An Empirical-informed Work Process Model for a Combined Approach of Agile, User-Centered Design, and Lean Startup

Maximilian Zorzetti
Ingrid Signoretti
maximilian.zorzetti@acad.pucrs.br
ingrid.manfrim@acad.pucrs.br
School of Technology
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, Brazil

Eliana Pereira
eliana.pereira@restinga.ifrs.edu.br
Federal Institute of Rio Grande do Sul
Porto Alegre, Brazil

Larissa Salerno
Cassiano Moralles
Michele Machado
larissa.salerno@acad.pucrs.br
cassiano.mora@acad.pucrs.br
michele.machado@edu.pucrs.br
School of Technology
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, Brazil

Ricardo Bastos
Sabrina Marczak
bastos@pucrs.br
sabrina.marczak@pucrs.br
School of Technology
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, Brazil

ABSTRACT

Organizations worldwide have been adopting software development methods that deviate from the standard Agile approach in order to overcome some of its failings—mostly by mixing Agile with other concepts. It has been reported that combining it with Lean Startup and User-Centered Design results in a very powerful development approach, and academic research has developed some high-level process models for it. However, in-depth, demonstrated work process models for this combined approach do not exist, which makes it hard for inexperienced professionals to start using it. We aim to present a process model that reflects how two industry teams that use the combined approach do their work. We performed a case study where we investigated their day-to-day work using daily observations, semi-structured interviews, and focus group sessions; and mapped a set of the activities, techniques, and work products that both teams use daily. We then had them visually represent their work process with the help of our mapping. The result was a software development process model that encompasses Agile, Lean Startup, and User-Centered Design concepts, which can be used as a starting point for those who want to adhere to such a development approach. We reflect on how the process model was conceived and how it could be used by other organizations.

SBQS'20, December 1–4, 2020, São Luís, Brazil
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8923-5/20/12…$15.00
https://doi.org/10.1145/3439961.3439967

CCS CONCEPTS

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

KEYWORDS

agile software development, lean startup, user-centered design, process model, case study

ACM Reference Format:
Maximilian Zorzetti, Ingrid Signoretti, Eliana Pereira, Larissa Salerno, Cassiano Moralles, Michele Machado, Ricardo Bastos, and Sabrina Marczak. 2020. An Empirical-informed Work Process Model for a Combined Approach of Agile, User-Centered Design, and Lean Startup. In *19th Brazilian Symposium on Software Quality (SBQS'20), December 1–4, 2020, São Luís, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3439961.3439967

1 INTRODUCTION

There is a rise in the number of organizations that choose to adopt mixed development methods instead of a pure Agile approach, which is not a surprise as it has been suggested that some of its limitations (e.g., lack of user involvement [29] and difficulty in figuring out the right product to build [13]) can be overcome by combining it with other methods [36]. A noteworthy "method combo" is to add Lean Startup and User-Centered Design (UCD) into the Agile mix, as the former enables software developers to see that the user's needs are met and the latter introduces experiment-driven development, which mitigates risk and guides the generation of value to business stakeholders [35]. This combined approach (hereinafter referred to as such) has been the subject of research for some time now and its success and improvements upon regular Agile has been reported multiple times [14, 31, 32, 38] along with studies that propose a workflow for it [11, 14, 38].

There are, however, challenges in adopting the combined approach [31]. These challenges are exacerbated due to a lack of documentation on how to run the combined approach itself. While the aforementioned workflow studies exist and do provide an initial understanding on how to run the approach, they do not delve into specifics or do not report on a real setting use case—and so a practical, demonstrated starting point that shows what to actually use from each development pillar (Agile, Lean Startup, and UCD) does not currently exist. Surely, experienced professionals can work with the understanding presented in said studies and might even prefer them instead of a more prescriptive process, but for people new to the software development game that want or need to be capable of innovation (which the combined approach is great for [14]), adopting the approach is not a straightforward process and the cost of making mistakes could spell doom to their efforts, or worse, the entirety of their organizations.

To fill in this knowledge gap, this paper reports on the case study of a large multinational IT company undergoing a transformation to the combined approach. We report on two software development teams that received training on the approach from a consulting firm (and have shown expertise on it) and highlight their specific development workflow as described by themselves. We first mapped the concepts used in their day-to-day work and later had both teams describe their work process while taking the resulting mapping into account. Our study reveals the activities, roles, techniques, and work products used by both teams and can be used as a starting point for other organizations to implement their own workflow.

The remainder of this paper is organized as follows. Section 2 discusses software development process models and the use of the combined approach of Agile, Lean Startup, and UCD in software development. Section 3 details the research method undertaken in this study. Section 4 presents our findings, namely the process model for the combined approach; Section 5 deliberates on our findings. Section 6 discusses similar studies. And Section 7 concludes the study, examines threats to its validity, and considers future work.

2 BACKGROUND

2.1 Software Development Process Models

A software development process prescribes the activities that must be carried out when creating software products. It recommends a set of roles, work products, and workflows to facilitate and organize transforming software requirements into actual software applications [18]. Although software development processes were already steadily evolving over the past few decades, agile methods significantly changed the whole environment: they were a first step in the direction of lightweight processes, which usually have less rules and practices or are just easier to follow. Their core idea is that agile teams decide themselves the activities and work products to use in their project—the developers make opportunistic decisions based on their knowledge at the time and are fully prepared to alter their process immediately based on user or customer demands as well as on their own increasing understanding of the process' use.

Even though development processes are lightweight when using agile methods, process models must first exist to use them. A process model is a representation of a software process that prescribes how things should be done in contrast to the process itself, which is really what happens [25]. For instance, in the Extreme Programming (XP) [4] process model there are twelve development practices that are performed in an iterative and incremental life cycle—these practices span the whole development cycle and encompass technical and management aspects.

2.2 Agile, Lean Startup, and UCD

As Agile was extensively used in the past two decades, its shortcomings became more apparent [29]. For instance, it does not provide much assurance that the right software is being developed, business-wise [13] nor does it guarantee customer involvement [3]. Some authors state that Agile alone is not enough to tackle business-level issues [36] and that current practices are not fast enough to support the desired feedback speed of product designers [6]. Combining it with other approaches to development has been suggested as a way to fix these issues [36], and combining it specifically with Lean Startup and UCD has shown great promise [31], with benefits ranging from increased code quality and trust among team members [31] to rapid feedback and reduced development effort [35].

