Information and Software Technology 141 (2022) 106718

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Improving Agile Software Development using User-Centered Design and Lean Startup

Maximilian Zorzetti, Ingrid Signoretti, Larissa Salerno, Sabrina Marczak *, Ricardo Bastos

School of Technology, PUCRS, 6681 Ipiranga Avenue, Building 32, room 504, Porto Alegre, RS, Brazil

ARTICLE INFO

Keywords:
Agile Software Development
Extreme Programming
User-Centered Design
Lean Startup
Case study

ABSTRACT

Context: Agile methods have limitations concerning problem understanding and solution finding, which can cause organizations to push misguided products and accrue waste. Some authors suggest combining agile methods with discovery-oriented approaches to overcome this, with notable candidates being User-Centered Design (UCD) and Lean Startup, a combination of which there is yet not a demonstrated, comprehensive study on how it works.

Objective: To characterize a development approach combination of Agile Software Development, UCD, and Lean Startup; exposing how the three approaches can be intertwined in a single development process and how they affect development.

Method: We conducted a case study with two industry software development teams that use this combined approach, investigating them through interviews, observation, focus groups, and a workshop during a nine-month period in which they were stationed in a custom-built development lab.

Results: The teams are made up of user advocates, business advocates, and solution builders; while their development approach emphasizes experimentation by making heavy use of build-measure-learn cycles. The combined approach promotes a problem-oriented mindset, encouraging team members to work together and engage with the entire development process, actively discovering stakeholders needs and how to fulfill them. Each of its approaches provide a unique contribution to the development process: UCD fosters empathy with stakeholders and enables teams to better understand the problem they are tasked with solving; Lean Startup introduces experimentation as the guiding force of development; and Extreme Programming (the teams' agile method) provides support to experimentation and achieving technical excellence.

Conclusion: The combined approach pushes teams to think critically throughout the development effort. Our practical example provides insight on its essence and might inspire industry practitioners to seek a similar development approach based on the same precepts.

1. Introduction

Agile methods are defined by their adaptability and flexibility when developing software products [1], having emerged at a time where software engineering methods proved not sufficiently equipped to deal with the constant change and unpredictability of the software market [2]. As agile methods were extensively used these past two decades, several studies have reported their shortcomings, ranging from difficulties in problem understanding and solution finding [3] and in adding business value [4] to lack of attention to design and architectural issues [4]. As the software industry continues to evolve so does what is needed to thrive in it, and "fixing" agile methods can be a way to achieve proper innovation and avoid the launch of undesirable products [5]. Vilkki [6] suggests that agile methods should be combined with other methods (hereinafter also referred to as "pillars") to overcome their weakness, with a noteworthy "method combo" being that of combining them with UCD [7] and Lean Startup [8], as the former enables software developers to see that the user's needs are met and the latter introduces experiment-driven development, mitigating risk and guiding the generation of value to business stakeholders [9]. 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 methods have been reported [8,10,11]. However, there still is a lack of studies thoroughly examining a demonstrated instance of the combined approach.

* Corresponding author.
E-mail addresses: maximilian.zorzetti@acad.pucrs.br (M. Zorzetti), ingrid.manfrim@acad.pucrs.br (I. Signoretti), larissa.salerno@acad.pucrs.br (L. Salerno), sabrina.marczak@pucrs.br (S. Marczak), bastos@pucrs.br (R. Bastos).

https://doi.org/10.1016/j.infsof.2021.106718
Received 15 October 2020; Received in revised form 6 August 2021; Accepted 12 August 2021
Available online 28 August 2021

Adopting the combined approach can be a costly journey, as it demands not only adherence to practices and technical capabilities for it to work, but employee mindset change as well. The combined approach advocates for the autonomy of teams [12] so they can effectively solve problems, which can clash with existing organizational mechanisms and culture, hindering the adopters' chance of a successful implementation of it. Although there are prerequisites mentioned in literature to adopting the combined approach and studies presenting models of it [8,10,13], we still know little about what changes take place at the development team level and how each pillar is manifested throughout the development process. Motivated by the need to discuss these intricacies and the combined approach as a whole, we conducted a case study of two software development teams from a large multinational IT company undergoing an adoption process to a combined approach upheld by Extreme Programming (XP) [14], UCD, and Lean Startup; with the intent of answering two research questions:

RQ1. How do teams use the combined approach in practice?
RQ2. How does the combined approach influence development?

To do so, we report on these teams' custom instance of the combined approach from a team-level perspective, addressing the combined approach itself in several aspects: the general environment and mindset changes it brings about and the influence of each of its pillars on the development process; and, as an extended version of a previous paper [15], we report on new findings concerning the roles that make up the teams and the phases of their development workflow. Our study dives into the forefront of software development using the combined approach, providing a holistic account of its inner workings and topical insights on its use from the perspective of the teams themselves.

The remainder of this paper is organized as follows. Section 2 discusses the use of the combined approach of Agile, UCD, and Lean Startup in software development. Section 3 describes the case setting. Section 4 details the research method undertaken in this study. Section 5 presents the combined approach as used by the two development teams, describing their roles and workflow. Section 6 examines the presence of the combined approach's three pillars during development. Section 7 discusses our findings, addressing existing literature; and examines the study's limitations. Finally, Section 8 concludes the study and considers future work.

2. Literature review

2.1. Background

Agile methods range from generic project management frameworks (e.g., Scrum [16]) to software engineering approaches that provide specific software development practices (e.g., XP [14]), such as pair programming. As agile methods have been extensively used in the past two decades, their shortcomings became more apparent [17]. Some authors argue that agile methods alone are not enough to tackle business-level issues [6] and that they provide insufficient customer involvement [18]. Combining agile methods with other approaches to development has been suggested as a way to fix these issues [6], and combining them specifically with Lean Startup and UCD has shown great promise as it seemingly increases the engagement of business stakeholders and end users while also enabling rapid experimentation, among other benefits [9,11].

In this combined approach, 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" cycle, where business hypotheses are evaluated through experimentation [19], enabling companies to pivot away from ideas that data suggests to be unfruitful and persevere on the ones more likely to succeed. 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 cycle) and studies have reported on it being a great driving force when developing software [21,22].

To ensure that the software not only meets business demands but also the users', the use of UCD [7] enables developers to understand the users' real needs [23] and create solutions that are generally regarded as having improved usefulness, usability, and user satisfaction [24]. 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 [25]. 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 [26], Design Thinking [27], and the Double Diamond,[2] the last of which encompasses two working spaces—one for exploring an issue more widely or deeply (divergent thinking) and one for taking focused action based on the learning from the previous phase (convergent thinking).

Notably, combining any of the three pillars can be quite challenging despite their many similarities and ways they can complement one another. For instance, even though agile methods in general have been criticized as being lackluster on the process of devising an initial design [28], merging them with UCD practices can be difficult due to the former's focus on getting working software quickly and the latter's needs for upfront design planning. These incompatibilities can span to a more philosophical and principled level as well, such as with the generalist views present in Agile in contrast to the specialist views of UCD [23], or Lean Startup's quantitative methods [19] and UCD's qualitative approach [29].

The three pillars deal with different aspects of product development, namely design (UCD), product management (Lean Startup), and development itself (Agile). Combining them into a single approach is Pivotal Labs, which proposes principles and ceremonies based on the three pillars, with XP as its agile method. Pivotal Labs' main goal is to assist teams in building software products that deliver meaningful value for users and their business, offering a framework and starting point for any team to discuss its needs and define its own path towards software development. To do so, it suggests the adoption of a cross-functional team composed of three main roles: Product Designer, Product Manager, and Software Engineer; the latter specialized in an Anchor role, responsible for bridging the engineers with users and business stakeholders. It also proposes that the team finds its own work "rhythm" by collectively deciding on which ceremonies and workflows derived from the three pillars to adopt, revisiting them constantly. Pivotal Labs does not provide a prescriptive set of work practices (which would go against the Agile philosophy); rather, it suggests the following principles: that teams should learn by doing, work in a co-located fashion in order to facilitate coordination and fast feedback loops among team members, promote constant collaboration, take the lead and own the product development cycle, and find their own sustainable pace having in mind that product development is a "marathon" and not a sprint.

2.2. Related work

While Pivotal Labs proposes a philosophical way of working, scientific literature presents models that abstract the combined use of Agile, UCD, and Lean Startup in software development—with each model having unique choices for implementing each pillar (e.g., using Design

[1] UCD originally defined four phases: observation, ideation, prototyping, and testing [7].

