Practical Implications from a Preliminary Theory of Simplicity in Agile Software Development Based on a Qualitative Study

Wylliams Barbosa Santos
Federal University of Pernambuco
University of Pernambuco
Recife, Pernambuco, Brazil.
wbs@cin.ufpe.br

José Adson O. G. Cunha
Exact Sciences Department
Federal University of Paraíba
Rio Tinto, Paraíba, Brazil.
adson@dcx.ufpb.br

Hermano Moura
Centre for Informatics
Federal University of Pernambuco
Recife, Pernambuco, Brazil.
hermano@cin.ufpe.br

Tiziana Margaria
Lero - the Irish Software Research Centre, University of Limerick
Limerick, Ireland.
tiziana.margaria@lero.ie

***Abstract*—*Context:* Several research works emphasise that the concept of simplicity is, in itself, by far not a simple concept, mainly because there are many perspectives on the perception of simplicity. *Purpose:* To understand how the agile team interpret their experiences in agile software projects considering the simplicity issues in agile software development. *Method:* Semi-structured interviews were carried out with project managers and software engineers within a software development company. The data was analysed using grounded theory techniques. *Results:* A set of categories (lightweight process, knowledge acquisition, personal communication, time-consuming, and product with value) that affect the simplicity in agile software development were extracted. Relationships among categories were used to construct propositions that explain the simplicity phenomena. Finally, implications for practices and recommendations are also addressed. *Conclusion:* The results show that a better understanding of the implications of simplicity on agile software development may contribute to the projects' successes.**

***Index Terms*—Simplicity, Agile Software Development, Qualitative Study.**

I. Introduction

Agile software development has had a huge impact on how software is developed worldwide [1]. The need for distribution of responsibility and initiative to support adaptation to change is a familiar territory of agile approaches [2].

Academic research in UK [3] called Rethinking Project Management (RPM) highlights the need for new possible future research about the developing practice, as the need for new thinking in the areas of project complexity, social process, value creation, project conceptualisation and practitioner developer. Based on the evolution of project management thinking, Moura and Skibniewski [4][5] presented the Software Project Framework (SPF), which introduces the definition of its elements to verify how project management and related research have evolved over the years and to identify related trends. Simplicity is one of the 14 dimensions (directions) for advancing research proposed by the SPF.

Agile software development represents an alternative to the heavyweight methodologies. It puts less emphasis on up-front and strict control and relies more on informal collaboration, coordination, and learning [6]. The business goals are achieved through practices, principles, and values focused on people and interactions, working software, customer collaboration, responding to change, and continuous improvement [7]. These practices embody the adaptability, flexibility and self-organisation.

A guiding principle behind agile software development is the idea of keeping things simple. The Agile Manifesto defines it this way: "Simplicity - the art of maximizing the amount of work not done - is essential." [8]. For example, the XP values lead us to ask, "What is the simplest thing that could possibly work?" [9].

In line with the agile manifesto, agile software development has proven to be an important set of methods in promoting simplicity issues. Although there is a variety of methodologies and frameworks of agile software development [10][9][11][12][13][14], few academic studies directly address simplicity [15][16][17]. This study aims to provide the practical implications based on the preliminary results of an ongoing research focused on understanding how project managers and software engineers interpret their experiences about simplicity in agile software development in the workplace.

A qualitative case study in a medium software company in England was conducted, which delivers bespoke software and consultancy, in which project managers, scrum masters and software engineers were interviewed to collect data about their perception in practice regarding simplicity phenomena.

The remainder of this paper is organised as follows. Section II introduces the background and related work of simplicity in various areas and domains, including philosophy, Information and Communications Technology, and in the context of Agile Software Development. Section III presents the methodological framework and the case study conducted with practitioners from a software development company in England. The results, which includes the provisional theory of simplicity and

its practical implications, are presented in Sections IV and V. Section VI addresses the theory in action, besides limitations, validity and reliability of this study. Finally, in Section VII, the conclusions of the study are presented, and directions for future works are pointed out.

II. Background

The study of simplicity is an interdisciplinary endeavour with many dimensions, concepts and attributes. The concept of simplicity is, in itself, by far not a simple concept because there are many perspectives on the perception of simplicity [16], described in the following subsections.

A. Simplicity in Philosophy

Simplicity principles have been proposed in various forms by theologians, philosophers, and scientists, from ancient to modern times. There is a widespread philosophical presumption that simplicity is a theoretical virtue. This presumption that simpler theories are preferable appears in many guises [18][19]. According to Gambrel [19], *virtue* refers to the generic term commonly used for any character trait people wish to commend. In both common speech and philosophical discourse, the virtues refer to those qualities whose possession makes a person, a good person.

Additionally, following Nussbaum's schema [20], Gambrel and Cafaro [19] define simplicity as the virtue disposing us to act appropriately within the sphere of our consumer decisions. From this point of view, simplicity is a conscientious and restrained attitude toward material goods that typically includes (i) decreased consumption and (ii) a more conscious consumption, (iii) greater deliberation regarding our consumer decision, (iv) a more focused life in general, and (v) a greater and more nuance appreciation for other things besides material goods, and also for (vi) material goods themselves.

B. Simplicity in Information and Communications Technology

Several works analyse simplicity in the context of Information and Communications Technology (ICT), with Margaria and Steffen specifically connecting agility and simplicity [21], and further elaborations in [15] [22]. According to Norman [23] [24] and Maeda [25] [26], the idea of getting less and paying more seems to contradict sound economic principles. Margaria and Hinchey [22] argue that simplicity is a mindset, a fundamental way of approaching solutions, an extremely wide-ranging philosophical stance in the world, and thus a deeply rooted cultural paradigm. The authors state that the culture of "less" can be profoundly disruptive, cutting out existing "standard" elements from products and business models, thereby revolutionising entire markets. In this sense, simplicity is inherently disruptive and can be a powerful thrust towards radical innovation.

Based on a systematic literature review and direct interaction with IT experts (individual interviews and focus group), Margaria, Steffen and Floyd [15][16] compiled a set of recommendations for possible lines of action, characterisation and dimensions of simplicity: the art of knowing, structure, orthogonality, size, transparency, communication, automation, context and subjectivity/felt complexity, which can be shared within and across these activities in a value network. Their reasoning and examples show that such dimensions rarely occur or are acted upon in isolation, so that in reality several dimensions play a role as initiator, mediator, or target of a simplicity-driven change. This understanding seems to suggest that agile Software development should be an ideal turf for embracing simplicity.