Tackling business-related issues is Lean Startup, an entrepreneurship methodology that focuses on developing a business plan iteratively through the use of a "build-measure-learn" loop, where business hypotheses are evaluated through experiments [24], enabling companies to pivot away from ideas that data suggests to be unfruitful and persevere on the most likely to succeed and yield profits. Continuous experimentation lies at the core of Lean Startup, and embracing it to develop software demands technological capabilities (e.g., continuous deployment) and organizational support (e.g., culture) [20]. Although not specifically a software development method, it follows the idea of lightweight processes (via the build-measure-learn loop) and studies have reported on it being a great driving force when developing software [12, 39].

To ensure that the software not only meets business demands but also the users', the use of UCD [22] enables developers to understand the users' real needs [27] and create solutions that are generally regarded as having improved usefulness, usability, and user satisfaction [37]. The intensity of user involvement varies, from consultation of their needs and participation in usability testing, to having the user actively participate in the design process [1]. Although originally a somewhat structured approach to design[1], as it stands UCD is an umbrella term for design philosophies, processes, and practices that focus on centering the user at the heart of the design space, therefore encompassing terms like Human-Centered Design [10] or Design Thinking [5].

One successful example of the combined approach is fashion retailer Nordstrom's Discovery by Design [14]. The creation of this development methodology was undertaken in an iterative fashion by a dedicated innovation team by combining Agile with Lean Manufacturing, Lean Startup, and Design Thinking: as the team recognized barriers to innovation (e.g., failure for the team to gain traction inside the organization and products seeming too "business-centric"), they incorporated ideas from the aforementioned pillars into their work process in order to overcome them. The team with its "iterative mindset, relentless focus on the needs of the customer, and

[1]UCD originally defined four phases to designing: observation, ideation, prototyping, and testing [22]

bias towards rapid experimentation, prototyping and testing" [14] emerged as a successful dynamic capability [34] for Nordstrom. Its process model, however, is described in high-level brush strokes so as to emphasize that the most valuable aspect of Discovery by Design is its mindset.

In a much more detailed fashion, Dobrigkeit, de Paula, and Uflacker [11] describe InnoDev, a similar development approach that has Scrum [30] tying Design Thinking and Lean Startup together. Their work reports on various aspects of the stated development approach, from work flow activities and techniques to team size and scaling considerations. As they move to validate the model in the present, their work seem most promising.

3 RESEARCH METHOD

We conducted a case study [26] with two software development teams from a multinational company named ORG (name omitted for confidentiality reasons). We mapped their day-to-day work and had them draw their work process using our mapping as guidance.

3.1 Case Setting

ORG has development sites in the USA (headquarters), India, and Brazil. With over 7,000 employees and responsible for about 1,200 internal software products, the IT department started its agile transformation in 2015 and moved to the combined use of Agile, Lean Startup, and UCD principles in late 2017. Before adopting the combined approach, ORG had a well defined roadmap for software product improvements based on an annual budget negotiated among business departments and organized into software projects. High-level business features were prioritized and decided upon by business personnel to later be transformed into software requirements by IT software project teams. The project deadlines were strict and defined by quarter, i.e., every four months the project teams delivered a set of software features to new or existing software products to the organization's internal customers.

With the introduction of an agile transformation in 2015, project teams used Scrum as the guiding development framework—although some participants of this study reported that the strict quarterly deadlines made it waterfall-like. In 2017 they decide to hire Pivotal Software Inc. consulting to support their transformation to a Pivotal Labs-like approach. Pivotal Labs[2] proposes a "team rhythm" that encompasses development, design, and product management; areas that are fulfilled with principles and ceremonies from XP, UCD, and Lean Startup, respectively. Pivotal Labs' main goal is to help teams to build software that deliver meaningful value for users and their business. It offers a framework and a starting point for any team to discuss its needs and define its own path towards software development, including roles, practices, and work products.

3.1.1 The Teams. We observed *in loco* two software development teams (see Table 1) from ORG's financial department located in Brazil. Both teams were built as a catalyst to prove the worth and spread the use of Pivotal Labs throughout the company and have been rated as high-performance and proficient in its use. To achieve this, some members underwent an immersive Pivotal Labs hands-on training at the company headquarters over the supervision of Pivotal Software Inc. consulting personnel before coming back to Brazil to teach the others. Both teams' members have reported on several perceived benefits on using their new work process [31, 35].

Table 1: Participants' Profiles

Team	*Role*	*IT Work Exp. (yr.)*	*Company Exp. (yr.)*
A	Product Designer	27	10
A	Product Manager	21	6
A	Product Manager	16	7.5
A	Software Engineer	21	8
A	Software Engineer	20	11
A	Software Engineer	6	1
A	Software Engineer	5.5	4
B	Product Designer	5	4
B	Product Manager	23	10.5
B	Product Manager	19	0.5
B	Software Engineer	15	11
B	Software Engineer	10	4
B	Software Engineer	7	7
B	Software Engineer	5	5

Team A is responsible for a software product that manages, calculates, and generates data about company projects related to equipment and service delivery. The product manages general project information, such as personnel assignment and time spent on tasks. The application also generates profit data for each project, which is consumed (along with the rest of the data) by the accounting department. Team A is composed of: one Product Designer, two Product Managers, and four Software Engineers.

Team B is responsible for a software product that consumes data from other ORG applications (including Team A's) to calculate the cost of equipment developed in Brazil. The application generates reports for internal accounting, such as inventory reports. The team was also working on automating the validation process for the data coming from each source. Team B is composed of: one Product Designer, two Product Managers, and four Software Engineers.

These teams worked for 9 months in a dedicated lab that follows Pivotal Labs' collaborative work environment recommendations (e.g., single large table for pair-wise work, large screen TV for reports and news, large whiteboards for idea development and information sharing, and a meeting room that turns into an entertainment space for leisure time). The lab is located on PUCRS's campus grounds and was specifically built for ORG teams as an experimental learning environment.

3.2 Data Collection

We observed both teams for a 9-month period, executing several data collection procedures throughout it. Initial perceptions of the teams were collected using typical case study instruments. A *questionnaire* was used to identify the profile of each participant, while *daily observation sessions* were used to shadow team members and attend team ceremonies to learn about their approach to software development and the responsibilities of each of their roles. Two rounds of *semi-structured interviews* were used to specifically gather the team members' perceptions on the combined approach, on

[2]https://tanzu.vmware.com/labs

role changes, on interactions between roles, and on the impact of changes on their work routine. Sporadic interviews were used to follow up on unclear aspects of their day-to-day work unveiled in the observation sessions (e.g., alternate naming conventions). All interviews were voice recorded and transcribed for analysis.