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

Thinking for UCD). For instance, a successful example of the referred approach is fashion retailer Nordstrom's Discovery by Design [8]. The creation of this multidisciplinary innovation approach was undertaken in an iterative and "organic" fashion by a dedicated innovation team by combining Agile with Lean Manufacturing, Lean Startup, and Design Thinking. Spearheaded by an employee passionate about Agile Software Development and Lean Manufacturing principles, the innovation team fully embraced the principles of the Agile Manifesto [1] and started to prototype new ideas, but soon found that their efforts failed to gain traction inside Nordstrom, as Agile and Lean Manufacturing lacked the tools to validate ideas before handing them over to the business. By adopting the Lean Startup method afterwards, the team turned itself into a successful "discovery vehicle" that rapidly tested and validated new ideas by means of measuring customer demand, albeit with their products seeming a little too much "business-centric". Given their positive experiences with interacting with end users during this iteration of their development method, they eventually decided to incorporate Design Thinking [26] into it in addition to adopting a more risk-tolerant attitude, completing their method into what they called Discovery by Design. The team with its "iterative mindset, relentless focus on the needs of the customer, and bias towards rapid experimentation, prototyping and testing" [8] emerged as a successful dynamic capability [30] for Nordstrom. The Nordstrom case is a good example of how the combined approach is successful in supporting software teams in keeping up with market and user needs, providing a solid source of evidence to its success.

Another similar development approach to the one reported on this study is Lean UX [31], which is grounded on agile methods (typically Scrum), Design Thinking, and Lean Startup, focusing on the integration of the design process with product development and utilizing principles derived from its three pillars (e.g., cross-functional teams from Design Thinking, permission to fail from Lean Startup, and getting out of the "deliverable business" from Agile). Other similar models include the aforementioned Discovery By Design [8], as well as Converge [10] and InnoDev [13]; which propose a software development approach that merges Agile, Design Thinking, and Lean Startup. Discovery By Design and Converge have their development process characterized by heavy use of Design Thinking during their initial phases, followed by build-measure-learn cycles and pivoting decision points; and are backed by empirical data collected from an industry case and an undergraduate development team, respectively. InnoDev is similar, providing a scoping phase at the beginning of its process; although it explicitly uses Scrum and is not yet backed by a real case example.

3. Case setting

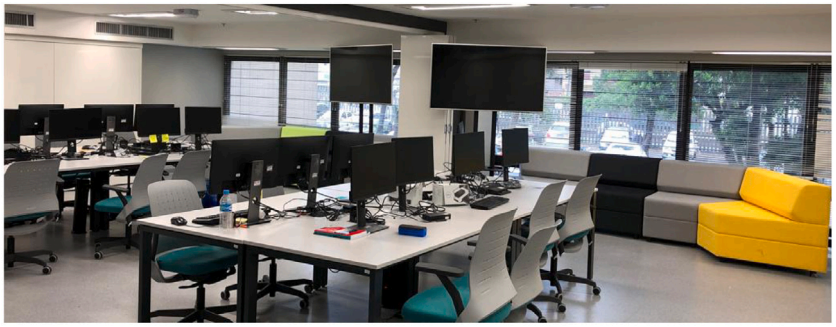
Our research group has studied two industry teams from the moment they arrived back in Brazil at PUCRS' campus after a three months-long period (December 2018 to February 2019) of "learn-on-the-job" training at ORG's[3] USA headquarters to December 2019 when the teams migrated back to ORG's Brazilian work site. The teams were put together by ORG as a piloting experience on the use of the combined approach. ORG managers and HR selected among the most talented team members in Brazil and sent them for a collaborative work experience with Pivotal Labs consultants in the USA. The four consultants composed the teams during this three months-long period mentoring them in decision making in light of Pivotal Labs principles. These invited guests acted as full-time Product Designer, Product Manager, and Software Engineers (two of them), participating in product development activities (i.e., they were co-owners of the products under development) and decisions involving both technical (e.g., which architectural design was best for the problem at hand) and work- and process-related aspects (e.g., how to best design and conduct an experimental study or what responsibilities certain roles entailed). Since ORG decided on adopting Pivotal Labs and hired consultancy to support its transformation from agile methods to the combined approach as part of the company's strategic plan, having the teams being temporarily moved to the USA headquarters alongside Pivotal Labs representatives helped in their immersion and enabled face-to-face contact with customers and end-users, these being from products for internal use of ORG. Moreover, it allowed for a snowballing strategy to train new people: after the period experimenting at PUCRS and maturing their newly-acquired skills, these teams' members would return to their original work premises and train their peers while on the job. The transformation process, its reasoning, activities, and benefits and challenges are described in detail in the work of Signoretti et al. [11].

Upon arrival in Brazil, the teams were promptly moved to work on our campus in a laboratory specially designed for collaborative work (see Fig. 1), mirroring the infrastructure suggested by Pivotal Labs (i.e., one large table for each team with notebooks for members to work in pairs, a large TV screen at the head of each table for presentations and screen sharing when necessary, and separated working booths in two corners of the main space for one-to-one meetings or private conversations). The lab also encompassed a large meeting room with an even larger table easily flipped into a ping-pong table for leisure time. It also offered a coffee area in which the teams could sit around during meals. These spaces contained several whiteboards and had writable walls to allow for sketching and writing notes as the teams saw fit.

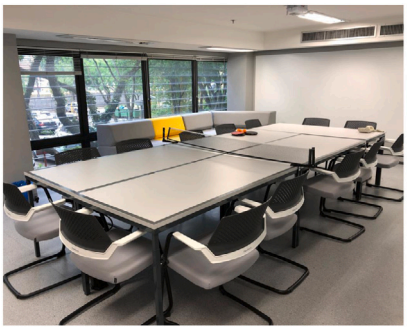
Ever since the teams started working at the PUCRS lab, we have analyzed several aspects of this case, from comparing the teams' use of the combined approach to existing literature [32] to analyzing how experimentation takes place during their development process [9] and what are the success and failure factors to adopting the combined approach from the teams' perspective [12]. As previously mentioned, our goal is to better comprehend the combined approach with two main contributions in mind: practice-wise, to help ORG in devising and rolling out a reduced-cost transformation effort throughout their organization by understanding the intricacies of doing so; and academic-wise, to conceptually develop, in the long-run, what we dubbed as an "acceleration" model [33]—a model to assist in such kind of transformation effort by identifying what knowledge (e.g., on experimentation, on developing empathy with the user, on pair programming) is required for a software team to develop a certain product using the combined approach, diagnosing the gap between the required and the actual team knowledge, and offering support to follow-up on actions aiming to fill in this knowledge gap. Our under-development acceleration model is fueled by the data collected on the teams' and other similar cases. We present the organization and the teams next.

The organization. ORG has development sites in the USA (headquarters), Brazil, and India, has over 7,000 employees, and is responsible for about 1,200 internal software products. The organization started an agile transformation in 2015, with teams using Scrum as the guiding development framework. From this time and on, it became common (but not organization-wide) to get more team members (e.g., developers, software architects, and testers) engaged with the business feature-to-software requirement translation. In late 2017, the transformation strategy changed as the CEO understood that the organization should improve their user experience by focusing on products. ORG then switched from a worldwide road map to focus on a product-oriented mindset, also adding XP to the Scrum agile tool package. This change demanded of teams more in-depth knowledge of their users and business needs. For this reason, the organization decided to invest in a combined approach of XP, UCD, and Lean Startup from mid 2018 onwards. This is when ORG reached out to PUCRS and decided on funding a dedicated experimental research lab of which this project is part of and that the teams were stationed in for about a year (three months to set up the lab configuration/room renovation and nine months working before moving back to the Brazilian ORG office as previously mentioned).

[3] Large IT multinational company. Name omitted due to confidentiality reasons.



(a) Collaborative workspace



(b) Meeting room

(c) Coffee area

Fig. 1. Photos of the laboratory.

Table 1
Participants' profile.

Team	Role	Overseas training	IT work Exp. (yr.)	Organization Exp. (yr.)
A	Product manager	Yes	21	6
A	Product manager	No	16	7.5
A	Product designer	Yes	27	10
A	Software engineer	No	6	1
A	Software engineer	Yes	21	8
A	Software engineer	No	5.5	4
A	Software engineer	Yes	20	11
B	Product manager	Yes	19	0.5
B	Product manager	No	23	10.5
B	Product designer	Yes	5	4
B	Software engineer	Yes	10	4
B	Software engineer	No	15	11
B	Software engineer	No	7	7
B	Software engineer	Yes	5	5
–	Transformation lead	–	12	7

The teams. We observed *in loco* these two software development teams (see Table 1), which originated from ORG's financial department located in Brazil and closely supported one another given the dependencies between their product scopes. Both teams are composed of two Product Managers, one Product Designer, and four Software Engineers. These teams were supervised by a senior manager assigned as the sponsor of the transformation process.

Team A is responsible for a software product that generates and manages data about organization 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 calculates the associated costs of services offered by the products sold by ORG and displays this information to internal ORG consumers, while also generating profit data for each project, which is consumed (along with the rest of the data) by the accounting department. Team A is tasked with integrating all existing operations of the product into a single application that fulfills user needs and business expectations.

Team B is responsible for a software product that consumes data from other ORG applications (including Team A's) to calculate the average cost of equipment developed in Brazil. The application generates reports for internal accounting, such as inventory reports. The team also works on automating the validation process for the data coming from each source. Team B has to research current product processes to automate them into the application.

4. Research method

We conducted a single-case study [34] with the two software development teams described in the previous section. As both teams worked closely together, sharing the same physical work environment and forming the focal point of ORG's adoption of the combined approach in Brazil, we considered them as a single unit of analysis. We closely investigated both teams with the intent of discovering two things: how these teams use the combined approach (**RQ1**), and in what ways the combined approach influences development (**RQ2**).

