PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL MATONE CHANIN

A FRAMEWORK TO DESIGN SOFTWARE STARTUPS COURSES IN SOFTWARE ENGINEERING UNDERGRADUATE CURRICULUM

Porto Alegre
2020

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

A FRAMEWORK TO DESIGN SOFTWARE STARTUPS COURSES IN SOFTWARE ENGINEERING UNDERGRADUATE CURRICULUM

RAFAEL MATONE CHANIN

Doctoral Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. Rafael Prikladnicki
Co-Advisor: Prof. Afonso Sales

Porto Alegre
2020

Ficha Catalográfica

C456f Chanin, Rafael Matone

A framework to design software startups courses in software engineering undergraduate curriculum / Rafael Matone Chanin. – 2020.
158 f.
Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rafael Prikladnicki.
Co-orientador: Prof. Dr. Afonso Henrique Corrêa de Sales.

1. Software Engineering Education. 2. Startup Education. 3. Entrepreneurship Education. 4. Active Learning Methodology. 5. Software Startups. I. Prikladnicki, Rafael. II. Sales, Afonso Henrique Corrêa de. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

Rafael Matone Chanin

A Framework to Design Software Startups Courses in Software Engineering Undergraduate Curriculum

This Doctoral Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on December 16, 2020.

COMMITTEE MEMBERS:

Dr. Alfredo Goldman Vel Lejbman (IME/USP)

Dr. Lucia Maria Martins Giraffa (PPGEDU/PUCRS)

Dr. Sabrina dos Santos Marczak (PPGCC/PUCRS)

Dr. Rafael Prikladnicki (PPGCC/PUCRS - Advisor)

Dr. Afonso Henrique Corrêa de Sales (PPGCC/PUCRS – Co-Advisor)

I dedicate this work to my family.

“Don’t worry about failures, worry about the chances you miss when you don’t even try.”
(Jack Canfield)

ACKNOWLEDGMENTS

First of all I would like to thanks my advisor, Dr. Rafael Prikladnicki, and my co-advisor, Dr. Afonso Sales, for all their support, advice, and endless patience during my research. I would also like to mention the contributions given by Dr. Alan Santos, who gave me a great support in the beginning of this journey.

I would like to thank my committee members, Dr. Sabrina Marczak, Dr. Lúcia Giraffa and Dr. Alfredo Goldman for their helpful feedback during my PhD qualification, and for taking their time and effort to help me with my work[1].

I am also grateful to the SSRN (the Software Startup Research Network). This group helped me a lot in the development of my thesis. I would like to make a special reference to Jorge Melegati, Dr. Xiaofeng Wang, and Dr. Pekka Abrahamsson.

Finally, to the most important people on this planet: my wife Roberta, and my daughters Luisa, and Nina. Thank you for being there for me. This work is for you.

[1]This work was achieved in cooperation with HP Brasil Indústria e Comércio de Equipamentos Eletrônicos LTDA. using incentives of Brazilian Informatics Law (Law nº 8.2.48 of 1991).

UM FRAMEWORK PARA CRIAR DISCIPLINAS DE SOFTWARE STARTUPS NO CURRÍCULO DOS CURSOS DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

RESUMO

Os avanços tecnológicos possibilitaram que pessoas desenvolvessem produtos e serviços que podem alcançar milhões de pessoas ao redor do mundo. Estas iniciativas, que são criadas em um cenário de extrema incerteza, são chamadas de *startups*. Facebook e Google são exemplos de empresas de sucesso que um dia foram *startups*. No entanto, a maioria das startups quebra nos seus primeiros anos de vida. No contexto educacional, apesar das universidades estarem adaptando seus currículos para inserir conceitos relativos a startups, o desafio é: como prover uma experiência real para que alunos de engenharia de software desenvolvam *software startups*? Para buscar resolver este problema, o objetivo desta pesquisa é propor um *framework* que pode servir de guia para professores entregarem conteúdos referentes a software startups. Este trabalho foi dividido em três fases: exploratória, proposta inicial, e avaliação e evolução. Na fase exploratória, serão aprofundados os conhecimentos em startups de software e na forma como esse conteúdo é ensinado para os estudantes. Isso foi feito através de estudos empíricos, como um mapeamento sistemático da literatura, *surveys*, e estudos de casos. A partir destes estudos foi proposto um *framework* inicial, que foi chamado de *Challenge Based Startup Learning*. Finalmente, na fase de avaliação e evolução será conduzido um painel com especialistas com o intuito de obter feedback para poder melhorar e evoluir o framework. Dessa forma, esta pesquisa tem como objetivo contribuir nas formas e técnicas de se ensinar software startups no contexto dos currículos de graduação de engenharia de software.

Palavras-Chave: Ensino em Engenharia de Software, Ensino em Startup, Ensino em Empreendedorismo, Metodologias Ativas, Startups, Startups de Software.

A FRAMEWORK TO DESIGN SOFTWARE STARTUPS COURSES IN SOFTWARE ENGINEERING UNDERGRADUATE CURRICULUM

ABSTRACT

The advances in technology have enabled people to develop services and products that can reach millions of people around the world. These endeavours, that are developed under high uncertainty, are called startups. Facebook and Google are examples of successful companies that once were startups. However, most startups fail in their first years of life. From an education standpoint, even though universities are adapting their curricula in order to embrace startup education content, the challenge still remains: how can we provide real world experiences for software engineering students to develop relevant software startups? In order to fill this gap, the goal of this research is to propose a framework that can serve as guidance to faculty to better deliver software startup content to software engineering students. The work was divided into three research phases: exploratory, initial proposal, and evaluation and evolution. In the exploratory phase we deepen our understanding on software startup, as well as on how these concepts are taught to students. This was pursued through empirical studies, such as systematic mapping review, surveys, and case studies. From these results, an initial proposal of the framework was developed - the Challenge Based Startup Learning. Finally, in the evaluation and evolution phase, we conducted an expert panel in order to gather feedback from specialists, so we could improve the framework. Therefore, this research aims at contributing to the educational community when it comes to teaching software startups in the context of undergraduate software engineering curriculum.

Keywords: Software Engineering Education, Startup Education, Entrepreneurship Education, Active Learning Methodology, Startups, Software Startups.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1. INTRODUCTION

In the last decade, we have witnessed significant advances in technology, specially after the popularization of the Internet. Today, any person with software development skills is able to create applications that can be reached by millions (and even billions) of people [GPU+15]. Companies such as Google, Netflix, and WhatsApp are examples of these applications.

These endeavours, that are developed under high uncertainty, are called *startups* [BD12]. Most startups follow the *Lean Startup* methodology, which combines short software development cycles with constant interaction with users [Rie11]. The goal is to reduce risk by focusing on constant learning [CSS+18]. A startup needs to find a business model as quickly as possible, otherwise it may run out of resources before turning itself into a company. Therefore, a startup must focus on understanding what customers need, what they expect, and how much they are willing to pay for a solution [Col05].

Unfortunately, the majority of the startups do not survive the first two years of their existence [GPU+15]. External factors, such as competition and market instability, definitely account for this result. However, internal factors also play a key role in this regard [KMN08]. Teams that lack experience working with real projects and real customers tend to feel the pressure when results do not come up as expected. In addition, technical founders usually lack the business experience and knowledge that is necessary to run a successful company.

A startup priority is to find its business model, causing software quality to end up not being a major concern [GUP+14]. The sooner a startup can present its product or service to customers, the earlier it can get feedback to adjust the system. Therefore, failure must be accepted as part of the process [GUP+14].

From an education perspective, entrepreneurship has already been identified as a key component that needs to be further explored [FHL+17]. However, teaching entrepreneurship is not an easy task, since the focus is more on soft skills (such as teamwork, creativity, and self-awareness), rather than on hard skills. In addition, Fagerholm *et al.* [FHL+17] have also pointed out that it is not clear how education institutions can provide entrepreneurial content to engineers without jeopardizing the learning on core principles (such as programming).

Several software engineering/computer science programs have focused on entrepreneurship in the last years ([dCA13, Har15, KAP+06]). In addition, several technology-related courses are adapting their curricula in order to include startup/entrepreneurship content [DR08]. One of the biggest challenges reported on these studies is the lack of a realistic environment for student to work on their startups [PML+15]. Since the main goal of a startup is to solve real customers' problems, faculty must find ways to provide real challenges to students.

In this context, there is an opportunity to further study how entrepreneurship and software startup concepts, such as Lean Startup and Customer Development, are being taught to software engineering students. This process allow us to identify best practices as well as gaps and improvement points. As a result, a framework that can serve as guidance to faculty to better deliver software startup content to students is proposed. In order to do so, the research question proposed for this thesis is:

How can we teach software startup related concepts to software engineering students?

In order to answer the main research question of this study, we broke down this goal into the following questions:

- **RQ1**: *What is the state-of-art in literature in regards to software startup education for software engineering students?*
- **RQ2**: *Which tools, models, methodologies and frameworks are applied in a software startup education context for software engineering students?*
- **RQ3**: *What are the reported best practices in regards to teaching software startup for software engineering students?*
- **RQ4**: *What are the benefits and challenges of software engineering students learning the Lean Startup approach?*

1.1 Goals and Activities

The goal of this research is to create a framework to help faculty and the academic community to teach software startup content. The framework will be developed in the software engineering context. In order to do so, we proposed the following activities:

- Study software startup related concepts in order to understand them in depth;
- Conduct a systematic mapping study on software startup education for software engineering students;
- Understand how software engineering students perceive software startup concepts;
- Understand how instructors around the world teach software startup to software engineering students;
- Create a framework to teach software startup to software engineering students;

- Evaluate and refine the proposed framework;
- Propose an implementation of the framework in software engineering programs in Brazil;
- Document and report results to the scientific community.

1.2 Document Structure

The remainder of this document is organized as follows: Chapter 2 presents the background on fundamental concepts. Chapter 3 describes the proposed research methodology. Chapters 4, 5, 6, and 7 present the experimental studies undertaken in the first phase of this research. Chapter 8 introduces the initial proposal for this study. Chapter 9 depicts the evaluation and evolution of the proposed framework. Finally, Chapter 10 shows our conclusion and final thoughts.

