiting the physical environment where the digital artifact is going to be deployed; Ideas, or Divergent Inquiry and Ideation, aimed at generating ideas; and Design Products, or Convergent Design, focused on producing solutions for selected ideas.

 # DT models with 4 working spaces

Dunne and Martin (2006) suggest a DT model with 4 working spaces: Generate Ideas, Predict Consequences, Test and Generalize. It explores abductive, deductive, and inductive thinking, in addition to testing. The authors integrated DT into a framework that enhances the user experience's current design by integrating three design approaches - DT, design for the user experience, and agile software development.

IBM DT is a 4-working spaces DT model that provides a framework to write requirements, organize teams, and track project progress, including constant end-user feedback (Lucena et al., 2016). This model comprises the following working spaces: Understand, which establishes empathy with users to help to understand them and their problems' context; Explore, which focuses on the generation of new innovative ideas; Prototype, which generates artifacts intended to answer questions to solve the problem; and Evaluate, which solicits users for feedback about the prototype created.

IBM DT introduces the hill concept, creating a new way to express user needs into project requirements. Each hill articulates a clear objective and contains a defined scope to be achieved in a release and must be written to solve a specific and clearly defined user problem. The hills also describe intersections between user expectations and business requirements and consist of: "Who": Describes a specific user; "What": describes a problem that needs to be resolved; and "How": a measurable target resulting from the completion of the hill (Lucena et al., 2016).

HCAW (Human-centered Agile Workflow) model (Glomann, 2017) organizes the working spaces in cycles, where an initial Cycle Conception is proposed organized into a Research phase that starts by receiving from business personnel a problem scenario used by the team to research to understand the problem better. This understanding is recorded in an Insight Report, which is the foundation for the Ideation phase, in which a Co-Creation Workshop is hosted to generate ideas later prototyped in the Prototyping phase and evaluated in the Evaluation phase.

Double Diamond DT model organizes DT in 4 working spaces: Discover, Define, Develop and Deliver Council (1944). Challiol et al. (2019) explain that Discover and Develop are the 2 working spaces exploring divergent activities since they allow open possibilities (problem thinking and solutions development). The other 2 working spaces, Define and Deliver, explore convergent thinking, defining the problem, and selecting the suitable solution. For instance, Levy (2018) used the Double Diamond in the academic context to teach and practice empathy activities with students from 2 different courses. The authors' goal was to assess how empathy is explored in projects linked to wicked problems. Challiol et al. (2019) used this model with students as an approach aiming to teach mobile application development for the domain of co-location games. Hehn et al. (2020) discuss the Double Diamond in the industry domain proposing a framework for integrating DT and Requirements Engineering.

Luchs (2015) proposes a DT model with 4 working spaces: Discover, for capturing customer's insights, followed by Define working space when the team creates a definition of the problem based on the insights gathered previously. In the Create working space, the team develops a solution to next evaluate them with the stakeholders getting feedback in the Evaluate working space.

#DT models with 5 working spaces

D.school DT model from Stanford University (d.School Stanford, 2004) was the most cited DT model in the literature. *D.school DT* is composed by 5 working spaces: Empathize, Define, Ideate, Prototype and Test. Empathize aims to observe and view users and their behavior in the context of their lives, interacting with them and living their experiences; Define is used to synthesize the empathy findings into insights and scope a specific and meaningful challenge; Ideate relates to the generation of design and solution alternatives, while Prototype is the phase in which ideas are transformed into physical artifacts for getting feedback from users and refining solutions in the Test phase.

D.school DT model was used by Mutuku and Colaco (2012). The authors applied *D.school* DT model to create and design solutions based on insights related to government data, focusing on Open Data solutions, for the government of Kenya. Pham et al. (2018) present another example of *D.school DT* model, using it in the format of workshops with students to learn how to develop solutions and generate ideas for mobile applications development.

Meinel and Leifer DT model organizes DT with the working spaces (Keighran and Adikari, 2016): (re) Define, to establish the problem to be solved; Needfinding and Benchmarking, to understand the users and explore the design space; Brainstorm, to generate ideas; Prototype, to build prototypes of the ideas, and; Test to learn with the generated solution. This model shapes DT in a cyclic mode, e.g., the first activity (re)Define helps to restart the process seeking for improvement in the solution. For instance, Keighran and Adikari (2016) introduced this model in a study involving DT to develop strategies that satisfy a high-performance team structure. Adikari et al. (2016) used these DT working spaces to explore how the inspired co-design approach on DT could be used to generate ideas and prototyping to create design solutions agreed by the user for an intended information system.

 # DT models with 6 working spaces

Driving Board (Newman et al., 2015) represents DT as an iterative cycle organized into 6 working spaces: Develop, Present and Provoke, Explore, and Reflect. This cycle is preceded by an Approach phase, in which stakeholders and team share information about the problem context and user experiences, and concludes with Escape. This working space allows the participants to conclude that the problem has been sufficiently explored and that the list of elicited requirements indeed reflects the solution for the identified problem. The working spaces aim to define low-fidelity prototypes to give a taste of what could be used to explore the discussed ideas

(Develop); present the latest versions of prototypes to encourage further ideation through the participants' feedback (Present and Provoke); use the prototypes as boundary objects to help focus in the problem space and provide a technological scope to facilitate the further exploration of the problem (Explore), and promote the reflection upon provided feedback aiming to improve or propose new prototypes until the solution is reached (Reflect).

Hasso Plattner Institute (HPI) model organizes DT into 6 working spaces (Hasso-Plattner-Institute): Understand, when the team sets the problem space; Observe, when the team gains a consistent view and empathizes with the users and stakeholders; Define the Point of View, which serves to define the point of view and in which the knowledge gained will be collated and summarized; Ideate, when the team subsequently generates a variety of solution possibilities, then selects a focus; Prototype, which serves in the development of concrete solutions that can then be tested on the appropriate target group (Test). For instance, da Cruz Júnior and do Nascimento (2016) applied HPI DT model as a design approach to create e-learning objects. The authors argue that the integration of this model with Scrum allowed the discovery of the users' needs, and the production of a solution that was tested by the users. Luedeke et al. (2018) used the HPI model to develop a tool for the automotive industry.

Nordstrom DT model proposes an integration between DT, Lean Startup, and Agile software development exploring the following 6 working spaces: Define the challenge, when a multidisciplinary team of developers and designers gathers information about the challenge to be solved and defines the sequence of activities to seek the solution; Observe People, when the team records what stakeholders do and think; Form Insights working space is used to generate insights in order to drive the discussion of potential solutions; Frame opportunities is a working space used to create a visual representation to understand the collected data by highlighting key relationships and developing the solution strategy. Next, the Brainstorming Ideas working space allows brainstorming solution ideas and testing the chosen solution under real conditions to learn how it works in practice (Try Experiments). de Paula and Araújo (2016) used the Nordstrom DT model for the development of a mobile game.

 # DT models with 7 working spaces

Hiremath and Sathiyam (2013) introduced a DT model for the creation of innovative solutions, setting DT in 7 working spaces: Scope, which is concerned with planning customer and stakeholder interactions, project member's availability, and planning the DT activities efficiently; Research, working space for the team learning about the problem space by observing and interviewing end-users and finding their needs and motivations; Synthesis, working space used by the teams to identify connections on the data gathered from research and making statements about their understanding of the problem; Ideation, working space for ideating, sharing, voting, selecting and generating feasible ideas; Prototyping, working space used by the team to create physical ideas to represent the solution; Validation, working space used to meet the users and validate the prototyped ideas, and Presentation, the last working space used to show the produced design blueprint for the stakeholders and establishing the solution ready for production.

Sandino et al. (2013) propose a DT model for the development of interactive real-time applications. It organizes DT in 7 working spaces: Define is a working space used to the definition of a series of constraints (e.g., how faithfully a simulator must reflect reality) that will guide the subsequent work; Explore is used for the team to gather information about potential users, their needs, and previous solutions to the same problem; Ideate is a working space when the team collaboratively identifies relevant issues and generates as many ideas as possible to get answers to the identified needs. Next, Prototype working space is when the ideas are prototyped and refined through iterative discussions; Select is the working space used to choosing the candidate solution that might solve the problem; Implement is the working space used by the team to implement the candidate solution, and in Review working space the team keeps track of whether it fits the purposes that it was conceived for, once the product is introduced to the users, and identifies possible areas of improvement. Canedo and Parente da Costa (2016) applied this set of working spaces on the development of two software solutions for the Brazilian Army.

RQ3: What DT techniques are used in software development?

This RQ aims to map what DT techniques are used in software development. DT as a set of techniques represent the perspective of DT as a toolbox as proposed by Brenner et al. (2016). Table 9 shows the techniques that we found in this systematic mapping study.

Pusca and Northwood (2018) discuss that DT can be understood by two main and core moments: problem and solution moments. The former focuses on the problem identification and formulation, while the latter, the solution moment, seeks for proposing and validating solutions. The authors also argue that the problem space "is considered an analytic sequence in which the designer determines all of the elements of the problem and specifies all of the requirements and the constraints", while the solution space "consists of synthesis and analysis sequences in which several possible concepts are evaluated to find the best solution for the problem". Therefore, following the statements presented by Pusca and Northwood (2018), we organized the DT techniques according to the problem space and solution space. Also, we point out a column "Not mentioned", which lists the papers that mention DT techniques but without indicating if the technique serves the problem or the solution space.

Table 9 also lists DT techniques that we mapped in literature for both problem and solution spaces. It means that teams using DT in software development are concerned with the understanding of the problem, and also in proposing the right solution (Sedano et al., 2019).