We report on the responsibilities of the roles that constitute the teams and their workflow to answer **RQ1** (see Section 5); and on an overview of the teams' mindset using the combined approach and the influence of its pillars on the development process to answer **RQ2** (see Section 6). We collected data on both teams during the nine-month period they were stationed in the development lab dedicated to experimentation at our university's campus. We describe the data collection and analysis methods next.

4.1. Data collection

We used multiple data sources to conduct our study. We detail each data collection procedure next.

Questionnaire. To kick-off data collection and break the ice with the participants, we used a brief questionnaire to collect the participants' profile (see Table 1) during a welcome meeting followed by a coffee break to celebrate the ORG-PUCRS partnership in this research project.

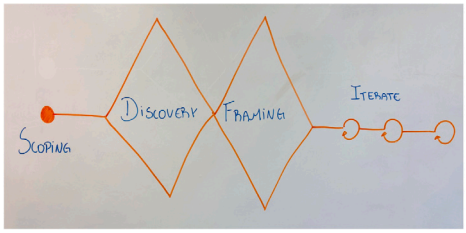


Fig. 2. Team B's development workflow outline [15].

Semi-structured interviews. Next, during the first month, we conducted an initial round of interviews with each participant to learn about their perceptions of the combined approach. In the third month, we conducted a second round of interviews with each participant to uncover their perceptions of role changes, interactions between roles, and the impact of the combined approach on their work routine. We invited all team members to reflect upon roles and responsibilities changes in the very last month they worked in our lab as part of our farewell party. This was done during a coffee break as people were invited to voluntarily post notes on the coffee space walls while we, researchers, checked on these notes and chit-chatted with the participants about them. Throughout the nine months of their stay, if time saw fit, we conducted interviews to learn more about diverse aspects (e.g., purpose of specific techniques) unveiled in our daily observations sessions. These interviews had their scripts prepared by the research team members as the research evolved and were validated with the senior researchers before being conducted.

Daily observation. We observed team ceremonies (e.g., retrospectives), stakeholder meetings (e.g., user interviews), and the participants' overall work routine. Observations were conducted on a daily-basis for two to four hours a day during the first three months and spaced out to a couple of times a week either decided on a random basis or whenever the team members announced that something new (e.g., end user bringing new needs to discussion) or interesting (e.g., an unforeseen outcome of an experimental study) was going on. We also shadowed each role during the fourth month to gather in-depth knowledge about their responsibilities and intricacies of how they performed their activities.

Focus group sessions. We conducted eight focus group sessions. In the first six sessions, which occurred in the third and fourth months, we discussed each of the pillars (2 sessions for each pillar) and confirmed data collected through other procedures (e.g., we discussed in-depth what we learned about the Product Designer role and the UCD pillar). For the seventh session, also in the fourth month, we had all the participants confirm our understanding of the teams' perceptions about the benefits and challenges of the combined approach [11]. In the last session, in the fifth month, we had the participants discuss the elements of each pillar as they perceived them (e.g., activities and techniques).

Workshop. In the seventh month, we had both teams clarify and illustrate their development workflow (see Section 5.2) in a two-part workshop.[4] In its first time slot, we gathered the teams together and asked them to confirm our understanding of their workflow activities, roles, and techniques [35]; in addition to explaining their context of use. In its second time slot, we had them sketch a rough outline (see Fig. 2) of their development workflow and discuss it in-depth, such as by examining similarities and differences between both teams.

All interviews, focus group sessions, and the workshop were voice recorded and transcribed for analysis upon agreement with the teams. We also took notes during all data collection activities, including during the observation and shadowing procedures. Interviews lasted 30 min on average, focus group sessions ran for about 1.5 h each, and both workshop time slots lasted approximately 1.5 h each.

4.2. Data analysis

We analyzed data following the content analysis procedure by Krippendorff [36], organized into the following steps: organization and pre-analysis, reading and categorization, and recording the results. Using Atlas.TI,[5] we first read the dataset, extracted text excerpts and marked them as codes. These codes were revisited and grouped into larger codes, forming categories. Fig. 3 depicts an example of the procedure. This iterative process was conducted by two researchers and was revised constantly by two senior researchers to mitigate any limitation or bias in our analysis.

We used the results of this analysis to provide a descriptive snapshot of the teams' combined approach as a whole. While some categories are made explicit (e.g., *shared responsibilities* in Section 6), we aggregated most of them to better illustrate some of the more prevalent concepts of the combined approach (e.g., roles in Section 5). That said, we can map specific collection procedures to each of our findings. For instance, we combined data obtained from observation with the workshop data to form a complete picture of the teams' work process. Our major findings are: the description and responsibilities of the **Roles** that make up the teams; the development **Workflow** used by the participants; the **Pillars' Influence** on the development process; and the **Teams' Impressions** of the combined approach, in regards to their mindset and its overall "feel". Table 2 maps each major research finding to data collection procedures—do note that the data analysis was constant (i.e., always the same strategy) and iterative (i.e., conducted and reported in a weekly-basis in the research team meeting).

During the case study's nine-month period, our research team held weekly 1.5 h meetings to organize, summarize, and review newfound data in insight cards and notes in order to discuss new understandings of the study and plan the following week's data collection procedures. Due to confidentiality reasons, the dataset cannot be shared, however, it is important to note that the research project underwent an Ethics Approval procedure with each party (ORG and PUCRS) and that the studied team members voluntarily signed a Research Consent Form. Our research team signed a non-disclosure and confidentiality term with ORG upon project approval with PUCRS.

5. A workflow for the combined approach

In this section we answer **RQ1** (How do teams use the combined approach in practice?) by characterizing the combined approach as used by the ORG teams. Section 5.1 details the roles that make up a combined approach development team. Section 5.2 examines the workflow followed by the teams.

Our interactions with the teams did not reveal any significant difference in their use of the combined approach, as both teams agreed with each others' opinions on operationalization matters during the data collection procedures in which both were present (focus group sessions and the workshop). Our analysis of the interviews also suggested this.

5.1. Roles

Inspired by Pivotal Labs, the teams are composed of three roles, Product Designer, Product Manager, and Software Engineer; each embodying the general aspects of the combined approach's pillars—UCD (design), Lean Startup (product development), and XP (development), respectively. Other essential roles embedded in the pillars, such

[4] We explicitly differentiate the focus group sessions from the workshop for clarity as the latter had the concrete objective of creating an artifact.

[5] atlasti.com

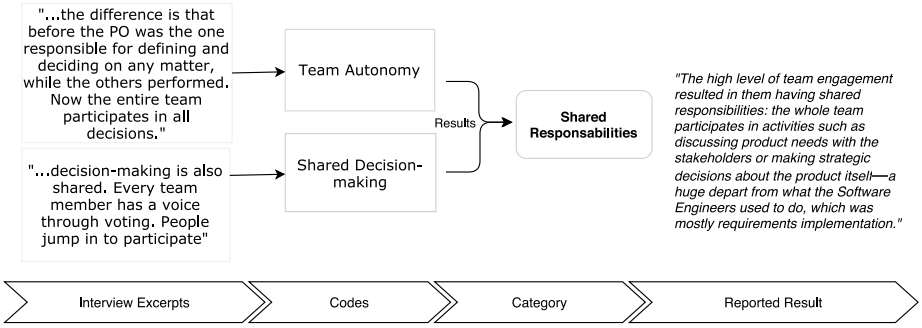


Fig. 3. Code analysis example.

Table 2
Mapping of major research findings to data collection procedures.

Finding	Semi-Structured Interview Questions	Daily Observation	Focus Group Sessions	Workshop
Roles	What is your role on the team and what are your main responsibilities? Could you tell us more about the role's responsibilities? How have these responsibilities changed over the transformation? i.e., what is new?	✔	Sessions 1–6 discussing the pillars	—
Workflow	—	✔	—	One workshop to discuss and outline their workflow using the combined approach
Pillars' Influence	Could you describe a typical work day?* What differences did you observe in retrospect regarding the previous work model and the current one in the transformation so far?** In your perception, what are the technical factors that should be adopted and practiced to support a good delivery process?	✔	Sessions 1–6 discussing the pillars Session 7 on benefits and challenges of the approach Session 8 on elements of the pillars	—
Teams' Impressions	Could you describe a typical work day? (*same question) Did the activities you perform on the previous work model change? What were these changes? Were new responsibilities assigned? What differences did you observe in retrospect regarding the previous work model and the current one in the transformation so far? (**same question) In your opinion, what are the factors that influence your team's engagement? In your perception, what factors contribute to the relationship of trust established between the stakeholders and the team? In your perception, what factors have contributed to a closer relationship with stakeholders and users? What contributes to the team's autonomy?	✔	Session 7 on benefits and challenges of the approach	—

as XP's Tester, are not explicitly used: any essential development activities are performed by the three aforementioned roles. For instance, testing was conducted by Software Engineers. By having roles coupled to the pillars, teams have an easier time answering questions about the product they are developing: if it solves the users' problem, if it helps the business, and if it is feasible to develop it. A common team is comprised of at least one member on each of these three roles, typically with a greater number of Software Engineers as they do the bulk of development work while in pairs. We describe each role next.

