# 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

Cassiano Moralles[1](✉), Matheus Vaccaro[1], Maximilian Zorzetti[1], Eliana Pereira[2], Cássio Trindade[1], Bruna Prauchner[1], Sabrina Marczak[1], and Ricardo Bastos[1]

[1] MunDDoS Research Group – PPGCC – School of Technology, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Porto Alegre, RS, Brazil
{cassiano.mora,matheus.vaccaro,maximilian.zorzetti, bruna.prauchner}@acad.pucrs.br, {cassio.trindade,sabrina.marczak,bastos}@pucrs.br

[2] Instituto Federal do Rio Grande do Sul (IFRS), Porto Alegre, RS, Brazil
eliana.pereira@restinga.ifrs.edu.br

**Abstract.** The agile development of software requires new approaches to serve users and end customers. The combination of Lean and User-Centered Design with Agile gives software development a competitive advantage. Given the novelty and scarcity of studies on such combined use in software development, as part of our long-term research that aims to develop a maturity model to accelerate the transformation from agile to the use of the combined approaches, we posed as our first step to identify what are the underlying concepts involved on the use of agile, lean, and user-centered design. We first conducted multiple literature reviews to identify the concepts for each of the individual approaches to then conduct an empirical study in order to identify what is considered useful by two software teams of a multinational IT company that are going through such a transformation for about 6 months. Our study revealed that there are concepts from literature not yet considered in practice and the other way around, there are practiced concepts not found in literature. For now, we hypothesize that this is due to the early maturing process of the studied teams. We believe that this is an initial contribution that can be of help for other teams enduring this challenging transformation process. Our research will next investigate how the three approaches relate to one another in order to provide a unique and consolidated combined model of concepts that will further be used as the skeleton of our maturity model.

**Keywords:** Agile development · Lean · User-Centered Design · Organizational transformation · Empirical study

P. Meirelles et al. (Eds.): WBMA 2019, CCIS 1106, pp. 25–40, 2019.
https://doi.org/10.1007/978-3-030-36701-5_3

## 1 Introduction

The adoption of agile methodologies has become an industry standard in the past years. Although these methodologies prepare teams to be more adaptive and to keep a closer contact with clients and customers, some authors (e.g., [19]) argue that agile needs to be combined with other approaches in order to provide better guidance for agile teams to improve their understanding of the problem at hand so as to provide more aligned solutions and to keep the customer engaged. To that end, Pivotal Software, Inc.[1] (henceforth referred to as Pivotal) has developed a three-pronged approach to software development: Pivotal Labs.

Pivotal Labs [8] combines certain aspects of Extreme Programming (XP) [1,2], Lean [12,13], and User-Centered Design (UCD) [4,11,15]. Ideas and practices from each of these methodologies are used to tackle different aspects of software development: XP handles the technical activities; Lean mitigates the risk of building the wrong software; and UCD guarantees the software solves an end-user real problem. Software development teams that have adopted this approach show increased productivity and efficacy [16,17], while in our study experience also reporting increased levels of satisfaction and happiness at work.

Apart from the argumentation for the combined use of the approaches from literature (e.g., [19]) and the reports from Pivotal customers, little is known on how to proceed to use XP, Lean, and UCD together. There are, however, comparable studies presenting frameworks that integrate Agile, Lean Startup, and Design Thinking. Grossman-Kahn and Rosensweig present Nordstrom's Innovation Labs model for innovation, *Discovery by Design* [7], while sharing lessons learned from building an innovation capability from the ground up. With a great focus on the needs of the customers and an iterative mindset, they perform rapid experimentation, prototyping, and testing cycles based on the core mindsets and tools from those three methodologies to create innovative products. Dobrigkeit, de Paula, and Uflacker present a software development process called InnoDev [5] based specifically on Scrum, Lean Startup, and Design Thinking. Its process is divided into three phases: Design Thinking, Initial Development, and Development; each composed of a list of activities, roles, deliverables, and techniques. Still, to the best of our knowledge, there are no papers on how a software team should start the journey of a combined adoption of XP, Lean, and UCD, and on how to identify that the team is maturing.

To fill in this gap, we have set up a three-years long research project to investigate the matter. Our main goal is to, at the end of this period, define a maturity model to help software teams through this transformation. Secondarily, we aim to define an assessment method (or, a health check) for identifying the maturity the team presents at a certain moment in time. To do so, our first step is to identify what are the underlying concepts that represent the combined

[1] https://pivotal.io.

use of XP, Lean, and UCD. For that, we conducted a series of literature review studies[2] to identify the concepts from literature and from Pivotal Labs[3].