Simplicity: Simple things are easy to create, maintain and understand. Although it seems obvious to create simple solutions most developers tend to choose overcomplicated ones. It is in fact a psychological issue to "do the simplest thing that could possibly work" [27].

C. Simplicity in Agile Software Development

Agile software development methods emerged as a practice-led approach and have become the predominant and popular mode of development in software industry [28]. According to Dingsøyr et al. [29], the articulation of the Agile Manifesto in 2001 brought unprecedented changes to the software engineering field, by valuating the individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [7][11].

In order to satisfy these values, some principles[1] should be respected, including "Simplicity, the art of maximising the amount of work not done – is essential". In essence, agile methods emphasise simplicity. The goal is to get user feedback quickly by delivering software at short increments, even if it covers only a subset of the expected functionality [17].

In his book, Meyer [17] affirms that who has ever obtained an initial solution to a problem of any kind, found it complex and tried to simplify it. Achieving simplicity often means adding work, sometimes lots of it. This study provided an empirical evidence about simplicity on work and led towards a theory of simplicity in agile software development that can help researchers and practitioners to better understand the benefits of simplicity in this context.

Additionally, Meyer [17] discusses that the following goals are worthy in software engineering, but they arise in different contexts and lead to different principles: (i) proponents of rigorous, elegant programming techniques, (ii) avoiding unneeded work which leads to such principles as "Eliminate waste" and "Decide as late as possible" in Lean software development [12]. These two views meet, but not necessarily in the way agile authors [7] would like.

From the agile team's perspective, Santos et. al [30] defined simplicity as "the theoretical virtue disposing the team towards an analytic attitude that leads agile projects to be successful". This definition is supported by a literature review covering models related to simplicity in different research areas. Based on that, a conceptual framework was developed, which was then triangulated through a focus group with six ASD experts.

[1] http://agilemanifesto.org/principles.html

As observed by Margaria, Steffen and Floyd [15][16], the community of researchers and practitioners believes that the philosophy of simplicity is strategically important, yet still insufficiently understood.

In this space, this research work provides empirical evidence about simplicity at work and leads towards proposing a first model of simplicity in agile software development that can help researchers and practitioners to better understand the benefits of simplicity in this context.

III. Method

This section describes the research methodology adopted in this study. Several factors make empirical research in software engineering particularly challenging as it requires studying not only technology but its stakeholders' activities while drawing concepts and theories from social science [31].

This study aim is to understand how practitioners interpret simplicity in the agile software development projects and how these interpretations shape their work towards simplicity. In this sense, this research adopts techniques of grounded theory [32], following the roadmap for building theories from case study proposed by Eisenhardt [33].

A. Getting Started

According to Eisenhardt, without a research focus, it is easy to become overwhelmed by the volume of data. In this sense, the research focus is defined by the following initial research question:

How is simplicity understood by the agile team?

We performed a comprehensive literature review of simplicity in several areas, covering studies of simplicity in (i) philosophy (Section II-A), (ii) information communication and technology (Section II-B), and (iii) agile software development (Section II-C).

B. Selecting the Case

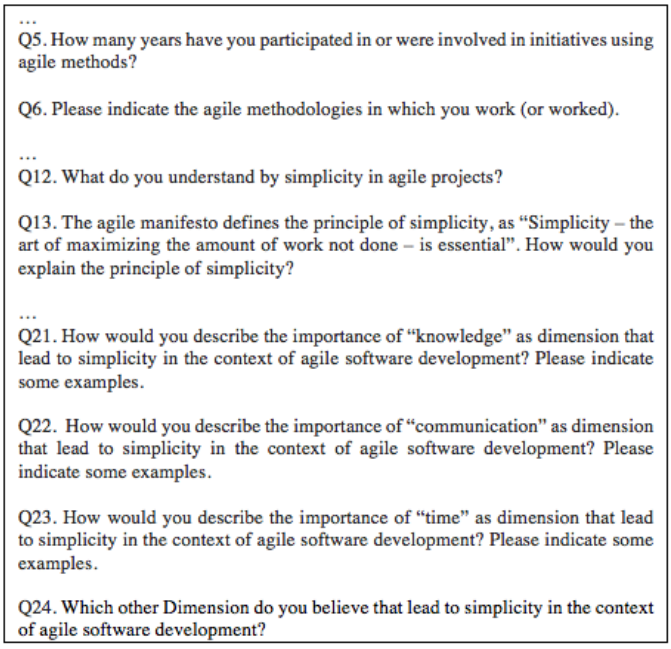
To build a theory from case studies, purposeful sampling consists of specifying the set of entities from which the research sample is to be drawn [33]. In this sense, we selected a company which:

- adopts agile methodologies in most projects, based on the concepts of adaptability, flexibility and self-organisation;
- achieves its organisation-level business goals through practices, principles, and values focused on people and interactions, working software, customer collaboration, responding to change, and continuous improvement;
- has a large number of simultaneous projects, of different size, technology, domain, scope;
- and provided us with full access to all the data and individuals necessary for our interviews.

Considering these essential requirements, we selected a medium sized British corporation in the software development sector. We provide a characterisation of this company in Section IV-A.

C. Crafting Instruments and Protocols

Semi-structured interviews were performed with experienced practitioners. The interview script was composed of open-end questions (Figure 1), structured as (i) respondent demographic profile, (ii) simplicity in agile software development, and (iii) key dimensions of simplicity. Initially (see Q5, Q6), the interview guide encompasses quick questions, aimed at exploring experience and the background of the participants. The next phases were presented in a funnel model [34], beginning with general questions focused on understanding the broad aspects of simplicity in agile software development (see Q12, Q13), which were refined toward more specific questions.


...
Q5. How many years have you participated in or were involved in initiatives using agile methods?

Q6. Please indicate the agile methodologies in which you work (or worked).

...
Q12. What do you understand by simplicity in agile projects?

Q13. The agile manifesto defines the principle of simplicity, as "Simplicity – the art of maximizing the amount of work not done – is essential". How would you explain the principle of simplicity?