Product designer. The Product Designer (PD) is a multi-disciplinary facilitator that enables the team's communication with the user. Their number one priority is to represent the user and to contribute to the development process by delivering value in the form of design decisions. A PD does this kind of work throughout the whole development process, not just during the initial phases of development when the product is still uncertain. The PD actively uses techniques such as exploratory research, interviews, and journey maps to get to know the user and "extract" their essence, working closely with the Product Manager so as to be an advocate for the user's favor whenever possible.

> *"The PD helps the team in conducting techniques like affinity and journey mapping and in identifying the user's problem in general. They suggest the use of techniques that foster the team's empathy with the user, and also help the team in addressing user pain points through the product. They work closely with the Product Manager".*
>
> [Team A]

A PD needs to be empathetic with their users while also helping the rest of the team in developing the same empathy towards them: by validating the team's perception of what the user's problem is through the use of various techniques (e.g., wireframes, low and high fidelity prototypes, and mockups) and direct user validation (e.g., A/B tests), the team gets to understand the user's pain points, consequently having an increase in empathy towards them. This process helps in guaranteeing that the team shares the same vision of the problem and the solution, in addition to addressing the user's needs.

"We need to build a relationship with stakeholders, showing them what we are doing and our results so far. This way we become more trusting of one another, as both sides are aligned and understand what each other is doing and their needs".

[Team B]

Product manager. The Product Manager (PM) acts as a business representative, helping the team to deliver products with meaningful value to the organization. As the UCD pillar pushes to have the user at the center of the development process, the PM must understand and address their needs from a business perspective, acting as a moderator that balances their needs with the expectations of stakeholders. As a business advocate, the PM constantly considers the product's impact on the business as a whole, and thus works closely with the PD to iron out any issues that may arise. The PM helps the team in defining the vision of the product, their strategy to solving the user's problem, and encourages the use of experimentation, defining assumptions that originate from business concerns. They schedule meetings with the stakeholders to verify if the team is aligned with their goals, ensuring that the team is treading in the right direction and delivering the right product to the organization.

"This role helps the team to address the business needs in the product through value mapping, creating and validating assumptions, helping the team manage the product backlog, and also helping during experiments. A PM helps the team focus on the problem and work with a discovery-oriented approach".

[Team B]

Software engineer. The Software Engineer (SE) is a software developer responsible for implementing solutions and the environment in which they are developed in. The majority of technical tasks (i.e., programming) are performed by the SE, who constantly strives for technical excellency by mastering development techniques and staying up-to-date on modern software trends so as to better support and manage the constant changes that experiments require. The SE also actively participates in other non-technical activities, such as user interviews and experiment definitions, contributing with their technical perspective.

"The software engineer has the responsibility of guaranteeing the environment to the develop the solution on, implementing a pipeline that employs continuous delivery and integration. The SE also participates in each decision made by the team since the product's conception, while also participating in user interviews, stakeholder meetings, and other ceremonies".

[Team B]

A specialization of it, the Anchor, also following Pivotal Labs guidelines, is an engineer that assists in overall communication by serving as a bridge from the SEs to the user and business stakeholders. The SE that takes on this role does not necessarily have to be the most experienced one on the team, but the role's heavier emphasis on soft skills calls for a moderate amount of knowledge of non-technical issues.

"One of the roles is the Anchor: a Software Engineer with great communication skills".

[Team B]

Both teams decided on rotating this role from time to time as a means to promote knowledge sharing among team members and familiarity with users.

"Since we don't want users to think that they should go back to the model in which we had focal points to mediate conversation with business people, we are rotating the Anchor role among ourselves to avoid attachment to a certain person and help each one of us to get to know our product's users better".

[Team A]

5.2. Workflow

Both teams reported that their workflow (see Fig. 4) is divided into three major phases, as they progressively gather an increased understanding of the problem they are trying to solve and what type of solution might fit best. Their work approach gradually morphs into a typical development approach more akin to "standard" agile methods, though it never completely does so. Although reported in a sequential manner, the teams stated that they sometimes move back and forth between phases, as dictated by the situation they find themselves in, conferring an "organic" attribute to problem solving.

Their workflow begins with the **Scoping phase**, in which teams aim to discover the scope of their work by interacting with stakeholders. Just as the product development cycle begins, the UCD pillar promptly starts exerting its influence as the teams host a meeting to explore the totality of the vision they will embark in by discussing the problem to be solved, mapping stakeholders, expectations, etc., in addition to determining and securing resources, such as specialized tools or extra personnel. They also take the opportunity to outline an approach to solve the stakeholders' demands by means of a brief brainstorming session, setting the pace of development.

"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".

[Team B]

By mapping all the initial elements that make up the context of the problem they will work on, the teams obtain a slate that effectively defines their work for the upcoming time period by pointing towards the "leads" that they will have to follow in order to properly satisfy stakeholder demands: stakeholders to interview, a problem vision and problem statement (the "big picture" and specifics of the stakeholders' demands, respectively) to explore, and an initial strategy to figure out how to solve the problem.

Next comes the **Discovery and Framing phase**, in which teams use the British Design Council's Double Diamond to refine their understanding of the problem to solve and to determine the right solution to develop, given the entire context the problem is situated in. It is in this phase that the Lean Startup pillar starts to emerge, as the concept of the build-measure-learn cycle is introduced and immediately takes a hold of the development process—its pervasive use conferring the same data-driven approach to decision-making that makes up the pivot-or-persevere mechanism of Lean Startup.

"The Discovery stage enables the whole team to understand the problem meticulously. Our goal in the Framing stage is to identify multiple solutions to the problem we are solving".

[Team B]

In its first stage, *Discovery*, the team focuses on attaining a deeper understanding of the user's problem and finding its root causes by running build-measure-learn cycles. Their activities in this stage are not experiments as one would do in a straight Lean Startup approach, but the teams still use the build-measure-learn nomenclature as they *"build interview scripts, measure the efficacy of the script, and learn to do better next time"*. By repeatedly employing UCD practices (e.g., collecting feedback with interviews) they obtain a more specified problem statement to work on, in addition to refining their vision of the context that the problem is a part of. Throughout cycles, they settle on a persona of the user and establish a list of assumptions as to what the root problem is, eventually uncovering metrics that assist them in defining what the most likely root problems are, which are then ranked and prioritized.

With a properly defined problem, the teams begins the second stage, *Framing*, in which they explore different solution ideas through the use of experimentation, culminating in the selection of a solution idea

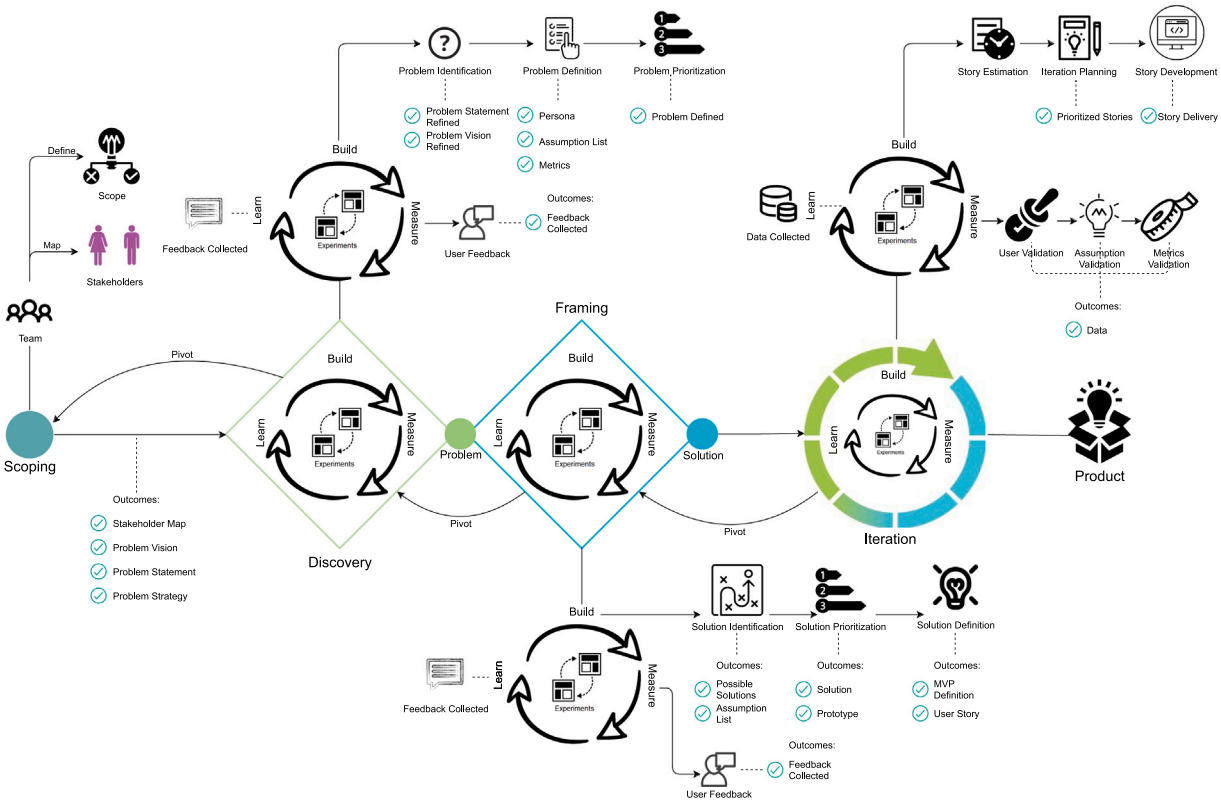


Fig. 4. Workflow overview of the combined approach.

that is decidedly better than the others, as suggested by experimental data. To identify the proper solution to the problem at hand, the teams again make use of build-measure-learn cycles to generate a list of possible solutions and the core assumptions that each one holds true (e.g., refactoring a spreadsheet system will improve its response time significantly). As the teams iterate, they experiment with prototypes and collect user feedback to narrow their solution choices down to the ones most likely to succeed. The Product Designer role is heavily accentuated in this phase, acting as a constant facilitator for UCD practices (e.g., user interviews) and guiding the team to the proper understanding of the user's needs. After defining the proper solution to the problem, they establish a Minimum Viable Product (MVP) by writing User Stories.