We conducted six *focus group sessions* to discuss the activities, techniques, roles, and work products of the pillars of the combined approach as perceived by the teams in order to map their work process. We conducted two sessions for each pillar and each lasted 1.5 hours on average. We then conducted a *workshop* with both teams. At the start of the workshop, we handed them our mapping and asked each team (apart from the other) to visually represent their work process model on a whiteboard. After both teams stabilized on a process model, we had them discuss and explain together their work process in its entirety while we took notes and recorded their discussion. We then converted their free-form drawings of the work process model into Business Process Model and Notation (BPMN) [2] using Bizagi Modeler[3] to guarantee syntactic validity and draw.io[4] for styling. We chose BPMN as it was similar to the free-form notation used by the teams. We also highlight that perceptions and opinions were very similar between the different participants, with no significant divergences found between them.

3.3 Data Analysis

We conducted Krippendorff's [19] content analysis procedure using a qualitative approach to the ethnographic content analysis, where we focused on the narrative description of the situations, settings, and perspective of both teams' members in their day-to-day work. As we use recording/coding units, we organized the analysis into the following steps: organization and pre-analysis, reading and categorization, and recording the results with ATLAS.ti 7.0[5]. We first read our observation notes and the transcriptions of the interviews and focus group sessions, extracted text excerpts, and marked them as codes. These codes were revisited and grouped into larger codes, forming categories—concepts to be included in our mapping efforts. The identified concepts were then listed and organized in a document to be presented for the teams during the workshop, and revised afterwards to update our understanding of their work processes. We constantly reviewed our coding and mapping schemes with two seniors researchers (the last authors of this paper) aiming to mitigate any limitations or bias in our analysis. Both senior researchers also reviewed the questionnaire and interview scripts and supported the piloting of these instruments for face and content validity with an external researcher with previous experience working with agile teams in the industry.

4 RESULTS

4.1 Roles and Techniques

The participants reported on the three roles that make up their teams, as per Pivotal Labs guidelines:

Product Designer, a facilitator that enables the team in communicating with the user, typically by conducting interviews and promoting the use of techniques to understand and foster empathy towards the user (e.g., a journey map);

Product Manager, a person that provides the business vision of things, helping the team in addressing business needs and establishing assumptions to experiment on; and

Software Engineer, a software developer responsible for implementing solutions and the environment in which they are developed, in addition to participating in decision making and other activities, such as user interviews. A specialization of it, the *Anchor*, is an engineer that resolves technical and non-technical issues by serving as a bridge from the engineers to the user and business stakeholders. Each team has one Anchor, and the Software Engineer that takes on this role does not necessarily have to be the most experienced one on the team.

Throughout their development process, the teams use an ever-increasing set of techniques (Table 2) in an opportunistic fashion (i.e., a specific technique is used given a development situation in which it is most suitable). Both teams continuously seek self-improvement, and as such both are constantly studying recent trends and looking for new techniques to add to their toolbox.

Table 2: Techniques use by the teams.

Pillar	***Techniques***
Agile (XP)	Behavior-Driven Development
	Code Review
	Continuous Integration
	Pair Programming
	Planning Poker
	Refactoring
	Small Releases
	Spike
	Test-Driven Development
	Unit Testing
UCD	Affinity Map
	Brainstorm
	Design Studio
	Dirty Map
	Discovery & Framing
	Ethnographic Research
	Hope and Fears [17]
	How Might We
	Interview
	Journey Map
	Mockup
	Persona
	Prototype
	Service Blueprint
	Two by Two Matrix
	User Flow
	Wireframe
Lean Startup	Continuous Delivery
	Customer Archetype
	Lean Canvas [21]
	Leap of Faith
	Minimum Viable Product

[3]https://www.bizagi.com/en/platform/modeler
[4]https://drawio-app.com/
[5]https://atlasti.com/

4.2 Process Model

The teams defined three sequential phases for the entirety of the development process, in order: *Scoping*, which aims to discover the scope of their work; *Discovery and Framing*, which consists in refining the problem to solve and then determining the right solution to solve it; and *Iteration*, in which the chosen solution is repeatedly developed upon and properly implemented.

As stated, the use of techniques occurs in an opportunistic fashion, although some are specifically used in certain activities. We describe the phases in detail and highlight such activities next.

4.2.1 Scoping Phase. The teams put their efforts on the start of a project towards trying to understand what the problem to solve is—“We have a meeting with stakeholders to understand what will be the problem that we will work with, understanding our work scope and problem, trying to understand what it all entails." All team members are involved in this phase, in addition to stakeholders actively participating. They start with a kick-off meeting in which stakeholders discuss their needs, with the development teams mapping each and every stakeholder alongside their demands. In this meeting, project resources, such as specialized tools or extra personnel, are determined and secured; in addition to the execution of a brief brainstorm session that seeks to outline their approach to developing a solution, setting the pace of the project. Hopefully, they end up with a defined overview of the work scope—in case they do not, the teams host a meeting in which each team member shares their opinions so far on the current scope through a Hopes and Fears technique, revealing the teams' expectations and doubts about the project [17]. At the end of this phase, teams could have four work products already: a stakeholder mapping, a product/problem statement, a product/problem vision, and a product/problem strategy (some teams begin working on new products and others on problems for already established products, hence the product/problem distinction). Figure 1 presents the *Scoping* phase.

4.2.2 Discovery and Framing Phase. Both teams reported the use of a variant of the well-known “Double Diamond” model by the Design Council[6]. In the *Discovery and Framing* phase, as it is called, the Product Designer role is essential as it becomes more active than usual by constantly conducting interviews, researching, and generally acting as a facilitator that guides the team to the proper understanding of the user's needs. Nevertheless, the rest of the team is also involved in all the activities of this phase. The teams highlight the use of the build-measure-learn loop—“We are constantly building something: a problem understanding, a possible solution, a MVP, etc. We create assumptions for each build process to measure the effectiveness of the deliverable, and we learn in each delivery if we are treading the right path.” The teams say the loop is present during the whole process of product development.

The first stage of this phase, *Discovery* (see Figure 2), is focused on understanding the problem at hand on a deeper level, finding its possible root causes and prioritizing them by probability of being the correct cause—“We have the [root] problem prioritization that was defined and identified by the team. The discovery stage enables the whole team to understand the problem meticulously.”