2. THEORETICAL BACKGROUND

This chapter presents the background on the core concepts related to this research: digital entrepreneurship, which is broken down into Software Startups, Lean Startup, and Customer Development, entrepreneurial education for the 21st century, including several active learning methodologies, and entrepreneurship in software engineering education.

2.1 Digital Entrepreneurship

In this section, we explore three important concepts related to digital entrepreneurship: *software startups Lean Startup* and *Customer Development*.

2.1.1 Software Startups

Though software startups have recently had a large economic impact across the globe, there is no clear consensus on what exactly a software startup is [UAW+16]. Startups are not simply small, new companies seeking to grow into larger corporations, nor is there a clear point after which a startup has clearly grown into a mature company. Despite the lack of a consensus on an exact definition, some shared understanding of characteristics that define startups does exist.

Startups operate under a lack of resources, both in terms of time, manpower, and finances [PGU+14, Sut00]. They largely rely on external funding especially early on in their lifecycles, and have little to no prior operating history [Bla13, Sut00]. Though not every single startup is a software startup or even focused on technology-based products, startups by definition are often nonetheless considered to be software or more generally tech companies, especially by practitioners [Bla13]. Software startups specifically, however, operate in particularly volatile markets, using current top-of-the-line technologies to engineer innovative products and services [HHO03]. This, combined with the scarcity of resources, leads to software startups generally operating under highly uncertain conditions [Rie11].

Perhaps the most important difference between a conventional small business and a startup is that startups are characterized by clear intentions for high growth. While small companies generally wish to grow, and will usually do so if presented with a clear opportunity, startups are founded with plans for high potential growth from their inception. Indeed, startups typically seek a particularly highly scalable business model [Bla13]. In the case of software startups in particular, this is further highlighted by the digital nature of software: digital goods are easily distributed or sold world-wide.

Another important characteristic of a startup is that startups are temporary: a startup does not want to keep being a startup. A startup will either fail somewhere along the way or grow into a mature organization. Though it is unclear when exactly a startup ceases to be a startup, drawing from the definition of Blank [Bla13], one could argue that a startup ceases to be a startup when it has found the highly scalable and sustainable business model it sought.

For the purpose of this study, we consider startups to be temporary organizations seeking a highly scalable business model. Software startups, on the other hand, are considered startups that deliver value primarily through software. For instance, though Uber is a taxi company, it nonetheless delivers its value to its customer through the software used to access the service; after all, it does not actually own a single taxi.

Software startups are typically associated with success stories such as that of the aforementioned Uber. However, the majority of software startups fail [Cro02], with some estimates citing numbers as high as 95%. Despite their high rate of failure, software startups have had a notable impact on the economies of more developed countries, especially in the last decade [UAW+16]. As a result of recent technological advances, an average supermarket laptop can now be used to develop software which can then be hosted in the cloud, whereas twenty years ago the cost of developing and distributing software was much higher. This sharp decrease in required resources in software development has resulted in an increasing number of software startups.

As software startups have become more numerous and impactful at an international economic level [UAW+16], they have also become increasingly relevant from the point of view of education. It is not uncommon for software engineering students to involve themselves in software startups both during their studies and after graduation. In fact, software startup practitioners in general tend to be inexperienced [KCM+14, Sut00]. Just as entrepreneurship in general is taught in educational institutes across the globe, startups as one of its facets have grown prominent enough to warrant unique focus. As established in this section, startups differ from conventional small companies, making generic entrepreneurship education not fully applicable to them.

In terms of business, whereas founding a conventional company would see one write a detailed business plan for investors and perhaps take out a loan cover material costs as well, startups prefer one-page-long business model canvases over business plans and are far more focused on acquiring outside funding through short public talks referred to as pitches. Though startups are not completely unlike conventional small businesses at their core, startup entrepreneurship has grown into a sub-culture with its own community and jargon. Startup events across the globe attract famous practitioner speakers and also successful startups that bring motivational success stories for up-and-coming startup practitioners.

Startup incubator organizations and various startup-related societies support startups during various stages of their lifecycles. As a result, startup companies use constructs

that differ from conventional business vocabulary and have their own practices, for instance, in terms of searching for investments.

In regards to software development processes, software startups have been shown to develop software differently from small, medium-size and large corporations [PGU+14]. Software startups often use varying agile methods and practices, or even develop software purely ad hoc [PGU+14]. Similarly, software startups are characterized by particularly high levels of technical debt. As time-to-market is essential and the lack of resources forces software startups to develop quickly, software startups often accumulate technical debt. After all, in the case of failure, which is the fate of most software startups [UAW+16], that technical debt will never have to be addressed. Of course this issue is not exclusive to startups. Software projects from all sorts suffer from technical debt; however, due to the nature of startups, this is some almost impossible to avoid.

Just as organizations such as startup incubators and various startup event organizations have sprouted to support the high number of emerging software startups, some scholars have also begun to devise and carry out startup-related university courses. Whereas business and entrepreneurial education in general has a long-standing history in the academia, startup and software startup education as its subset is still in its infancy.

2.1.2 Lean Startup

Inspired by the Customer Development and Lean Manufacturing, Eric Ries proposed the Lean Startup [Rie11]. This methodology has the premise that every startup is a set of experiments to test hypotheses the startup makes about a given business idea. In order to do so, startups build Minimum Viable Products (MVPs) [Li07].

An MVP is used to collect feedback from potential users/customers in order to identify and to model the next experiment [GUP+14]. One of the common issues with this approach is to understand what “minimal” and “viable” means [LT16]. The development of an MVP should focus on the smallest effort in order to test a given assumption of the business model and to obtain maximum validated learning. Figure 2.1 presents the Lean Startup process.

In addition to the process itself, the Lean Startup methodology is based on a set of 5 principles:

1. *Entrepreneurs are everywhere*: there is no special setting to work on a startup. It can be in an office, in a garage, and even from home;

2. *Entrepreneurship is management*: a startup is not just a technical product. It requires management as any other organization;

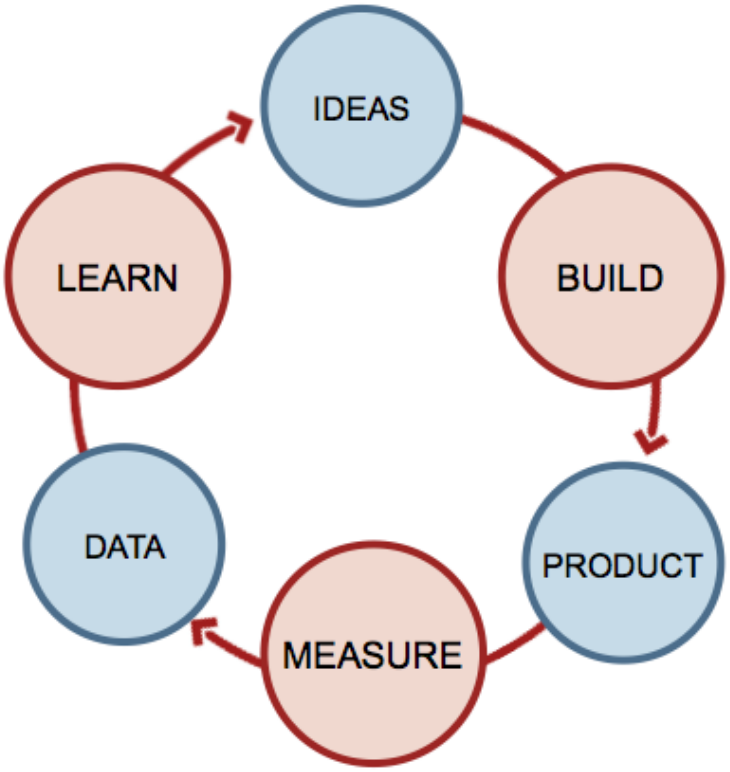

Figure 2.1 – Lean Startup process [Rie11].

3. *Validated learning*: the goal of a startup is to learn about a given context/market. The mindset should be focused on creating and validating experiments;

4. *Innovation accounting*: the focus should be on measuring progress, setting up milestones, and on prioritizing task;

5. *Build-measure-learn*: all processes should be geared to accelerate the feedback loop.

Despite of the Lean Startup popularity, it is an approach proposed in a non-scientific book based on anecdotal evidences. Nevertheless, according to Frederiksen and Brem [FB17], there is substantial evidence in scientific literature to support the Lean Startup efficacy.

2.1.3 Customer Development

Steve Blank [Bla13] proposed and created the Customer Development process based on the premise that most startups fail from lack of customers, rather than product development issues. He argues that there are several processes to manage product development, but there is no formal process to manage customer development.

The Customer Development process is divided into four phases (see Figure 2.2):

- *Customer Discovery*: state and validate both the problem and the customer hypotheses. A startup should only move to the next phase once there are evidences that a group of customers are willing to pay to solve a given problem;
- *Customer Validation*: develop and test a sales process. If the startup is not able to validate the model, it has to go back to the discovery phase. This move is called *pivot*;
- *Customer Creation*: once the business model is validated, it is time to grow the customer base in order to validate the business feasibility;
- *Company Building*: put formal management into place and create growth strategies to scale the business.

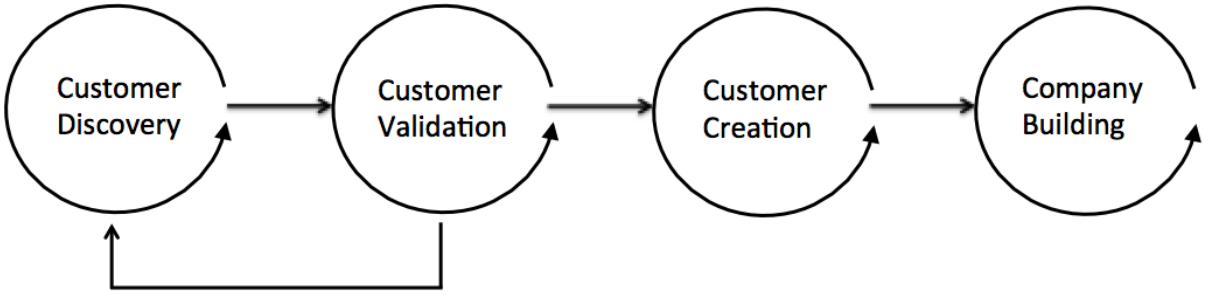

Figure 2.2 – Customer Development process [Bla13].