The final phase is the **Iteration phase**, in which the solution chosen in the previous phase is repeatedly developed upon and properly implemented. This phase is characterized by its heavy use of XP and similarity to typical agile development, which includes the use of popular practices such as planning poker and development iterations. As the teams estimate, prioritize, develop, and deliver User Stories validated by stakeholders in weekly meetings, the teams still constantly push for new experiments: they establish new assumptions and devise new experiments to gather data that might steer the course of the solution (e.g., validating the need for a new feature), all in service of improving it overall.

> *"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".*
>
> [Team A]

Experiment and user feedback data points are used together to plan out the next steps in developing the solution, and this is done until all stakeholder demands have been satisfied. If at any point the teams find out through new data that either the problem or the solution they have been working on is faulty in some way, they pivot back to either the *Discover* or *Framing* stages to explore new alternatives.

Table 3
Key aspects of using the combined approach.

Aspect	Summary
Roles	Each pillar is upheld by a specific role. **Product designer** (UCD): an advocate for the user, assists the team in exploring user needs through the use of UCD techniques, such as personas and user journey mapping. **Product manager** (Lean Startup): an advocate for business stakeholders, assures that the solution being developed is in the best interests of the organization and assists the team in defining experiments to do so. **Software engineer** (XP): implements the solution itself and sets up their development environment with CI/CD capabilities to better support experimentation, while also actively participating in the activities proposed by the other two roles. One Software engineer is designated as Anchor, and has to bridge the communication gap between engineers and user and business stakeholders.
Workflow	The workflow of the combined approach is organized into three sequential phases, but teams can return to any previous phase of development if they find what they are working on to be unfitting. **Scoping**: An overview of the problem to be solved is defined and affected parties are mapped. Organization resources required for development are secured. **Discovery & framing**: Development immediately turns its focus towards experimentation. The problem space is explored with the user and a sensible solution that fits the organization is determined. **Iterations**: The solution is developed iteratively as with common agile methods, with users and business stakeholders being consulted weekly.

Table 3 shows a summary of the teams' operationalization matters.

6. Influence of the combined approach on development

In this section we answer **RQ2** (How does the combined approach influence development?) by reporting on the teams' perception of the combined approach. Section 6.1 reports on how the teams see the influence of the pillars during development. Section 6.2 describes the teams' impressions of the combined approach overall, explaining its perspective to software development.

6.1. Influence of the pillars

Each of the pillars that make up the combined approach brings a set of qualities to the development process, from ceremonies and techniques to methods and attitudes. In this section we outline the influence of each pillar from the perspective of the ORG teams.

6.1.1. User-centered design

The teams considered the use of a variant of the British Design Council's Double Diamond framework to be the most significant change imparted by the UCD pillar. They emphasized that the use of this framework is a consequence of the problem-oriented perspective and that the whole team participates in its activities. Its use promotes a closer relationship between them and the stakeholders, increasing the stakeholders' trust of the team.

> *"Using UCD techniques and being more empathetic with our stakeholders makes them feel essential to the development process, consequently encouraging them to contribute and talk with us. The business considers us to be very important to their success, while our users see us as problem solvers. We gain their trust when we show interest in providing a product that solves their problems".*
>
> [Team B]

The framework enables the teams to explore the problem they were tasked with solving, obtaining a deep understanding of it, and to discover several potential solutions to it. Having stakeholders position themselves as empathetic partners was reported to be essential to this process, more so than in the other phases of development. The participants state that constantly working with stakeholders and collecting their feedback is vital to promoting their engagement and discovering every relevant detail about the problem at hand:

> *"We use stakeholder feedback as a tool to refine and redefine problem definitions and priorities. Being aligned with the stakeholders' needs makes them more confident about our work. We work together with stakeholders, ensuring that we are discovering the right path to guide the product's development".*
>
> [Team A]

6.1.2. Lean startup

One of the most powerful concepts drawn out from Lean Startup from the teams' point of view is the use of experimentation—the build-measure-learn cycle. The participants state the cycle to be an ubiquitous mechanism throughout development:

> *"We use build-measure-learn all the time, in any part of our process. For example, in a user interview. If we are defining the interview script, we are building a script. We measure the script's value by checking after the interview if we collected the right data or not. For instance, a stakeholder answers our questions, but our questions did not bring about the answers we were looking for. This process allows us to learn from our failures and create a more assertive script, so we can be more accurate next time. Build-measure-learn is applicable to any product development activity".*
>
> [Team A]

Experimentation lies at the core of the build-measure-learn cycle. The teams make assumptions about everything concerning development, from what might be the cause of the stakeholders' problem to how efficient a solution could be, and then experiment in order to confirm or refute them, assisting them in understanding the problem and finding out optimal solutions:

> *"At the foundation of the build-measure-learn cycle is the concept of experimentation. We work with a problem-oriented mindset because experimentation allows it. For instance: in the beginning, we have a simple view of a problem and this leads us to start making assumptions, executing experiments using prototypes or other techniques. The results enable us to measure things about the problem and to refute or accept our assumptions. At the end of the cycle, we learn from the results and restart the loop, refining our vision of the problem".*
>
> [Team A and B]

By confirming or refuting assumptions, the teams can "pivot" away from wrong problem understandings and bad solutions, reducing development waste that would have been spent on exploring them. This practice of "pivoting" confers a strategic element to the adaptability already present in regular agile methods, as teams now have experimental data to fall back on and make more assertive decisions:

> *"Experiments give us the conditions to understand if we are doing a smart strategy for our product or not. Sometimes, the strategy defined for the long-term may not be valid anymore. Also, our relationship with the stakeholders is an essential factor to persevering in a strategy or pivoting to another direction".*
>
> [Team B]

Additionally, experimentation provides the teams with more room for failure as it makes wrong courses of action be stopped at early stages, with experimental data also giving them the pointers they need to fix mistakes quickly:

> *"Product development is uncertain and very susceptible to failure. Nevertheless, what matters is the speed at which we will react to those things. The experimentation at the core of build-measure-learn gives us room to fail but also allows us to fail and fix things quickly. We do not need to wait until the end of an iteration to discover that we do not understand the stakeholder's needs".*
>
> [Team A and B]

The participants also shared an unusual experiment case which resulted in a non-software solution, an experience that solidified the benefits of experimentation in the eyes of the team:

> *"Our users were facing problems when using spreadsheets with a substantial amount of data. They were spending a lot of time in this one application, usually about three days between calculating what they needed and waiting for the results. Based on that and after analyzing the application's architecture for possible performance bottlenecks, our assumption was that maybe the problem was with the host machine's performance and not with the application that generates the spreadsheet itself. So we defined a hypothesis that a computer with better specifications would be able to handle the workload, and got to confirm it. After that, we suggested that the host machine needed some new parts and we managed to solve the problem without having to write a single line of code. This experience showed us the importance of experimentation—and more than that, it showed us that sometimes the solution can be a non-software one, which is a huge breakthrough for us".*
>
> [Team A]

By solving the users' problem without delving into proper software development, Team A managed to save the organization a considerable amount of money that would have been spent on development expenses and man-hours. The "out-of-the-box" solution further consolidated how they were seen as problem solvers rather than just software developers.

6.1.3. Extreme programming

The XP pillar strives to provide technical excellency to the developers by instituting practices such as pair programming, unit testing, and test-driven development. The teams highlight how they benefit from it, but in regards to pair programming, what is interesting to note is how this technique was revisited into pair-wise work: the teams work in pairs most of the time, be it programming, developing interview scripts, or defining new assumptions to experiment on.

> *"We benefit from pair programming in many ways: from accelerating the learning process of a new engineer, to promoting improvements in code quality. We actually work in pairs in most activities, on the single table that we have. Our productivity increased despite all the odds".*
>
> [Team B]

The teams reported significant change to their work process when adhering to XP due to the new ceremonies. Of note is the "office stand up" in which both teams (each working on different problems) discuss technical hurdles and give status updates on their work to one another.