Brainstorming, Empathy map, Interview, and Personas

Table 9
Design Thinking Techniques Used in Software Development

Techniques	Space		
	Problem Space	Solution Space	Not mentioned
AEIOU	109		
A Beginner's mind		119	
Acceptance Test		109 037 092	
Affinity Diagram	115	099	
As-is scenario map	126	109	100 111
Behaviour Map	006 115		
Behavioural Archeology	006 115		
Benchmarking			111
Bodystorming	118	115 003 122	
Brainstorming	112 115	020 043 045 051 054 057 076 088 089 099 107 116 095 002 004 007 023 075 081 115 123 015 022 039 087 105 126 016 068 092 122	024 073 079 120 026 121 055
Brainwriting		085 086 083 101 091	
Blueprint	081 087 101 018 087 115		120
Business Model Canvas	115	101	
Card sorting	007		111
Cost-benefit matrix		083	
Conceptual Map (cognitive)	115	088	024
Crazy eights		079 101	
Customer Journey Map	046 054 099 116 049 065 075 101 102 091 115 119 018 035 092	081	098 120 055
Day in the life	085		
Desk Research	088 081 092		
Dot Voting		086 107 091	
Eliminate-reduce-raise-create grid		085 002	
Empathy Map	017 045 051 083 098 050 075 081 091 109 115 118 003 080 035		111 120 055
Epic	097		
Error Analysis	006		
Ethnography	006 115 116		031 084
Expectation Test		109	
Exploratory Research	054 088 112 081 102 115 092		
Feedback matrix		054 023 050 055 091	
Field Studies	009 102		
Fishbone			100
Five Fingers		119	
Five Human Factors	060		
Five Whys	037 119		016
Fly on the wall	006 115		
Focus Group	076 114	080	031
Generative Sessions	081		
How can (might) We?	054 107 099 081 053	124	121
Ideas Menu	081		
Interview	054 076 088 089 107 023 037 065 067 099 110 116 074 075 081 091 115 124 015 003 022 039 105 126 025 097 122 123	076 107 112 123 080	014 071 073 120 121 025 117
Insight Cards	014 112 115 092		024 120
I Wish/I like feedback		057	120

Divergent Space: 000 Academic 000 Industry 000 Innovation | 000 Theoretical
Convergent Space: 000 Academic 000 Industry 000 Innovation | 000 Theoretical
Space Not Mentioned: 000 Academic 000 Industry 000 Innovation | 000 Theoretical

Table 9
Design Thinking Techniques Used in Software Development (Continuation)

Techniques	Space		
	Problem Space	Solution Space	Not mentioned
Letter to grandMa	119		
Matriz CSD			120
Mind Mapping	107 112 081 109 022	115 123	024 111 120
Motivation Matrix	115		
Now, How, Wow		091	
Observation	085 002 023 067 035 097 122	076 123 105	073 079 025
Personal Inventory	119		
Personas	017 045 046 054 083 085 086 112 116 007 009 049 059 065 074 075 091 098 101 102 115 118 123 003 018 078 126 053 122		014 031 010 071 079 113 120 026 108 121 055 117
PEST	060		
(Pitch) Presentation		085 091	120
Positioning Matrix		081	
Priorization Grid		050 118	100
Proof-of-concept		051 058 053	111
Prototyping (paper or low fi)		046 051 054 057 083 085 086 107 116 002 007 023 034 049 067 075 081 102 119 123 003 015 018 126 005 035 053 068 092	062 073 079 113 120 026 005 084
Protoyping (medium-fi)		076 102 039 105 097	025 026
Prototyping (physical/hi-fi)		054 076 085 099 110 112 090 102 114 115 123 015 039 126	062 113 026 005 025 111
Power of ten		119	
Questionnaire	088 006 023 081 115 105	076 099 116 006	
Role playing		107 091 092	073
Sailboat			100
Service Walkthrough		101	
Shadow	124 025 122		
SIPOC Review			100
Social network Mapping	006		
Stakeholder Map	050 102 115 118 018 122	081	111 071
StoryBoard	046 023 075 073	054 060 049 067 034 091 115 118	024 031 072 120 055
Storytelling	066 115 022 035 122	003	120
Survey	089 076 081 006 123	089 006	
(Systematic) Literature Review	085 080 025		
Touchpoint matrix	115		
Trends Matrix	060		
Try it yourslef	006	115	
Usability (user) test		046 076 116 023 065 102 114 126	
Use case diagram		067	
User Stories		097	010
Venn Diagrams			122
Workshop	088 089 004 081 015 092	089 004 052 100 105 097	031 025
world cafe	057		
Yes, but/ Yes and then.. game	057		
5w2h	049		
30 Second Sketch		119	
2x2 Matriz			122
6-3-5 method		123	

Divergent Space: 000 Academic 000 Industry 000 Innovation | 000 Theoretical
Convergent Space: 000 Academic 000 Industry 000 Innovation | 000 Theoretical
Space Not Mentioned: 000 Academic 000 Industry 000 Innovation | 000 Theoretical

are the most cited techniques in the search of a better problem understanding (or problem space). For proposing solutions and validate them (or solution space), the most cited techniques were Prototyping (on different fidelity levels), Usability testing, and Workshops. Prototyping is the DT technique that assumes the role of the protagonist. Although the prototypes' level of fidelity varies, they offer support to DT practitioners to visualize the ideas and learn with them to improve a solution (Dobrigkeit and de Paula, 2019). Hehn et al. (2020) claim that Prototyping helps to promote user interaction with the proposed solution. Araújo et al. (2019) advocates that prototypes are a well-known technique that helps validate the ideas created in DT activities. Alhazmi and Huang (2020) mention that prototypes allow the stakeholders to interact with the solution.

Still looking at the DT techniques, we found 19 techniques that were used in both problem and solution spaces. Such techniques are: Affinity Diagram, As-is Scenario Map, Bodystorming, Brainstorming, Business Model Canvas, Conceptual Map, Customer Journey Map, Focus Group, How Can We?, Interview, Mind Mapping, Observation, Questionnaire, Stakeholder Map, Storyboard, Storytelling, Survey, Try It Yourself, Workshop. It means that DT techniques are flexible helping the practitioners to solve the problem using the techniques according to their needs.

Also, there are well-known techniques used in software development activities, such as Epics, Use case diagrams, and User stories. Although these are not DT techniques, they were reported as adopted techniques in the use of DT in software development. This finding "blend in" techniques from DT itself and software development, like Requirement Engineering, for example (Dobrigkeit et al., 2017).

Given the DT techniques that we listed in Table 9, to choose among them may be challenging. Thus, in RQ4 we present our findings of how the literature reports the decision criteria used by practitioners for choosing DT techniques.

RQ4: What is reported about the DT techniques selection in software development?

This research question aimed to explore how literature describe the factors that contributes to the decision-making of DT techniques selection to use in software development.

We did not find any paper that explicitly discusses on how the professionals decide on which techniques to use. On the other hand, we have figured out that the DT techniques are often chosen according to the goal that the DT practitioner wants to achieve (or DT activity). Goals in this context are encapsulated by the DT working space being conducted by the teams. Table 10 lists the papers which mention that DT techniques were selected taking into account the DT working space being conducted.

We have also investigated what possible criteria are used by practitioners in selecting DT techniques. The selection criteria for the techniques we have identified are:

- Application goal: goal that the DT practitioner aims to achieve using a specific technique;
- Application time: time required to apply a technique;
- Familiarity: level of knowledge about a technique, but without use it;
- Comfortable to use it: how comfortable the practitioner feels herself to apply a technique;
- Stakeholder's information: information available about the stakeholder's and how to explore her participation on the DT activities;
- Problem information: information available about the problem and how to explore it;
- Previous experience: level of experience on the use of a technique.

Table 11 shows the criteria for selecting DT techniques that we identified in the literature and the respective papers that mention them.

Souza et al. (2020) argue that even DT specialists recommend the use of DT techniques toolkits to guide DT practitioners on the selection of DT techniques based on DT models' working spaces. The authors also mention that the selection of techniques might be made by exploring questions such as "What tools can I use to understand people?". It indicates that the Design Thinking practitioners' goals determine what techniques to use.

Dobrigkeit et al. (2020a) point out in a study with practitioners that the selection of DT techniques uses information that allow identifying which techniques require a short application time, are easy to put into practice and understand how to use, and developers feel comfortable using.

Chasanidou et al. (2015) discuss that the DT techniques help to facilitate the creation of innovative software solutions. The authors consider selecting the methods and tools that are most fit, and knowing them may improve the results generated for software engineering activities through the possibility of choice between the alternatives that involve the user more effectively in the software development process. Brenner et al. (2016) claim that the deployment of appropriate methods is a success factor of DT in projects, while Carlgren et al. (2016) argue that the use of the right techniques allows to collect the 'right' solution's requirements.

De Paula et al. (2020) consider that it is important to do an efficient pre-work on the selection of DT techniques, investigating the stakeholders in detail, taking the time to know their needs, and choosing the appropriate innovation techniques for ensuring the production of solutions meeting the user's expectations. The authors accepted an IBM's challenge to apply a DT light-version, where time constraints were decisive for selecting the techniques.

Harriet et al. (2019) propose a card-based set of DT techniques called Innodeck. Innodeck provides information about the techniques supporting the use of DT for innovation. The authors mention that DT is composed of various techniques and selecting which technique to use in the working spaces depends on the problem and the people involved

Table 10
Selection of DT techniques

Technique selection	Publications
According to the goal to be achieve (encapsulated as DT working spaces)	001 014 017 020 024 043 045 046 051 054 057 076 077 083 085 086 088 089 099 107 112 002 004 006 007 009 023 034 037 049 050 052 058 059 065 066 067 074 075 081 090 091 093 095 098 101 102 109 115 003 015 018 022 039 078 080 087 105 005 016 019 025 030 035 053 068 092 094 097

Research type: Empirical research - 000 Academic 000 Industry 000 Innovation context| Non-empirical research - 000 Theoretical

Table 11
Criteria used by Practitioners for Selecting DT Techniques

Criteria	Publications
Application goal	115
Application time	100 119
Familiarity	119 018 094
Comfort of use	119 094
Stakeholders' information	100 094
Problem information	094
Previous experience	073

Research type: Empirical 000 Academic 000 Industry 000 Innovation Non-empirical 000 Theoretical

in the project. For instance, the authors mention it is worth the professionals selecting those methods that they are familiar, comfortable with, and consider suitable for their challenges. Hehn et al. (2020) propose a framework comprising 40 artifacts to be generated using DT integrated to RE that indicates a set of techniques that can be used and when to select each one.

Dobrigkeit and de Paula (2019) investigated a global software company and concluded that the professional' experience plays an essential role in understanding DT and, consequently, as it is used or as the techniques are selected. The authors mention that the developers who participated in the study described DT as a process or toolbox because they had, in general, less DT experience than the managers or designers. On the other hand, the designers consider DT as a mindset that fosters the professionals' to problem-solving.

Also, the DT techniques usage in software development is discussed by Souza et al. (2020). The study presents an experience report addressing the challenges faced by software engineering students for selecting and using DT techniques. Initially, the authors introduced 15 DT techniques to a group of students. After the execution of a mobile application development challenge, the students used only 6 techniques. The authors reported that the students did not select more techniques due to the lack of clarity in understanding how a certain technique works and how to use the techniques into the established scenario. Then, to handle these challenges, in a second round, the authors proposed DTA4RE (*Design Thinking Assistant for Requirement Engineering)*[11], a tool to aid the selection of DT techniques. DTA4RE recommends DT techniques according to user needs, based on a question-answering form. After the second round, the students mentioned that the tool helped them to choose techniques to solve the problem they had at hand. The students selected 12 techniques of the 15 presented.