If a startup is not able to validate its business model before running out of resources, it will not succeed. Hence, it is very important to perform an effective customer discovery process [BCC16]. This can be done by interviewing potential customers, using marketing strategies, such as developing landing pages or online advertisements to validate a given value proposition, or even by manually delivering the product/service to the customer (the concierge approach).

2.2 Entrepreneurial Education for the 21st Century

Entrepreneurs are a heterogeneous group. Therefore, there is a need for pedagogical approaches that encompasses all types of students; from those less creative, to the ones that are capable of developing amazing solutions [FHL+17]. Berglund and Wennberg [BW06] suggest the implementation of flexible educational structures, embracing both individual and group needs.

Gorman *et al.* [GHK97] developed a ten-year literature review on entrepreneurship education and they found out interesting insights. One of them is that entrepreneurship education should focus on developing students' soft skills *et al.* [GHK97]. The idea behind

this conclusion is that most startups do not fail because they can not (technically) build their products; they fail from lacking managerial and personal skills. The second insight, which is connected to the first one, is that education should be focused on concrete experiences through active participation on projects and tasks.

In addition, Gorman *et al.* [GHK97] research clearly indicates that the vast majority of knowledge and skills required to become an entrepreneur can be taught. Moreover, teaching methods and strategies can be improved through active participation. Another interesting take away from this research is that even though educational programs can, at least, encourage students to pursue a career as a business owner, small business owners resist startup training. Because of that, the authors suggest that startup training and education should occur in an environment where the university has control over the process. Even though it may seem to be a good idea to involve students in real startup organizations, there is a risk of jeopardising learning, specially if the startup does not have mature processes implemented.

In regards to teaching methodologies, experiential learning seems to be the most acknowledged approach in this context. Problem-Based Learning, Project-Based Learning, Task-Based Learning, and Challenge Based Learning are just a few examples of these frameworks. These approaches engage students and improve their performance [DSVDBS05, IPSL09]. Moreover, they move students from being a spectator to being actively engaged into solving real world problems [Inc08, Inc09]. The following sections briefly describe the four aforementioned approaches.

2.2.1 Problem-Based Learning

The Problem-Based Learning methodology (PBL) is centered on working on open-ended problem instead of presenting content to students [MG09]. In this process, there is no single solution to a given problem; students must collaborate in order to develop their own solution according to their resources and skills [İB13]. In order to do so, they must not only develop the skills to look for the knowledge needed to solve the problem, but they also need to develop soft skills, such as communication and leadership.

In this context, instructors should prepare classes in such a way that students feel comfortable asking as many question as they need. The problem can be presented in any format: text, image, or video. This process must present a situation that contains a problem. If possible, this scenario should be broad and open with little or no specification.

Once the problem is presented and clarified, students develop hypothesis and a plan in order to learn more about the situation so they can find the solution. Of course it is good if students find a solution, but the process is more important than the solution itself; the learning happens by asking questions and by researching the topics around the problem.

In sum, the PBL process can be describe in the following stages:

1. The situation is presented to students;
2. The problem is defined;
3. Student create hypothesis and define a work plan;
4. Research;
5. Lessons learned and conclusions regarding the problem.

Throughout this process students must become more independent and active. Instructors should act as facilitators; they must stimulate and engage students as well as help them in case they get stuck or if they are going to the wrong direction. Instructors should also provide advice and assistance. This means that instructors play an important role in this context. Since students are partially responsible for their learning, if there is no guidance the learning process may be jeopardise.

2.2.2 Project-Based Learning

Project-Based Learning (PRBL) is very similar to the Problem-Based Learning methodology. However, in this case, students work on a specific project, rather than on a problem [FW03]. Thomas [Tho00] argues that by working on a project, students must learn planning, investigation, execution, and decision-making. The biggest challenge on implementing this approach in an educational environment lies on changing the “traditional” roles of both students and instructors. While students must take action into the learning process, instructors become tutors [IPSL09].

Another difference between PBL and PRBL is that the former focuses on the acquisition of knowledge, while the later focuses on the application of knowledge. Since knowledge is acquired during the course of the project, PRBL is prone to the same problem as PBL, which is not going deep enough into the content.

In order to avoid this issue, students need to be assessed during the process so instructors can understand whether knowledge is being acquired or not. However, assessing is hard in both PBL and PRBL since exams, for instance, usually focus on technical content and not on the learning process. Thus, instructors must find the right approaches and tools to measure not only hard skills, but also soft skills.

According to Pardo [Par14], instructors should observe students throughout the process and ask for reports and presentations. Although observations are somehow subjectives, this approach is better than just assessing student though an exam.

2.2.3 Task-Based Learning

The Task-Based Learning methodology (TBL), as the name suggests, focus the learning process by working on tasks. This approach is similar to the Project-Based Learning one, but tasks are considered smaller portions of work if compared to projects. The learning process is broken down into the pre-task activity (when students are exposed to the task), the planning, the execution, and the evaluation [FF09].

According to Fetaji [FF09], this approach has three main advantages:

1. Learning designed around tasks is more effective than traditional memory-based or purely apprenticeship-type learning;
2. Learning structured around tasks is an efficient approach to learning;
3. Task-based learning is likely to lead to more relevant and appropriate education.

Fetaji [FF09] also mentions that TBL offers action and reflection. In other words, students can put their learning into practice and can also reflection on the learning process.

2.2.4 Challenge Based Learning

Challenge Based Learning (CBL) [NCT16] was developed by educators working with *Apple Inc.* [NC08] and has been implemented both in educational and corporate environments. It is a learning framework based on solving real world challenges and problems. From an education standpoint, students obtain knowledge by working with technology on open-ended problems and challenges in collaborative and (when possible) heterogeneous teams.

During the CBL learning process, professors/lecturers, students and other stakeholders work together as active collaborators. Divergent thinking and creativity are stimulated throughout the whole process. Moreover, the focus is not only on the final deliverable (the challenge solution), but also on the learning process itself. Students and lecturers must reflect from time to time on their learning/teaching evolution.

The CBL framework is divided into three interconnected phases: *Engage*, *Investigate* and *Act*. Each phase includes a different set of activities:

- ***Engage***:

  Big Idea: a broad concept that can be explored. It has to be a topic that is engaging for students;

Essential Question: the question related to the *Big Idea* that students want to explore;

Challenge: a call to action derived from the essential question. It should be actionable and exciting.

- ***Investigate***:

Guiding Questions: questions related to the challenge. Includes everything that needs to be learned;

Guiding Activities and Resources: list of activities and resources that can help students pursue the challenge;

Analysis: sets the foundation to develop the solution to the challenge.

- ***Act***:

Solution Development: based on the learnings from the previous steps, the solution is implemented;

Evaluation: verifies if the solution has addressed the challenge or if it needs refinement.

Johnson and Adams [JA11] have showed that the use of active learning methodologies improves students' learning when compared to traditional methods. Additionally, the engagement and the soft skills acquired during the process is also perceived as a big advantage not only for students, but also for other stakeholders involved.

2.2.5 Methodologies' Comparison

Santos [San16] developed a comparison among several active learning methodologies in his PhD thesis. Table 2.1 presents the most important aspects to be observed regarding these methodologies.

Attribute	CBL	PBL	PRBL	TBL
Focus on students' autonomy	X	X	X	
Teamwork	X	X	X	X
Reflection	X	X	X	X
Everyone is a learner	X			
Work on real world challenges	X			
Use of technology	X			

Table 2.1 – Active methodology comparison.

It is important to point out that even though only CBL was designed to work with technology, the other methodologies have also been adapted to work on a technology environment [San16]. The same is valid for the "real world challenges" attribute. PBL, for instance, is often adapted to work with real world problems. The point is that CBL is the only methodology that explicitly prescribes all aforementioned attributes. In addition, CBL seems to fit well with the characteristics of a software startup, since it aims at solving a real world problem by using technology. Because of these reasons, we chose CBL as the active methodology to further explore the objectives of this work.

2.3 Entrepreneurship in Software Engineering Education

Nowadays, business education is becoming so relevant that some universities across the world have already recognized the importance of providing entrepreneurial skills to engineering students [PML+15]. The fact is that being technically competent is not enough. Understanding customers' needs and knowing how to deliver real value to people is a must in this new economy. Even though some initiatives can be perceived in the academic world, the lack of business education to software engineering students has been a noted issue for a long time. Back in 2000, Shaw [Sha00] already observed that software engineering students rarely faced non-technical issues. In 2009, a survey of 119 Spanish students reported that they were not receiving enough training in innovation and entrepreneurship [ESRTCBS09]. Several courses have focused on entrepreneurship for software engineering students in the last years [dCA13, Har15] and other studies focused on the importance of business concepts for technical students [KAP+06].

Some entrepreneurship courses teach emerging methodologies for startup development like Lean Startup [Har15, Nob11, WKA16] and Customer Development [CPF+17]. Fitzgerald and Stol [FS17] argue that these methodologies are consequence of the need of

a closer integration between business needs and development. However, despite of consisting interesting concepts and ideas, these methodologies can be difficult to comprehend and to apply in practice [BOBL13], which may hinder software engineering students to grasp business skills by following the courses.

In addition, when dealing with real users/customers, students need to be creative in order solve problems that may arise. In other words, students must not only develop software development skills, but also many soft skills, such as critical thinking, adaptability, and teamwork. This combination enables students to be prepared to deliver software that is not only technically well-design, but that also delivers value to a group of people [PML+15].

Encouraging working with rapid and iterative development cycles, along with constant feedback as well as self-evaluation is harder in an academic setting, due to students' lack of experience [GSX15]. However, in order to meet current market demands in term of software development skills, it is necessary to combine creativity and innovation along with technical knowledge [DKI+10].