> *"We tried to be more aligned with ourselves, and the ceremonies are useful for that. We continued doing the stand up meeting, retrospective, and planning. However, we now have an office stand up to be more connected to other teams—also, the ceremony nomenclature changed from sprint to iteration. In the planning sessions, we choose if we must have more than one session—for example, a pre-iteration meeting. Finally, we have weekly sessions with all stakeholders to strengthen our relationship with them further".*
>
> [Team A and B]

The teams highlighted the use of the already in-place practice of establishing a continuous integration and continuous delivery (CI/CD) pipeline, as they considered it essential in enabling their problem-oriented approach and in improving the morale of Software Engineers:

> *"A CI/CD pipeline is crucial to handle the changes. It promotes faster feedback and helps us validate stories on the production environment. CI/CD makes software engineers feel more accomplished, leading to better quality code later".*
>
> [Team B]

Lean Startup's build-measure-learn cycle was reported to directly impact how they deal with development iterations, as the development of the product is completely ingrained with the sense of thinking systematically and conducting experiments. The participants state that their adoption of the combined approach is successful due to their opportunistic use of the different pillars:

> *"Even though our process is represented in a sequential or continuous vision of the combination of the three methodologies, our daily use of it is adapted. If we are doing development iterations and notice that the problem is not well defined, we find it okay to come back to the Discovery and Framing framework and start again. Alternatively, if we defined some assumptions and discovered that the product or problem vision is not well aligned with them, we can redefine these assumptions. This is the secret of the adoption, applying the combined approach organically".*
>
> [Team A]

The combined approach gives the teams the tools they need to systematically solve problems and expects them to engage in pursuing solutions without relying on a strict process, thus having the team think critically and drive the development effort themselves.

6.2. Teams' impressions

The combined approach has been described by the study participants as an "organic" way to solve problems, in contrast to the waterfall-like approach they were used to employing at ORG before the transformation to Agile, and even still when compared to Agile itself given the combined approach's exploratory and experimental nature. Before and whilst diving into actual software programming, the teams constantly question the problem they are actively solving, clearing their doubts only by trusting "real" sources of truth: user feedback and other data acquired through experimentation. The teams use this experimental data to fuel their product discovery process.

> *"We aggregate value to our products by using experimentation. We explore the problem that the business wants us to, and by the end of it, we address their needs through a product".*
>
> [Team A]

The constant questioning changes the way in which teams approach issues (e.g., stakeholder demands, architectural decisions) to a strategic one, as they look to be as effective as possible with their solutions. This in turn leads to an interesting outcome, as they realize that not all problems are to be solved by software: although a rare occurrence, the teams have reported solving some stakeholder demands without having to write a single line of code.

The chosen pillars of XP, UCD, and Lean Startup respectively provide direct support to development, product design, and product management issues with the myriad of techniques that they each refer to. Our interactions with the teams, however, suggest that its not the techniques themselves that make the combined approach prosper, but rather the "team environment" (i.e., shared mindset) that enables their use in an effective and opportunistic fashion.

A problem-oriented mindset. The ORG team members stated that one of the most impactful changes experienced during their transition to the combined approach was that of moving to a discovery-based approach to development, which pushes them to actively search for the answers they need and the questions that need answering themselves, making them embrace a problem-oriented perspective instead of one of "implementing software requirements" as they previously used to:

> *"Previously, we usually received a set of predefined software requirements. We implemented these requirements and considered our work to be done, not knowing if the underlying problem was actually solved. Now, we investigate, understand, and participate in the problem".*
>
> [Team A]

The participants mentioned that this change in mentality was a challenge at first, as it directly affects the team's whole attitude towards development: the new mindset demanded the teams to start acting as the main actors of product development and not just as those who operationalize it—whereas before business people would provide software requirements directly, the teams now had to interact with stakeholders and discover the requirements themselves.

High levels of team engagement. The teams' perception of their commitment to the entire software development process has improved considerably since the adoption of the problem-oriented mindset. Early on, the teams recognized the need to move to a discovery-based development approach that provides an improved understanding of the product as a whole, paving the way for more business-aligned solutions. This realization led them to understand that several new aspects of development depend directly on their involvement, and that everyone on the team should be engaged with them.

> *"Everyone needs to understand the product, not just the Product Designer or the Product Manager—the Software Engineer is no longer isolated. The entire team needs to know why the product is working and understand the vision behind it. Everyone is always up-to-date".*
>
> [Team B]

Shared responsibilities. The high level of team engagement resulted in the teams having shared responsibilities: the entire team participates in activities such as discussing product needs with the stakeholders or making strategic decisions about the product itself—a huge departure from what the Software Engineers used to do, which was mostly requirements elicitation and implementation. By establishing a relationship between the whole team and the stakeholders, having a consistent problem understanding throughout the team becomes easier; and having the Software Engineers actively contribute to the product conversation can provide novel technical (or non-technical) insights, hopefully resulting in a better solution overall.

> *"We have the responsibility to guarantee the environment to develop the solution on, setting up our pipeline with continuous delivery and integration. However, we are now responsible for discovering what we are supposed to be doing in a product-sense, participating in each decision made since the product's conception. To that end, we interview users ourselves, discuss business needs with stakeholders, and so on".*
>
> [Software Engineers from Team A and B]

The combined approach encourages an increased level of proactivity as the teams empathize with and get to see issues from the perspective of users and the business, pushing them to active problem-solving as they better understand the impact and fruitfulness of their work. Combined with the data-driven approach of Lean Startup, the teams are able to solve problems in ways beneficial to the whole organization, for instance, by taking into account the cost of development for different types of solutions to a given problem.

> *'Back when our work was based on release plans, there was no room whatsoever for us to experiment and fail. With our new continuous development and release approach, we can explore, test, and pivot to different solutions. The extra time we get from this provides us room for value-driven development".*
>
> [Team A]

As mentioned in our previous studies on this case, the teams reported a series of perceived benefits after adopting the combined approach, from increased code quality and trust among team members [11] to rapid feedback and reduced development effort [9]. The combined approach appears to be a suitable fit for ORG, as senior staff members continue to invest organizational resources in it with the intent of spreading it throughout the whole organization.

> *"Users are happy with the results, and senior managers are positive [the transformation] will work in the long run".*
>
> [Team B]

While the teams are consistently satisfied with the combined approach itself and its development results, they have expressed several concerns with ORG properly adhering to it due to culture clash [11]. The teams require a lot of autonomy to make proper use of the combined approach, but the rest of ORG is used to employing strict project deadlines and other rigid measures [12], affecting their development process. For instance, they were unable to deploy a solution for two months as managers claimed the production environment was unstable, and one of the participants feels like they will be pressured to abandon the combined approach if management does not take a stand for it at ORG's upper management decision levels [12].

Table 4 shows a summary of the combined approach's influences.

Table 4
Key aspects of the combined approach.

Aspect	Summary
Overview	The combined approach is a software development methodology backed by **XP**, **UCD**, and **Lean Startup**. Each of its pillar tackles different issues (development, product design, and product management, respectively), and together they encourage a **problem-oriented mindset**, pushing team members to **high levels of engagement** as they actively pursue assertive solutions through the use of experimentation and carry **shared responsibilities** when taking into account business needs and user concerns to improve developed solutions both in a usability and business sense.
Benefits	The ORG teams perceived a series of benefits when using this combined approach, from increased code quality and trust among team members [11] to rapid feedback and reduced development effort [9].

7. Discussion

The workflow used by the teams can be interpreted as a somewhat lightweight process, as the participants stated that they use it in an "organic" fashion, as both product-scale and moment-to-moment pivots (i.e., opportunistic use of techniques and phases) confer great adaptability to it. Their workflow provides a series of steps that when taken lead to the discovery and eventual solving of stakeholder problems, and as the decisions made during the development of a solution are backed by data, teams can work with confidence knowing that their efforts are being put to good use.

There are other studies that report on approaches similar to the teams' workflow in scientific literature [8,10,13]. While Converge [10] and InnoDev [13] were presented in a lower level of abstraction, reporting on roles and activities yet untested in an industry scenario, and Discovery by Design [8] in a higher level [8], reporting on their approach as a whole from an empirical perspective in the industry, our study strikes the balance between both ends of the spectrum by reporting on operationalization matters of the combined approach in addition to providing rich detail on the inner workings of the approach grounded from a team-level industry perspective.

Although the teams' workflow is not so different from the similar approaches, it has its particularities. Discovery by Design [8], Converge [10], and InnoDev [13] all bridge Agile and UCD by employing an upfront design phase, a commonplace "fix" to some Agile and UCD integration issues [37]. Unlike typical practice, however, the ORG teams explicitly involve all members in the so-called iteration 0 [37] instead of having designers "one iteration ahead" of engineers [37]. The teams' focus on experimentation also ends up improving the integration effort: with the entirety of the team perceiving their development effort as a series of experiments, it is easier to convince themselves to go back to a design phase and adapt their product when there is data justifying as much, mitigating any resistance to process and culture change one would expect [37], in addition to helping chunking design [37] down into experiments. This was not as transparent to ORG stakeholders external to the team, however, as they were still resistant to the combined approach [12], even if the change to a discovery-oriented perspective forced the teams to build more rapport with stakeholders to properly uncover problems and define effective solutions.