Q5: What are the key points to be aware of when using DT in software development?

DT fosters an open-minded environment for problem-solving, explores collaboration, and promotes creativity. It also aims to boost the user's participation in the software development process. However, DT is not a silver bullet and its use might establish a complex environment. For instance, De Paula et al. (2020) discuss points and counterpoints based on their experiences of using DT in the software industry. The authors indicate not only the benefits, but also the risks they figured out in projects with DT. Pereira et al. (2021) also investigated the perceived benefits, and the challenges faced by IT professionals when using DT for understanding the user's needs and proposing innovative software solutions. However, both studies do not indicate what are the key points the DT practitioners have to pay attention of when using DT in software development. Therefore, this question aims to bring some light through the map of what are the points to be aware of when using DT in software development.

Table 12 presents a set of points that DT practitioners should pay attention for using DT in software development. We categorized the key points into 4 categories, as follows: Problem and solution preconceiving, organizational and stakeholders' participation, DT techniques selection and results sharing, and Requirements Engineering integration.

The first category of attention points shows that DT practitioners have to be aware that 2 types of participants' preconceiving might difficult the adoption of DT in software development. Problem preconceiving is the first type of preconceiving (Sedano et al., 2019; Tannian, 2020). It means that the customers or even teams members may have a preformed understanding of the problem to be solved, which is known as Entrained Thinking. (Snowden and Boone, 2007) defines Entrained Thinking as a pre-established mindset that drives the team to an already conceptualized solution (lacking innovation) or even to a solution that does not solve the right problem. The second preconceiving that the Design Thinking practitioners have to pay attention to is the solution preconceiving (Sedano et al., 2019; De Paula et al., 2020). It happens when the customer presents the problem to be solved and also presents the solution she wants in advance.

[11] https://sites.google.com/site/dta4re/pagina-inicial

Table 12
Attention points for using DT in software development

Attention points	Publications
Problem and solution preconceiving	
Preconceiving problems (lack of problem understanding)	071 106
Preconceiving solutions (outdated ideas)	071 100
Organization and stakeholders participation	
Reach end users	086 007 081 114 065 071 073 097
Time pressure	086 071 100 102 114 119 029 097 106
Unavailability of resources (dedicated space for creativity, materials)	102 055 106
Lack of higher management engagement	071 073 029 106
Lack of collaboration	058 114 125
Lack of employee commitment	057 007 102 114 119 097 106
Mindset changing	007 102
Lack of knowledge in DT	051 058 125 106
DT techniques selection and results' sharing	
Select a correct combination of artifacts and their proper use	081 100 114
Share results from DT activities and ensure an effect on the final product	073
Requirement Engineering integration	
Lack of requirements traceability	065 081
Neglect of non-functional requirements	065 081
Imprecise effort estimates	065 081
Prioritization of requirements	065 081
Changing of requirements	081
Lack of documentation	054 081

Research type: Empirical 000 Academic 000 Industry 000 Innovation Non-empirical 000 Theoretical

The second category of attention points refers to the organizations and stakeholders participation in DT activities. It means that when using DT in software development, the DT practitioners have to pay attention to how the organizations commit with DT, and also to the user participation on DT activities. Literature reports that it might be difficult to reach the "proper" end-users for the problem that the team wants to solve, due several constraints such as people agenda, ethical issues, organization setup on business environment (Hiremath and Sathiyam, 2013; Sedano et al., 2019; Dobrigkeit and de Paula, 2019).

Time required for using DT techniques or conducting DT workshops is also a point to be considered on the use of DT in software development. Kongot and Pattanaik (2017) argue that "DT tools and techniques need a certain amount of time to be carried out. Customer's time is precious and often there is a need to improvise on the techniques and come up with a proposal that works given the time and audience". Alhazmi and Huang (2020) advocates that DT requires a lot of effort to address the users' needs. In addition, time pressure is provoked by customers that do not understand the design process and the value it has on the construction of an innovative and desired solution (Sedano et al., 2019).

Unavailability of resources might become a challenge that compromises the use of DT in software development. Therefore, it is an important point to be aware of before starting the use of DT (Hehn et al., 2020; Gürdür and Törngren, 2018; Levy, 2018). DT fosters a creativity environment that looks for innovative solutions, breaking paradigms and proposing solutions that might have never been thought before. However, a proper exploration of DT requires investment in dedicated spaces or materials for research, user analysis, brainstorming, ideas generation, prototyping and feedback collection.

Literature also show that for using DT in software development, a DT practitioner should be prepared to deal with lack of higher management engagement (Sedano et al., 2019; Dobrigkeit and de Paula, 2019; Gurusamy et al., 2016; Tannian, 2020), pressure to the teams to converge (Sedano et al., 2019), lack of collaboration, when the participants are not opened to give their ideas, or lack of experience in working on a multidisciplinary environment proposed by DT (Prasad et al., 2018; Kongot and Pattanaik, 2017; Jolak et al., 2020). In addition, other key points to consider when using DT are lack of employee commitment, that represents the team members may think that discussing the problem and proposing solutions is a waste of time for those working on the technical aspects of the solution Alhazmi and Huang (2020); lack of knowledge in DT, indicating that the use of DT in software development requires practice to be used properly. For example, Prasad et al. (2018) points out that organizations fail in using DT because they do not have enough theoretical knowledge in DT or in the integration of agile and DT principles. In addition, DT may be challenging to be performed because it requires a mindset changing, that is a key element on the proper exploration of DT but that might be achieved when the professionals have years of experience in DT (Dobrigkeit and de Paula, 2019).

The third category of attention points take into account when using DT refers to the selection of DT techniques and how to share the results obtained in DT activities. Using DT requires a process of decision-making for selecting what artifacts the team should elaborate on a project (Martins et al., 2019; De Paula et al., 2020; Tannian, 2020). For Tannian (2020), DT is not a cooking recipe, since it evolves several dimensions such as problem domain, team composition, level of DT expertise, cooperating users, available materials, market timing, among others. These dimensions promote a complex scenario of problem-solving. On the other hand, Dobrigkeit and de Paula (2019) argues that once a result is achieved with DT, it needs to be shared properly with the stakeholders. The authors pointed out that "results need to be shared with the whole team in a time-effective manner and are only interesting to most of the team once a certain feature is under development". Therefore, there is still room for research about how to effectively explore the decision-making of DT techniques in software development.

The last, but not less important category of key attention points that DT practitioners to consider when using DT is in regards to the integration of DT into RE activities. Researchers and DT practitioners have been considering DT as an easy-in integration approach to better perform RE. However, although there are different integration strategies between DT and RE as proposed by Hehn et al. (2020) such as Upfront DT, Infused DT or Continuous DT, when using DT the practitioners have to pay attention to the requirements traceability, to the requirements prioritization, to the requirements changing, and to the effort estimation (Martins et al., 2019; Hehn and Uebernickel, 2018). Minimal or no documentation are reasons that collaborate to fail, since DT does not foster the creation of an extensive documentation of requirements (Corral and Fronza, 2018; Martins et al., 2019; Hehn and Uebernickel, 2018). Furthermore, non-functional requirements are usually not in the spotlight when using DT, even though usability feedback are collected with end-users (Martins et al. (2019); Hehn and Uebernickel (2018)).

In Section 5 we discuss the main findings of our mapping study on the use of DT in software development.

5. Discussion

This section highlights the main findings of this Systematic Mapping Study. It also presents implications for research and practice of using DT in software development.

5.1. Main Findings

This systematic study maps the current use of DT in software development, aiming to explore the strategies that have been using for integrating it into software activities, the DT models and techniques used, what is reported about the DT techniques selection, and what are the attention points to be aware of when using DT in software development. Therefore, the main findings of our study are:

Finding 1: Design Thinking is integrated into software development through 3 integration strategies (Upfront, Infused and Continuous). Upfront is the most cited strategy.

Taking into account the integration strategies of Design Thinking in software development proposed by Hehn et al. (2020), we identified that the DT Upfront integration strategy has been the most cited strategy if compared to the Infused and Continuous strategies (73/127 papers).

This result shows that DT has been integrating into software development as a start activity to the Requirements Engineering, highlighting the search for the problem and the proposition of solutions. Therefore, development teams have been perceiving DT as a way to foster the formation of multidisciplinary teams, the search for a deep understanding of the problem, the ideation and prototyping of various possible solutions, and the collection of feedback with users before thinking about the technical requirements of the solution to be built. This perception indicates that software development teams have moved forward to understand what is needed to develop before starting the development process.

The use of the Upfront integration strategy has also shown that development teams recognize that even with the benefits resulting of the use of agile methods and the well-known Requirements Engineering techniques, there is still room about what the solution is expected. Thus, teams can put into practice the philosophy "fail fast and fail often" for developing the right solution.

DT can also be integrated into software development through the Infused DT strategy, in which DT is used to boost Requirements Engineering activities. In this strategy mentioned in 36 of the 127 papers selected, DT assists in activities already carried out by the development team to extract, analyze, and evaluate requirements. The infused strategy sets DT as a toolbox that supports the practices adopted by teams without requiring new steps in the software development process or even without forcing teams to modify the activities already performed.

The use of the Infused strategy also indicates that teams have realized the need to improve their Requirements Engineering activities. Thus, DT has been included as a mechanism that encourages exploration of the mindset of designers, promoting a deep understanding of the problem and prototyping a solution spam to then produce the solution that suits the user's needs.

DT was integrated into software development by the Continuous DT strategy in 17 of the 127 papers selected. In this strategy, DT integration exceeds the process of developing a solution and achieves levels of change in the organization's mindset. DT is seen not only by development teams as a set of techniques or as a process that foresees a set of working spaces, but it is understood as a way of thinking about disruptive, innovative solutions that meet the user's needs. Continuous DT provokes the development team, managers, and professionals who perform decision-making functions to realize that innovating is not an isolated activity but a way of acting and thinking.

Continuous DT has been discussed as a DT integration strategy in software development obtained due to DT practitioners' experience. Dobrigkeit and de Paula (2019) shows that the perception of DT as a mindset beyond a process or a toolbox is a perception of professionals who have more experience using DT or who perform management functions. On the other hand, the authors point out that for developers or professionals with less experience in DT, it is seen as a specific activity to solve a particular problem. Thus, this result indicates that companies will no longer use DT only as an upfront activity or as a means of improving what is already performed by Requirements Engineering in an infused strategy. Companies will explore DT to adapt thinking around the way they work to produce innovative solutions.