[6]https://www.designcouncil.org.uk/news-opinion/what-framework-innovation-design-councils-evolved-double-diamond

The *Framing* stage (see Figure 3), in turn, is associated with the possible solutions for the problem discussed in the *Discovery* stage: “Our goal in the *Framing* stage is to identify multiple solutions for the problem we defined. At the the end of it, we are able to have a proper solution definition.”

The two stages deal with different outcomes: the objective of *Discovery* is for all team members to further understand what is the problem that they are working to solve; while *Framing* has the aim of yielding a defined solution for the problem—“We must explore the Double Diamond concepts of convergence and divergence. We will be ready to advance to the *Framing* phase if we already converged on a specific problem to work on. And we will be ready to implement our solution if we already converged on a defined solution.”

The *Discovery* stage has three main activities to achieve its goal: Problem Identification, Problem Prioritization, and Problem Definition. The Problem Identification activity is comprised of a set of activities and is part of the build step of the build-measure-learn cycle. It starts with the creation of an interview script using Interview Preparation, which uses the previously developed stakeholder mapping, along with affinity mapping, topic mapping, and brainstorming techniques. At least one representative from each role must participate in the following Interview Conduction, which is typically led by a Product Designer. After analyzing interview data, the teams evaluate if they need to conduct more interviews—“If we conduct an interview and learn new insights about the problem, we must do more sessions of Interview Preparation using affinity mapping and brainstorming techniques to then perform new interviews, until we decide that we gathered enough information.”

The teams report that after acquiring interview data, they use the Pain Points Identification activity (with the help of the service blueprint and user flow techniques) to map their user's daily work routine so as to locate their paints points—“We are able to identify and answer our own questions about the process the user deals with, discovering the user's difficulties.”

After identifying pain points, a refined problem statement and a refined problem vision arise. The teams can now do the Problem Definition activity, which they start with a Persona Creation activity by using user flow, service blueprint, and journey mapping techniques. Next comes the Assumption Creation activity, which generates an assumption list to be validated through experiments in the *Framing* stage or the *Iteration* phase. Teams also perform a User Activities Mapping to map the user's process using a ethnographic research technique.

Finally, they define the proper problem to solve through the Problem Prioritization activity by using a two by two matrix and now-near-next techniques. They later Gather User Feedback to validate their findings and see if they are addressing the user's needs by solving this problem—“We are always getting user feedback to refine and re-prioritize problems.” If they find that the problem is correct and helps the user when solved, they move to the *Framing* stage, or restart the build-measure-learn cycle looking for a new problem definition.

The *Framing* stage (see Figure 3), also guided by the build-measure-learn loop, has two major activities: Solution Identification and Solution Definition. The Solution Identification activity starts by means of Idea Generation—“We meet to discuss what are the possibilities