Thus, it is crucial to implement an engaging and collaborative approach in order to help students understand what is takes to develop a real startup. One of the methodologies that combines the educational process with interactive learning and applied to everyday challenges, for instance, is the Challenge Based Learning (CBL) [NCT16]. This methodology works well not only in real world projects, but also in educational settings [SSFK18, SSFN15]. CBL encourages problem-solving through activities such as reflections, self and team evaluations and challenges carried out during the process [NC08].

There are a few publications that connect Challenged Based Learning and software development practices [BNRM17, SSFN15]. Agile practices are often used in a startup context. In this sense, our work was inspired by Santos *et al.* [SSFN15], which presented the first framework that formally combines CBL methodology and Scrum.

Santos *et al.* [SSFN15] explored a case study of an iOS development course that used CBL as the methodology for teaching. Even though there was no formal software development process put it place when the first course was offered, it became clear that student would need one. Therefore, a framework was proposed and implemented. In this framework, students do not start implementing their mobile application until they are done with their investigation. If something goes wrong during this step, it might be to late to go back and adapt the solution. Moreover, the model does not make any reference to outside users or customers. Even though this work presents a great methodology for teaching mobile application development, the lack of real users/customers interaction is clearly a gap. Teachers can assess whether students learned how to develop a given mobile application. However, nothing is mentioned regarding the validation of the proposed solution by real users. This is the gap that this work intend to explore.

2.3.1 Current Guidelines for the Brazilian SE Curriculum

In Brazil, there is a guideline provided by the Brazilian Computer Association [AZN+17] for all computer-related undergraduate programs. This document is a very useful resource for universities implementing such programs. In regards to software engineering programs, there is a specific track called “Entrepreneurship and Innovation” that encourage software engineering programs to develop entrepreneurship concepts not only within an organization, but also when creating a new company.

This document was created by a group formed by several Brazilian undergraduate institutions. The goal is to offer guidance and orientation regarding the professional formation in software engineering for new and existing programs.

The guidelines are based on a work proposed by Nunes *et al.* [NYN16], which focuses on competencies refinement. According to the authors, competency is the ability to articulate and to consolidate knowledge, abilities and atitudes, putting them into action by solving problems and by dealing with unpredictable situations in a given context. By analysing these competencies, it is possible to determine what need to be taught in the classroom.

Software engineering programs are somehow recent in Brazil. The first program was created and approved in 2008. Currently (at the time of writing this thesis), there are around 50 universities that offer software engineering programs, according to the National Higher Level Education Database Programs[1].

Even though different computer-related programs have different focuses, they all share a common knowledge base. Topics such as math, algorithms, and software development processes are present in all programs. When it comes to entrepreneurship-related knowledge, the following aspects are expected from software engineering graduates [AZN+17]:

- ability to reflect over a software project in order to understand its impact on people and on society;
- understand the economic and financial aspects associated with a new software project;
- recognize the importance of innovation and creativity as well as business opportunities;
- identify new business opportunities and develop innovative solutions.

In regards to tracks, the guideline suggests 7 main topics:

1. Computing fundamentais, math and processes;

[1] http://emec.mec.gov.br/

2. Entrepreneurship and innovation;
3. Professional practices and abilities;
4. Software processes and management;
5. Software requirements, analysis, and design;
6. Software development and testing;
7. Software quality.

For each of track, competencies and contents are derived and classified according to depth of knowledge suggested. The classification entails the following components:

- Knowledge: remembering a given content that was taught in class.
- Comprehension: understanding a given content that was taught in class.
- Application: using a given content in new and concrete situations.

As it can be observed by looking at the tracks, there is a clear focus on entrepreneurship and innovation in the guideline for the Brazilian software engineering curriculum. The goal of this track is to encourage students to create and develop innovative businesses within organizations as well as building a new company. Table 2.2 summarizes the competencies that should be developed along with the corresponding classification and content, which is described in the guidelines provided by the Brazilian Computer Association [AZN+17].

Competencies	**Classification**	**Content**
Identify business opportunities	Comprehension	Entrepreneurship
Create business models, transforming ideas into products or services	Application	Entrepreneurship, business modelling, and business model frameworks
Plan innovative projects	Application	Entrepreneurship, business planning
Fundraising for innovative projects	Comprehension	Entrepreneurship, and funding
Manage innovative businesses	Comprehension	Entrepreneurship, business administration, and marketing

Table 2.2 – Software engineering entrepreneurship track.

Regarding teaching methodologies, the guideline suggests that software engineering programs should focus on student-centered strategies so that student can learn by reflecting over practical activities. Project based learning, problem based learning, competency-based learning, and case studies are mentioned in the document as possible methodologies to be used in order to deliver a better experience to students.

Moreover, the author of these guidelines are also concerned about soft skills development, such as communication, negotiation, teamwork, leadership, and management. In today's world, it is vital to address such skills since the market is becoming more complex as well as more competitive.

In sum, as it can be observed, the guidelines regarding entrepreneurship are somehow generic; the document provides a vision for this track, but it does not explain how this concepts and contents could be applied in practice in a classroom environment. Therefore, in the next section we will understand how some of the software engineering programs in Brazil are interpreting and implementing this track in their contexts.

2.3.2 Entrepreneurship in the context of SE Programs in Brazil

In this section we will study a sample of the software engineering programs offered in Brazil in regards to their implementation of entrepreneurial concepts. We choose 7 programs to fully analyze. This sample was picked based on their overall evaluation by the Ministry of National Education. In Brazil, every higher education program (either private or public) is evaluated by a set of parameters, such as students' grade on a national exam (ENADE), number for MSc and PhD professors, facilities and class activities and resources. In some situation, government representatives visit the university in order to further evaluate the context. This process assign a grade between 1 and 5 to the program. Details about this process can be found in the Ministry of National Education website[2].

Given this explanation regarding higher education program evaluation, we selected 7 programs - 4 programs that currently hold the highest grade (5), and 3 programs holding an overall evaluation of 4. The rationale behind this approach is that we definitely need to analyze some of the best programs in the country, but it is also interesting to verify whether other good programs also provide interesting insights. Moreover, programs with low or no evaluation could mean that they are still new or under development. Therefore, we decided not to take those programs into account at this moment.

PUCRS

The software engineering program at Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)[3] was created in 2014, and currently holds a 5 on the aforementioned government assessment. One interesting aspect of this program is software engineering experimental agency. This lab was conceived to be a hands-on learning environment that (i) allows students to work on real projects, but always focusing on the learning process (ii) inte-

[2] http://emec.mec.gov.br

[3] https://www.pucrs.br/politecnica/curso/engenharia-de-software/

grates all concepts learned throughout the program; (iii) connects students with researchers as well as companies or other stakeholders that would be interested in working together on a software project. This is definitely an innovative and creative approach to teach software engineering concepts to students. Moreover, students are introduced to entrepreneurship concepts since they ended up working with startups and other similar organizations.

Aside from that, the program offers a 2-credit course, called Entrepreneurial Education, in which students learn entrepreneurial concepts. In addition, they study other important aspects, such as motivation and leadership. They also oversee practical aspects by looking at important tools that help entrepreneurs on modelling their businesses (such as the business model canvas), as well as the steps that need to be undertaken in order to do so.

In sum, software engineering experimental agency seems to be a great opportunity for students to develop entrepreneurial skills, although this is not the primary focus of the agency. In regards to course offerings, it looks like a 2-credit course can only give students an overview of the entrepreneurial process, with no time to actually to the learning into practice.

UnB

The software engineering program at Universidade de Brasília (UnB)[4] was created in 2008 and it also holds a 5 for its government assessment. One of the program's main objetives is to implement pedagogical practices that foster autonomy, creativity, critical thinking, entrepreneurial mindset, in order to give students the opportunity to think about developing new organizations.

The program offers a course called Project Development I and II, which accounts for 120 hours. According to our analysis, this course has a similar goal as the software engineering experimental agency offered by PUCRS. It is an opportunity for students to work on a project-based course so they can put their learning into practice. According to the course syllabus, the core focus is the development of soft skills, such as communication, and teamwork, as well as some hard skills related to software development life cycle. The document mentions that students have the opportunity to understand entrepreneurial concepts.

It looks like this program provides a great hands-on opportunity for students to practice their software engineering knowledge by working on real projects as well as on toy projects. However, it seems that entrepreneurial concepts are not at the core of this process.

[4] http://fga.unb.br/software/engenharia-de-software

UTFPR

At UTFPR (Universidade Tecnológica Federal do Paraná)[5], the software engineering program was created in 2014 and currently holds a 5 for its government assessment. Interestedly, one of the program's main objetives is exactly the same as the one of UnB, which is to implement pedagogical practices that foster autonomy, creativity, critical thinking, entrepreneurial mindset, in order to give students the opportunity to think about developing new organizations.

This objective is translated in this program as a 2-credit course called Entrepreneurship, in which students study business plan, business opportunity, and entrepreneurial profile. The program also offer 2 other courses as electives - Knowledge Management, and Leadership and Management - that are suppose to foster entrepreneurial mindset. However, by looking the courses syllabus the focus of both courses are mostly on established organization.

In this program, it looks like students briefly and superficially touch on entrepreneurial concepts throughout the program. In other words, they do not actually practice the process of developing a startup or any other new venture.

UFG

The software engineering program at Universidade Federal de Goiás (UFG)[6], which holds a 5 for its government assessment, was created in 2008. In the programs goals and objectives, there is no reference to entrepreneurship; the objetives are focuses on software engineering processes and practices.

In regards to courses related to entrepreneurship, none of the courses of this program focus directly on this concept. There is one course - Ethics, Rules, and Professional Behavior - that mentions in its syllabus a few topics and concepts related to the development of entrepreneurial mindset.

By looking at this program website, as well as its pedagogical project document, it looks like there is no focus on fostering the development of new startups or new ventures.

UFMS

At UFMS (Fundação Universidade Federal de Mato Grosso do Sul)[7], the software engineering program was created in 2017 and currently holds a 4 for its government assessment. One of the program's main objetives is to foster creativity and an innovation mindset, by understanding business perspectives and opportunities.