Finding 2: Design Thinking models organize DT in distinct working spaces and make it in a flexible and dynamic approach for problem-solving.

Brown and Wyatt (2010) define that the Design Thinking process is "a system of overlapping spaces rather than a sequence of orderly steps". Based on this statement, we identi-

fied a spam of DT models in this systematic mapping study. We found DT models ranging from 2 to 7 working spaces. However, despite the difference in the number of working spaces, the models use convergent and divergent thinking to understand the problem and evaluate the proposed solution.

D.school and Brown were the most DT models reported in the literature, respectively. The former organizes DT in 5 working spaces, starting from empathy, which encourages the interaction and observation of users and their behavior in the context of the problem, going to the synthesis of the problem representing the foundation of the understanding of the previous working space. Also, d.school DT model suggests ideation as a space for proposing several ideas to find proposals for innovative solutions, prototyped and evaluated by the user. These 5 working spaces lead teams to move from the problem space to the solution space.

The DT model proposed by Brown organizes the transition from the problem space to the solution space starting with the Inspiration working space, which is similar to the first two working spaces of the D.school model. The goal of it is to understand the problem according to the needs and challenges of the end-users. In the Brown's model, the ideation working space acts similarly to the ideate prototyping working spaces in d.school, where ideas and prototypes for these ideas are generated. Finally, the Implementation working space foreseen in the Brown model corresponds to the Test working space of the d.school model, in which end-users evaluate ideas.

Thus, although there are differences between DT models, DT presents a flexibility regarding working spaces, allowing the teams to generate innovative solutions. The amount of working spaces of a DT model does not limit the use of DT for understanding the user's needs, creating team engagement, exploring creativity and proposing solutions that are appropriate to the problem presented. Therefore, there is no single silver bullet DT model that must be followed as also reported by Waidelich et al. (2018), where the authors performed a literature review on DT models applied for general purposes. It is worth mentioning that there are no studies comparing different DT models regarding the working spaces used to generate innovative software solutions.

*Finding 3:*A large number of DT techniques can be applied to software development in the problem space or in the solution space. Techniques can be selected independently of the integration approach used, and the large number of techniques creates a decision-making problem.

DT has been exploited by practitioners as a toolbox (Brenner et al., 2016). In our mapping study we identified 83 techniques used in DT working spaces when integrated into software development, 51 of them for the problem exploration (problem space) and 45 techniques for the solution proposal (solution space). In addition, we have identified that there are techniques that have been using both spaces (19 techniques), reinforcing the dynamic and flexible nature of DT that explores the designers' mindset for problem-solving.

The most cited DT techniques in software development are Brainstorming, Empathy Map, Interview, Personas, Prototyping. These techniques allow the collaborative proposition of various ideas to solve the problem, to map the user's needs, to obtain data about the problem and the solution, to represent the user through a fictional character, and to sketch the solution, respectively. However, the wide variety of DT techniques gives the practitioner the option to explore them as needed and test them in conjunction with other software development practices already performed by RE or by the development method used, such as the Agile methods.

Our results also show that the DT techniques are not linked to DT integration approaches in software development. If the team wishes to integrate DT using the Upfront, the Infused, or the Continuous strategy, it can explore the full set of techniques and select those it considers suitable for the problem at hand. In addition, the flexibility provided by DT has also allowed the integration of techniques already known in Requirements Engineering, such as user stories, use cases diagrams, and epics. Thus, we can highlight that DT and ER can be complementary activities so that their techniques can be used together and thus enable the software requirements to be appropriately established.

Therefore, Design Thinking from the perspective of a toolbox helps DT practitioners as a practical way to understanding what the user's needs are, and also developing a solution that is viable and feasible. However, although the large number of DT techniques collaborates with teams in software development activities, it creates an endeavor in regard to the decision-making process for selecting the techniques. Thus, understanding how the selection of techniques is performed by professionals and researchers is important for the use of the DT in software development.

Finding 4: DT practitioners usually consider the goals they want to achieve for selecting DT techniques. Goals in this context are encapsulated as DT working spaces. Application time, familiarity, comfortable to use, stakeholder's information, problem information and previous experience are also criteria used to guide the selection of DT techniques.

Despite the importance that professionals have given to DT in software development, we did not identify any work in the literature that investigates in depth what is the decision-making process of DT techniques. Our study identified that some papers indicate that the practitioners have selected the techniques based on the working space of a selected DT model. This finding confirms that the DT models help the choice of techniques and that the perspective of DT as a process is taken into account by teams (Brenner et al., 2016).

However, it is necessary to know which criteria contribute to the decision-making of which DT techniques the practitioner will put into practice in order to help the software development. We identified that practitioners use the information they have about the techniques as criteria. This information can include the application time required by the technique, which indicates how the team should prepare itself and how much time, for example, the stakeholders should plan to participate in the activities of DT, the familiar-

ity that the practitioner has with the technique, that indicates how much the practitioner knows the technique even without having applied it before, and comfortable to use, that indicates how apt the professional feels to apply the technique in practice with the team multidisciplinary available.

Other criteria that support practitioners with the decision of which DT techniques to select involve the information that the practitioner has about the stakeholders, allowing one to explore the most of each participant, or about the problem, which enables the practitioner to choose techniques that better explore the problem and propose effective solutions to it. Previous experience is another criterion cited for selecting DT techniques, which considers the result of earlier uses of certain techniques and helps to use the technique again.

The technique selection criteria we have identified in this systematic mapping study is an initial set of indications of how practitioners decide which techniques they intend to use. However, we emphasize that there is a lack of studies investigating what are the criteria used by professionals to select the techniques that could encourage the development of solutions for helping novice professionals to establish techniques that foster the exploration of DT in software development in the most effective way.

Finding 5: DT boosts software development, but it is not a silver bullet. Although DT has been easy-in integrated to software development, there are key attention points that DT practitioners have to be aware of when using DT.

Although literature has been showing that DT brings benefits to software development, it also shows that IT professionals have to be aware of key points to extract such benefits of the DT's usage (see Table 12).

Preconceiving problems or preconceiving solutions are 2 points that need a certain attention on the use of DT in software development. DT practitioners have to handle with problems already defined or solutions pre-proposed by customers, even before starting the use of DT. A mindset changing could be a way to walk around these preconceiving issues, but it may be just achieved after a long time experiencing DT in software development. Pressure provoked by the market time or higher management also represents endeavors to be solved when using DT in software development.

DT has collaborated to improve RE activities by providing a space to do a deeper collection and understanding of the users' needs. However, reach the "right" end-users could require a big effort to the organization due time or domain constraints. In addition, the integration between RE and DT is not all a bed of roses. Literature has pointed that DT for RE might requires attention to lack of requirements traceability, to the neglecting of non-functional requirements, to the non-ability to do a proper effort estimation, and to the difficult of doing a requirements prioritization by do not consider a detailed requirements' documentation.

Literature also points that it is important to develop a creative mindset and to establish empathy by the user, as well as to move from a specific problem solution to a problem-solving mindset. Creating a multidisciplinary team composed by professionals with different backgrounds might be a way to achieve the benefits of DT. However, some companies are still resistant to bring its employees to work full-time in DT activities. Bringing the user together is yet another point to consider when using DT. In addition, is also necessary to find proper tools to support the use of DT, that will provide the proper setup to understand the real problem and propose innovative solutions.

Therefore, there is still a long way to go. Academia and industry can work together to identify which models and techniques are best fit to a certain scenario, customer profile, and application areas (e.g., problem is blurry, problem is well defined but no indication of solution is previously known, a solution already in place is no longer attending the user needs and requires reconsideration).

5.2. Implications for Research and Practice

For researchers, we believe that this systematic mapping of literature contributes by synthesizing and advancing what is known about the use of Design Thinking in software development through a reproducible research process that follows the guidelines defined for literature mapping. Furthermore, our study shows that there is room for further research on the topic, indicating that it is still possible to explore unaddressed questions such as the decision-making process of selecting DT techniques, which criteria professionals have used to do it, and what are the challenges to be handled. The challenges represent open opportunities for new researches on the integration of DT in software development.

For practitioners, this SMS provides information to know what strategies can be used to integrate DT into software development (Upfront, Infused or Continuous), helping them to guide the DT process through different DT models and working spaces, and also supporting teams to know what DT techniques are available, what criteria are considered for the selection of techniques, and what are the challenges that could be faced when applying DT in software development. Therefore, we believe that our study contributes to the state-of-the-practice by delivering information that can assist DT practitioners in software development activities.

Inspired by the results we obtained in this SMS, and considering the discussions we have presented, we list below a set of short takeaways that we extracted from Literature on the use of Design Thinking in software development.

Takeaway 1: Integrate DT with RE without forgetting that it is important to create documentation to support requirements activities. Also, try to consider non-functional requirements.

Teams using DT for boosting RE activities have to keep in mind that documentation is a key element for success. It helps to manage the requirements traceability, priorization, effort estimation and requirements' changing.

DT focuses on discover and deliver an innovative solution for a given problem. In software development, a prototyped and validated solution is proposed as input for a Lean or Agile method, that seek to develop the proposed solution. Therefore, in addition to identify the users' needs, it is also

important to start the elicitation of non-functional requirements (usability, security, privacy, etc) since the user is collaborating to the creative process.

Takeaway 2: Prepare yourself and your team for deal with preconceiving problems and solutions. Act as a ice breaker and foster the customer to think outside the box.

DT has creativity and collaboration as 2 of its core values. Therefore, ideas coming from the user should be very well received. However, try to make the user think in a disruptive way, providing a space that allows a deep understanding of the problem and the creation of solutions that had not been thought of before. Try to promote an environment that is considered food for thought.

Takeaway 3: Do a pre-work well. Try to select techniques and models that support you to deal with organizational constraints and users' engagement.

Select a DT model as your DT guide (not strict) and try to use a set of DT techniques that helps you to engage the participants (stakeholders, higher management, etc). For instance, if you do not know how to select the techniques, look for computational resources that suggest DT techniques according to other professionals' experiences. Computational resources such as DTA4RE (Souza et al., 2020; Parizi et al., 2020) and Helius support you on the selection of DT techniques for software development.

Discuss with the higher management in a Upfront approach that DT brings benefits to RE activities, to the software development process, and also to the organizational' mindset. Show that employee commitment, team collaboration and a mindset changing can represent a path to achieve better solutions that fit with the users' needs.

Takeaway 4: Share the DT's workshops results in a proper way.