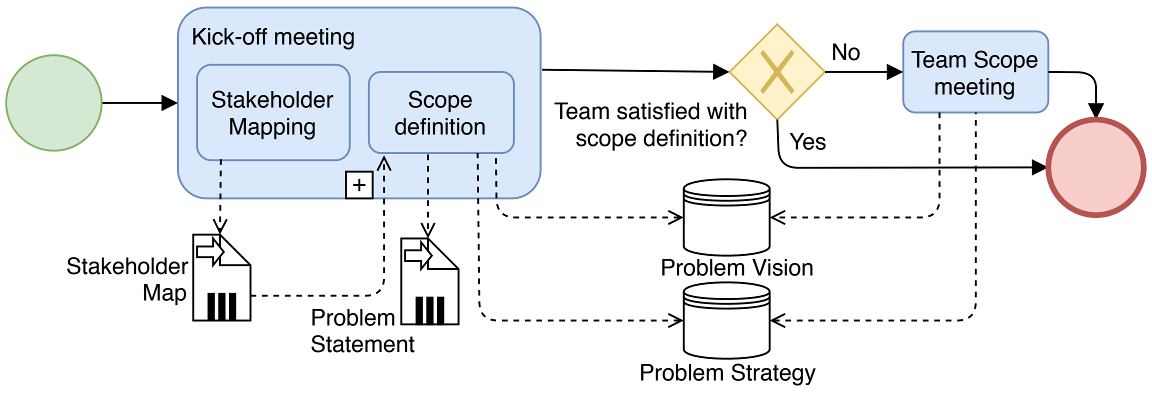

Figure 1: Scoping Phase

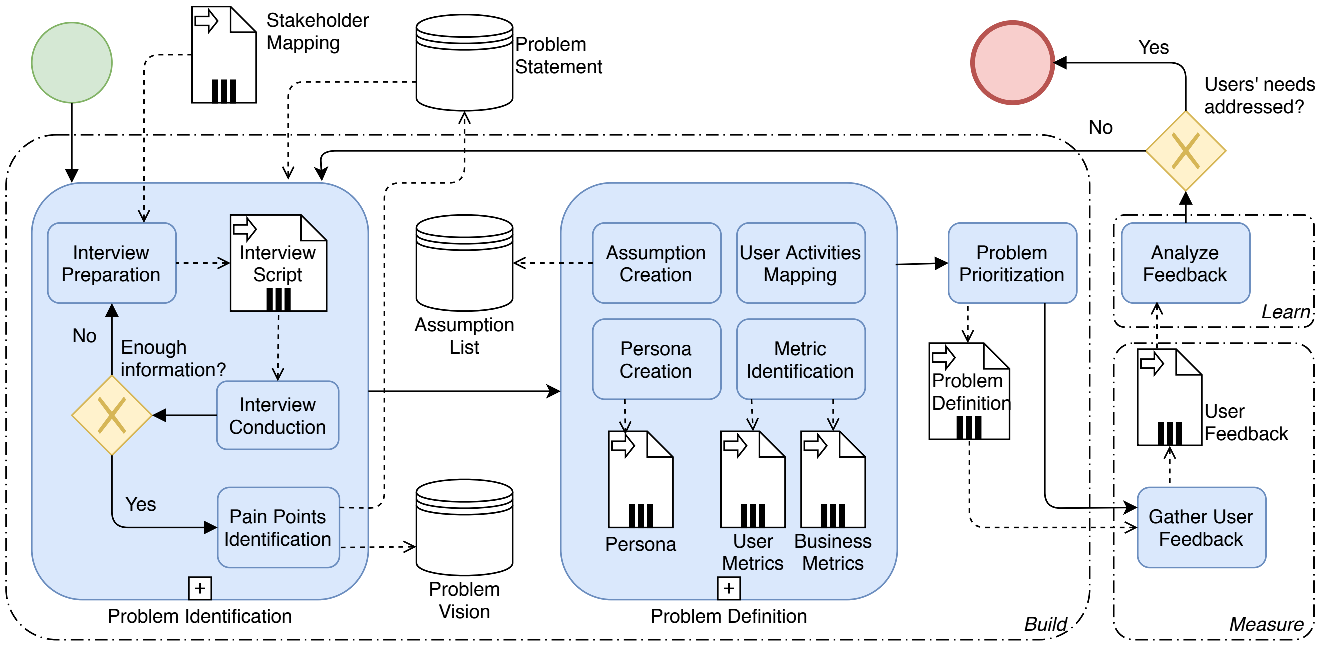

Figure 2: Discovery Stage

to solving the problem." The use of two by two matrices and brainstorming generate an exhaustive idea list, which is organized by means of the Idea Cluster activity using the affinity mapping technique. Afterwards, they have a list of possible solutions which are turned into assumptions to validate later. To perform the Solution Definition activity, teams run experiments to validate assumptions and gather data on solutions, in addition to directly prioritizing them by using two by two matrices, now-near-next, and how might we techniques—" 'How might we' mixed with some brainstorming allows us to identify how we can solve the defined problem."

After prioritizing solutions, they can create prototypes with wireframe and mockup techniques and gather feedback on them with users. If they are confident enough with their findings they can move forward and Define a MVP, or do the whole build-measure-learn cycle to explore more possible solutions if need be. With a MVP definition, they finally move to the Iteration phase.

4.2.3 Iteration Phase. The teams estimate the user stories on their product backlog using planning poker prior to prioritizing them in an Iteration Planning Meeting, which can be preceded by an optional Pre-Iteration Planning Meeting for general discussion on user stories and pre-prioritization. Meanwhile, they set up their whole development pipeline for continuous integration and delivery purposes. After prioritizing stories, story development begins (i.e., coding), focusing on small releases. The teams use several software development techniques (unit testing, pair programming, code reviews, and behavior-driven development) and make use of