...
Q21. How would you describe the importance of "knowledge" as dimension that lead to simplicity in the context of agile software development? Please indicate some examples.

Q22. How would you describe the importance of "communication" as dimension that lead to simplicity in the context of agile software development? Please indicate some examples.

Q23. How would you describe the importance of "time" as dimension that lead to simplicity in the context of agile software development? Please indicate some examples.

Q24. Which other Dimension do you believe that lead to simplicity in the context of agile software development?

Fig. 1. Interview guide extract

The general questions encouraged relevant and unbiased reflections bringing more details when answering the specific questions. The set of innovative research topics on the concept of simplicity identified by Margaria, Steffen and Floyd [15][16] identified dimensions that lead to simplicity. Thus, some questions (see Q21, Q22, Q23) embodied in the interview guide aim at characterising these dimensions of simplicity in the context of agile software development. The last question of our instrument (see Q24) explores additional possibilities of dimensions that lead to simplicity in agile software development.

The instrument was pre-tested with two pilot interviews to get some practice in interviewing and also quickly learn which questions might need rewording or might yield useless data. As a result, a few changes in the sequence of questions improved the final interview guide. These pilot test subjects are not part of the subject identified in the Participants Selection (Section III-D).

D. Entering the Field

According to our exploratory research strategy [33], we purposedly sampled six quite different practitioners. We selected various types of roles, such as software engineers, team leaders and project managers with different genders, ages and levels of education (Section IV). We considered only professionals who are experts in agile projects with more than 3 years of agile experience.

The participants were interviewed at their workplace. The data was collected over seven weeks in November and December of 2015 and January of 2016. The interviews lasted 60 minutes on average and were recorded with an MP3 player. The use of audio recording ensured an identical replication of the content of each interview, which facilitated analysis. The audio data, transcriptions and fields notes were securely stored on a password-protected computer. Each session started with an overview of the objectives of the study and full explanation about the nature of participation.

E. Analysing Data

According to Eisenhardt [33], analysing data is the heart of building a theory, but it is also considered the most difficult phase. Initially, the audio data of the interviews were transcribed by the investigator using oTranscribe[2]. Aiming to keep the anonymity and confidentiality, just the named investigators had access to the verbatim data collected during the interviews. According to the ethics issues (Section III-G), identifying participant information was stored separately from the survey instrument data so that the participant data is private.

We used ATLAS.ti[3] tool to analyse and synthesise the data, which were labelled thorough qualitative coding (open coding) to distill, identify similarities and sorts them to describe the phenomenon of simplicity in agile software development. Charmaz [35] clarifies that coding means that we attach labels to segments of data that depict what each segment is about (Figure 2).

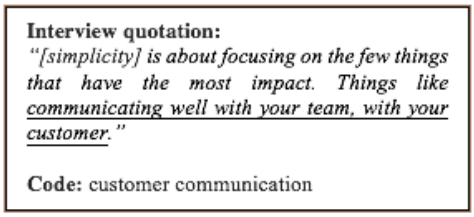


Fig. 2. Open Coding: building Codes

In sequence, the constant comparative method of qualitative analysis [36] was adopted to compare each code from the same interview and those from other interviews. As we continuously compared the codes, many fresh concepts emerged (Figure 3). As the process of data analysis progressed, relationships among categories (Figure 4) and memos were written to keep us involved with the analysis, thus helping to increase the level of abstraction of our ideas about the codes, concepts, categories and possibly even relationships. These memos forced us to look beyond impressions and see evidence through multiples lenses.

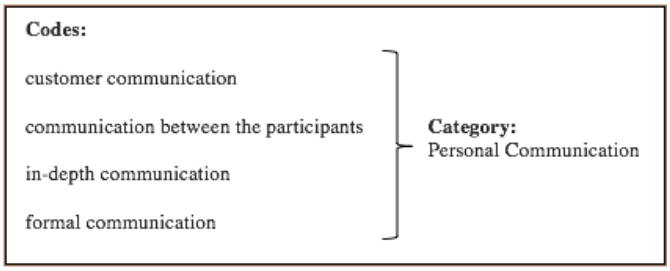


Fig. 3. Open Coding: Building Categories

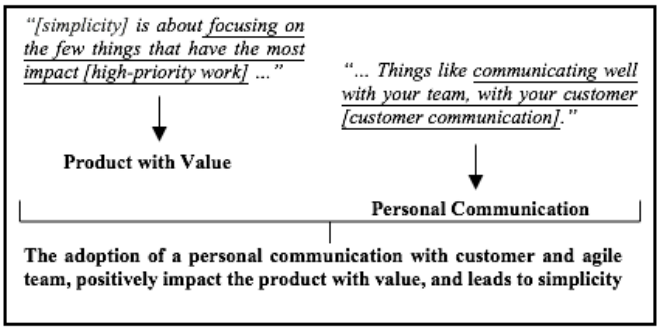


Fig. 4. Axial Coding: Building Relationships

F. Enfolding Literature

Shortly after the qualitative analysis, Eisenhardt [33] also considers that comparison of the emergent concepts or theory with extant literature is an essential feature of theory building.

In order to identify different phenomena, we deeply analysed and discussed our evidence and results, considering the broad range of studies, outlined in Section II. This way, we increase the confidence of emerging theory.

G. Research Ethics

We followed the norms of the Irish Institute of Health Science of Ireland that regulates research with human subjects. We applied the research to the Faculty of Science and Engineering Research Ethics Committee Board, and the ethical approval was obtained on 30th November 2015. The official approval allowed the conduction's interviews for the period from 23rd November 2015 to 31st October 2016.

Before the interviews, the consent forms and information sheet were presented. The consent forms were signed assuring anonymity, confidentiality of data and the right for the subjects to interrupt and withdraw without having to explain or give a reason at any time. All the subjects agreed to participate in this study and gave their written informed consent.

[2]http://otranscribe.com
[3]http://atlasti.com

IV. Results: Provisional Theory of Simplicity

This section provides detailed contextual information regarding the selected software company and participants involved in the study. In the sequel, the results relating the characterisation of simplicity, as well the central story that explains simplicity in agile software development are described.