[5] http://portal.utfpr.edu.br/cp/engenhariadesoftware
[6] https://inf.ufg.br/p/30140-engenharia-de-software
[7] https://www.facom.ufms.br/engenharia-de-software/

The program offers 2 courses related to entrepreneurship. The first one is a 2-credit course called Entrepreneurship. The goal of this course is to teach basic concepts, and to motivate students to become entrepreneurs. It also touches on building sustainable and scalable business models by developing a business plan. The second one is a 4-credit course called Introduction to Administration, which focuses on organizations environment, strategies, planning, and management.

Even though there are 6 credits in this program designed for entrepreneurial activities, the 4-credit course (Introduction to Administration) is clearly focused on established companies processes. The only moment in which students in fact study and work with entrepreneurial concepts is on the Entrepreneurship course.

UniCesumar

The software engineering program at Centro Universitário de Maringá (UniCesumar)[8] was created in 2013 in an online format only. One of the program's objetive is to implement pedagogical practices that foster autonomy, creativity, critical thinking, entrepreneurial mindset, in order to give students the opportunity to think about developing new organizations (the same as the one of UnB and UTFPR).

The program offers in the last semester a 4-credit course called Entrepreneurship. This course focuses on explaining the importance of entrepreneurship by developing and creating a business plan for a company. It also touches on legal and ethic aspects related to running an organization. Finally, the course also entails planning, organization, coordination, leadership in regards to production, marketing, finance and human resources.

By analyzing the program website as well as its pedagogical project document, it looks like the focus (in regards to entrepreneurship) on established organization, and not on startups or new ventures.

UniPampa

The software engineering program at UniPampa (Universidade Federal do Pampa)[9], which holds a 4 for its government assessment, was created in 2009. One of the program's main objetives is to train students into applying their knowledge in an innovative and entrepreneurial way, contributing to the development of the society.

The program offers one 4-credit course called Administrations and Entrepreneurship, which focuses on applying management and administrative strategies into decision making processes. The course also touches on business model and on business plan. Interestingly, this program also offers a 2-credit course as elective, in which students learn

[8] https://www.unicesumar.edu.br/ead/cursos-graduacao/engenharia-de-software/
[9] http://cursos.unipampa.edu.br/cursos/engenhariadesoftware/?page_id=3

startup concepts and business model in the context of software engineering. This is was the only reference to the term "startup" in all 7 programs analyzed in this thesis.

2.3.3 Software Engineering Programs Samples Analysis

After analyzing each of the selected program, we mapped them out against the expected competencies and classifications depicted in Table 2.2. The result of this process is shown in Table 2.3. The labels presented in the table are depicted as follows:

- K: Knowledge (remembering a given content that was taught in class).
- C: Comprehension (understanding a given content that was taught in class).
- A: Application (using a given content in new and concrete situations).
- N: Not Application (in case there is no clear reference to that competence).

Expected Competencies	Expected Classification	PUCRS	UnB	UTFPR	UFG	UFMS	UniCesumar	UniPampa
Identify business opportunities	**C**	K	K	C	N	C	C	K
Create business models, transforming ideas into products or services	**A**	C	C	C	N	C	C	C
Plan innovative projects	**A**	C	C	C	N	C	C	C
Fundraising for innovative projects	**C**	N	N	N	N	N	N	N
Manage innovative businesses	**C**	C	C	C	N	C	C	C

Table 2.3 – Software engineering entrepreneurship track by program.

As it can be observed in Table 2.3, there are several gaps between the expected competencies suggested by the guidelines of the Brazilian Software Engineering Curriculum, and the actual ones explored in the 7 programs analyzed. The most important gap based on the collected data, which is presented in all programs, is the lack of application in regards to creating business models, transforming ideas into products or services, and in planning innovative projects. It can also be noticed that none of the programs analyzed touch on fundraising for innovative projects. Even though this topic is not as relevant as the others since we are working with first time entrepreneurs, it would be better if this topic was as least mentioned to students at some point.

It is important to mention that this analysis was done based on the documents provided by each university, as well as their websites. Moreover, it was performed only by the author of this thesis. Therefore, there is a threat to validity to this process regarding the research bias.

Finally, the guidelines of the Brazilian Software Engineering Curriculum, developed by the Brazilian Computer Association [AZN+17], provide great insights and guidance on how to build an effective software engineering curriculum. As already mentioned in this

section, this document mentions the importance of student-centered approaches, active-based methodologies, and experiential learnings strategies. These methods not only help engaging students, but they also touch on important soft skills that need to be developed. Nevertheless, this document does not address examples on how these strategies can be implemented in a classroom environment. On one hand, this is positive, since it leaves the floor open for universities to be creative and to adapt these guidelines according to their reality. On the other hand, if the university has no background or previous experience with a given topic, it might lead to an ineffective implementation of a given track.

3. RESEARCH METHODOLOGY

In this chapter, we discuss aspects related to the research methodology followed in this thesis. In Section 3.1 we explain the research methods used in this work. Section 3.2 depicts the research design.

3.1 Methodological Background

In this research, the following research methods were used: systematic mapping study, case study, survey, and expert panel. Each of these methods is explained as follows.

3.1.1 Systematic Mapping Study

According to Petersen *et al.* [PFMM08], "*a software engineering systematic map is a defined method to build a classification scheme and structure a software engineering field of interest*." Based on these scheme an analysis is drawn, which focuses on frequencies of publications for the designed categories. Thus, the context of the research field can be determined. In addition, other insights can be taken depending on defined research questions.

The first stages of a systematic mapping study are very similar to those of a systematic literature review [BTBK08]. However the research question tend to be much broader, in order to address the wider scope of this type of study. Theses stages are detailed as follows [BTBK08]:

1. The search stage: identification of primary studies that may contain relevant research results;

2. The inclusion/exclusion stage: selecting the appropriate primary studies from these after further examination;

3. The bias/validity: if appropriate, performing a quality assessment of the selected studies.

In the context of this research, a systematic mapping study was performed in the beginning of the process so we could understand the field in order to identify opportunities and gaps. This process addresses **RQ1**, **RQ2**, and **RQ3**.

3.1.2 Case Study

A case study is an empirical inquiry that investigates a contemporary phenomenon within its context, especially when the boundaries between phenomenon and context are not clearly evident [Yin03]. Its designs can be single-case or multiple-case studies, and they may entail a single unit or multiple units of analysis [Yin03].

In the context of software engineering, case studies are relevant for the evaluation of software engineering tools and methods, since they can avoid the problems that are usually associated with experiments [SDJ07]. In addition, case studies offer in-depth understanding of the context of a certain phenomena, and can reveal cause-effect relationships among events [ESSD08].

Data sources from case studies can be derived from a variety of ways. For instance, quantitative data involve classes and numbers. Qualitative data (from observations and interviews) can offer great insights due to the amount of information involved. When researchers are working on an initial investigation, the case study is called “exploratory”, whereas when they are testing existing theories the case is called “confirmatory” [ESSD08].

In this thesis, a case study was performed during our study on student’s perception on lean principles (addressing **RQ4**), as well as on the application of the Challenged Based Learning methodology in an entrepreneurship course focusing on teaching software startups (addressing **RQ2**).

3.1.3 Survey

As the name suggests, survey research is the process of conducting research using surveys that are sent to respondents by the researchers [Bab90]. The data collected from this process is then analyzed in order to draw conclusions. Surveys are appropriate when the focus of the research is on the “what”, “how”, or “why” of a given situation. In addition this method also applies when it is not possible to control independent and dependent variables.

Data collection is usually done by one of these methods: phone, email, online form or face-to-face. Each of these approaches give researches advantages as well as disadvantages. For instance, a face-to-face survey takes more time and it is harder to perform, but give researchers more thorough data. On the other hand, online forms are easier to distribute, but usually do not retrieve in-depth data.

Surveys are relatively inexpensive; a lot can be done online and almost for free. As already mentioned, surveys are flexible since they can be done in a variety of ways. Finally, since it is possible to run an anonymous survey, respondents can be more candid, bringing valuable information to researchers.

In this research, a survey was performed in our study on student's perception on lean principles (addressing **RQ4**), as well as on our software startup education study (addressing **RQ3**).

3.1.4 Expert Panel

An expert panel is an exploratory study that aims at the strengths and weaknesses of given method, process, model, practice, or technique [BHB+05]. It entails a group of experts recognized in at least one of the fields addressed by the proposed research. Data from this process helps to construct models or frameworks and to validate them [SC01].

The idea of using expert to evaluate a given process is very valuable, since experts can help prevent requirement defects [BHB+05]. Moreover, due to their expertise in the field, they can make wise and useful recommendations on why a given process is good or bad. Even though experts might share different opinions on a given subject, conclusions and recommendations are usually drawn by consensus.

The value of expert knowledge is recognized by the capture expert judgment, the ability of experts to predict techniques to prevent requirement defects and in their analysis of the accuracy of several methods of estimating project effort [BHB+05]. It reaches conclusions and recommendations through consensus [Slo03].

In the context of this thesis, the expert panel was used to evaluate the initial proposed framework for teaching software startup concepts to software engineering students.

3.2 Research Design

In this section, we depict the research design used in this thesis. Figure 3.1 presents the selected methods and the timeline for the execution of this study. The process was inspired by a methodology proposed by Shull *et al.* [SCT01], and further extended by Mafra *et al.* [MBT06]. In order to answers the main research questions of this work - *How can we teach software startup related concepts to software engineering students?* - we proposed a constructive approach. This process allows us to evaluate the effectiveness of a given approach under realistic conditions by identifying strengths, as well as improvement opportunities [SCT01].