Given this context, in this paper we present a case study of two software development teams from a multinational IT company named ORG (name omitted for confidential reasons). Due to a recent drive to modernize the company from the inside out, these two teams have recently adopted Pivotal Labs, having learned the approach from Pivotal itself. As part of the partnership between our research group and ORG, we have them stationed in a local modern software development lab within the University campus where teams from ORG spend 3 months working in this isolated environment that was intentionally designed to serve our research purposes and to allow the teams to work without interference from others that have not on-boarded the transformation process yet. The main goal of this report is *to present the mapping between literature and practice regarding the underlying concepts that relate to XP, Lean, and UCD in order to provide initial insights to those that aim to endure the same transformation process.* As a next step, we will consolidate such concepts, from literature and practice, into a conceptual model and, in the long run, this conceptual model will be used as the skeleton of a maturity model.

The remainder of this paper presents the mapping between literature and our case study teams' experience, highlighting and discussing the differences between both perspectives.

## 2 Research Method

We conducted a multiple case study [14] on the two ORG software development teams stationed in our Software Development Lab from April to June, 2019, as presented next.

### 2.1 Case Setting

ORG has software product development sites in the USA (headquarters), India, and Brazil. With over 7,000 employees and responsible for about 1,200 software products, the IT department started its agile transformation in 2015 and moved to the combined use of Agile, Lean, and UCD principles in late 2017. The adopted approach was inspired by the Pivotal Labs [8] methodology, which proposes a

[2] A journal article consolidating the review on the concepts of the 3 approaches and what maturity models are published on the topic is under review, thus we cannot cite it for now. We would like to note that we found no maturity model addressing the 3 approaches together nor for the combination of 2 of them, but we did find 19 models for agile maturity and 5 for lean maturity alone.

[3] We developed an executive report on findings from this study that is of ORG use only. Due to our confidential research agreement with the organization we cannot disclose this document, but we would like to note that there is little on the matter and that what we report in this paper is representative of what is publicly available in blogs, websites, etc of Pivotal customers.

"team rhythm" composed of principles and ceremonies based on the 3 aforementioned approaches. It also suggests the adoption of a cross-functional team, called balanced team, composed of three main roles: Product Designer, Product Manager, and Software Engineer. Pivotal Labs' main goal is to help teams to build software products that deliver meaningful value for users and their business. Thus, it offers a framework and initial starting point for any team to discuss its specific needs and define its own way towards software development.

We had two development teams from ORG's Brazilian financial sector stationed in a modern software development lab inside the University's campus. Of the total 16 team members, we interacted with the 8 that participated in a Pivotal hands-on immersion training in the USA. Team A is responsible for a software product that calculates the cost of associated services offered by the products sold by ORG and displays this information to ORG consumers. Team B is responsible for the software product that gathers information about these services from other ORG software products and stores them for Product A to use. These teams spent 3 months in the USA working directly with Pivotal Labs consultants, who played roles in the software development process as hands-on mentors to the ORG members. Afterwards, both teams spent 3 months working at the University's dedicated lab, which is equipped with 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 ideas' development and information sharing, and a meeting room that turns into an entertainment space for leisure time). This last stage is when the data collection and analysis took place.

### 2.2 Data Collection and Analysis

We used 3 data sources: a questionnaire to collect the participants' profile (name, role, responsibilities, and time working in IT and at ORG); observations to learn about their day to day activities; and focus group sessions to gather information on their perceptions about the transformation, the training experience, the benefits and challenges of the Pivotal Labs approach; and to discuss the concept mapping between literature and what we observed them doing in practice.

Altogether, we performed six focus group sessions that lasted in average 1 hour with the 8 members that worked in the USA. Their profiles are shown in Table 1. Meetings were voice recorded and transcribed for thematic analysis [3,6,18]. Of those six meetings, we used two sessions for each approach. We first presented them the concepts from their practice in order to clarify whether we comprehended them correctly and then we presented the concepts from literature in order to identify the completeness of our observations from practice. By discussing the literature, team members could present us with concepts that we might have missed or misunderstood. We considered the work of Kent Beck [1,2] as literature for XP; Lean Startup [13] and Lean Software Development [12] for Lean; and the work of Norman [11], Brown [4], and Salah, Paige, and Cairns [15] for UCD. We based our definition of literature on existing Pivotal work and an initial observation of the teams. For instance, although Pivotal Labs advocates

**Table 1.** Participants' profile

| ID | Role | Training | IT Work Exp (years) | Company Exp (years) |
|---|---|---|---|---|
| P1 | Software Engineer | Enabler | 10 | 4 |
| P2 | Product Manager | Enabler | 19 | 0.5 |
| P3 | Product Designer | Enabler | 27 | 10 |
| P4 | Software Engineer | Enabler | 21 | 8 |
| P5 | Product Manager | Enabler | 21 | 6 |
| P6 | Product Designer | Enabler | 5 | 4 |
| P7 | Software Engineer | Enabler | 20 | 11 |
| P8 | Software Engineer | Enabler | 5 | 5 |

for the use of Lean Startup (misnaming it as only "Lean"), we observed the use of Lean Software Development concepts, so we decided to consider it as part of the literature for Lean.