A. Context Description

1) The Software Company: Based on the first part of our semi-structured interview and document analysis (Section III-C), the organisational demography profile was collected and analysed. It is a UK-based software development company, founded in 2000, with offices in London and Bristol. By the time of this study, the company had about 110 skilled practitioners. The company is classified as a for-profit organisation and small (defined in the EU recommendation 2003/361)[4]. It focuses on software consultancy, software development, software quality assurance, testing services, and software support. Such services are provided to variety types of software, including mobile phone applications, websites, e-commerce, and business applications.

2) The Participants: As mentioned in Section III-D, six skilled practitioners with different roles (software engineer, scrum master, project manager, lead developer, test leader, and designer) were purposely sampled, to achieve maximum variation in data collection. Table I compiles the participants' demographic profile. Due to anonymity and ethical issues (Section III-G), the participants are labeled by *P1* to *P6* codes. Table I presents a total of six individuals with professional experience in agile software development - the time of involvement with agile methodologies of the interviewers vary from six to ten years. Among them, four participants are male and two female. All of them worked or have been working in projects which use an agile process based on Scrum and Kanban, with exception to *P2* who were involved with Scrum, Kanban, eXtreme Programming (XP), Lean Software Development (LSD), and Feature-Driven Development methodologies.

TABLE I
Profile of Participants

	Role	Education	Experience (years)	Agile
P1	Software Engineer Scrum Master	BSc	6 to 10	Scrum Kanban
P2	Project Manager Scrum Master	MSc MBA	6 to 10	Scrum Kanban XP LSD FDD
P3	Project Manager Scrum Master Lead Developer	BSc	6 to 10	Scrum Kanban Lean
P4	Software Engineer Scrum Master Test Lead	MSc	6 to 10	Scrum Kanban Lean
P5	Software Engineer Designer	BSc	6 to 10	Scrum Kanban LSD
P6	Software Engineer Scrum Master	BSc	6 to 10	Scrum Kanban

B. Characterising Simplicity in ASD

Participants were asked about their understanding of simplicity in agile software development. The findings of this step are presented here. For each quote, the following format was adopted: *[P participant number]*. From the agile team's perspective, simplicity in ASD is broadly related to different kind of categories, such as (i) **lightweight process**, (ii) **knowledge acquisition**, (iii) **time-consuming**, (iv) **product with value**, (v) **personal communication**, (vi), and **reduction**. These categories are better discussed in the following excerpts.

According to participants, simplicity is broadly related to **lightweight process**. It means that the overhead of the process is kept as small as possible, to maximise the amount of productive time available for getting useful work done.

> *"I would say that simplicity in agile is about not having too much complicated process."* [P4]

> *"('simplicity in ASD) it has to be straight full to do if it's a process, you know it has to be not taking a lot of different steps."* [P5]
> *" (...) suppose like if an agile method is simple, there won't be too much processes involved."* [P3]

One of the benefits of agile software development is that organisations are capable of significantly reducing the overall risk associated with software development. Agile also focus on **reduction** of the amount of overhead in communication and documentation:

> *"Or we can even break it up. Well, they're kind of, let's say, a first goal and how we get to that and break into little tiny simple steps."* [P6]
> *"(...) it comes to mind lean in particular because at lean you got a concept model and anyone wants to reduce as much as possible."* [P2]

Additionally, participants reinforce the connection of simplicity with the essence of lean principles [37], such as eliminate waste, amplify learning, and deliver as fast as possible. Strengthened by the spirit of Lean, [P2] claims that:

> *"(simplicity in ASD) absence of that which does not add value (...) anyone wanna reduce that as much as possible. I think that the principles of lean really embodies what I think simplicity should be."* [P2]

When the team prioritise things that add knowledge at the beginning of the project, they could speed up the pace of understanding the requirements and therefore lower risks of misunderstanding [38]. Participants emphasised that **knowledge acquisition** is widely related to simplicity.

> *"Oh, I think there's probably two aspects to that (simplicity in ASD). One is that something has to be easy to understand (...)"* [P5]

Essentially, the agile team needs kick-start by understanding their readiness for project's context and focusing resources most effectively to minimise **time-consuming** (time spent on

[4]https://goo.gl/u0IPLr

process overhead). According to participants, time management also influences simplicity to avoid unnecessary time-consuming and downtime, as seen in the following quotes:

> *"(simplicity in ASD) it has to be quick to do."* [P5]
> *"(...) suppose like if an agile method is simple than it won't take me long. So it will be quick for me to use and either won't do the wrong thing."* [P3]

Fundamentally, agile teams must ensure consistent delivery of a product with value to customers. It makes sure that product or service always is of the highest quality possible [14]. According to participants, focusing on **product with value** leads to simplicity, as seen in the following quotes:

> *"(simplicity in ASD) about focusing on the few things that have the most impact. Kind of focusing on what the project is meant to achieve."* [P4]
> *"Yeah, it is getting down to what is the client optimal tools and follow one of those, and one of the simplest steps we can imagine with the least amount of work we can do to get that attempt done."* [P6]

In particular, the agile methodologies focus more on effective communication, collaboration, and coordination within a dynamic team environment than up-front planning and documentation [7]. According to evidence, **personal communication** with team and customer also promotes simplicity, as seen in the following quotes:

> *"(simplicity in ASD) is also about focusing on things like communicating well with your team, with your customer."* [P4]

C. Understanding of the Agile Manifesto

The participants answered questions regarding their understanding and explanation of the principle in the agile manifesto. The statement of simplicity "Simplicity – the art of maximising the amount of work not done – is essential" [7]. Generally speaking, participants agreed that this principle is aligned with the grounded categories of (i) **lightweight process**, (ii) **knowledge**, (iii) **time-consuming**, (iv) **product with value**, (v) **personal communication**, (vi), and **reduction** as follows:

The majority of participants understand the principle of simplicity from the perspective of product value, as exemplified in summary in the following excerpts:

> *"So I think you want to maximise the amount of work that doesn't add value - not done."* [P2]
> *"(...) if you're not wasting time doing, spending effort, doing things that aren't important. So I think you are aligned with that far."* [P3]

Furthermore, participants reinforce the connection of simplicity's principle with the essence of Lean [37].

> *"(...) I think it comes back to lean right? You need to maximise the work not done without compromising the results."* [P2]
> *"That's quite like Lean sort of. The art of maximising the amount of work not done. I feel like it's a good thing to aim for. Like, before you even do anything you should think of whether it is worth doing."* [P4]

On the other hand, participants also express their concern about the statement. P5 declares that the principle of simplicity is itself not simple - it is a convoluted definition.