In addition to have proper space and materials for conducting Design Thinking workshops, be aware that it is important to share the artifacts produced in those workshops consistently. The team will be engaged on the software development process if it has access to detailed data obtained collaboratively through processes of convergence and divergence. Therefore, define a way to ensure that all the participants have access to the DT's results (problem understanding and solution proposal).

6. Final Remarks

DT has been used as an UCD approach for involving the user in the process of development of innovative software solutions. It helps software teams to understand the users, fostering creativity and practicing both convergent and divergent thinking. Therefore, knowing how DT can be practiced in software development and what are the techniques, models and how selectors can collaborate with the improvement of software development process results.

In this article, we presented a study for characterizing and advancing what is known about the use of DT in software development. We aimed to answer the main Research Question: "*How has DT been integrated into software development, what models and techniques are used, how are DT techniques selected, and what are the key points to be aware of when using DT in software development?*"

Following the guidelines proposed by Petersen et al. (2008); Wohlin et al. (2012); Felizardo et al. (2016), we performed a reproducible Systematic Mapping Study, supplemented by a forward snowballing. We ran automatic searches in 6 digital libraries (ACM, IEEE Xplore, Science Direct, Scopus, Springer, and Wiley Online Library) resulting in a total of 3386 papers retrieved. Then, we made a process of papers' selection and obtained 127 papers.

By analyzing the selected papers, we presented and discussed that the 3 strategies for integrating DT in software development proposed by Hehn et al. (2020) (Upfront, Infused, and Continuous) are explored in the literature. Upfront is the most cited integration strategy, showing that the teams are concerned about discovering the solution to be developed before starting the development process itself. We also identified 16 models that organize DT from 2 to 7 distinct working spaces, guiding the practitioners to conduct DT activities in software development. Stanford D.School and Brown are the 2 most cited DT models in the literature.

We identified 83 techniques used in DT activities for problem understanding (problem space) and solutions proposal (solution space) by evaluating the DT techniques cited in the literature. We also identified that 19 techniques, such as Brainstorming, Workshop, Observation, and Interview, were used in both problem and solution spaces, showing the flexibility and the dynamic nature of DT.

Therefore, given the large number of DT techniques that has been using for software development motivates a decision-making problem. Considering this, we looked for the criteria used by the practitioners to select the DT techniques to be used in software development. We identified an initial set of 7 decision criteria, including application time, familiarity, comfortable to use it, stakeholder's information, problem information, and previous experience using a technique. However, this initial set represents only a first effort to identify which criteria are used to select techniques, indicating that there is research space to deepen the understanding of practitioners' decision-making to select DT techniques.

We also mapped 16 attention points that DT practitioners have to be aware of when using DT for boost software development activities. These attention points comprehends the integration with RE activities, the preconceiving for both problem and solution domains, the organizations and participants mindsets and engagement, the selection of DT techniques, and about the way of sharing the results achieved with the use of DT among team members.

Next, inspired on the findings we obtained with our SMS, we proposed 4 takeaways aiming to help those interest on the use of DT in software development to explore such approach and extract its benefits. Thus, an IT professional who aims to use DT in software development should consider the following: (1) integrate DT with RE without forget-

ting that it is important to create documentation to support requirements activities. Also, try to consider non-functional requirements; (2) prepare yourself and your team for deal with pre-conceiving problems and solutions. Act as a ice-breaker and foster the customer to think outside the box; (3) do a pre-work well. Try to select techniques and models that allow you to deal with organizational constraints and users' engagement; and (4) share the DT workshops results in a proper way. It is important to know how to explore this problem-solving approach properly for achieving success. Therefore, we believe our results represent a seed for a use more effective of DT in software development.

Our study contains limitations and threats to validity inherent to a literature review (Kitchenham and Charters, 2007; Wohlin et al., 2012). As a construct validity, our research questions may not provide complete coverage of all the papers that present DT and software development. As internal validity, related to studies selection and data extraction, two authors performed the process of paper selection suggested by Kitchenham and Charters (2007). We searched into 6 digital libraries containing the majority of the high-quality papers in software engineering. This research strategy was reviewed by 2 senior researchers. As conclusion validity, we extracted data and made discussions following the protocol we defined before starting the extraction process.

As future work, we intend to investigate the research gap about the selection criteria of DT techniques that explore the decision-making scenario of DT practitioners. We will seek to understand how DT can be exploited to collaborate with software development and produce solutions that meet the user's needs and are feasible, viable, and desirable.

Acknowledgements

We thank PUCRS BPA 2019, 2020 and 2021 projects (Programa de Bolsas Pesquisa Alunos da PUCRS/Chamada Geral 1/2019, 1/2020 and 1/2021) and CNPq PIBIC 2019/2020 project (Programa Institucional de Bolsas de Iniciação Científica do CNPq) for the undergraduate research assistant scholarships. Tayana Conte thanks CNPq (Grant 314174/2020-6) and Sabrina Marczak thanks CNPq (Grant 307177/2018-1). Rafael Parizi thanks CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil) for the financial support (Code 001).

References

Adikari, S., Keighran, H., Sarbazhosseini, H., 2016. Embed Design Thinking in Co-Design for Rapid Innovation of Design Solutions, in: Proceedings of the International Conference of Design, User Experience, and Usability, Springer, Toronto, Canada. pp. 3–14.

Adikari, S., McDonald, C., Campbell, J., 2013. Reframed Contexts: Design Thinking for Agile User Experience Design, in: Proceedings of the International Conference of Design, User Experience, and Usability, Springer, Las Vegas, USA. pp. 3–12.

Ahmad, R., Chyi, W.A., Sarlan, A., Kasbon, R., 2014. Guiding Novice Database Developers in Database Schema Creation, in: Proceedings of the Conference on e-Learning, e-Management and e-Services, IEEE, Melbourne, Australia. pp. 64–69.

Ahmed, B., Dannhauser, T., Philip, N., 2018. A Lean Design Thinking Methodology (LDTM) for Machine Learning and Modern Data Projects, in: Proceedings of the Computer Science and Electronic Engineering, IEEE, Colchester, United Kingdom. pp. 11–14.

Alhazmi, A., Huang, S., 2020. Integrating Design Thinking into Scrum Framework in the Context of Requirements Engineering Management, in: Proceedings of the International Conference on Computer Science and Software Engineering, ACM, Beijing, China. pp. 33–45.

Almeida, F.V., Canedo, E.D., d. Costa, R.P., 2019. Definition of Indicators in the Execution of Educational Projects with Design Thinking Using the Systematic Literature Review, in: Proceedings of the Frontiers in Education Conference, IEEE, Covington, USA. pp. 1–9.

Araújo, C.M.M.S., Miranda Santos, I., Dias Canedo, E., Favacho de Araújo, A.P., 2019. Design Thinking Versus Design Sprint: A Comparative Study, in: Proceedings of the International Conference on Design, User Experience, and Usability, Springer, Orlando, USA. pp. 291–306.

Arayalert, S.C., Puttinaovarat, S., 2020. Designing Mangrove Ecology Self-Learning Application Based on a Micro-Learning Approach. International Journal of Emerging Technologies in Learning 15, 29–41.

Araújo, R., Anjos, E., Silva, D.R., 2015. Trends in the Use of Design Thinking for Embedded Systems, in: Proceedings of the International Conference on Computational Science and Its Applications, IEEE, Banff, Canada. pp. 82–86.

Arbieto-Batallanos, C., Villanueva-Montoya, L., Chavez-Ponce, D., Alfonte-Zapana, R., Córdova-Martínez, M., 2019. Mobile Application Based on Design Thinking for Teaching Kinematics, in: Proceedings of the International Congress on Educational and Technology in Sciences, CEUR-WS, Arequipa, Peru. pp. 257–266.

Asante, G., 2018. Effective Design Methodologies. Design Management Review 29, 10–15.

Avalos, M., Larios, V.M., Salazar, P., Maciel, R., 2017. Hackathons, Semesterathons, and Summerathons as Vehicles to Develop Smart City Local Talent that via heir Innovations, in: Proceedings of the International Smart Cities Conference, IEEE, Wuxi, China. pp. 1–6.

Azab, A., Mostafa, N., Park, J., 2016. Ontimecargo: a Smart Transportation System Development in Logistics Management by a Design Thinking Approach, in: Proceedings of the Pacific Asia Conference on Information Systems, Chiayi, Taiwan. pp. 1–44.

Bargh, M.S., Choenni, S., 2019. Towards Applying Design-Thinking for Designing Privacy-Protecting Information Systems, in: Proceedings of the International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, IEEE, Los Angeles, USA. pp. 196–202.

Berger, A., 2011. Design Thinking for Search User Interface Design. Central Europe Workshop Proceedings 763, 1–4.

Bordin, S., De Angeli, A., 2016. Communication Breakdowns in the Integration of User-centred Design and Agile Development. Springer. pp. 137–161.

Brad, S., Brad, E., Homorodean, D., 2019. CALDET: A TRIZ-Driven Integrated Software Development Methodology, in: Proceedings of the International TRIZ Future Conference, Springer, Marrakesh, Morocco. pp. 400–416.

Braz, R.d.S., Merlin, J.R., Freitas Guilhermino Trindade, D., Eduardo Ribeiro, C., Sgarbi, E.M., Junior, F.d.S., 2019. Design Thinking and Scrum in Software Requirements Elicitation: A Case Study, in: Proceedings of the International Conference on Human-Computer Interaction, Springer, Orlando, USA. pp. 179–194.

Brenner, W., Uebernickel, F., Abrell, T., 2016. Design Thinking as Mindset, Process, and Toolbox. Springer. pp. 3–21.

Brown, T., 2008. Design Thinking. Harvard Business Review 86, 84–95.

Brown, T., Wyatt, J., 2010. Design Thinking for Social Innovation. Development Outreach 12, 29–43.

Buchanan, R., 1992. Wicked Problems in Design Thinking. Design Issues 8, 5–21. URL: http://www.jstor.org/stable/1511637.

Budiarto, A., Kacamarga, M.F., Suparyanto, T., Purnamasari, S., Caraka, R.E., Muljo, H.H., Pardamean, B., 2018. SMARTD Web-Based Monitoring and Evaluation System, in: Proceedings of the Indonesian Association for Pattern Recognition International Conference, IEEE, Jakarta, Indonesia. pp. 172–176.

Burchardt, C., Maisch, B., 2018. Advanced Agile Approaches to Improve Engineering Activities. Procedia Manufacturing 25, 202–212.

Canedo, E.D., Parente da Costa, R., 2016. The Use of Design Thinking in Agile Software Requirements Survey: A Case Study, in: Proceedings of the International Conference of Design, User Experience, and Usability, Springer, Las Vegas, USA. pp. 642–657.