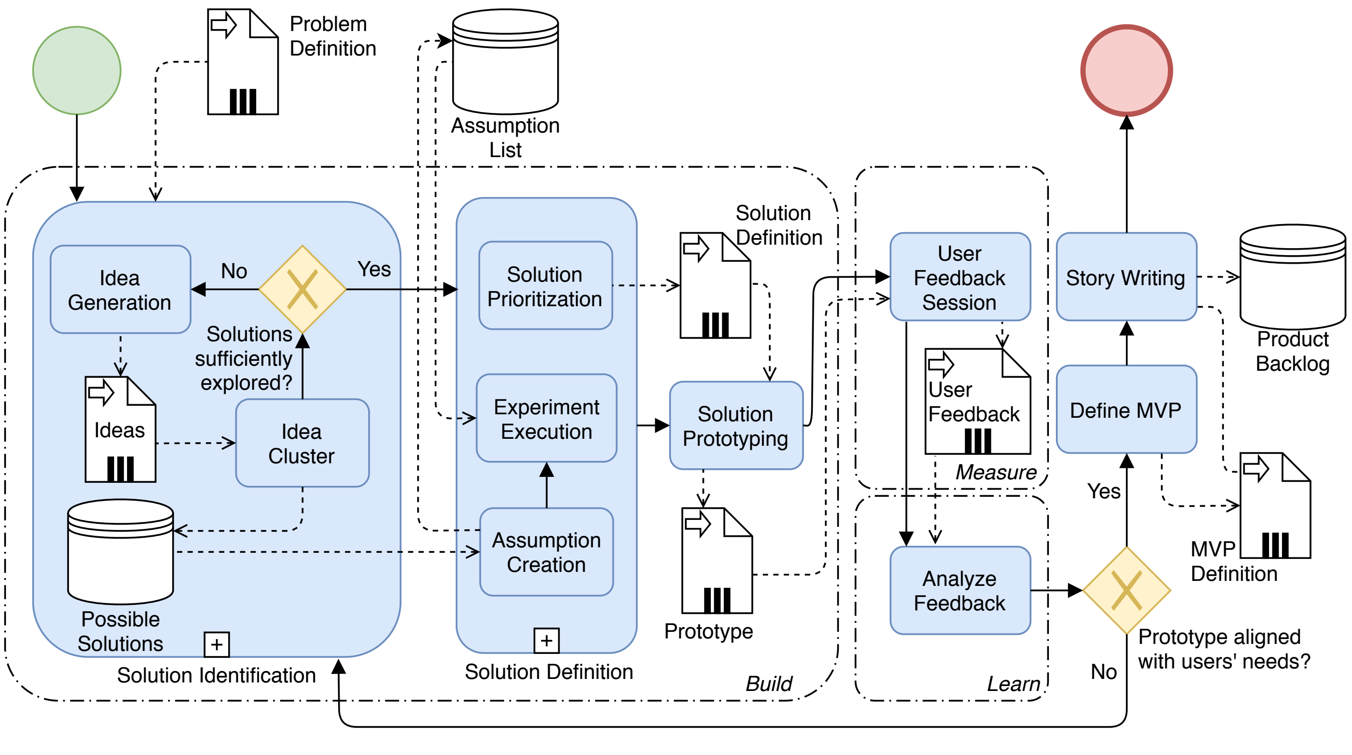

Figure 3: Framing Stage

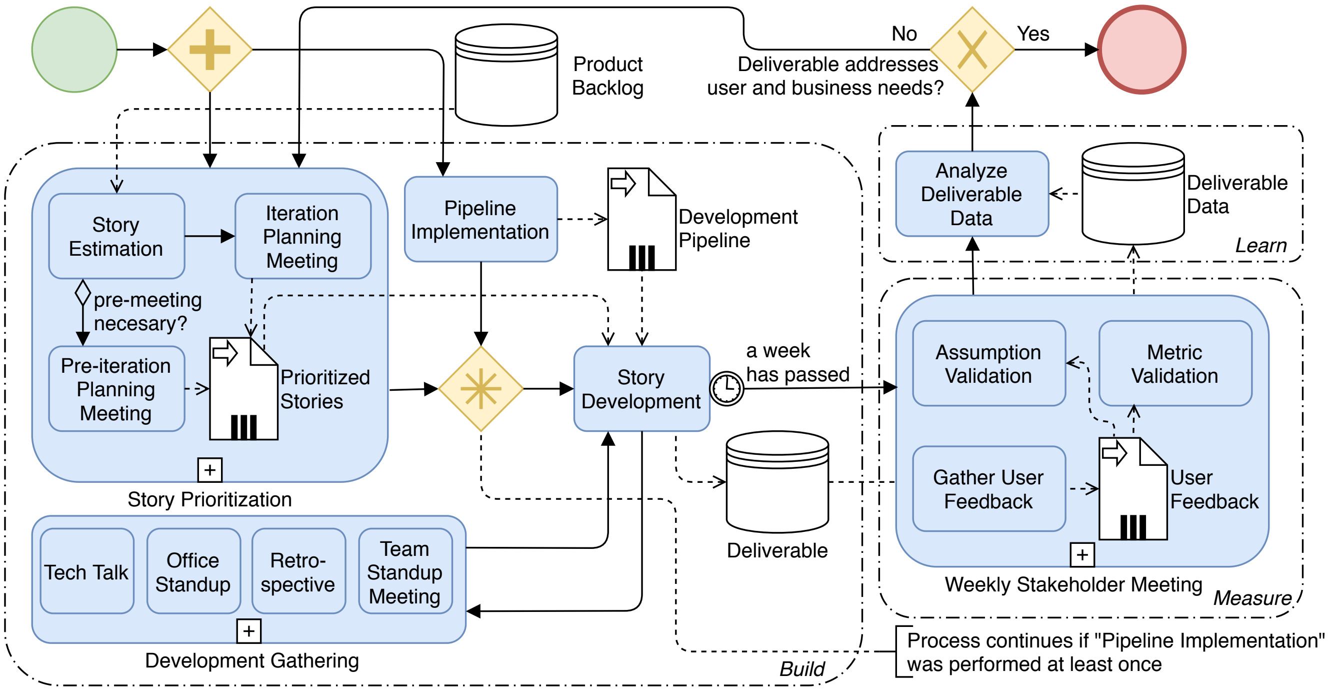

Figure 4: Iteration Phase

different types of meetings for process improvement and information sharing and synchronization purposes. Of note is the Office Standup, in which occurs inter-team knowledge sharing of their daily work, and Tech Talks, occasional meetings in which members presents new technology or practices to add to their repertoire.

The teams meet with stakeholders weekly—"It is a moment for the team to share their work and get feedback. We also use it to define our next steps." They use stakeholder feedback data to validate their assumptions and other metrics, enabling the teams to learn more about their own deliverable and make the call on whether the current deliverable needs more polish or new features and if they should start a new iteration to continue working on it. The Iteration phase is executed repeatedly until all business and user needs are addressed.

5 DISCUSSION

The process itself (see Figure 5) can be interpreted as being lightweight, as a major aspect of the teams' development process is the focus on "problem solving", which can really just be an excuse to break free from the established process and rules in place in order to get to the root cause of a given problem. Most of the time, however, their process provides the necessary steps to find and solve an existing problem by virtue of the heavy focus on experimentation—the build-measure-learn cycle is present throughout all of the development process, aside from the initial *Scoping* phase.

The constant use of the build-measure-learn cycle shifts the responsibility of asserting product validity away from the customer (as it typically occurs in Agile) and into experiment data itself, as developers set target metrics that their solution must adhere to, which, to be fair, might be decided together with or involve the customer. Nevertheless, the resulting solution is developed with confidence as it is backed by data. This is in line with previous studies that use the combined approach: InnoDev's authors state that "each [of its] phase[s] can be seen and implemented as a build-measure-learn cycle" [11], corroborating with having Lean Startup as a driving force to development in the case of Discovery by Design [14]. The use of experimentation also reduces costs by reducing the amount of development effort spent on unwanted products and features [39]. Given ORG's corporate environment, this works wonders to convince upper management of the decisions made by the development teams, which to them hopefully results in increased trust and more freedom to work the way they think is best—such developer empowerment has been stated as a way to reduce development waste, as empowered developers make better decisions by taking into account the scenario they find themselves in [23].

As structured as the process is, both teams previously highlighted how it feels "organic" and how their mindset of problem solving is crucial to the combined approach. This suggests that their process is not as structured as it seems, but they might view it as such given their old work process was waterfall-based. A comparison to lightweight processes is an apt one, and to startups as well. Souza et al. [33] conducted a case study with 10 early-stage startups from the All Saints Bay startup ecosystem and modeled their software development practices, most of which are present in ORG's teams day-to-day work, such as practices to speed-up development, high priority on user experience, and an evolutionary

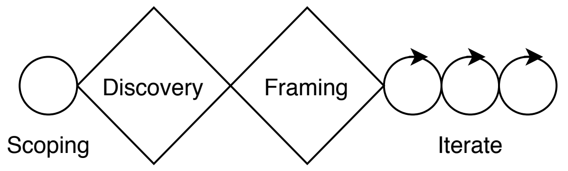

Figure 5: Overview of the process model used by the teams. Adapted from Signoretti et al. [32]

approach to development (i.e., constant use of prototypes and user feedback gathering). Unsurprisingly, the teams deviate from the findings of Souza et al. in aspects that are typical of the startup scenario and that are "remedied" by ORG resources and policies (e.g., accumulating technical debt to save on development costs or lack of human or financial resources). This suggests that the rapid, risk-taking development mindset of startups can work on larger organizations, arguably even better than on startups themselves.

Extensively exploring the problem space early on leads to a better definition of requirements, in addition to laying the groundwork for exploring alternative solutions should the use of experimentation point towards them. We see a common ground shared between ORG's use of the combined approach and InnoDev [11] and Discovery by Design [14] in that they all have a bigger emphasis on UCD activities on the initial stages of a project. As such, this suggests that a user-centric mindset is pivotal to the combined approach.

6 RELATED WORK

Before the Lean Startup book [24], there already existed interest in combining UCD with Agile practices, as seen in DT@Scrum [15] and Hildenbrand and Meyer's work [16]. After the book launch in 2011, we can date the earliest model as soon as 2012, due to Lean Startup's ideas being available before launch on the author's blog, from which they spread throughout the industry.