> *" (...) it's kind of a convoluted way of putting it. It's not a simple way of defining simplicity. It's not a simple definition."* [P5]

D. Relating Practices and Building Propositions

The relationships among categories that emerged from the data analysis were organised in propositions, each one describing a particular view of the phenomenon, resulting in four key relationships.

Proposition 1: The inclination towards **lightweight process** enhances **personal communication**, reduce **time-consuming** tasks, and leads to **simplicity in ASD**.

In general, agile methods are very lightweight processes that embody less process, employ short iteration cycles; actively involve users to establish, prioritise, and verify requirements; and rely on tacit knowledge within a team as opposed to documentation [14]. Participants identified that enhancement of personal communication and reduction of time-consuming are caused by the promotion of lightweight process.

> *"(...) not having too much complicated process. Again about focusing on the few things that have the most impact. Things like communicating well with your team, with your customer."* [P4]

Proposition 2: The encouragement to **knowledge acquisition** promotes **simplicity in ASD**.

The agile development environment is considered as a platform for the extraction of knowledge without extra effort, overcoming cultural and psychological barriers [39]. Participants identified knowledge acquisition as an important tool that leads to simplicity in ASD and helps the longer term productivity and flexibility of the team, as seeing in the following quotes:

> *"(...) if your knowledge that comes out it's really useful, that certainly makes things simpler."* [P6]
> *"The more you understand about what you are trying to achieve and about what your system can do; that leads you to get to a solution faster."* [P5]

Proposition 3: The adoption of a **personal communication** with customer and agile team, positively impact the **product with value**, and leads to **simplicity in ASD**.

The personal communication among the team members and customers were highlighted as crucial to achieve simplicity in ASD and ensure effective feedback, as seen in the following quotes:

> *"(...) communication is definitely on the top of the pyramid of customer involvement"* [P2]
> *"(...) the less communication you have the more difficult it is to do anything"* [P5]

Proposition 4: The optimisation of **time-consuming** positively promotes on development of **product with value**, and leads to **simplicity in ASD**.

According to participants, the high-quality of product (value to the customer) is a consequence of a focused time prioritisation, towards a thoughtful reduction of unnecessary work done upfront.

> *"(...) if something is simpler in the context of ASD it should take less time. But in that, it should, say to break the things down in stories and stories smaller, simpler then each story will take less time to develop. They'll take less time to test, they'll take less time to understand the requirements"* [P2]

Table II outlines the highest ranking categories emerged from the axial coding. The level of empirical groundedness indicates the number of quotes (frequency of occurrence) that substantiates the existence of that category. In what concerns theoretical density, it states the number of codes linked with each code. As is evident from Table II, **lightweight process** (first row), gets the highest density (38) and highest groundedness (14).

TABLE II
GROUNDEDNESS AND DENSITY OF CODES

Categories / Sub-categories	Density	Groundedness
Lightweight Process	38	14
Reduction	8	9
Knowledge Acquisition	3	6
Time-consuming	6	11
Product with Value	7	3
Personal Communication	12	6

E. Building the Provisional Theory

The propositions presented in the previous section were combined to build the central story that explains a model of simplicity in agile software development. The relationships between the categories were analysed and focused using selective coding [40] to provide the overall theoretical picture. In this instance, the analysis showed that **lightweight process** emerged as the fulcrum category of the study. This central story is presented below and is illustrated in Figure 5.

Furthermore, when analysing the interrelationship between these categories, we find that there are two clusters of goals, organised in externally-oriented simplicity goals and internally-oriented simplicity goals. Firstly, composed by **lightweight process**, **knowledge acquisition**, **time-consuming**, **personal communication**. Secondly, aiming for **product with value**. This structure is inspired by the study proposed by Agerfalk [41], which grouped internal and external goals of the agile manifesto.

V. RESULTS: IMPLICATIONS FOR PRACTICE

The provisional theory is proposed to be an analytical tool to understand the simplicity phenomena in agile software development that aims to be useful and reflective in its approach to both, researchers and practitioners. In this sense, practitioners could benefit the agile team and organisation that desire to achieve simplicity through directions promoted by categories (elements of theory), which underpin the agile team towards a better focus on agile practices related to those categories.

According to Figure 6, the theoretical level (bottom) focuses on understanding simplicity in agile software development from grounded data. However, the theory (Figure 5) has an high-level abstraction and do not describe the practices related to those categories and propositions.

The second level sheds light on the relative importance of various agile practices; hence, it provides useful insights to the agile team to identify best practices which leads to simplicity in their projects. Furthermore, both levels have a symbiotic relationship, in which practices (level two) are specifically connected with categories of provisional theory (level one). In other words, the second level (implications for practices) is instrumental in helping the agile team to sort and select a set of best practices. Additionally, it is reflexive in recommending attitudes and behaviours.

Having emerged the provisional theory [42] considering the simplicity issues in ASD, we conducted a Systematic Mapping Study (SMS) [43][44] about simplicity in ASD to get a sense of how the emerged categories impacts into practices.

1) Lightweight Process: An important direction towards a *lightweight process*, is through the *simple design* and *refactoring* [45]. In order to reach an optimised process for projects, Hussain et al. [45] reflected it has to be tailored to the nature of each team and project in order to provide the benefits it promises. In this sense, from the very beginning the team has to keep the design as simple as possible. Refactoring of code also contributes in keeping the design simple.

These practices that lead to simplicity is also supported by Hunt [8]. Hunt and Thomas [8] argue that unfortunately many developers have a knack for making one of two errors: (i) Oversimplifying something that really is complex; (ii) Overcomplicating something that should be easy. As a solution, Hunt and Thomas boost simplicity by implementing essential features or framework functionality and by taking steps (such as constant refactoring) to avoid future problems that might require extra work.

In line with Hunt and Thomas [8], Shore [46] also endorses the focus on *continuous design* through simplicity and continuous improvement. It makes the code better and more maintainable over time, rather than less (simple but not simplistic design).

2) Personal Communication: Agile software development changes the nature of collaboration, coordination, and communication in software projects. It involves a radically new approach to decision-making in software projects, since project teams deliver working software in short iterations, which results in more frequent, short-term decisions, compared with a traditional software development approach [47].