## 3 Results

To facilitate the presentation of the large number of concepts that we identified from literature and later mapped to the case study teams' practice, we introduce these concepts in tables, one per approach (we divided Lean into Lean Startup and Lean Software Development, due to them being radically different). When a concept from literature was not reported by the teams, we indicated "—" in the Case Study table column. Similarly, the concepts identified in practice and lacking in literature are indicated with "—" in the Literature table column. We also organized these concepts into categories, which we name "elements" as per the literature perspective, namely: Activity, Role, Work Product, and Technique/Practice. An exception is the Lean Software Development approach, which organizes itself into Principles that can be realized by Tools, which in turn make use of Concepts as shown in Table 4.

### 3.1 Extreme Programming

We observed a few differences with our mapping as presented in Table 2. When considering the Activities, the teams put aside their categorization, having them distributed throughout the project life cycle (e.g., BDD as an strategy to validate acceptance tests). With regards to Roles, given that ORG adopted the concept of Balanced Teams from Pivotal Labs, there are three main roles, namely: Product Manager, Product Designer, and Software Engineer. These interchange job responsibilities with XP defined roles, adding them up as stated by some team members: *"The role of Product Designer encompasses more attributions than a Designer"* (P2, P5, P6). A balanced team is described as "a global movement of people who value multidisciplinary collaboration and iterative delivery focused on customer value as a source for innovation" [9]. This concept is used to complement an agile team, as it places the product-focused team members, such as product managers and designers, on equal footing with the team's technical-focused members through a set of core values instead of the definition of explicitly defined roles, events, and artifacts [10]. This allows for a Shared Context. There is also an additional role—Anchor, played by an experienced Software Engineer who, in addition to coding full-time, acts as a resource for the rest of the development team for supporting the resolution of technical and non-technical issues; *"The Anchor role is not necessarily played by the most experienced team member. It is the professional who can talk about the product and about engineering in the same language"* (P1, P2), and *"The Anchor can also be the colleague who will remain in the product team for a longer time, to become a focal point"* (P7). Also, *"The anchor can represent the team in a meeting for clarifications with the user, avoiding the need to send the entire team for this discussion"* (P2). Specific to ORG is the Consultant role, who supports the team, belongs to the Services team, and is responsible for infrastructure and databases.

Considering the Work Products, User Stories can be proposed *"at any time, any role can propose a feature or story"* (P4), however, some members explained that *"We do informally categorize them into Bugs, Features, and Chores"* (P5). *"Bugs are defects that we need to fix, regardless of how they were identified, and Features are new additions to the software product"* (P3). Chores, on the other hand, are a new specialization to indicate that something needs to be done but does not add value to the software: *"We observed some unnecessary processes, in our opinion, and questioned the customers. No one knew what they meant. We just decided alongside with the customers to remove them from the system. We will do this when time allows"* (P7). It is important to note that despite its categorization, all User Stories are now driven by Problem-resolution rather than requirements (as it has been for the past two decades): *"We don't start from the elicitation or clarification of requirements. We now focus on discussing with the customers and users what are the problems they have"* (P8). The Product Backlog is also specialized. The new Ice Box concept is used to indicate User Stories that were either not prioritized yet or were put on hold for some reason, *"Any story can be put on hold in the Ice Box"* (P4), *"We had situations where the business told us that a user story was necessary, but we left it in the Ice Box after realizing it was not relevant. The project evolved and with time the user also realized it was of no use and ended up satisfied with our decision"* (P2). Also, *"We use it as a way to record ideas to avoid forgetting them"* (P4, P8).

When considering the Techniques and Practices, the major mindset change we observed is the fact that the ORG teams do not focus on Releases. They do plan an Iteration as a means to set up expectations with users but they do not estimate efforts or set due dates, *"The term release is used only as a team control mechanism to set the users' expectation and provide visibility"* (P3, P6). This is possible because they are in constant contact with the users, although they