Each step of the proposed methodology is organized as follows:

- **Phase 1 - Exploratory**: The first step consisted of executing four studies. The goal of *Phase 1* was to deepen our understanding on software startup related concepts, as well as on how these concepts are taught to students. By identifying best practices

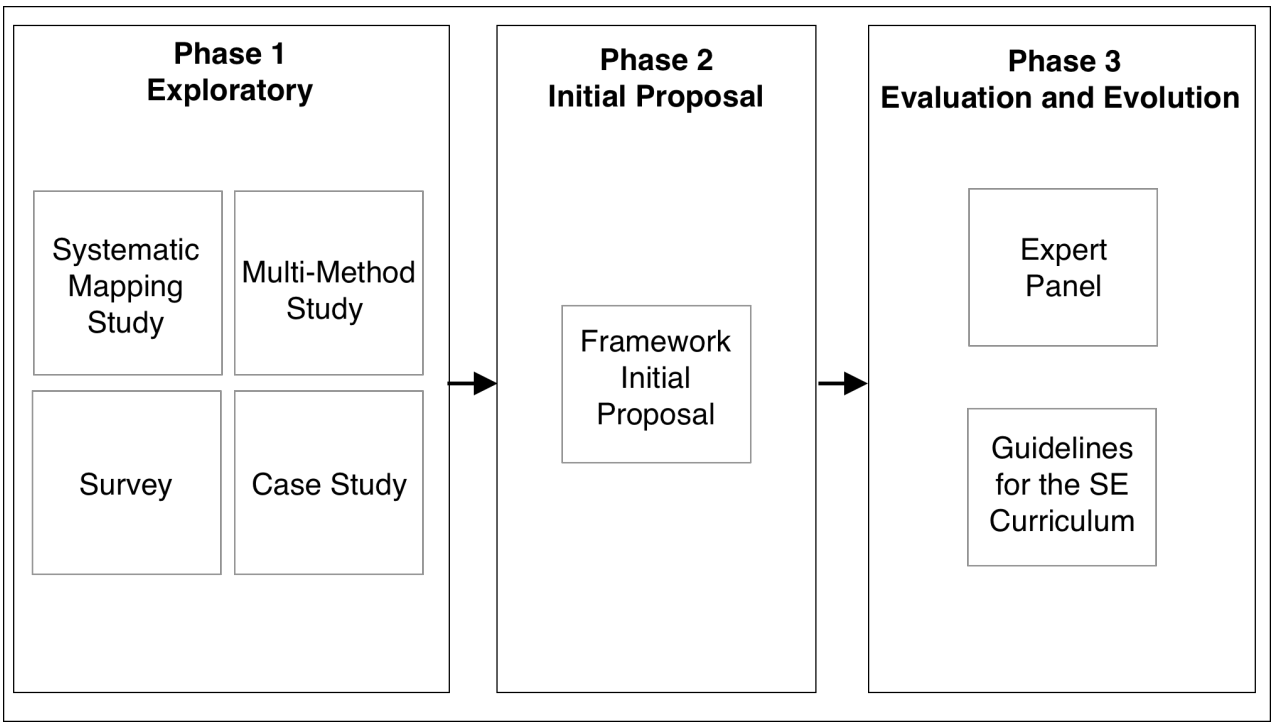

Figure 3.1 – Research methodology.

and improvements opportunities, it would be possible to develop a framework that can help faculty deliver software startup concepts to students in a consistent manner. In addition, we studied teaching methodologies in order to identify which one(s) is(are) best suited for teaching software startups.

During this phase, a systematic mapping review on software startup education was performed in order to identify how this topic is being reported in the literature. In order to do so, we will follow the guidelines proposed by Petersen *et al.* [PFMM08] and by Budgen *et al.* [BTBK08].

In addition, we ran three other experimental studies. The first one was a multi-method study (two case-studies and a survey) aimed at understanding how students perceive software startup concepts. The goal was to identify whether there was a gap between theoretical and practical understanding of such topics.

The second experimental study was a survey on faculty that teaches software startup to software engineering students. The goal was to collect best practices as well as lessons learned.

Finally, the third experimental study was a case study on the application of the Challenged Based Learning methodology in an entrepreneurship course focusing on teaching software startups. The goal was to understand how the methodology helps students to grasp software startup concepts, as well as to retrieve aspects that could be improved in order to better deliver value in a classroom environment.

These studies are presented and detailed in Chapters 4, 5, 6, and 7 respectively. Once we found enough evidence in the literature and on the experimental studies described above, an initial proposal of the framework was developed. The goal of this initial proposal was to have a starting point for further discussions and analysis.

- **Phase 2 - Initial Proposal**: Based on the findings discovered in the previous phase, an initial proposal named *Challenge Based Startup Learning* was developed in order to be further evaluated. This initial proposal is detailed in Chapter 8;
- **Phase 3 - Evaluation and Evolution**: The third phase of this research included the evaluation and evolution of the initial proposal. In order to do so, we conducted an expert panel. This approach is a recognized way to perform an initial evaluation of a proposal [SC01]. The reason for that is that an expert panel allows interviewers to analyze the responses from interviewees in a less rigid way. In this sense, our goal was to gather insights from experts regarding the framework, and also to look for improvements opportunities. This study is presented in Chapter 9. At the end of this process, we proposed a set of recommendations for software engineering programs in Brazil to teach software startup concepts by using the Challenge Based Startup Learning framework.

4. SYSTEMATIC MAPPING STUDY

The goal of this study was to identify the main academic contributions on software engineering education in the software startup context. In order to do so, we perform a systematic mapping study [BTBK08, PFMM08] aimed at:

- understanding the state-of-the-art research on software startup development education for software engineering students;
- collecting best practices and methodologies used on software startup education for software engineering students;
- identifying gaps for future studies.

From an initial set of 268 papers, we have identified 39 primary studies worth analyzing from 5 distinct scientific databases.

4.1 Systematic Mapping Review Research Methodology

The goal of this systematic mapping study is to determine and characterize the state-of-the-art on software startup development education. In order to do so, we carried out this systematic mapping following the recommendation of the most influential researchers in this area [BTBK08, KBB11, PFMM08]. Figure 4.1, which was adapted from Petersen *et al.* [PFMM08] presents the process undertaken to perform the systematic mapping study. The remainder of this section depicts the planning of each step of this study.

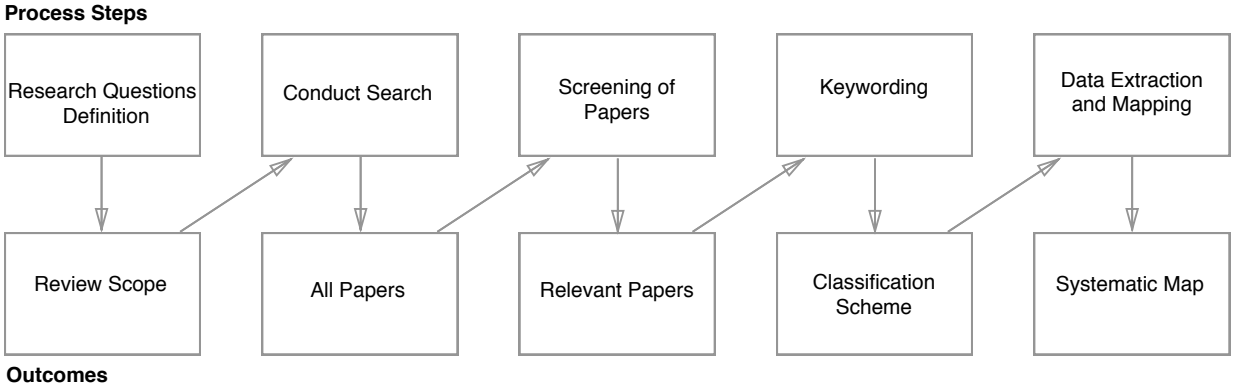

Figure 4.1 – Systematic mapping process.

Research questions definition

This study addresses the following questions of this thesis:

- **RQ1**: *What is the state-of-art in literature in regards to software startup education for software engineering students?*
- **RQ2**: *Which tools, models, methodologies and frameworks are applied in a software startup education context for software engineering students?*
- **RQ3**: *What are the reported best practices in regards to teaching software startup for software engineering students?*

RQ1 is a general research question, that will be explored by running the systematic mapping study. The purpose of RQ2 is to understand if there are tools, models, methodologies and frameworks that are somehow new to the technology education context and that have been used specifically to teach software startups processes. With RQ3 we intend to analyze which teaching strategies and best practices are put in place. Since software startups have the goal to solve real-world problems [Bla13], there is a possibility that non-traditional teaching approaches could be used.

4.1.1 Data sources and search strategy

With regard to identifying the primary studies for this systematic mapping study, we designed a search string following the guidelines proposed by Kitchenham and Charters [KC07]. The search string is composed by the *population*, *intervention* and *outcomes* expected. We omitted the *comparison* and the *context* structures since we were focusing on a more exploratory research. The final search string used in this study is presented in Table 4.1.

Population	(Software Engineering OR Software Development)
	AND
Intervention	(Software Startup OR Startup OR Entrepreneurship)
	AND
Outcome	(Education OR Undergraduate OR Graduate OR
	Teaching OR Educating OR Training)

Table 4.1 – Search string.

The search strategy is summarized in Table 4.2. The databases sources were chosen based on the list proposed by Kitchenham and Charters [KC07]. Two databases (Citeseer library and Inspec) were left out of this research due to difficulties in using these platforms; results were not matching the string search criteria at all. In addition, Google

Scholar was also used, but only to double check the results. In regards to the publication period, we decided to begin in 1998 since this is the time in which the concept of software startup, as defined by Ries [Rie11], started to be formed and studied.

Databases searched	ACM Digital Library
	IEEExplore
	Scopus
	EI Compendex
	Science@Direct
Selection Criteria	available online
	written in English
	from 1998 to May 2018
	in: Journals/Conferences/Workshops/Symposiums
	4 pages minimum
Search applied to	Title
	Abstract
	Keywords

Table 4.2 – Search strategy.

Moreover, we only included papers that were accepted in journals, conferences, workshops and symposiums. Keynote presentations, extended abstracts and papers with less than 4 pages were also excluded from the research criteria since they usually do not present in-depth analysis. The focus was to identify papers that could present at least preliminary studies on the research topic. After following all these steps, we came across 268 papers. Table 4.3 presents the number of publication retrieved from each database.