According to the provisional theory (Figure 5), **personal communication** is being empirically identified as an important way to achieve simplicity. From this analysis, some agile

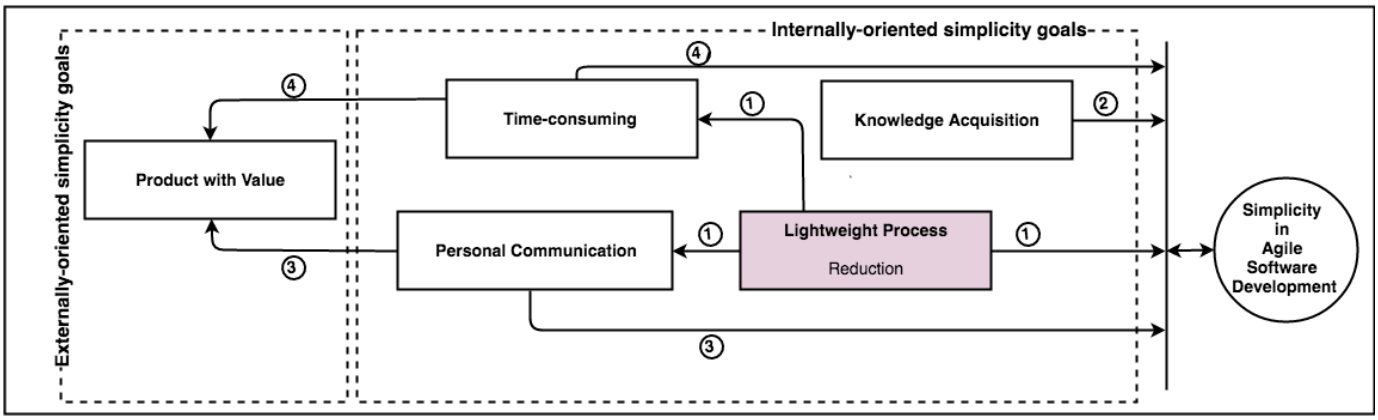


Fig. 5. Provisional Theory (Central Story of Simplicity in Agile Software Development)

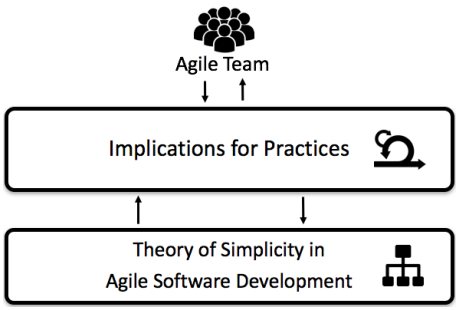


Fig. 6. Implication for Practices Usage

practices could be refocused as a vehicle to promote an effective communication with the team and customer, such as daily stand-up meetings, video conversation, and instant messaging rather than just mediated communication (e.g. bug tracking or Kanban board).

Besides the benefits related to *simple design* and *refactoring*, Hussain [45] also highlight practices related to *personal communication*, as instruments which contribute to improve not only our process but also to increase the overall morale of the team, such as: sitting together, face-to-face communication, feedback, stand-up meetings, the planning meetings, pair programming and reflection meetings.

According to Ambeler [48], modeling is a way to think issues through before you code because it lets you think at a higher abstraction level. The Agile Model Driven Development (AMDD) suggests you to work with the simplest (not just simple) tools. Ambler advocates the use of tools that strengthen the team's *personal communication*, such as whiteboards and paper, when work with users to explore and analyze their requirements. Amble emphasizes that "simple tools are easy to work with, inclusive (my stakeholders can be actively involved with modeling), and flexible, and they're not constraining". In this sense, with AMDD, programmers write the code progressively in step with the models and promotes an evolutionary approach, in which implementation occurs iteratively and incrementally.

3) Knowledge Acquisition: Findings from a systematic mapping study of simplicity in agile software development [44] meets *action learning* [49] as an evidence related to **knowledge acquisition**. The action learning cycle is by definition an iterative process and is never accurately conveyed as a single cycle of action. Instead, the action learning cycle should be represented as a continuous - possibly never-ending - process (continuous iterative loop of activities: plan, act, observe and reflect) [50].

4) Time-consuming: With respect to time, our provisional theory supports that *time-consuming* is one category that leads to simplicity in ASD. In this direction, Ambler [48], also suggest Test-driven development (TDD) as a way to think issues through before you code - AMDD. It lets the practitioner think at a higher abstraction level, saving time incrementally.

5) Product with Value: Several studies [51][49] suggest that agile projects can incorporate changes more easily and demonstrate business value more efficiently than traditional projects. In addition, our provisional theory addresses *product with value* as a consequence of overall categories that came up into our provisional theory of simplicity, which is classified as an externally-oriented simplicity goal (see Figure 5).

VI. Discussion

In this section, we address the theory in action, limitations, validity and reliability of our results.

A. Theory in action

The idea of instantiating the theory of simplicity make it easier for the agile team to tune the selected practices into their projects, following some recommendation:

1) Embody practices: So, we recommend that practitioners carefully study their projects' characteristics and try to incorporate those practice with the aim of enhancing simplicity in

their projects. Theses agile practices can be combined with overall agile projects and methods.

2) Rethinking (Mindset): In case these practices are already in usage, an important step is relating those themes (Figure 5) to the current agile practices. In a preliminary study, Santos et. al [30] defined simplicity as "the theoretical virtue disposing the team towards an analytic attitude that leads agile projects to be successful". This definition is supported by a conceptual framework was developed, which was then triangulated through a focus group with six ASD experts.

The conceptual framework of simplicity in agile software development is an invitation to practitioners to do what they already do, but to do so more consciously. This consciousness can make a substantial difference in real situations. From this perspective, rethinking means committing oneself to a course of action where plausible analysis exist, in order to reexamine the adopted practices.

B. Addressing Limitations, Validity and Reliability

This article presents the results of a qualitative case study to build a provisional theory to explain the phenomena of simplicity in software projects aiming to be a guide to future investigation. The validity and reliability of our results are discussed from the perspective proposed by Merriam [52].