Current literature reports on several models that aim to combine Agile, Lean Startup, and UCD, such as Discovery by Design [14] (see Section 2.2). Unlike ORG's model use of the higher-level UCD, such models specifically use Design Thinking as it provides a more concrete approach to design. Table 3 shows a comparison of the origins and depth of the existing academic literature models. We highlight that our model has the most thorough description among the related work, in addition to being evaluated empirically with an industry case, a leverage only shared by Discovery By Design [14].

7 CONCLUSION

We have reported on a case study of two software development teams from multinational IT organization ORG, which is currently undergoing a transformation to the combined approach. The case study consisted of the use of observations, semi-structured interviews, and focus group sessions, followed by Krippendorff's content analysis procedure [19]. Our findings outline the ORG teams' roles and techniques, and highlight a software process model that shows the activities and work products that they use daily. This model can be used as a starting point for organizations to model a custom instance of the combined approach; alternatively, organizations can adhere to it if their circumstances are similar to ORG's.

Table 3: Comparison of similar models. "—" stands for "unspecified".

Model	*Foundation and Development*	*Evaluation*	*Representation*	*No. of Techniques*	*Roles*	*Work Products*
Converge [38]	Based on previous work combining Agile, Lean Startup, and Design Thinking and adjusted through the opinions and empirical observation of an undergraduate student team	Empirically evaluated with an undergraduate student team	Text descriptions, high-level diagram	9	Developer (inferred), Project Lead	—
Discovery by Design [14]	Iteratively built upon by the innovation team of a large fashion retailer as difficulties and needs arose by combining Agile and Lean Manufacturing principles, Design Thinking, and Lean Startup	Empirically evaluated along its development on an industry case	High-level diagram	5	—	—
InnoDev [11]	Combines DT@Scrum [15] and MoIT [8] into a process model that spans Design Thinking, Lean Startup, and Scrum	Surveys and workshops (not executed yet)	Text descriptions, high-level diagram	31	InnoDev Team, InnoDev Facilitator, Process Master, Product Owner, Project-specific Expert, Project Sponsor Scrum Master	Yes
MoIT [8]	Based on empirical observations and opinions of two student teams running Discovery by Design and Hildenbrand and Meyer's model [16]	Empirically evaluated with an undergraduate student team [9]	Text descriptions, high-level diagram	8	—	—
ORG	Mapped from two industry teams' accounts of their specific use of Pivotal Labs, which spans Extreme Programming, Lean Startup, and User-Centered Design	Empirically evaluated with two industry teams from a multinational organization	Text descriptions, high-level diagram, process model (BPMN)	32	Product Designer, Product Manager, Software Engineer	Yes

As seen in ORG's case, top-level staff are impressed with the combined approach as they requested it to be spread throughout the organization. However, there are several software development teams in the organization due to its size, which makes organization-wide adoption efforts difficult given the huge experience and expertise disparity between teams: the steps taken for a team to adopt the combined approach might not work for another, and adhering to the reported process might not yield the same benefits reported by this study's teams, as its generalizability is not guaranteed (an inherent characteristic of case studies). Not only that, but understanding what makes the combined approach tick is not a straightforward task. Given the novelty of the combined approach, helpful instruments in this regard (such as maturity models) do not exist. The model as described in this paper eases this aspect by showcasing how the use of a fully-adopted combined approach should look like, but does not provide insight on how to achieve it. Such undertaking would best be left to a proper adoption instrument, which perhaps could be developed using our model itself as its basis.

Regarding any empirical study, our study has limitations. To mitigate construction validity concerns, we used multiple data sources to triangulate findings and had senior researchers accompany each step of the study. In regards to external validity, we can not assert that the model is usable in other scenarios. For our modeling efforts, we used automated digital tools to validate their syntactic quality, while semantic quality was strengthened due to the supervision and revisions by aforementioned senior researchers.

As for future work, seeing as the teams used to work with different approaches and likely represented their work process following a structured, activity-centered approach, an interesting exercise would be to conduct a similar study but instruct the teams in using tools to represent their work process in an unstructured fashion, such as Azzurra [7]. Should results point towards the approach's mindset being vital, a tool to support the adoption of the approach could focus either on its process or on capturing its mindset. Another course of action would be to evaluate the efficacy of the process model, such as by checking it for process smells [28].

ACKNOWLEDGMENTS

We thank the study participants and acknowledge that this research is sponsored by Dell Brazil using incentives of the Brazilian Informatics Law (Law no. 8.2.48, year 1991).

REFERENCES

[1] Chadia Abras, Diane Maloney-krichmar, and Jenny Preece. 2004. User-Centered Design. In *Encyclopedia of Human-Computer Interaction*. SAGE, Bainbridge Island, USA.

[2] Thomas Allweyer. 2010. *BPMN 2.0*. BoD, Norderstedt, Germany.

[3] María Bastarrica, Germán Espinoza, and Jacqueline Sánchez. 2018. Implementing Agile Practices: The Experience of TSol. In *Proceedings of the Int'l Symposium on Empirical Software Engineering and Measurement*. ACM, Oulu, Finland, 1–10.

[4] Kent Beck. 1999. Embracing Change with Extreme Programming. *Computer* 32, 10 (Oct. 1999), 70–77.

[5] Tim Brown. 2008. Design Thinking. *Harvard Business Review* 86 (07 2008), 84–92.

[6] Alistair Cockburn. 2019. Cutting-Edge Agile: Opening Statement. *Cutter Business Technology Journal* 32 (2019), 1.

[7] Fabiano Dalpiaz, Evellin Cardoso, Giulia Canobbio, Paolo Giorgini, and John Mylopoulos. 2015. Social specifications of business processes with Azzurra. In *Proceedings of the International Conference on Research Challenges in Information Science*. IEEE, Athens, Greece, 7–18.

[8] Danielly Ferreira Oliveira de Paula. 2015. *Model for the Innovation Teaching (MoIT): um modelo baseado em Design Thinking, Lean Startup e Ágil para estudantes de graduação em computação*. Master's thesis. Universidade Federal de Pernambuco, Recife, Brazil.

[9] Danielly F. O. de Paula and Cristiano C. Araújo. 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 Int'l Conf. on Human-Computer Interaction*. Springer, Toronto, Canada, 30–34.

[10] Design Kit. 2015. *The Field Guide to Human-centered Design*. IDEO, Palo Alto, California, USA.