**Table 2.** Extreme Programming Literature and Case Study Mapping

| Literature | Element | Case Study | |
|---|---|---|---|
| Coding | **Activity** | Coding | |
| Designing | | — | |
| Testing | | — | |
| Listening | | Interviews | |
| — | **Role** | Anchor | |
| Consultant | | Services Consultant | |
| Coach | | Product Manager | |
| Tester | | Product Designer/Software Engineer | |
| Programmer | | Software Engineer | |
| Tracker | | Software Engineer | |
| Manager | | Product Manager | |
| Doomsayer | | Product Manager | |
| Big Boss | | Product Manager | |
| Customer | | Product Manager | |
| Bug Fix | **Work Product** | Problem-based User Story | Bug |
| User Story | | | Feature |
| | | | Chore |
| Product Backlog | | Product Backlog | Current |
| | | | Ice Box |
| Iteration Backlog | | — | |
| Release Planning | **Technique / Practice** | — | |
| Iteration Planning | | Pre-Iteration Planning Meeting (Pre-IPM)/ Iteration Planning Meeting (IPM) | |
| Customer Approval | | User Feedback | |
| Pair Programming | | Pair Programming | |
| Acceptance Test-Driven Development | | — | |
| Test-Driven Development | | Test-Driven Development | |
| Customer Tests/ On-Site Customer | | — | |
| Continuous Delivery | | Continuous Delivery | |
| Refactoring/ Design Improvement | | Refactoring | |
| Continuous Integration | | Continuous Integration | |
| Planning Game | | Planning Game | |
| Estimation by Example | | — | |
| — | | Behavior-Driven Development (BDD) | |
| Spike | | Experiments | |
| Daily Meeting | | Daily Stand-Up | |
| Stand-up Meeting | | Office Stand-up/Team Stand-up | |
| Whole Team | | Balanced Team | |
| Collective Ownership/ Collective Code Ownership | | Collective Ownership | |
| Coding Standards | | S.O.L.I.D. | |
| 40 Hours per Week/Sustainable Pace | | Sustainable Pace | |
| Constant Feedback | | Constant Feedback | |
| Simple Design | | Simple Design | |
| Metaphor | | Metaphor | |
| Small Releases | | Small Increments | |
| Retrospective | | Retrospective | |
| — | | Tech Talks | |
| — | | Shared Context | |
| — | | Team Agreement | |
| — | | Question Actual Process | |
| — | | Information Repository | |

are not On-Site Customers but *"they are nearby"* (P1). They also believe that a good way to constantly collect User Feedback is by using BDD: *"We use BDD to have better communication with our users. We validate our acceptance tests, which in turn validate the users' perspectives"* (P5). Spikes, simple programs to explore potential solutions, often not good enough to keep, are used in a slightly different way than proposed in literature. The ORG teams use spikes as a resource for their experimentation of hypotheses, *"We work up to 4 hours if needed to build a spike to experiment our theories and explore possibilities"* (P2).

Other smalls adjustments to concepts from literature are: instead of only writing code in accordance with rules (Coding Standards), the teams use SOLID, the mnemonic acronym for five design principles intended to make software designs more understandable, flexible, and maintainable—Single responsibility, Open–closed, Liskov substitution, Interface segregation, and Dependency inversion principle. Tech Talks meetings focus on exposing a subject of interest to the teams and other colleagues who want to learn something new. These meetings can be of technical nature or comprise any other aspect. Team Agreement refers to any kind of decision the team makes that will take longer than 30 min to be implemented and therefore is worth discussing and recording. Question Actual Process is the mindset *"we learned from Pivotal; they instigated us to be investigative all the time by asking questions when we see fit"* (P2, P5). And, finally, Information Repository is used as a resource to maintain a shared context *"where everyone has access to information about the problems we are trying to resolve. Currently we are using Slack to make it easier"* (P3, P6).

### 3.2 Lean

**Lean Startup.** Overall, we identified that the Lean Startup (LS) concepts used by the ORG teams are all heavily centered around conducting experiments, as stated by a Software Engineer, *"Let's conduct an experiment to validate if this approach will be better. If it works, let's proceed. How do we know this? Through experimentation."* (P7), and being able to make informed decisions, as exemplified by a Product Manager: *"The team compiled the results and sent them to the stakeholders saying: 'look, these are the results and this is what we've learned. What are we going to do with it? Do you want to follow this approach or the other one? The decision is yours."'* (P2).

As part of the Activity element, we find some activities related to LS principles, e.g., Building Experiments, Measuring Results, and Learning being directly associated with the Build Measure Learn Cycle. A similar phenomenon happens in the Work Product section, where Iterate, Escalate, Persevere, and Give Up are outcomes of the principle Validated Learning. The 31 techniques presented in Table 3 were extracted from the Lean Startup book by Eric Ries [13].

We found that ORG teams use a subset of the activities mapped from the literature. The core experimentation cycle activities related to the principles of Build Measure Learn and Innovation Accounting are used normally, however Formulating the Business Model and Hypotheses is approached in a different