Construct validity in qualitative research is related to the precise and clear-cut definition of constructs that is consistent with the meanings assigned by the research participants. Although we constantly compared and contrasted our construct definitions with the literature, a member checking can be conducted in the future to ensure our interpretations were consistent with those of the participants.

Internal validity, or *credibility*, is related to the extent that the results match reality. To increase credibility, we sampled participants with different roles in software projects, as described in Table I. The preliminary results were discussed between the authors to refine the categories and the propositions. A limitation is that we used only interviews as data collection method.

Reliability refers to the extent that the results can be replicated. Although we do not expect all our findings to be directly applicable to other contexts, it is possible to learn from the case description and decide to what extent the results can be applied or transferred to other situations. We tried to provide a rich description of the research method, the context in which the research was performed, and the results themselves.

Finally, a common challenge in qualitative studies is to reach *theoretical saturation*. In this study, we interviewed six participants with different points of view and perceptions about the studied phenomenon. Although they contributed to a rich description of the phenomenon, we aim to replicate our protocol in other organisations.

VII. Conclusion and Future Works

In this article, we presented the main results of a qualitative study focused on understanding how practitioners interpret simplicity in the agile software development projects and how these interpretations shape their work towards simplicity. This qualitative study was performed in a small software company in England that had just over 110 skilled practitioners.

Qualitative research techniques were applied to collect, analyse and synthesise data via interviews. Qualitative coding techniques (open coding, axial coding and selective coding) were employed to identify the categories that lead to simplicity in agile software development. From the interpretation of data, we constructed propositions that describe the relationships, and also build a provisional theory that explains the phenomena.

In summary, the obtained results showed that simplicity aligned with the grounded categories of lightweight process, knowledge acquisition, time-consuming, personal communication, aiming for a product with value. This consciousness can make an substantial difference in real situations: the deliberate use of simplicity theory in ASD can direct the attention to aspects of projects and agile team practice that might otherwise not be thought about, and hence it can influence the actions that follow, and the eventual results that might be achieved. As usual, when building mindsets, awareness sharpens the sight, especially in critical situations.

The next phase of the investigation is to apply the protocol in other organisations as well as to use the member checking technique to check the consistency of our interpretations and to raise the theoretical level. Additionally, we urge companies to participate in research projects that target goals relevant regarding simplicity in their agile software projects. Action research is a reflective process of progressive problem solving led by individuals working with professionals to improve the way they address issues and solve problems, while simultaneously contributing to new theoretical knowledge [32]. It would be one way to organise further collaboration between industry and researchers that would be highly pertinent to the topic of simplicity in ASD.

Acknowledgements

The authors would like to acknowledge the Brazilian National Research Council - CNPq (142296/2013-9), Brazil′s Science without Borders Program (205663/2014-1), the SFI grant 13/RC/2094 to Lero - the Irish Software Research Centre (www.lero.ie), University of Pernambuco - UPE, and University of Limerick for the support of this research.

References

[1] T. Dingsoyr, F. O. Bjornson, and F. Shull, "What do we know about knowledge management? Practical implications for software engineering," *IEEE Software*, vol. 26, no. 3, pp. 100–103, 2009.

[2] D. J. Fernandez and J. D. Fernandez, "Agile Project Management - Agilism Versus Traditional Approaches," *Journal of Computer Information Systems*, vol. 49, pp. 10–17, 2008.

[3] M. Winter, C. Smith, P. Morris, and S. Cicmil, "Directions for future research in project management: The main findings of a UK government-funded research network," *International Journal of Project Management*, vol. 24, no. 8, pp. 638–649, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.ijproman.2006.08.009

[4] H. Moura, "Software Project Framework," Federal Univerity of Pernambuco, Recife, Pernambuco, Tech. Rep., 2011, accessed on February 21, 2015.

[5] H. Moura and M. Skibniewski, "The Evolution of Management Thinking," in *International Research Network on Organizing by Project (IRNOP)*, 2011.

[6] T. Dybå, T. Dingsøyr, and N. B. Moe, "Agile Project Management," in *Software Project Management in a Changing World*, G. Ruhe and C. Wohlin, Eds. London: Springer Berlin Heidelberg, 2014, ch. 11, pp. 277–300. [Online]. Available: http://link.springer.com/10.1007/978-3-642-55035-5

[7] K. Beck, M. Beedle, A. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for Agile Software Development," 2001, http://agilemanifesto.org. Accessed on January 27, 2017. [Online]. Available: http://agilemanifesto.org

[8] A. Hunt and D. Thomas, "The trip-packing dilemma," *IEEE Computer Society*, vol. 20, pp. 106–107, 2003.

[9] K. Beck, *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley, 2004.

[10] J. Highsmith and K. Orr, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. New York, NY, USA: Dorset House Publishing Co., Inc., 2000.

[11] K. Schwaber and M. Beedle, *Agile Project Management With Scrum*. Washington: Microsoft Press, 2004.

[12] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[13] S. R. Palmer and M. Felsing, *A Practical Guide to Feature-Driven Development*, 1st ed. Prentice Hall, 2001.

[14] A. Cockburn, *Agile Software Development*. Addison-Wesley Professional, 2001.

[15] B. D. Floyd and S. Bosselmann, "ITSy - Simplicity Research in Information and Communication Technology," *Computer*, vol. 46, no. 11, pp. 26–32, Nov. 2013.

[16] T. Margaria, B. Steffen, and B. D. Floyd, "ITSy – Recommendation Document," University of Postdam, Postdam, Tech. Rep., 2011, accessed on January 27, 2017.

[17] B. Meyer, *Agile! The Good, the Hype and the Ugly*. Zurich, Switzerland: Springer, 2014.

[18] A. Baker, "Simplicity. The Standard Encyclopedia of Philosophy," 2013, accessed on February 03, 2017. [Online]. Available: http://plato.stanford.edu/archives/fall2013/entries/simplicity/

[19] J. C. Gambrel and P. Cafaro, "The Virtue of Simplicity," *Journal of Agricultural and Environmental Ethics*, vol. 23 VN-r, pp. 85–108, 2009.