[11] Franziska Dobrigkeit, Danielly de Paula, and Matthias Uflacker. 2019. *InnoDev: A Software Development Methodology Integrating Design Thinking, Scrum and Lean Startup*. Springer, Potsdam, Germany, 199–227.

[12] Fabian Fagerholm, Alejandro Sánchez Guinea, Hanna Mäenpää, and Jürgen Münch. 2017. The RIGHT model for Continuous Experimentation. *Journal of Systems and Software* 123 (2017), 292–305.

[13] Peggy Gregory, Leonor Barroca, Helen Sharp, Advait Deshpande, and Katie Taylor. 2016. The challenges that challenge: Engaging with agile practitioners' concerns. *Information and Software Technology* 77 (2016), 92–104.

[14] Benjamin Grossman-Kahn and Ryan Rosensweig. 2012. Skip the silver bullet: driving innovation through small bets and diverse practices. *Leading Through Design* (2012), 815–829.

[15] Franziska Häger, Thomas Kowark, Jens Krüger, Christophe Vetterli, Falk Übernickel, and Matthias Uflacker. 2015. *DT@Scrum: Integrating Design Thinking with Software Development Processes*. Springer, Switzerland, 263–289.

[16] Tobias Hildenbrand and Johannes Meyer. 2012. *Intertwining Lean and Design Thinking: Software Product Development from Empathy to Shipment*. Springer, Berlin, Heidelberg, 217–237.

[17] IBM. 2020. Enterprise Design Thinking – Hopes and Fears. https://www.ibm.com/design/thinking/page/toolkit/activity/hopes-and-fears Accessed on 29-Oct-2020.

[18] I. Jacobson, G. Booch, and J. Rumbaugh. 2001. *The Unified Software Development Process*. Addison Wesley, Upper Saddle River, USA.

[19] Klaus Krippendorff. 2013. *Content Analysis - 3rd Edition: an Introduction to Its Methodology*. SAGE, Thousand Oaks, USA.

[20] Eveliina Lindgren and Jürgen Münch. 2016. Raising the Odds of Success: The Current State of Experimentation in Product Development. *Information and Software Technology* 77 (04 2016), 80–91.

[21] A. Maurya. 2012. *Running lean: Iterate from plan A to a plan that works*. O'Reilly Media, Sevastopol, Crimea.

[22] Donald A. Norman and Stephen W. Draper. 1986. *User Centered System Design; New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates Inc., New Jersey, USA.

[23] Mary Poppendieck and Tom Poppendieck. 2003. *Lean Software Development: An Agile Toolkit*. Addison-Wesley, Boston, USA.

[24] Eric Ries. 2011. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business, New York, USA.

[25] Colette Rolland. 1998. A Comprehensive View of Process Engineering. In *Proceedings of the International Conference on Advanced Information Systems Engineering (Lecture Notes in Computer Science, Vol. 1413)*. Springer, Pisa, Italy, 1–24.

[26] Per Runeson and Martin Höst. 2008. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering* 14, 2 (19 Dec 2008), 131.

[27] Dina Salah, Richard Paige, and Paul Cairns. 2015. Patterns for integrating agile development processes and user centred design. In *Proceedings of the European Conference on Pattern Languages of Programs*. ACM, Kaufbeuren, Germany, 1–10.

[28] Edison J. Santos, Rita Suzana Pitangueira Maciel, and Cláudio Sant'Anna. 2018. A Catalogue of Bad Smells for Software Process. In *Proceedings of the Brazilian Symposium on Software Quality*. ACM, Curitiba, Brazil, 1—-10.

[29] Schön et al. 2017. Key challenges in agile requirements engineering. In *Proceedings of the Int'l Conf. on Agile Sw. Development*. Springer, Cologne, Germany, 37–51.

[30] Ken Schwaber and Jeff Sutherland. 2017. *The Scrum Guide*. Vol. 21. Scrum Alliance, Westminster, USA. https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf Retrieved on 2020-Aug-15.

[31] Ingrid Signoretti, Sabrina Marczak, Larissa Salerno, Augusto de Lara, and Ricardo Bastos. 2019. Boosting Agile by Using User-Centered Design and Lean Startup: A Case Study of the Adoption of the Combined Approach in Software Development. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*. IEEE, Porto de Galinhas, Brazil, 1–6.

[32] Ingrid Signoretti, Larissa Salerno, Sabrina Marczak, and Ricardo Bastos. 2020. Combining User-Centered Design and Lean Startup with Agile Software Development: A Case Study of Two Agile Teams. In *Proceedings of the Int'l Conference on Agile Software Development*. Springer, Copenhagen, Denmark, 39–55.

[33] Renata Souza, Karla Malta, Roselane Silva, Paulo Masiero, Eduardo Almeida, and Ivan Machado. 2019. A Case Study about Startups' Software Development Practices: A Preliminary Result. In *Proceedings of the Brazilian Symposium on Software Quality*. ACM, Fortaleza, Brazil, 198–203.

[34] David J. Teece. 1998. Capturing Value from Knowledge Assets: The New Economy, Markets for Know-How, and Intangible Assets. *California Management Review* 40, 3 (1998), 55–79.

[35] Bruna Prauchner Vargas, Ingrid Signoretti, Maximilian Zorzetti, Sabrina Marczak, and Ricardo Bastos. 2020. On the Understanding of Experimentation Usage in Light of Lean Startup in Software Development Context. In *Proceedings of the Evaluation and Assessment in Sw. Eng.* ACM, Trondheim, Norway, 330–335.

[36] Kati Vilkki. 2010. When Agile is Not Enough. In *International Conference on Lean Enterprise Software and Systems*. Springer, Helsinki, Finland, 44–47.

[37] Karel Vredenburg, Ji-Ye Mao, Paul W. Smith, and Tom Carey. 2002. A Survey of User-Centered Design Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Minneapolis, Minnesota, USA, 471–478.

[38] Bianca H. Ximenes, Isadora N. Alves, and Cristiano C. Araújo. 2015. Software Project Management Combining Agile, Lean Startup and Design Thinking. In *Proceedings of the International Conference on Design, User Experience, and Usability: Design Discourse*. Springer, Los Angeles, USA, 356–367.

[39] Sezin Gizem Yaman, Myriam Munezero, Jürgen Münch, Fabian Fagerholm, Ossi Syd, Mika Aaltola, Christina Palmu, and Tomi Männistö. 2017. Introducing continuous experimentation in large software-intensive product and service organisations. *Journal of Systems and Software* 133 (2017), 195–211.