Canedo, E.D., Pergentino, A.C.D.S., Calazans, A.T.S., Almeida, F.V., Costa, P.H.T., Lima, F., 2020. Design Thinking Use in Agile Software Projects: Software Developers' Perception, in: Proceedings of the International Conference on Enterprise Information Systems, Scitepress, Online. pp. 217–224.

Carell, A., Lauenroth, K., Platz, D., 2018. Using Design Thinking for Requirements Engineering in the Context of Digitalization and Digital Transformation: A Motivation and an Experience Report. Springer. pp. 107–120.

Carlgren, L., Rauth, I., Elmquist, M., 2016. Framing Design Thinking: The Concept in Idea and Enactment. Creativity and Innovation Management 25, 38–57.

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

de Carvalho Souza, C.L., Silva, C., 2015. An Experimental Study of the Use of Design Thinking as a Requirements Elicitation Approach for Mobile Learning Environments. CLEI Electronic Journal 18, 1 – 6.

Challiol, C., Borrelli, F.M., Mendiburu, F.I., Rouaux Servat, C.M., Goin Plexevi, F., Orellano, D.H., Gomez-Torres, E., Gordillo, S.E., 2019. Design Thinking's Resources for in-situ Co-Design of Mobile Games, in: Proceedings of the International Conference on Information Systems and Computer Science, IEEE, Quito, Ecuador. pp. 339–345.

Chasanidou, D., Gasparini, A.A., Lee, E., 2015. Design Thinking Methods and Tools for Innovation, in: Proc. of the Design, User Experience, and Usability: Design Discourse, Springer, Los Angeles, USA. pp. 1–12.

Chongwatpol, J., 2020. Operationalizing Design Thinking in Business Intelligence and Analytics Projects. Decision Sciences Journal of Innovative Education 18, 409–434.

Corral, L., Fronza, I., 2018. Design Thinking and Agile Practices for Software Engineering: An Opportunity for Innovation, in: Proceedings of the Special Interest Group on Conference on Information Technology Education, ACM, Fort Lauderdale, USA. pp. 1–6.

Council, D., 1944. The design process: What is the double diamond? URL: designcouncil.org.uk/.

Coutinho, E.F., Gomes, G.A.M., José, M.A., 2016. Applying design thinking in disciplines of systems development, in: Proceedings of the Euro American Conference on Telematics and Information Systems, IEEE, Cartagena, Colombia. pp. 1–8.

da Cruz Júnior, G.G., do Nascimento, R.L.S., 2016. Desenvolvendo um Objeto de Aprendizagem Virtual para o Ensino da Programação Web Mobile com o Design Thinking, in: Anais do Congresso Regional sobre Tecnologias na Educação, CEUR-WS, Natal, Brasil. pp. 1–12.

Cruzes, D.S., Dybå, T., 2010. Synthesizing Evidence in Software Engineering Research, in: Proceedings of the International Symposium on Empirical Software Engineering and Measurement, ACM, Bolzano, Italy.

Daniëls, N.E.M., Hochstenbach, L.M.J., van Bokhoven, M.A., Beurskens, A.J.H.M., Delespaul, P.A.E.G., 2019. Implementing Experience Sampling Technology for Functional Analysis in Family Medicine – A Design Thinking Approach. Frontiers in Psychology 10, 2782.

Darrin, M.A.G., Devereux, W.S., 2017. The Agile Manifesto, Design Thinking and Systems Engineering, in: Proceedings of the International Systems Conference, IEEE, Quebec, Canada. pp. 1–5.

De, S., 2020. A Novel Perspective to Threat Modelling using Design Thinking and Agile Principles, in: Proceedings of the International Conference on Parallel, Distributed and Grid Computing, IEEE, Waknaghat, India. pp. 31–35.

De, S., Vijayakumaran, V., 2019. A Brief Study on Enhancing Quality of Enterprise Applications using Design Thinking. International Journal of Education and Management Engineering 9, 1–26.

De Paula, T.R., Santana Amancio, T., Nonato Flores, J.A., 2020. Design Thinking in Industry. IEEE Software 37, 49–51.

Diaz Intal, G.L., Senoro, D., Palaoag, T., 2020. User Experience Design for Disaster Management Mobile Application Using Design Thinking Approach. ACM, Osaka, Japan. pp. 7–13.

Dobrigkeit, F., Pajak, P., de Paula, D., Uflacker, M., 2020a. DT@IT Toolbox: Design Thinking Tools to Support Everyday Software Development. Springer. pp. 201–227.

Dobrigkeit, F., de Paula, D., 2019. Design Thinking in Practice: Understanding Manifestations of Design Thinking in Software Engineering, in: Proceedings of the European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, Tallinn, Estonia. pp. 1059–1069.

Dobrigkeit, F., de Paula, D., Carroll, N., 2020b. InnoDev Workshop: A One Day Introduction to Combining Design Thinking, Lean Startup and Agile Software Development, in: Proc. of the Conf. on Software Engineering Education and Training, IEEE, Munich, Germany. pp. 1–10.

Dobrigkeit, F., de Paula, D., Uflacker, M., 2019. InnoDev: a Software Development Methodology Integrating Design Thinking, Scrum and Lean Startup, in: Design Thinking Research. Springer, pp. 199–227.

Dobrigkeit, F., de Paula, D., et al., 2017. The Best of Three Worlds- The Creation of InnoDev a Software Development Approach that Integrates Design Thinking, Scrum and Lean Startup, in: Proceedings of the International Conference on Engineering Design, Design Society, Vancouver, Canada. pp. 1–10.

Dobrigkeit, F., Wilson, M., Nicolai, C., et al., 2018. Adding Scrum-style Project Management to an Advanced Design Thinking Class, in: Proc. of the NordDesign, Design Society, Linköping, Sweden. pp. 1–12.

Dunne, D., Martin, R., 2006. Design Thinking and How it will Change Management Education: An Interview and Discussion. Academy of Management Learning & Education 5, 512–523.

Díaz, P., Aedo, I., Cubas, J., 2014. CoDICE: Balancing Software Engineering and Creativity in the Co-design of Digital Encounters with Cultural Heritage, in: Proceedings of the Workshop on Advanced Visual Interfaces, ACM, Como, Italy. pp. 1–4.

El-Sharkawy, S., Schmid, K., 2011. A Heuristic Approach for Supporting Product Innovation in Requirements Engineering: A Controlled Experiment, in: Proc. of the Int'l Conference on Requirements Engineering: Foundation for Software Quality, Springer, Essen, Germany. pp. 1–16.

Felizardo, K.R., Mendes, E., Kalinowski, M., Souza, É.F., Vijaykumar, N.L., 2016. Using Forward Snowballing to Update Systematic Reviews in Software Engineering, in: Proceedings of the International Symposium on Empirical Software Engineering and Measurement, ACM/IEEE, Ciudad Real, Spain. pp. 1–6.

Ferreira, B., Conte, T., Barbosa, S.D.J., 2015. Eliciting Requirements Using Personas and Empathy Map to Enhance the User Experience, in: Proceedings of the Brazilian Symposium on Software Engineering, IEEE, Belo Horizonte, Brazil. pp. 1–10.

Ferreira, B., Silva, W., Barbosa, S.D., Conte, T., 2018. Technique for Representing Requirements using Personas: a Controlled Experiment. IET Software 12, 280–290.

Freitas, R., Peres, S., Fantinato, M., Steinbeck, R., Araújo, U., et al., 2013. Experimenting with Design Thinking in Requirements Refinement for a Learning Management System, in: Proceedings of the Brazilian Symposium on Information Systems, SBC, João Pessoa, Brazil. pp. 182–193.

Gama, K., Alencar, B., Calegario, F., Neves, A., Alessio, P., 2018a. A Hackathon Methodology for Undergraduate Course Projects, in: Proceedings of the Frontiers in Education Conference, IEEE, San Jose, USA. pp. 1–9.

Gama, K., Castor, F., Alessio, P., Neves, A., Araújo, C., Formiga, R., Soares-Neto, F., Oliveira, H., 2018b. Combining Challenge-Based Learning and Design Thinking to Teach Mobile App Development, in: Proceedings of the Frontiers in Education Conference, IEEE, San Jose, USA. pp. 1–5.

Gamble, M.T., 2016. Can Metamodels Link Development to Design Intent?, in: Proceedings of the International Workshop on Bringing Architectural Design Thinking Into Developers Daily Activities, ACM, Austin, USA. p. 14–17.

Genzorova, T., Corejova, T., Stalmasekova, N., 2019. How Digital Trans-

formation can Influence Business Model, Case Study for Transport Industry. Transportation Research Procedia 40, 1053 – 1058.

Glomann, L., 2017. Introducing 'Human-Centered Agile Workflow' (HCAW) – An Agile Conception and Development Process Model, in: Proc. of the International Conference on Applied Human Factors and Ergonomics, Springer, Los Angeles, United States. pp. 1–10.

González, C.S.G., Fleitas, Y.d.C.B., 2015. Promoting creativity and innovative thinking in software engineering teaching: a case study, in: Proceedings of the IEEE Global Engineering Education Conference, IEEE, Tallinn, Estonia. pp. 989–995.

González, C.S.G., González, E.G., Cruz, V.M., Saavedra, J.S., 2010. Integrating the Design Thinking into the UCD's Methodology, in: Proc. of the Engineering Education Conference, IEEE, Madrid, Spain. pp. 1–4.

Goodspeed, R., Riseng, C., Wehrly, K., Yin, W., Mason, L., Schoenfeldt, B., 2016. Applying Design Thinking Methods to Ecosystem Management Tools: Creating the Great Lakes Aquatic Habitat Explorer. Marine Policy 69, 134–145.

Goudsouzian, L.K., Riola, P., Ruggles, K., Gupta, P., Mondoux, M.A., 2018. Integrating Cell and Molecular Biology Concepts: Comparing Learning Gains and Self-efficacy in Corresponding Live and Virtual Undergraduate Laboratory Experiences. Biochemistry and Molecular Biology Education 46, 361–372.

Guinan, P.J., Parise, S., Langowitz, N., 2019. Creating an Innovative Digital Project Team: Levers to Enable Digital Transformation. Business Horizons 62, 717 – 727.

Gürdür, D., Törngren, M., 2018. Visual Analytics for Cyber-physical Systems Development: Blending Design Thinking and Systems Thinking, in: Proceedings of the NordDesign Conference, Design Society, Linköping, Sweden. pp. 1–15.