**Table 3.** Lean Startup Literature and Case Study Mapping

| **Literature** | | **Element** | **Case Study** | |
|---|---|---|---|---|
| Formulating the Business Model and Hypotheses | | **Activity** | Understanding the Problem | |
| | | | Defining the Team's Vision of the Problem | |
| | | | Establishing the Team's Strategy to Solve the Problem | |
| | | | Mapping Everyone Affected by the Problem (Users and Stakeholders) | |
| | | | Formulating Hypotheses | |
| Build Measure Learn (Principle) | Building Experiments | | Build Measure Learn (Principle) | Building Experiments |
| | Measuring Results | | | Measuring Results |
| | Learning | | | Learning |
| Innovation Accounting (Principle) | Establish the Baseline | | Innovation Accounting (Principle) | Establish the Baseline |
| | Tune the Engine | | | Tune the Engine |
| | Pivot or Persevere | | | Pivot or Persevere |
| Running the Engine of Growth | | | — | |
| Pivot or Persevere Meeting | | | — | |
| Entrepreneur | | **Role** | Team | |
| Ideas (Hypotheses) | | **Work Product** | Ideas (Hypotheses) | |
| Product | | | Product | |
| Data (Metrics and Measurements) | | | Data (Metrics and Measurements) | |
| Validated Learning (Principle) | Iterate | | Validated Learning (Principle) | Iterate |
| | Escalate | | | — |
| | Persevere | | | Persevere |
| | Give Up | | | Give Up |
| | — | | | Double Down |
| Split Tests | | **Technique / Practice** | — | |
| Small Batches | | | Small Batches | |
| Triple "A" Metrics (Actionable, Accessible, Auditable) | | | — | |
| Customer Development | | | Customer Development | |
| 5 Whys | | | 5 Whys | |
| Customer Advisory Board | | | — | |
| Falsifiable Hypotheses | | | — | |
| Product Owner | | | — | |
| Accountability | | | — | |
| Customer Archetypes | | | Customer Archetypes | |
| Cross-Functional Teams | | | Balanced Teams | |
| Smoke Tests | | | — | |
| Continuous Deployment | | | Continuous Deployment | |
| Usability Tests | | | — | |
| Real-Time Monitoring & Alerting | | | Real-Time Monitoring & Alerting | |
| Customer Liaison | | | Customer Liaison | |
| Funnel Analysis | | | — | |
| Cohort Analysis | | | — | |
| Net Promoter Score | | | — | |
| Search Engine Marketing | | | — | |
| Predictive Monitoring | | | — | |
| Unit Tests | | | Unit Tests | |
| Continuous Integration | | | Continuous Integration | |
| Incremental Deployment | | | Incremental Deployment | |
| Free & Open-Source | | | — | |
| Cloud Computing | | | — | |
| Cluster Immune System | | | — | |
| Just-In-Time Scalability | | | — | |
| Refactoring | | | Refactoring | |
| Developer Sandbox | | | — | |
| Minimum Viable Product | | | Minimum Viable Product | |

way, since the team's goal is to solve the company's problems instead of creating a sustainable business: the team focuses on understanding the problem at hand, so that they can build a common understanding and a strategy to tackle it. A Product Manager says: *"When we are identifying a problem, we contact* *the stakeholders in order to understand what the problem we're dealing with is. We spent the whole morning discussing everything we thought was related to the problem, and everything that could be a problem, until we reached a final statement. After that, as a team, we defined the vision and the strategy that we were going to use to solve the problem..."* (P2). We did not identify the explicit usage of Pivot or Persevere Meetings and Engines of Growth.

We did not identify any explicit categorization of roles in the literature. Eric Ries often refers to the ones conducting the scientific method of the LS as Entrepreneurs. In our case study, this role is taken by the Team as a whole.

Regarding Work Products, the main difference found is that the teams did not mention Escalate as an informed decision based on the outcome of an experiment, i.e., a Validated Learning outcome. A Software Engineer says that *"The decisions normally are: you can abandon that track of work; you can persevere, and continue to work on that; you can pivot, change the direction and try to investigate it in another way; or you can even double down on it, things are going the right way but we want it to go faster, so we put more engineers to work on it."* (P4) Among the 31 presented Techniques, we identified that the team actively uses 13 of them. Most notably, we found that the concept of Cross-functional Teams is mapped to Balanced Teams, as previously mentioned in the XP approach (Sect. 3.1).

**Lean Software Development.** Most of what is presented by the Poppendiecks [12] is used in some way by the ORG development teams as seen in Table 4. We observed that the Iteration tool is used differently: ORG teams disregard the use of *fixed time-boxes*, as *"[stakeholders do not impose deadlines], unless there's a compliance or interlocking deadline already in-place"* (P5), although a Software Engineer adds that *"stakeholders have target dates or launch windows for the final solution, and we aim to deliver it all by then"* (P4). Additionally, Iterations have an open scope, says a Software Engineer: *"We might have decided to work on two User Stories for a given Iteration, but if something—anything—comes up mid-iteration, we reshuffle our priorities and work on something else"* (P8).

For Synchronization purposes, the teams prefer the use of *spanning application* instead of *matrix*: upon being asked if they develop a system by sketching out its components and then splitting the team to work on each, a Software Engineer responded, *"No, we make experiments—a whole slice of a solution, comprised of the full technology stack, to see if it works. If it does, we expand upon it"* (P4). In regards to the team's decision making process (Making Decisions), all decisions are made exclusively through the interpretation of experiment results, disregarding the Poppendiecks' *intuitive decision making* and *simple rules* (we called this *experiment-based decisions*). Although an expert's intuition can influence the decision or open up more options, the final say comes from experimentation, as stated by a Software Engineer: *"We needed to insert a lot of data into a database, and it was taking too long with our current technology stack. I developed a solution using another technology stack that I was sure was going to perform better. As I thought, it did, so we started using it"* (P1).