[20] M. C. Nussbaum, "Non-Relative Virtues: An Aristotelian Approach," *Midwest Studies In Philosophy*, vol. 13, no. 1, pp. 32–53, 1988. [Online]. Available: http://dx.doi.org/10.1111/j.1475-4975.1988.tb00111.x

[21] T. Margaria and B. Steffen, "Simplicity as a Driver for Agile Innovation," *Computer*, vol. 43, no. 6, pp. 90–92, Jun. 2010.

[22] T. Margaria and M. Hinchey, "Simplicity in IT: The Power of Less," *Computer*, vol. 46, no. 11, pp. 23–25, Nov. 2013.

[23] D. Norman, "Simplicity is highly overrated," *Interactions*, vol. 14, no. 2, pp. 40–41, 2007.

[24] ——, "The way I see it: Simplicity is not the answer," *interactions*, vol. 15, no. 5, pp. 45–46, Sep. 2008.

[25] J. Maeda, *The Laws of Simplicity*, 1st ed. The MIT Press, 2006.

[26] ——, "Laws of Simplicity: design, business, technology, life," 2012, http://lawsofsimplicity.com/. Accessed on February 04, 2017. [Online]. Available: http://lawsofsimplicity.com/

[27] M. Lippert and S. Roock, "Adapting XP to Complex Application Domains," in *8th European Software Engineering Conference*. Vienna, Austria: ACM New York, NY, USA, 2001, pp. 316–317.

[28] B. Fitzgerald, M. Musia, and K.-J. Stol, "Evidence-based decision making in lean software project management," *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*, pp. 93–102, 2014.

[29] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1213–1221, 2012.

[30] W. Santos, A. Cunha, H. Moura, and T. Margaria, "Towards a Conceptual Framework of Simplicity in Agile Software Development: A Focus Group Study," in *8th Brazilian Workshop on Agile Methods (WBMA), Agile Brazil*, Springer, Ed. Pará, Brazil: Springer, 2017.

[31] C. Wohlin and A. Aurum, "Towards a decision-making structure for selecting a research design in empirical software engineering," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1427–1455, 2014.

[32] O. Badreddin, "Thematic Review and Analysis of Grounded Theory Application in Software Engineering," *Advances in Software Engineering*, vol. 2013, pp. 1–9, 2013.

[33] K. M. Eisenhardt, "Building theories from case study research," *The Academy of Managment Review*, vol. 14, no. 4, pp. 532–550, 1989.

[34] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: guidelines and examples*, 1st ed., Wiley, Ed. United States of America: Wiley, 2012.

[35] K. Charmaz, *Constructing grounded theory: a practical guide through qualitative analysis*. London: SAGE Publications Inc., 2006, vol. 10.

[36] B. Glaser, "The Constant Comparative Method of Qualitative Analysis," *Social Problems*, vol. 12, no. 4, pp. 436–445, 1965. [Online]. Available: http://www.jstor.org/stable/798843

[37] M. Poppendieck and T. Poppendieck, *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley Professional, 2006.

[38] P. Malmquist, *Agile Leadership*, 2013.

[39] M. Levy, D. Telekom, L. Ben, and T. Israel, "Knowledge Management in Practice : The Case of Agile Software Development Orit Hazzan Department of Education in Technology and Science," pp. 60–65, 2009.

[40] A. L. Strauss, *Qualitative Analysis for Social Scientists*. San Francisco, CA, United states: Cambridge University Press, 1987.

[41] P. J. Agerfalk, "Towards Better Understanding of Agile Values in Global Software Development," in *11th International Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'06)*, Luxembourg, 2006, p. 8. [Online]. Available: http://ceur-ws.org/Vol-364/

[42] W. B. Santos, A. Cunha, H. Moura, and T. Margaria, "Towards a Theory of Simplicity in Agile Software Development: A Qualitative Study," in *43rd Euromicro Conference on Software Engineering and Advanced Applications*, IEEE, Ed. Vienna, Austria: IEEE Computer Society Press, 2017.

[43] B. Moreira, W. Barbosa Santos, I. Júnior, H. Moura, and T. Margaria, "Simplicidade no Desenvolvimento Ágil de Software : Resultados Preliminares de um Mapeamento Sistemático da Literatura," in *XIII Brazilian Symposium on Information Systems (SBSI), 4th Workshop on Information Systems Undergraduate Research (WICSI)*. Lavras, MG, Brazil: Brazilian Computer Society (SBC), 2017, pp. 89–92.

[44] W. B. Santos, B. Moreira, I. Júnior, H. Moura, and T. Margaria, "Simplicidade no Desenvolvimento Ágil de Software: Um Mapeamento Sistemático da Literatura," in *Simposio Argentino de Ingeniería de Software (SAIS), 46th Jornadas Argentinas de Informática (JAIIO)*, Córdoba, Argentina, 2017.

[45] Z. Hussain, M. Lechner, H. Milchrahm, S. Shahzad, W. Slany, M. Umgeher, and T. Vlk, "Optimizing Extreme Programming," in *International Conference on Computer and Communication Engineering*. Kuala Lumpur, Malaysia: IEEE Computer Society Press, 2008, pp. 1052–1056.

[46] J. Shore, "Continuous Design," in *IEEE Computer Society*, vol. 21, no. 1. Los Alamitos, CA, USA: IEEE Computer Society Press, 2004, pp. 20—-22.

[47] J. Cunha, H. Moura, and F. Vasconcellos, "Decision-Making in Software Project Management: A Systematic Literature Review," in *International Conference on Project MANagement (ProjMAN)*, vol. 100, 2016, pp. 947–954.

[48] S. Ambler, "Agile Model Driven Development Is Good Enough," *IEEE Software*, vol. 20, pp. 71–73, 2003.

[49] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, vol. 50, no. 9-10, pp. 833–859, aug 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0950584908000256

[50] A. Heinze, G. Fletcher, T. Rashid, and A. Cruz, *Digital and Social Media Marketing: A Results-Driven Approach*, 1st ed. Routledge, 2016.

[51] T. Dyba and T. Dingsoyr, "What Do We Know about Agile Software Development?" *IEEE Software*, vol. 26, no. 5, pp. 6–9, sep 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5222784/

[52] S. B. Merriam, *Qualitative Research: A Guide to Design and Implementation*, 2nd ed., ser. Jossey-Bass higher and adult education series. United States of America: John Wiley & Sons, 2009. [Online]. Available: https://books.google.com.br/books?id=tvFICrgcuSIC