Gurusamy, K., Srinivasaraghavan, N., Adikari, S., 2016. An Integrated Framework for Design Thinking and Agile Methods for Digital Transformation, in: Proceedings of the International Conference of Design, User Experience, and Usability, Springer, Toronto, Canada. pp. 1–9.

Harriet, K., Monika, K., Verena, P., 2019. InnoDeck: Card based Innovation Support-A Modular Human-Centered Approach to Facilitate Innovation Workshops, in: Proceedings of the International Joint Conference on Knowledge Discovery,Knowledge Engineering and Knowledge Management, SciTePress, Vienna,Austria. pp. 83–91.

Hasso-Plattner-Institute, D.S., . Hasso-plattner-institute: School of design thinking. URL: https://hpi.de/en/school-of-design-thinking/design-thinking.html.

Hehn, J., Mendez, D., Uebernickel, F., Brenner, W., Broy, M., 2020. On Integrating Design Thinking for Human-Centered Requirements Engineering. IEEE Software 37, 25–31.

Hehn, J., Uebernickel, F., 2018. The Use of Design Thinking for Requirements Engineering: An Ongoing Case Study in the Field of Innovative Software-Intensive Systems, in: Proceedings of the International Requirements Engineering Conference, IEEE, Banff, Canada. pp. 1–6.

Hehn, J., Uebernickel, F., Stöckli, E., Brenner, W., 2018. Designing Human-Centric Information Systems: Towards an Understanding of Challenges in Specifying Requirements within Design Thinking Projects, in: Proceedings Of The Multikonferenz Wirtschaftsinformatik, Leuphana University of Lüneburg, Luneburg, Germany. pp. 1–12.

Hiremath, M., Sathiyam, V., 2013. Fast Train to DT: a Practical Guide to Coach Design Thinking in Software Industry, in: Proceedings of the International Federation for Information Processing, Springer, Cape Town, South Africa. pp. 1–8.

Husaria, A., Guerreiro, S., 2020. Requirement Engineering and the Role of Design Thinking, in: Proceedings of the International Conference on Enterprise Information Systems, Scitepress, Online. pp. 353–359.

Jensen, M.B., Lozano, F., Steinert, M., 2016. The origins of Design Thinking and the Relevance in Software Innovations, in: Proceedings of the International Conference on Product-Focused Software Process Improvement, Springer, Trondheim, Norway. pp. 1–4.

Jolak, R., Wortmann, A., Liebel, G., Umuhoza, E., Chaudron, M.R.V., 2020. The Design Thinking of Co-Located vs. Distributed Software Developers: Distance Strikes Again!, in: Proceedings of the International Conference on Global Software Engineering, ACM, Seoul, Republic of Korea. p. 106–116.

Kah-Hoe, Wang, S.M., 2014. Design Thinking for Usability Evaluation of Cloud Platform Service, in: Proc. of the Int'l Conf. on Applied System Invention, IEEE, Chiba, Japan. pp. 247–250.

Kahan, E., Genero, M., Oliveros, A., 2019. Challenges in Requirement Engineering: Could Design Thinking Help?, in: Proceedings of the International Conference on the Quality of Information and Communications Technology, Springer, Ciudad Real, Spain. pp. 79–86.

Kawano, A., Motoyama, Y., Aoyama, M., 2019. A LX (Learner EXperience)-Based Evaluation Method of the Education and Training Programs for Professional Software Engineers, in: Proceedings of the International Conference on Information and Education Technology, ACM, Aizu-Wakamatsu, Japan. pp. 151–159.

Keighran, H., Adikari, S., 2016. Developing High-Performing Teams: A Design Thinking Led Approach, in: Proceedings of the International Conference of Design, User Experience, and Usability, Springer, Toronto, Canada. pp. 1–12.

Kitchenham, B., Charters, S., 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Software Engineering Group Department of Computer Science Keele University 2, 512–523.

Kloeckner, A.P., Libânio, C.d.S., Ribeiro, J.L.D., et al., 2017. Design Thinking Methods and Techniques in Design Education, in: Proceedings of the International Conference on Engineering and Product Design Education, Building Community: Design Education for a Sustainable Future, IED, Oslo, Norway. pp. 1–6.

Kongot, A., Pattanaik, M., 2017. Empowering Project Managers in Enterprises - A Design Thinking Approach to Manage Commercial Projects, in: Proceedings of the Conference on Human Computer Interactions, Springer, Mumbai, India. pp. 189–197.

Kuula, S., Haapasalo, H., Kosonen, J., 2020. Three Phases of Transforming a Project-Based IT Company Into a Lean and Design-Led Digital Service Provider. IEEE Software 37, 41–48.

Landis, J.R., Koch, G.G., 1977. The Measurement of Observer Agreement for Categorical Data. Biometrics 33, 159–174.

Larsen, L.J., 2018. Juicing the Game Design Process: Towards a Content Centric Framework for Understanding and Teaching Game Design in Higher Education. Educational Media International 55, 231–254.

Lee, W.S., Sohn, S.Y., 2019. Discovering Emerging Business Ideas Based on Crowdfunded Software Projects. Decision Support Systems 116, 102–113.

Levy, M., 2017. Promoting the Elicitation of Usability and Accessibility Requirements in Design Thinking: Using a Designed Object as a Boundary Object, in: Proceedings of the International Requirements Engineering Conference Workshops, IEEE, Lisbon, Portugal. pp. 1–4.

Levy, M., 2018. Educating for Empathy in Software Engineering Course, in: Proc. of the Int'l Conference on Requirements Engineering: Foundation for Software Quality, CEUR-WS, Utrecht, Netherlands. pp. 1–9.

Levy, M., Hadar, I., 2018. The Importance of Empathy for Analyzing Privacy Requirements, in: Proceedings of the International Workshop on Evolving Security & Privacy Requirements Engineering, IEEE, Banff, Canada. pp. 1–5.

Levy, M., Huli, C., 2019. Design Thinking in a Nutshell for Eliciting Requirements of a Business Process: A Case Study of a Design Thinking Workshop, in: Proceedings of the International Requirements Engineering Conference, IEEE, Jeju Island, South Korea. pp. 351–356.

Liikkanen, L.A., Kilpiö, H., Svan, L., Hiltunen, M., 2014. Lean UX: the Next Generation of User-centered Agile Development?, in: Proceedings of the Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational, ACM, Helsinki, Finland. pp. 1–6.

Lucena, P., Braz, A., Chicoria, A., Tizzei, L., 2016. IBM Design Thinking Software Development Framework, in: Proceedings of the Brazilian Workshop on Agile Methods, Springer, Curitiba, Brazil. pp. 1–12.

Luchs, M.G., 2015. A Brief Introduction to Design Thinking. John Wiley & Sons. chapter 1. pp. 1–12.

Luedeke, T.F., Köhler, C., Conrad, J., Grashiller, M., Sailer, A., Vielhaber, M., et al., 2018. CPM/PDD in the Context of Design Thinking and Agile Development of Cyber-Physical Systems, in: Proceedings of the NordDesign, Design Society, Linköping, Sweden. pp. 1–11.

Magare, A., Lamin, M., Chakrabarti, P., 2020. Inherent Mapping Analysis of Agile Development Methodology Through Design Thinking, in: Proceedings of International Conference on Data Science and Intelligent Applications, Springer, Gujarat, India. pp. 527–534.

Mahe, N., Adams, B., Marsan, J., Templier, M., Bissonnette, S., 2020. Migrating a Software Factory to Design Thinking: Paying Attention to People and Mind-Sets. IEEE Software 37, 32–40.

Malins, J., Maciver, F., 2016. From the Real to the Virtual: Developing Improved Software Using Design Thinking. Springer. chapter 16. pp. 337–349.

Martins, H.F., Carvalho de Oliveira Junior, A., Dias Canedo, E., Dias Kosloski, R.A., Ávila Paldês, R., Costa Oliveira, E., 2019. Design Thinking: Challenges for Software Requirements Elicitation. Information 10, 1–27.

Matz, A., Germanakos, P., 2016. Increasing the Quality of Use Case Definition Through a Design Thinking Collaborative Method and an Alternative Hybrid Documentation Style, in: Proceedings of the International Conference on Learning and Collaboration Technologies, Springer, Toronto, Canada. pp. 1–12.

McMahon, E., 2014. From Product Development to Innovation, in: Proc. of the International Conference of the American Society for Engineering Management, ASEM, Virginia Beach, USA. pp. 1–11.

Murugesan, L.K., Hoda, R., Salcic, Z., 2017. Identifying Design Features Using Combination of Requirements Elicitation Techniques, in: Proceedings of the International Workshop on Design and Innovation in Software Engineering, IEEE, Buenos Aires, Argentina. pp. 1–7.

Mutuku, L.N., Colaco, J., 2012. Increasing Kenyan Open Data Consumption: A Design Thinking Approach, in: Proc. of the International Conference on Theory and Practice of Electronic Governance, ACM, New York, USA. pp. 1–4.

Nalepa, G., Fontana, R.M., Reinehr, S., Malucelli, A., 2019. Using Agile Approaches to Drive Software Process Improvement Initiatives, in: Proceedings of the European Conference on Software Process Improvement, Springer, Edinburgh, UK. pp. 495–506.

Nedeltcheva, G., Shoikova, E., 2018. Innovation through Design Thinking, User Experience and Agile: Towards Cooperation Framework, in: Proceedings of the International Conference Telecom, NSTC, Sofia, Bulgaria. pp. 42–49.

Nedeltcheva, G.N., Shoikova, E., 2017. Coupling Design Thinking, User Experience Design and Agile: Towards Cooperation Framework, in: Proceedings of the International Conference on Big Data and Internet of Thing, ACM, London, England. pp. 1–5.

Newman, P., Ferrario, M.A., Simm, W., Forshaw, S., Friday, A., Whittle, J., 2015. The Role of Design Thinking and Physical Prototyping in Social Software Engineering, in: Proceedings of the International Conference on Software Engineering, IEEE, Florence, Italy. pp. 1–9.

Ortiz-Rojas, M., Maya, R., Jimenez, A., Hilliger, I., Chiluiza, K., 2019. A Step by Step Methodology for Software Design of a Learning Analytics Tool in Latin America: A Case Study in Ecuador, in: Proceedings of the Latin American Conference on Learning Technologies, San Jose Del Cabo, Mexico. pp. 116–122.

Palacin-Silva, M., Khakurel, J., Happonen, A., Hynninen, T., Porras, J., 2017. Infusing Design Thinking Into a Software Engineering Capstone Course, in: Proceedings of the Conference on Software Engineering Education and Training, IEEE, Savannah, USA. pp. 1–10.