**Table 4.** Lean Software Development Literature and Case Study Mapping

| **Literature** | | | **Case Study** | | |
|---|---|---|---|---|---|
| **Principle** | **Tool** | **Concept** | **Principle** | **Tool** | **Concept** |
| Eliminate Waste | Seeing Waste | | Eliminate Waste | Seeing Waste | |
| | Value Stream Mapping | | | Value Stream Mapping | |
| Amplify Learning | Feedback | | Amplify Learning | Feedback | |
| | Iteration | Negotiable Scope | | Iteration | Negotiable Scope |
| | | Team Commitment | | | Team Commitment |
| | | Fixed Time-Box | | | — |
| | Set-Based Development | Constraints | | Set-Based Development | Constraints |
| | | Multiple Options | | | Multiple Options |
| | Synchronization | Daily Build and Smoke Test | | Synchronization | Daily Build and Smoke Test |
| | | Spanning Application | | | Spanning Application |
| | | Matrix | | | — |
| Decide as Late as Possible | Making Decisions | Intuitive Decision Making | Decide as Late as Possible | Making Decisions | — |
| | | Simple Rules | | | — |
| | | — | | | Experiment-Based |
| | Options Thinking | | | Options Thinking | |
| | The Last Responsible Moment | | | The Last Responsible Moment | |
| Deliver as Fast as Possible | Cost of Delay | Economic Model | Deliver as Fast as Possible | Cost of Delay | Economic Model |
| | Pull Systems | Information Radiators | | Pull Systems | Information Radiators |
| | Queueing Theory | Small Work Packages | | Queueing Theory | Small Work Packages |
| | | Slack | | | Slack |
| | | Steady Rate of Service | | | Steady Rate of Service |
| | | Steady Rate of Arrival | | | Steady Rate of Arrival |
| Empower the Team | Expertise | Communities of Expertise | Empower the Team | Expertise | Communities of Expertise |
| | | Standards | | | Standards |
| | Motivation | Belonging, Safety, Competence, and Progress | | Motivation | Belonging, Safety, Competence, and Progress |
| | | Moderation | | | Moderation |
| | | Purpose | | | Purpose |
| | | Champion | | | — |
| | Self-Determination | Principles, Not Practices | | Self-Determination | Principles, Not Practices |
| | Leadership | Master Developer | | — | |
| Build Integrity In | Conceptual Integrity | Software Architecture | Build Integrity In | Conceptual Integrity | Software Architecture |
| | Perceived Integrity | Institutional Memory | | Perceived Integrity | Institutional Memory |
| | | Model-Driven Design | | | Model-Driven Design |
| | Refactoring | | | Refactoring | |
| | Testing | As-Built Test Suite | | Testing | As-Built Test Suite |
| | | Customer Tests | | | Customer Tests |
| | | Developer Tests | | | Developer Tests |
| See the Whole | Measurements | Information Measurement | See the Whole | Measurements | Information Measurement |
| | Contracts | Target-Cost Contracts | | — | |
| | | Time-And-Material Contracts | | | |
| | | Shared-Benefit Contracts | | | |
| | | Multistage Contracts | | | |

Concerning Motivation, ORG teams do not have a *champion*, a person that compels other members to work on a project. Instead, they all compel themselves to work, as stated by a Software Engineer: *"The empathy we feel for our work colleagues motivates us to work"* (P1). As for Leadership, we observed that the ORG teams do not have leaders at all: a Software Engineer points out that each role in the Balanced Team spearheads its respective domain (e.g., Software Engineers lead technical discussions), but also adds that *"all decisions are shared and made by the whole team"* (P4), while a Product Manager emphasizes that *"[even a rookie can make] the most experienced team member say 'You are right, let's do it your way."'* (P2). Finally, ORG teams dismiss the need for Contracts since they work for ORG itself: *"We do not sign any legal contracts. ORG decides what problems need solving, and these eventually trickle down to us"* (P2).

## 3.3 User-Centered Design

Table 5 shows the *Phases*, *Activities*, *Work Products*, and *Techniques* for UCD. The Phases, shown in the left side of the activities in the table, from literature, are related to the Double Diamond of *Design Thinking*. The idea is to perform the UCD activities inside of the *Finding the Problem* and *Finding the Solution* phases [11]. It is important to mention that we identified 77 techniques from the literature, however, Table 5 shows only the most cited techniques and those identified in our case study.

We found that the phases and activities of the UCD literature are the same used by the teams. In terms of Phases, the difference is the used terminology. We identified the *Finding the Problem* phase is the *Discovery* phase and the *Finding Solution* is the *Framing* phase. For the Activities, the first difference was in the *Testing* activity that originally (from literature) focus on validating the solution with the final users. In the studied teams, the solution is validated internally by the Product Designer (PD) and Product Manager (PM) roles in the *Seek Feedback* activity before being validated with the final users: *"The PM and PD will validate if the solution proposed is according to what was developed by the team. PM and PD validate before reaching the user. They will either accept or reject the story"* (P4); *"Sometimes we do not even have access to the user"* (P7). Another difference in terms of activities was that the teams perform an additional activity named *Communicate Early and Often*. This activity is related to the designer's pairing with the team members during product development: *"During implementation, the PD can pair with an engineer to ensure that this engineer has all the understanding he needs to develop"* (P4), *"It is the responsibility of the whole team to deliver the correct product"* (P5).