Database	**Papers**
ACM Digital Library (http://dl.acm.org/)	58
IEEExplore (http://ieeexplore.ieee.org/)	56
Scopus (https://www.scopus.com/)	67
EI Compendex (https://www.engineeringvillage.com/)	85
Science@Direct (http://www.sciencedirect.com/)	2
Total	268

Table 4.3 – Retrieved papers.

4.1.2 Screening of Papers

Once papers were retrieved from the databases, a spreadsheet was created in order to organize the information for the screening process. Table 4.4 summarizes the basic information collected from each of the 268 studies initially selected.

Information retrieved	**Explanation**
Database	ACM, Science@Direct, Scopus, IEEExplore, EI Compendex
Title	Paper title
Year	Year published
Authors	List of all authors
Type of forum	Journal, conference, workshop, symposium
Abstract	Paper abstract
Keywords	Paper keywords
Status 1	Duplicate
Status 2	Do not fit into the criteria
Status 3	Is relevant

Table 4.4 – Spreadsheet basic information.

The screening process started by excluding duplicates, which accounted for 76 items, leaving the spreadsheet with 192 papers. This was done by assigning the value “0” (zero) on the *Status 1* column in the corresponding row of the duplicates. After that, we followed the exclusion criteria. This process eliminated 77 items, leaving the spreadsheet with 115 papers. This was done by assigning the value “0” (zero) on the *Status 2* column in the corresponding row of those studies that did not fit into the criteria. Finally, in the last step of the screening process, we read the title, abstract and keywords in order to verify if the paper is relevant in regards to our research goal. This was done by assigning the value “0” (zero) on the *Status 3* column in the corresponding row of those papers we understand were not relevant to answer the research goal. At the end, our screening process led to 39 primary studies to be fully analyzed.

4.1.3 Keywording

According to Petersen *et al.* [PFMM08], “*keywording is a way to reduce the time needed in developing the classification scheme and ensuring that the scheme takes the*

existing studies into account". This process is illustrated in Figure 4.2. This figure was adapted from Petersen *et al.* [PFMM08]. It starts by reading abstracts of the primary studies in order to look for keywords that identifies the main contribution area of the paper. The goal is to create a set of categories in which papers can be combined. If meaningful keywords cannot be found by reading the abstracts, researches may look for them in the introduction and conclusion sections of the papers.

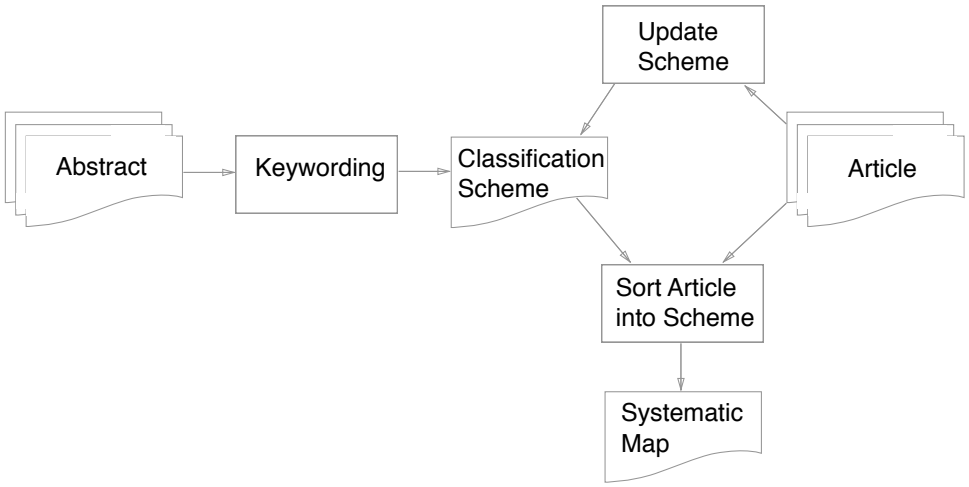

Figure 4.2 – Classification scheme workflow.

It is important to point out that the classification scheme can evolve and change during the systematic mapping process. As researches read the papers thoroughly, new categories/classifications may appear and others might merge or disappear.

4.1.4 Data extraction and mapping

The data extraction and mapping was performed by using two tools: a spreadsheet, and the software *Mendeley*[1]. *Mendeley* is an application that helps users manage papers for research purpose.

In addition to the information mentioned in Table 4.4, we added the following information to the spreadsheet:

- Focus Facet: the categories created during the classification scheme process;
- Contribution Facet: type of contribution. Based on a work from Shaw [Sha03], and from Paternoster *et al.* [PGU+14];
- Research Method: method used on the research (case study, survey, *etc.*);
- Research Type: type of research (adapted from Wieringa *et al.* [WMMR06]);

[1] https://www.mendeley.com

- Paper Rank: a grade from 0 to 10, based on the work from Salleh *et al.* [SMG11]. Details on Table 4.5;
- Contribution: important take aways from the study.

4.1.5 Threats to Validity

There are several threats in this process that could invalidate the systematic mapping. If the search strategy is not done correctly, the retrieved papers may not account for all studies that could answers the research questions. In addition, data extraction and paper classification is also an important step that should be done carefully. Finally, it is important to pay attention to researches' bias. In order to mitigate these threats, we followed the recommendations from Petersen *et al.* [PFMM08] during the whole systematic mapping process.

Aside from the systematic mapping process itself, we understand there are two important threats that requires attention. The first is related to publication bias. We are more likely to find papers reporting positive experiments regarding software startup education, rather than failure ones. It is very difficult to mitigate this risk since we only have access to published data, naturally. The second threat, and most important one, has to do with the interdisciplinary aspect of software startups. Since this topic overlaps with entrepreneurial education, this work could be missing relevant sources from the business area. Even though we are aware of this issue, we are not considering studies from other research areas.

4.1.6 Classification Scheme

In this section we detail the classification scheme proposed for this systematic mapping. We have defined five categories: focus, contribution, research method, research type, and paper rank.

The paper rank facet, based on the work of Salleh *et al.* [SMG11], is divided into 8 questions. The two questions directly related to RQ1 and RQ2 are worth 2 points. All other questions are worth 1 point. If a given question is fully answered by reading a study, it gets the total points for that question; if it is partially answered, it gets half of the points; if it is inconclusive or if it does not answer the question, it gets zero points. Once a study is graded, it falls into one of these three categories:

- High rank: 8 to 10 points;
- Medium rank: between 5 and 8 points;
- Low rank: 0 to 5 points.

Table 4.5, inspired by Paternoster *et al.* [PGU+14], presents the whole classification scheme for this study.

Category	Description
Research Method Facet	
Case Study	A report about an specific situation that has been studied [KPP95].
Empirical Study	Study based on empirical evidence [PPV00].
Experimental Study	A study in which an intervention is introduced to observe its effects [SHH+05].
Survey	A process to collect data, analyze the information and report the results [PK01].
Research Type Facet	
Evaluation Research	Evaluation of a technique that was put into practice.
Experience Paper	Personal experience of the author depicting how something has been done in practice.
Opinion Paper	Personal opinion on whether a certain technique is good or bad.
Philosophical Paper	New way of looking at an existing context.
Solution Proposal	The proposition of a solution to a problem.
Validation Research	New techniques being implemented in experiments, simulations or in practice.
Contribution Facet	
Advice/Implication	Recommendations based on personal opinions.
Framework/Method	Framework/Method used to teach (or to learn) software engineering in a software startup context.
Guidelines	Advices based on the research results.
Lessons Learned	Actionable advices derived from the obtained research results.
Model	Representation of a given context based on a conceptualized process.
Tool	Tools used to teach (or to learn) software engineering in a software startup context.
Focus Facet	
Teaching	The focus is on classroom education strategies.
Real Projects	Startup software development education is based on real-world project execution.
Multidiscipline	Multidisciplinary activities are mentioned as key to startup software development education.
Environment	The paper focuses on the environmental issues that affect startup software development education.
Paper Rank Facet	
References	Is the study well referred? (1 point)
Paper Goal	Is the goal clearly stated? (1 point)
Sample Observation	Data collection and sample strategy was carried out correctly? (1 point)
Method	The analysis methodology was well applied? (1 point)
Clear Description	Is the context of the study clearly described? (1 point)
Findings	Are findings credible? (1 point)
RQ2	Does the paper answer RQ2? (2 points)
RQ3	Does the paper answer RQ3? (2 points)

Table 4.5 – Classification scheme.

4.2 Results of the Systematic Mapping

The search process was carried out by following the process described in Figure 4.1. From an initial set of 268 papers identified through the search strategy, we have come across 39 primary studies. The systematic map overview is presented in Table 4.6.