Much of the teams' perceived improvements can be traced back to Lean Startup, as its pervasiveness of the build-measure-learn cycle (i.e., continuous experimentation) enables companies to make well-informed strategic decisions based on customer data [19]. This rationale of gathering information to make assertive decisions is used by the ORG teams to take sensible courses of action at every turn of product development, resulting in a more adequate solution both in a technical and business sense. The use of continuous experimentation brings about several benefits, such as deeper customer insights and reduced development effort [22], deeper insights on customer needs,

behavior, and priorities [22,38]; and possibly improving cycle times due to the required risk-tolerant approach that increases room for failure [39]. Running such experiments requires a certain degree of autonomy and freedom for the teams to explore business-level issues. Such empowerment can enhance the teams' morale [40] and overall efficiency, as they are better suited to handle some decisions due to being in direct contact with customers and technical matters [39].

Cross-functional teams have been reported as crucial to new product development for decades [41], and the teams pointed out the importance of having Product Designers amongst them: even though caring about and understanding the users' needs is a responsibility shared by the entire team, the Product Designer's expertise and dedicated focus in conducting UCD techniques is invaluable to having these needs well-understood. The presence of the Product Manager role mirrors this on the business side, with the relationship of IT and business stakeholders greatly improving as new products properly address their needs as well. Having Software Engineers present during product conception activities and during business decision-making meetings highlights how the role is not limited to technical skills in this discovery-oriented approach, and helps ease the burden of UCD activities [37] on the Product Designer.

While ORG has other teams working on products for external customers, the case study teams have only worked for internal customers so far. We cannot say for sure if the combined approach could work throughout (or if it is appropriate for) the whole company due to its sheer size and variety of software projects with different needs [42]. Nordstrom [8] addressed this by using the combined approach specifically for innovation—creating a self-contained team to test new ideas and products. The case study teams, however, specifically employ XP—which brings additional engineering practices in contrast to their old use of Scrum. Tessem [43] highlights how empowered agile teams accentuate the use of low-cost techniques that promote information flow in order to make better decisions, much like how the ORG teams share responsibilities among team members and employ pair-wise work, decreasing knowledge silos and enabling all members to contribute to decision-making, which should bring at least some net benefit to ORG.

Although we know that each software product comes with its own context and the combined approach as reported might not be well suited to some, we derived pragmatical recommendations to using it:

- Ensure that teams understand that they have to look at the development process through "problem-solving" lenses to make the combined approach work.
- Ensure that the team is working on several activities together and not having each pillar completely upheld by a single role.
- Have Product Designers and Product Managers work closely together to better align the user's needs with the organization's.

7.1. Limitations

As with any empirical study, ours is limited in several aspects (e.g., generalization concerns [34]). A limitation specific to our study is that we did not have explicit contact with the teams' customers due to our research contract with ORG, as they were located outside of Brazil. However, we were allowed to use any customer and user data collected throughout any of our data collection procedures with full awareness of these individuals. Even with the explicit absence of direct contact, we managed to observe several activities in which users and customers were involved, thus allowing us to realize how cooperative, in line with the transformation, and fond of the combined approach they were.

Our case study paints a somewhat idyllic picture of the combined approach. From a more cynic point of view, we could interpret the general positivity of the participants towards the combined approach as an attempt to convince upper management that the investments made on the teams were worthwhile, though we think this is quite unlikely given the extensive period we observed the teams; or that it was brought by subconsciously due to the dedicated lab's improved facilities and cheerful atmosphere compared to their old work site.

We made use of several techniques to improve the quality of our study, and describe them in regards to specific quality tests [34] next. We do not address internal validity concerns as our study is of an exploratory nature [34].

Construct validity. As a premise to construct validity, we defined our subject of change by asking each participant how they were used to developing software. This was essential as it provided us with a baseline to better understand what changed with the adoption of the combined approach. We had multiple researchers conduct the data collection and analysis procedures to reduce bias, and also used multiple methods and sources of data to triangulate our findings, in addition to using member checks (i.e., confirming our interpretations with study participants). We also had senior researchers validating draft case study reports with two keen study participants. Alongside the use of multiple data sources, both are tactics commonly used to address construct validity [34]. We made sure our research work was as transparent as possible to the teams, to avoid any misguided suspicions that we might have been harming the teams in some way, such as by conducting performance evaluations on behalf of ORG. Our research team coexisted daily with the teams throughout the extensive time period they were stationed at our campus. Indeed, some members of our research group even played ping-pong with the teams outside work hours. While this could have influenced our analysis somewhat, we think that the long exposure between participants and researchers led to increased trust, opening the teams to more honest dialogues about their situation.

External validity. The study was conducted in a single organization (see Section 3), posing a threat to its external validity. We sampled two teams developing two different products to mitigate this, although we analyzed them both as a single cohesive unit. We also highlight that both teams had the support of one another when using the combined approach, having been stationed in the same environment for this purpose. This fact could have made the use of the combined approach an easier and more fortunate endeavor—all negative statements reported by the teams were not made towards the combined approach itself, but instead to contextual organizational issues that interfered with it [12]. Nevertheless, the two teams had a substantial level of empowerment, granting them the autonomy to work as they saw fit in the two distinct software products they were each working on. We believe this makes our study more generalizable to software development teams inserted in different contexts, however we cannot assertively claim so as several factors need to be considered during the adoption of a development approach, from team maturity and organizational vision, to specifics of unique instances of the combined approach.

Reliability. Although case studies are seldom reproducible, we made sure to document the entire study [44]. We also made use of the aforementioned triangulation efforts to make our data more consistent and dependable.

8. Conclusion

We reported on a case study of two product-oriented teams using a development approach that combines Agile (XP), UCD, and Lean Startup. We characterized this combined approach through the perspective of both teams, revealing key elements that constitute it: the mindset of the teams when using it, the roles that make up the teams adhering to it, its workflow, and how its pillars affect development—while XP provides tools to achieve technical excellence, UCD helps draws in more knowledge from the user, and the constant experimentation from Lean Startup guides the development process. With the combined approach, teams are given the tools to gather information and experiment, compelling them to think critically upon experimental data in order to determine the next courses of action. For academics,

our study contributes with details on the essence of the combined approach backed by empirical evidence, providing a rich snapshot of the workflow and mindset of industry practitioners that use it.

We also reported the study from a pragmatic viewpoint by describing operational aspects of the combined approach, detailing the roles and workflow utilized by the teams, as well as providing practical recommendations to using it. Practitioners can take advantage of the study to kick-start their own adoption process of the combined approach, even more so if their organizational contexts are similar to ORG's.

For future work, we suggest an experiment to determine how the combined approach fares when compared to other development approaches (e.g., pure XP or InnoDev), so as to quantitatively determine its strengths and weaknesses. As for our endeavors, after nine months of studying these two teams, we are ready to assist ORG in assessing how mature other teams participating in the transformation rollout are, particularly from a self-evaluation perspective in order to reduce costs. As previously mentioned, we have been developing a model [33] that enables teams to perform context-informed assessments and identify what actions they can take to accelerate their transformation.

CRediT authorship contribution statement

Maximilian Zorzetti: Investigation, Data curation, Writing – review & editing. **Ingrid Signoretti:** Conceptualization, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Larissa Salerno:** Investigation, Formal analysis, Data curation, Writing – original draft. **Sabrina Marczak:** Conceptualization, Methodology, Validation, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Ricardo Bastos:** Conceptualization, Resources, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We acknowledge that this research is sponsored by Dell Brazil using incentives of the Brazilian Informatics Law (Law no. 8.2.48, year 1991). Sabrina Marczak thanks the financial support by CNPq, Brazil (grant no. 307177/2018-1).

References

[1] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al., Manifesto for agile software development, 2001.

[2] E.J. Barry, T. Mukhopadhyay, S.A. Slaughter, Software project duration and effort: An empirical study, Inf. Technol. Manag. 3 (1–2) (2002) 113—136, http://dx.doi.org/10.1023/A:1013168927238.

[3] T. Lindberg, C. Meinel, R. Wagner, Design thinking: A fruitful concept for IT development? in: Design Thinking: Understand – Improve – Apply, Springer, Berlin, Heidelberg, 2011, pp. 3–18, http://dx.doi.org/10.1007/978-3-642-13757-0_1.

[4] P. Gregory, L. Barroca, H. Sharp, A. Deshpande, K. Taylor, The challenges that challenge: Engaging with agile practitioners' concerns, Inf. Softw. Technol. 77 (2016) 92–104, http://dx.doi.org/10.1016/j.infsof.2016.04.006.

[5] R. Verganti, Leveraging on systemic learning to manage the early phases of product innovation projects, R&D Manag. 27 (4) (1997) 377–392, http://dx.doi.org/10.1111/1467-9310.00072.

[6] K. Vilkki, When agile is not enough, in: Proceedings of the International Conference on Lean Enterprise Software and Systems, Springer, Helsinki, Finland, 2010, pp. 44–47, http://dx.doi.org/10.1007/978-3-642-16416-3_6.