As for UCD Work Products and Roles, we did not observe any differences between those used in the team and the literature. The teams have the role *Product Designer* and produce a small set of work products connected with the applied techniques by them. Finally, we identify that of the 77 UCD techniques listed in literature, the teams have only used 7 of them so far. They also used 4 techniques we had not found in literature. Indeed, what we could obverse was that although they did not use a vast amount of techniques,

**Table 5.** User-Centered Design Literature and Case Study Mapping

| **Literature** | | **Element** | **Case Study** | |
|---|---|---|---|---|
| Finding the Problem and Finding the Solution Phases | Observation /Inspiration | **Activity** | Discovery and Framing Phases | Conduct Research |
| | Ideation | | | Generate Solutions |
| | Prototyping | | | Deliver Design Decisions |
| | Testing | | | Seek Feedback |
| | | | | Testing |
| | — | | | Communicate Early and Often |
| Designer | | **Role** | | Product Designer |
| Prototype | | **Work Product** | Prototype | |
| User Journey Map | | **Technique / Practice** | — | |
| Business Model Canvas | | | — | |
| Scenarios | | | — | |
| Stakeholders Mapping | | | — | |
| Persona | | | Persona | |
| Affinity Diagram | | | Affinity Mapping | |
| Blueprint | | | Blueprint | |
| Photo Journal | | | — | |
| Empathy Mapping | | | — | |
| Mind Mapping | | | — | |
| Storytelling | | | — | |
| Card Sorting | | | — | |
| Prototyping | | | Prototyping | |
| Ethnography | | | — | |
| Interview | | | Interview | |
| Brainstorming | | | Design Session | |
| 5 Why | | | — | |
| Point of View | | | — | |
| Questionnaire | | | — | |
| Usability Test | | | — | |
| Inspection | | | — | |
| Profiles | | | — | |
| Survey | | | — | |
| How Might We | | | How Might We | |
| — | | | 2x2 Prioritization | |
| — | | | Now, Near, Next | |
| — | | | Integration Research | |
| — | | | Design Studio | |

they are continually searching and studying new techniques, as argued by a Product Manager, *"Techniques are things that we keep looking for, studying, and eventually applying"* (P5). The teams pointed out *How Might We* and *$2 \times 2$ Prioritization* as the most used techniques: *"How Might We is one that we use a lot"* (P6). *"We always use How Might We because it helps us think about the value that solution will deliver"* (P5). Concerning *$2 \times 2$ Prioritization* they said: *"$2 \times 2$ to identify the pain points"* (P3), *"$2 \times 2$ can be used in prob-*

*lems, solutions"* (P5), *"2 × 2 to validate with user"* (P4), and *"2 × 2 is used as wild card"* (P7). They also mentioned the *Product Designer* as responsible for choosing better techniques for each situation: *"The Product Designer has the responsibility to attempt to identify the best technique to validate as fast as possible the teams' assumptions. The Product Designer tries to find the better technique to validate the idea"* (P6).

## 4 Discussion

Evolution is natural for software development with the adoption of new methodologies and technologies—the evolution of XP with the joint use of other approaches in the industry is evidenced in our research. The shared responsibilities, resulting of the multidisciplinary work, is a valid direction with increasing complexity in software development. More specifically, a balanced team works with no time-bound iterations (no Iteration Planning) as a mean to continuously deliver value to the customer. To do so, the main mindset change is to now focus on problems (Problem-based User Stories) rather than on requirements as the starting point of customer interaction and involvement.

As for Lean, the ORG teams seem to use a subset of LS and LSD that complements the use of one another. For instance, the teams' Decision Making does not use the concepts provided by LSD, instead, it uses *experiment-based decisions*, which is completely rooted in LS. Following this example is the lack of Leadership: since all decisions are experiment-driven, there is no need for a leader or boss figure among the team.

In regards to UCD, it seems that both teams only use a subset of its available tools, and understandably so, given how many of them there are. Not only that, but each team uses different sets of tools for their respective problems, indicating that there is really no be-all-end-all tool package to product design.

As a side note, we observed that both teams always seek to adapt techniques, practices, and roles to their context: they seem to have a "drive" to strive for the best way to do their jobs at all times. We believe this is a good indicator for the undergoing transformation of ORG. This also leads us to leave the following questions up in the air: are the differences identified in our mapping an issue? Since teams are always evolving, can our study be a good-to-have-at-hand document for consulting?