Parizi, R., Moreira da Silva, M., de Souza Couto, I., Pavin Trindade, K., Plautz, M., Marczak, S., Conte, T., Candello, H., 2020. Design Thinking in Software Requirements: What Techniques to Use? A Proposal for a Recommendation Tool, in: Proceedings of the Ibero-American Conference-American on Software-American Engineering, Curran Associates, Curitiba, Brazil. pp. 1–14.

Park, J., Mostafa, N.A., Han, H.J., 2020. "StoryWeb": A Storytelling-based Knowledge-sharing Application Among Multiple Stakeholders. Creativity and Innovation Management 29, 224–236.

de Paula, D.F., Araújo, C.C., 2016. Pet Empires: Combining Design Thinking, Lean Startup and Agile to Learn from Failure and Develop a Successful Game in an Undergraduate Environment, in: Proceedings of the International Conference on Human-Computer Interaction, Springer, Toronto, Canada. pp. 1–5.

Pendse, A., Amre, H., 2018. Software 4.0: 'How' of Building 'Next-Gen' Systems, in: Proceedings of the Innovations in Software Engineering Conference, ACM, Hyderabad, India.

Penzenstadler, B., 2020. When Does Design Help Thinking, and When Does Design Thinking Help? IEEE Software 37, 6–9.

Pereira, J.C., Russo, R.d.F., 2018. Design Thinking Integrated in Agile Software Development: A Systematic Literature Review. Procedia Computer Science 138, 775–782.

Pereira, L., Parizi, R., Prestes, M., Marczak, S., Conte, T., 2021. Towards an Understanding of Benefits and Challenges in the Use of Design Thinking in Requirements Engineering, in: Proceedings of the Annual ACM Symposium on Applied Computing, ACM, Virtual Event, Republic of Korea. p. 1338–1345.

Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic Mapping Studies in Software Engineering, in: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering, ACM, Bari, Italy. pp. 1–10.

Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update. Information and Software Technology 64, 1–18.

Pham, Y.D., Fucci, D., Maalej, W., 2018. A First Implementation of a Design Thinking Workshop during a Mobile App Development Course Project, in: Proceedings of the International Workshop on Software Engineering Education for Millennials, IEEE, Gothenburg, Sweden. p. 8.

Piras, L., Dellagiacoma, D., Perini, A., Susi, A., Giorgini, P., Mylopoulos, J., 2019. Design Thinking and Acceptance Requirements for Designing Gamified Software, in: Proc. of the International Conference on Research Challenges in Information Science, Brussels, Belgium. pp. 1–12.

Prasad, W.R., Perera, G., Padmini, K.J., Bandara, H.D., 2018. Adopting Design Thinking Practices to Satisfy Customer Expectations in Agile Practices: A Case from Sri Lankan Software Development Industry, in: Proceedings of the Moratuwa Engineering Research Conference, IEEE, Moratuwa, Sri Lanka. pp. 1–6.

Puad, N.H.M., Rahim, N.A., Firdaus, K.H.A., Sayedi, N., Mohadis, H.M., 2019. Designing a Persuasive Application for Behaviour Change with Children. Journal of Marketing and Information Systems 1, 1–7.

Pusca, D., Northwood, D.O., 2018. Design Thinking and its Application to Problem Solving. Global Journal of Engineering Education 20, 1–3.

Péraire, C., 2019. Dual-Track Agile in Software Engineering Education, in: Proceedings of the International Conference on Software Engineering, ACM/IEEE, Montreal, Canada. pp. 38–49.

Queiros, L.M., Da Silveira, D.S., da Silva Correia-Neto, J., Vilar, G., 2016. LODPRO: Learning Objects Development Process. Journal of the Brazilian Computer Society 22, 3.

Rajashekharaiah, K.M.M., Pawar, M., Patil, M.S., Kulenavar, N., Joshi, G., 2016. Design Thinking Framework to Enhance Object Oriented Design and Problem Analysis Skill in Java Programming Laboratory: An Experience, in: Proceedings of the International Conference on MOOCs, Innovation and Technology in Education, IEEE, Madurai, India. pp. 1–6.

Reddy, P.D., Iyer, S., Sasikumar, M., 2017. FATHOM: TEL Environment to Develop Divergent and Convergent Thinking Skills in Software Design, in: Proceedings of the International Conference on Advanced Learning Technologies, IEEE, Timisoara, Romania. pp. 1–5.

Rhinow, H., Meinel, C., 2014. Design Thinking: Expectations from a Management Perspective. Springer. pp. 239–252.

Rösel, A., 2016. Are We Ready for Disruptive Improvement?. Springer. chapter 5. pp. 77–91.

Sandino, D., Matey, L.M., Vélez, G., 2013. Design Thinking Methodology for the Design of Interactive Real-time Applications, in: Proceedings of the International Conference of Design, User Experience, and Usability, Springer, Las Vegas, USA. pp. 1–10.

Santos, H.R.M., Alves, C., 2017. A2BP: A Method for Ambidextrous Analysis of Business Process, in: Proc. of the International Conference on Enterprise Information Systems, SCITEPRESS, Porto, Portugal. p. 12.

Sarbazhosseini, H., Adikari, S., Keighran, H., 2016. Design Thinking Framework for Project Portfolio Management, in: Proceedings of the International Conference of Design, User Experience, and Usability,

Springer, Toronto, Canada. pp. 1–8.
Sedano, T., Ralph, P., Péraire, C., 2019. The Product Backlog, in: Proceedings of the International Conference on Software Engineering, ACM/IEEE, Montreal, Canada. pp. 200–211.
Senapathi, M., Drury-Grogan, M.L., 2021. Systems Thinking Approach to Implementing Kanban: A case study. Journal of Software: Evolution and Process 33, 22–32.
Senft, B., Rittmeier, F., Fischer, H., Oberthür, S., 2019. A Value-Centered Approach for Unique and Novel Software Applications, in: Proceedings of the International Conference on Human-Computer Interaction, Springer, Orlando, USA. pp. 366–384.
Snowden, D.J., Boone, M., 2007. A Leader's Framework for Decision Making. Harvard Business Review , 1–8.
Sohaib, O., Solanki, H., Dhaliwa, N., Hussain, W., Asif, M., 2019. Integrating Design Thinking into Extreme Programming. Journal of Ambient Intelligence and Humanized Computing 10, 2485–2492.
Sosnin, P., 2017. Question-answer Analysis in Design Thinking at the Conceptual Stage of Developing a System with a Software, in: Proceedings of the International Conference on Information, Intelligence, Systems Applications, IEEE, Larnaca, Cyprus. pp. 1–6.
Souza, A., Ferreira, B., Valentim, N., Correa, L., Marczak, S., Conte, T., 2020. Supporting the Teaching of Design Thinking Techniques for Requirements Elicitation Through a Recommendation Tool. IET Software 14, 693–701.
Souza, A.F., Ferreira, B.M., Conte, T., 2017. Applying Design Thinking in Software Engineering: A Systematic Mapping, in: Proceedings of the Ibero-American Conference on Software Engineering, Curran Associates, Buenos Aires, Argentina. pp. 1–14.
Souza, C.L.d.C., Silva, C.T., 2014. Use of Design Thinking in Requirements Elicitation of Mobile Learning in Virtual Environments, in: Proceedings of the Workshop on Requirements Engineering, PUC-Rio, Pucón, Chile. pp. 1–14.
d.School Stanford, 2004. Design Thinking - Bootcamp Bootleg Compilation. URL: https://dschool.stanford.edu/resources/the-bootcamp-bootleg. (accessed October 20, 2020).
Suzianti, A., Wulandari, A.D., Yusuf, A.H., Belahakki, A., Monika, F., 2020. Design Thinking Approach for Mobile Application Design of Disaster Mitigation Management, in: Proceedings of the Asia Pacific Information Technology Conference, ACM, Bali Island, Indonesia. p. 29–33.
Tannian, M.F., 2020. Embracing Quality with Design Thinking. The Future of Software Quality Assurance , 161–174.
Tellioğlu, H., 2016. Models as Bridges from Design Thinking to Engineering, in: Proceedings of the International Conference on Interfaces and Human Computer Interaction, IADIS, Madeira, Portugal. pp. 1–4.
Unnikrishnan, R., Hebert, M., 2005. Measures of similarity, in: Proceedings of the Workshops on Applications of Computer Vision, IEEE, Washington D.C., USA. p. 1.
Valentim, N., Silva, W., Conte, T., 2017. The Students' Perspectives on Applying Design Thinking for the Design of Mobile Applications, in: Proceedings of the International Conference on Software Engineering: Software Engineering Education and Training Track, IEEE, Buenos Aires, Argentina. pp. 1–10.
Vetterli, C., Brenner, W., Uebernickel, F., Petrie, C., 2013. From Palaces to Yurts: Why Requirements Engineering Needs Design Thinking. Internet Computing 17, 91–94.
Vianna, M., 2012. Design Thinking: Inovação em Negócios. Design Thinking.
Viera, A.J., Garrett, J.M., et al., 2005. Understanding Interobserver Agreement: the Kappa Statistic. Family Medicine 37, 360–363.
Waidelich, L., Richter, A., Kölmel, B., Bulander, R., 2018. Design Thinking Process Model Review. A Systematic Literature Review of Current Design Thinking Models in Practice, in: Proceedings of the International Conference on Engineering, Technology and Innovation, IEEE, Stuttgart, Germany. pp. 1–9.
Wieringa, R., Maiden, N., Mead, N., Rolland, C., 2006. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirements Engineering 11, 102–107.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. 1 ed., Springer.

Rafael Parizi is a PhD Student in the topic of Design Thinking for Software Engineering in the School of Technology at PUCRS University, Brazil. Parizi is also a Software Engineering senior instructor at the Federal Institute Farroupilha. Contact him at rafael.parizi@edu.pucrs.br.

Matheus Prestes holds a MsC from the School of Technology, PUCRS , Brazil. Prestes is also a senior systems analyst.
Contact him at matheus.plautz@edu.pucrs.br.

Sabrina Marczak is an Adjunct Professor in the School of Technology at PUCRS University, Brazil. She co-leads the MunDDoS Research Group, which mostly conducts industry-based research in Software Engineering. Her research is on Software Process Improvement and Human Aspects in Software Development. Contact her at sabrina.marczak@pucrs.br.

Tayana Conte is an Associate Professor in the Institute of Computing at the Federal University of Amazonas (UFAM), Institute of Computing (IComp). Her research interests include the intersection between Software Engineering and Human-Computer Interaction, Software Quality, Human-Centered Computing, and Empirical Software Engineering. Contact her at tayana@icomp.ufam.edu.br.