[7] D.A. Norman, S.W. Draper, User Centered System Design; New Perspectives on Human-Computer Interaction, Lawrence Erlbaum Associates, New Jersey, USA, 1986.

[8] B. Grossman-Kahn, R. Rosensweig, Skip the silver bullet: driving innovation through small bets and diverse practices, in: Proceedings of the DMI International Research Conference, Design Management Institute, Boston, USA, 2012, pp. 815–829.

[9] B.P. Vargas, I. Signoretti, M. Zorzetti, S. Marczak, R. Bastos, On the understanding of experimentation usage in light of lean startup in software development context, in: Proceedings of the Evaluation and Assessment in Software Engineering, ACM, Trondheim, Norway, 2020, pp. 330—335, http://dx.doi.org/10.1145/3383219.3383257.

[10] B.H. Ximenes, I.N. Alves, C.C. Araújo, 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, 2015, pp. 356–367, http://dx.doi.org/10.1007/978-3-319-20886-2_34.

[11] I. Signoretti, S. Marczak, L. Salerno, A.d. Lara, R. Bastos, 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, 2019, pp. 1–6, http://dx.doi.org/10.1109/ESEM.2019.8870154.

[12] I. Signoretti, M. Zorzetti, L. Salerno, C. Moralles, E. Pereira, C. Trindade, S. Marczak, R. Bastos, Success and failure factors for adopting a combined approach: A case study of two software development teams, in: Proceedings of the International Conference on Product-Focused Software Process Improvement, in: Lecture Notes in Computer Science, vol. 12562, Springer, Turin, Italy, 2020, pp. 125–141, http://dx.doi.org/10.1007/978-3-030-64148-1_8.

[13] F. Dobrigkeit, D. de Paula, 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, 2017, pp. 319–328.

[14] K. Beck, Embracing change with extreme programming, Computer 32 (10) (1999) 70–77, http://dx.doi.org/10.1109/2.796139.

[15] I. Signoretti, L. Salerno, S. Marczak, R. Bastos, Combining user-centered design and lean startup with agile software development: A case study of two agile teams, in: Proceedings of the International Conference on Agile Software Development, Springer, Copenhagen, Denmark, 2020, pp. 39–55, http://dx.doi.org/10.1007/978-3-030-49392-9_3.

[16] K. Schwaber, J. Sutherland, The Scrum Guide, Vol. 21, Scrum Alliance, 2011.

[17] E. Schön, et al., Key challenges in agile requirements engineering, in: Proceedings of the International Conference on Agile Software Development, Springer, Cologne, Germany, 2017, pp. 37–51, http://dx.doi.org/10.1007/978-3-319-57633-6_3.

[18] M. Bastarrica, G. Espinoza, J. Sánchez, Implementing agile practices: The experience of Tsol, in: Proceedings of the International Symposium on Empirical Software Engineering and Measurement, ACM, Oulu, Finland, 2018, pp. 1–10, http://dx.doi.org/10.1145/3239235.3268918.

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

[20] E. Lindgren, J. Münch, Raising the odds of success: The current state of experimentation in product development, Inf. Softw. Technol. 77 (2016) 80–91, http://dx.doi.org/10.1016/j.infsof.2016.04.008.

[21] F. Fagerholm, A.S. Guinea, H. Mäenpää, J. Münch, The RIGHT model for continuous experimentation, J. Syst. Softw. 123 (2017) 292–305, http://dx.doi.org/10.1016/j.jss.2016.03.034.

[22] S. Yaman, M. Munezero, J. Münch, F. Fagerholm, O. Syd, M. Aaltola, C. Palmu, T. Männistö, Introducing continuous experimentation in large software-intensive product and service organisations, J. Syst. Softw. 133 (2017) 195–211, http://dx.doi.org/10.1016/j.jss.2017.07.009.

[23] D. Salah, R. Paige, P. Cairns, Patterns for integrating agile development processes and user centred design, in: Proceedings of the European Conference on Pattern Languages of Programs, ACM, Kaufbeuren, Germany, 2015, pp. 1–10, http://dx.doi.org/10.1145/2855321.2855341.

[24] K. Vredenburg, J.-Y. Mao, P.W. Smith, T. Carey, A survey of user-centered design practice, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, Minneapolis, Minnesota, USA, 2002, pp. 471—478, http://dx.doi.org/10.1145/503376.503460.

[25] C. Abras, D. Maloney-krichmar, J. Preece, User-centered design, in: Encyclopedia of Human-Computer Interaction, SAGE, Bainbridge Island, USA, 2004, pp. 763–768.

[26] IDEO, The Field Guide to Human-Centered Design, IDEO, Palo Alto, USA, 2015.

[27] T. Brown, Design thinking, Harv. Bus. Rev. 86 (2008) 84–92.

[28] H. Obendorf, M. Finck, Scenario-based usability engineering techniques in agile development processes, in: Extended Abstracts on Human Factors in Computing Systems, ACM, Florence, Italy, 2008, pp. 2159–2166, http://dx.doi.org/10.1145/1358628.1358649.

[29] D. Norman, The Design of Everyday Things: Revised and Expanded Edition, Basic books, USA, 2013.

[30] D.J. Teece, Capturing value from knowledge assets: The new economy, markets for know-how, and intangible assets, Calif. Manage. Rev. 40 (3) (1998) 55–79, http://dx.doi.org/10.2307/41165943.

[31] J. Gothelf, Lean UX: Applying Lean Principles to Improve User Experience, O'Reilly, Sebastopol, USA, 2013.

[32] C. Moralles, M. Vaccaro, M. Zorzetti, E. Pereira, C. Trindade, B. Prauchner, S. Marczak, R. Bastos, On the mapping of underlying concepts of a combined use of lean and user-centered design with agile development: The case study of the transformation process of an IT company, in: Workshop Brasileiro de Métodos Ágeis, in: Communications in Computer and Information Science, vol. 1106, Springer, 2019, pp. 25–40, http://dx.doi.org/10.1007/978-3-030-36701-5_3.

[33] C. Moralles, M. Zorzetti, I. Signoretti, E. Pereira, M. Vaccaro, B. Prauchner, L. Salerno, C. Trindade, S. Marczak, R.M. Bastos, On the development of a model to support the combined use of agile software development with user-centered design and lean startup, in: Proceedings of the European Conference on Systems, Software and Services Process Improvement, in: Communications in Computer and Information Science, vol. 1251, Springer, Düsseldorf, Germany, 2020, pp. 220–231, http://dx.doi.org/10.1007/978-3-030-56441-4_16.

[34] R. Yin, Case Study Research and Applications: Design and Methods, SAGE, Thousand Oaks, USA, 2003.

[35] M. Zorzetti, I. Signoretti, E. Pereira, L. Salerno, C. Moralles, C. Trindade, M. Machado, R. Bastos, S. Marczak, A practice-informed conceptual model for a combined approach of agile, user-centered design, and lean startup, in: Proceedings of the International Conference on Product-Focused Software Process Improvement, in: Lecture Notes in Computer Science, vol. 12562, Springer, Turin, Italy, 2020, pp. 142–150, http://dx.doi.org/10.1007/978-3-030-64148-1_9.

[36] K. Krippendorff, Content Analysis: An Introduction to Its Methodology, SAGE, Thousand Oaks, USA, 2018.

[37] D. Salah, R.F. Paige, P. Cairns, A systematic literature review for agile development processes and user centred design integration, in: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering, ACM, London, England, United Kingdom, 2014, pp. 41–50, http://dx.doi.org/10.1145/2601248.2601276.

[38] M. Gutbrod, J. Münch, M. Tichy, How do software startups approach experimentation? Empirical results from a qualitative interview study, in: Proceedings of the International Conference on Product-Focused Software Process Improvement, Springer, Innsbruck, Austria, 2017, pp. 297–304, http://dx.doi.org/10.1007/978-3-319-69926-4_21.

[39] M. Poppendieck, T. Poppendieck, Lean Software Development: An Agile Toolkit, Addison-Wesley, Boston, USA, 2003.

[40] C.M. Riordan, R.J. Vandenberg, H.A. Richardson, Employee involvement climate and organizational effectiveness, Hum. Resour. Manag. 44 (4) (2005) 471–488.

[41] A. Griffin, PDMA research on new product development practices: Updating trends and benchmarking best practices, J. Prod. Innov. Manage. 14 (6) (1997) 429–458, http://dx.doi.org/10.1016/S0737-6782(97)00061-1.

[42] M. Yilmaz, R. O'Connor, P. Clarke, Software development roles: A multi-project empirical investigation, Special Interest Group Softw. Eng. 40 (2015) 1–5, http://dx.doi.org/10.1145/2693208.2693239.

[43] B. Tessem, Individual empowerment of agile and non-agile software developers in small teams, Inf. Softw. Technol. 56 (8) (2014) 873–889, http://dx.doi.org/10.1016/j.infsof.2014.02.005.

[44] I. Signoretti, Characterizing the Combined Use of Agile, User-Centered Design and Lean Startup: A Case Study of Two Software Teams (Master's thesis), Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil, 2020, URL http://tede2.pucrs.br/tede2/handle/tede/9783.