## 5 Final Considerations

As part of a long-term research project that aims to define a maturity model to help teams in their transformation to the combined use of XP, Lean, and UCD, we report in this paper our first step: the mapping between the identified underlying concepts from literature and those used by the observed ORG teams. From the comparison between the results from literature and our case study with the two teams that have been undergoing this transformation process for about 6 months, we found that the teams' use of Pivotal Labs is mostly aligned with the literature, but differs in some aspects, namely:

- All decisions are based on experiments, disregarding the intuition of experts;
- Lack of leaders, since the team inspires itself and shares decision making equally;
- There is an Anchor role, that bridges the understanding between business and engineering;
- Not all UCD techniques are used, but the teams are constantly seeking out to use new ones that might benefit their case.

This initial contribution can already be of use to software development teams aiming to endure such transformation. By revealing the concepts from literature, practitioners can have a broad overview of what they might have to deliberate on and can use, and by identifying what is being used by a maturing team that has been experiencing such transformation as part of a large multinational IT company, practitioners can envision some adjustments that have been proven to work so far. We note that our results are not generalizable nor are conclusive given the exploratory nature of our case study. However, they are a first step towards our main goal. We will continue observing other teams (there are 4 teams confirmed for the coming 6 months) in the University lab and contacting the past observed teams every 3 months as a means to identify how they mature throughout time. We expected to soon report on our to-be-proposed maturity model.

**Acknowledgement.** We would like to thank the study participants from ORG. Also, we thank ORG for the financial support (Brazilian Informatics Law n° 8.2.48 of 1991).

## References

1. Beck, K.: Embracing change with extreme programming. Computer **32**(10), 70–77 (1999). https://doi.org/10.1109/2.796139
2. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change, 2nd edn. Addison-Wesley, Upper Saddle River (2004)
3. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qual. Res. Psychol. **3**(2), 77–101 (2006). https://doi.org/10.1191/1478088706qp063oa. https://www.tandfonline.com/doi/pdf/10.1191/1478088706qp063oa. https://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa
4. Brown, T.: Design thinking. Harvard Bus. Rev. **86**, 84–92, 141 (2008)
5. Dobrigkeit, F., de Paula, D., Uflacker, M.: InnoDev: a software development methodology integrating design thinking, scrum and lean startup. In: Meinel, C., Leifer, L. (eds.) Design Thinking Research. UI, pp. 199–227. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-97082-0_11
6. Gregory, P., Barroca, L., Taylor, K., Salah, D., Sharp, H.: Agile challenges in practice: a thematic analysis. In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (eds.) XP 2015. LNBIP, vol. 212, pp. 64–80. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18612-2_6
7. Grossman-Kahn, B., Rosensweig, R.: Skip the silver bullet: driving innovation through small bets and diverse practices. In: Leading Through Design, p. 815 (2012)

8. Pivotal Software Inc.: Pivotal Labs (2019). https://pivotal.io/labs. Accessed 18 July 2019
9. Jarrell, J., Berner, I.: Balanced Team: A Balanced Approach to Product Design and Delivery (2014). http://www.balancedteam.org/. Accessed 18 July 2019
10. Jarrell, J., Berner, I.: Striking the Right Balance with Balanced Teams (2019). https://content.pivotal.io/white-papers/striking-the-right-balance-with-balanced-teams. Accessed 18 July 2019
11. Norman, D.A.: The Design of Everyday Things. Basic Books, New York (2002)
12. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit. Addison-Wesley, Boston (2003)
13. Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business, New York (2011)
14. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng. **14**(2), 131 (2008). https://doi.org/10.1007/s10664-008-9102-8
15. Salah, D., Paige, R.F., Cairns, P.: A systematic literature review for agile development processes and user centred design integration. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, England, pp. 5:1–5:10. ACM (2014). https://doi.org/10.1145/2601248.2601276
16. Sedano, T.: Sustainable Software Development: Evolving Extreme Programming, April 2017. https://doi.org/10.1184/R1/6723431.v1. https://kilthub.cmu.edu/articles/Sustainable_Software_Development_Evolving_Extreme_Programming/6723431
17. Sedano, T., Ralph, P., Péraire, C.: Sustainable software development through overlapping pair rotation. In: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2016, Ciudad Real, Spain, pp. 19:1–19:10. ACM (2016). https://doi.org/10.1145/2961111.2962590
18. Vaismoradi, M., Turunen, H., Bondas, T.: Content analysis and thematic analysis: implications for conducting a qualitative descriptive study. Nurs. Health Sci. **15**(3), 398–405 (2013). https://doi.org/10.1111/nhs.12048. https://onlinelibrary.wiley.com/doi/pdf/10.1111/nhs.12048. https://onlinelibrary.wiley.com/doi/abs/10.1111/nhs.12048
19. Ximenes, B.H., Alves, I.N., Araújo, C.C.: Software project management combining agile, lean startup and design thinking. In: Marcus, A. (ed.) DUXU 2015. LNCS, vol. 9186, pp. 356–367. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20886-2_34