1st Author (year)	Research method	Research type	Contribution	Focus	Paper Rank
Fagerholm (2017) [FHL+17]	Experiment	Experience Paper	Lessons Learned	Teaching	9.5
Génova (2016) [GG17]	Empirical Study	Philosophical Paper	Model	Teaching	9.5
Järvi (2015) [JTH15]	Case Study	Evaluation Research	Lessons Learned	Teaching	9.5
Schilling (2010) [SK10]	Case Study	Evaluation Research	Guidelines	Teaching	9.5
Adorjan (2017) [AM17]	Survey	Evaluation Research	Lessons Learned	Teaching	9.0
Buffardi (2017) [BRR17a]	Experiment	Evaluation Research	Guidelines	Multidiscipline	9.0
Izurieta (2016) [ITOGI16]	Case Study	Experience Paper	Framework/Method	Real Projects	9.0
Zaina (2015) [ZA15]	Case Study	Evaluation Research	Framework/Method	Teaching	9.0
Chesney (2014) [Che14]	Case Study	Evaluation Research	Guidelines	Real Projects	9.0
Currie (2011) [CDK11]	Case Study	Experience Paper	Framework/Method	Real Projects	9.0
Salas (2017) [Sal17]	Survey	Experience Paper	Lessons Learned	Teaching	8.5
de Lange (2016) [dLNKK16]	Case Study	Validation Research	Framework/Method	Environment	8.5
Zhang (2015) [Zha15]	Experiment	Solution Proposal	Model	Teaching	8.5
Daimi (2008) [DR08]	Empirical Study	Solution Proposal	Guidelines	Teaching	8.5
Joseph (2006) [Jos06]	Case Study	Evaluation Research	Lessons Learned	Multidiscipline	8.5
Boutell (2017) [BF17]	Survey	Evaluation Research	Lessons Learned	Real Projects	8.0
Devadiga (2017) [Dev17]	Survey	Experience Paper	Guidelines	Teaching	8.0
Ribeiro (2016) [RAF16]	Case Study	Experience Paper	Lessons Learned	Environment	8.0
Vitolo (2016) [VHB16]	Experiment	Evaluation Research	Lessons Learned	Multidiscipline	8.0
McMahon (2014) [McM14]	Case Study	Experience Paper	Lessons Learned	Teaching	8.0
Kaltenecker (2013) [KHH13]	Empirical Study	Philosophical Paper	Lessons Learned	Teaching	8.0
Chenoweth (2008) [Che08]	Survey	Experience Paper	Guidelines	Teaching	8.0
Buffardi (2017) [BRR17b]	Case Study	Solution Proposal	Framework/Method	Multidiscipline	7.5
Porter (2015) [PML+15]	Case Study	Solution Proposal	Lessons Learned	Multidiscipline	7.5
Nguyen-Duc (2016) [NDSA16]	Case Study	Evaluation Research	Model	Teaching	7.5
Bharadwaj (2014) [Bha14]	Experiment	Experience Paper	Framework/Method	Teaching	7.5
Breytenbach (2013) [BdVH13]	Case Study	Experience Paper	Lessons Learned	Environment	7.5
Ko (2017) [Ko17]	Empirical Study	Experience Paper	Advice/Implication	Environment	6.5
Heintz(2014) [HK14]	Case Study	Experience Paper	Lessons Learned	Teaching	6.5
Pauca (2012) [PG12]	Case Study	Solution Proposal	Framework/Method	Real Projects	6.5
Ford (2004) [FGW04]	Case Study	Experience Paper	Lessons Learned	Multidiscipline	6.5
Barbe (2010) [Bar10]	Empirical Study	Experience Paper	Model	Environment	6.5
Gross (2000) [Gro00]	Survey	Evaluation Research	Lessons Learned	Teaching	6.5
Q.-Sarmiento (2018) [QSEMD+18]	Case Study	Solution Proposal	Model	Teaching	6.0
Sun (2009) [SXT+09]	Empirical Study	Philosophical Paper	Model	Environment	5.0
Sarraipa (2016) [SFMJ+16]	Case Study	Solution Proposal	Advice/Implication	Environment	4.5
Engelsma (2014) [Eng14]	Case Study	Experience Paper	Lessons Learned	Real Projects	4.5
Rioja Del Rio (2014) [RMEDJ14]	Empirical Study	Opinion Paper	Tool	Teaching	4.5
Pauli (2008) [PLB08]	Case Study	Experience Paper	Lessons Learned	Real Projects	2.0

Table 4.6 – Systematic map overview.

Figure 4.3 presents the distribution of the primary studies according to the year they were published. It is clear that the majority of the studies are recent. More than half of the selected studies are from 2014 to May 2018. This result is somehow interesting, since studies on software startup started almost 10 years earlier [PGU+14]. It could be a sign that the interest for software startup education research has began not long ago.

In addition, the majority of the primary studies selected were published in conferences (see Figure 4.4). Even though the quality of a paper is not directly related to the type of forum it is published, the scientific community understands that studies published in journals tend to be more valuable than those in other forums. Only 5 out of the 39 studies in this systematic mapping review were published in journals. By combining this information with

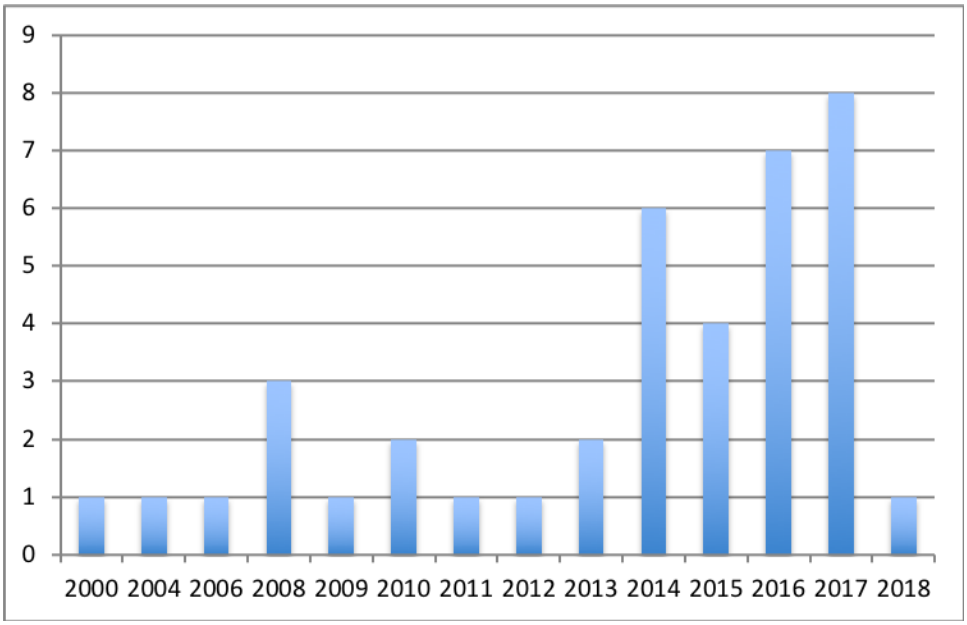

Figure 4.3 – Distribution of primary studies by year.

the data from Figure 4.3, we can see that, in fact, software startup education just started to be explored by the scientific community.

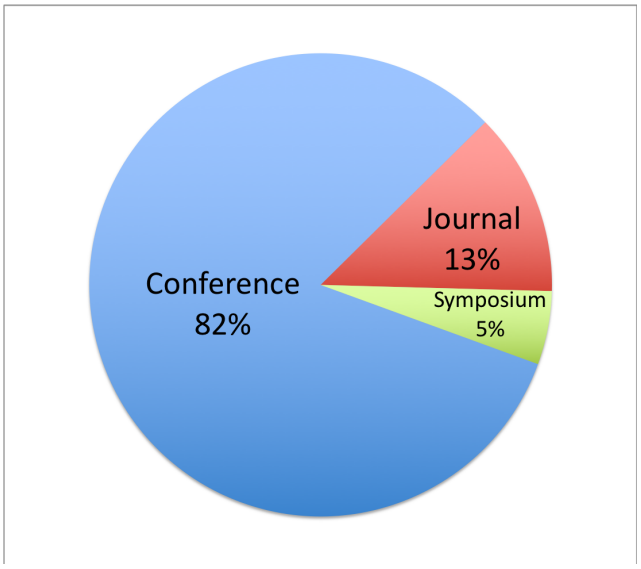

Figure 4.4 – Distribution of primary studies by type of forum.

Finally, we followed the guidelines from Petersen *et al.* [PFMM08] and used bubble plots to combine and compare the different facets that were defined in the classification scheme. In these plots, the axis correspond to categories taken from the scheme. The size of the bubble represents the number of primary studies in a given intersection. By doing so, it becomes easier to visualize and to analyze the data.

Figure 4.5 presents the first of the three bubble plots designed for this study. It combines the paper rank facet, with the research type facet and the focus facet. To begin with, most studies were considered as medium and as high rank. In regards to the research

type facet, there is only one validation research paper. This could be an indication of a gap in the software startup education context. The focus facet data indicates that most studies are related to teaching related content. This fact does not come as a surprise; on the contrary, it reveals that the results are aligned with the research questions proposed.

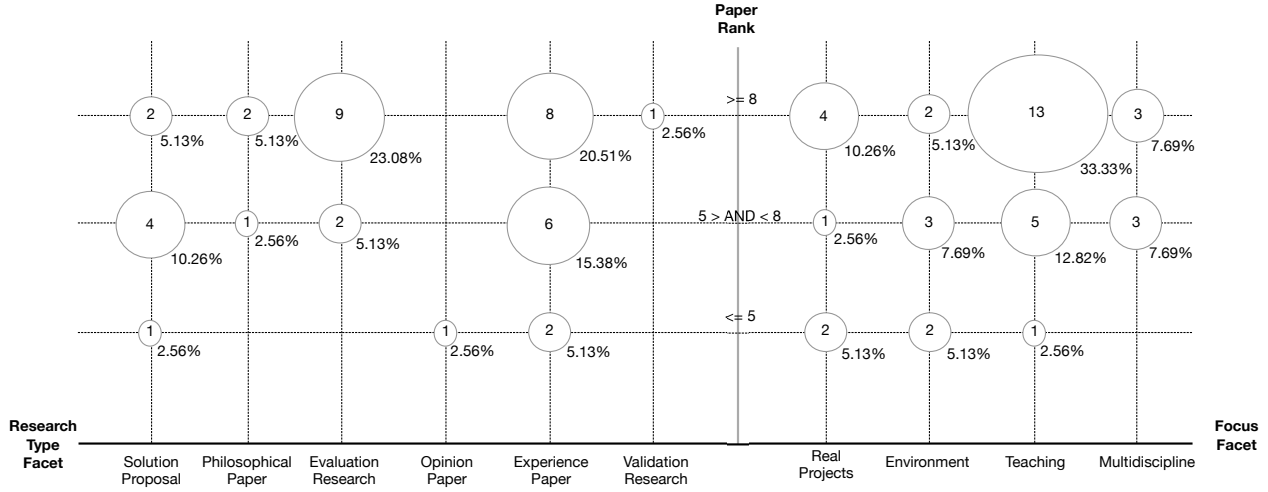

Figure 4.5 – Systematic map by paper rank, research type and focus.

In Figure 4.6 we show the bubble plot that combines the focus facet, with the research type facet and the contribution facet. Regarding the contribution facet, most studies derive lessons learned across all focus categories. On the research type facet, we see a similar behavior when it comes to experience papers; there is at least one paper for each focus facet that was categorized as experience paper.

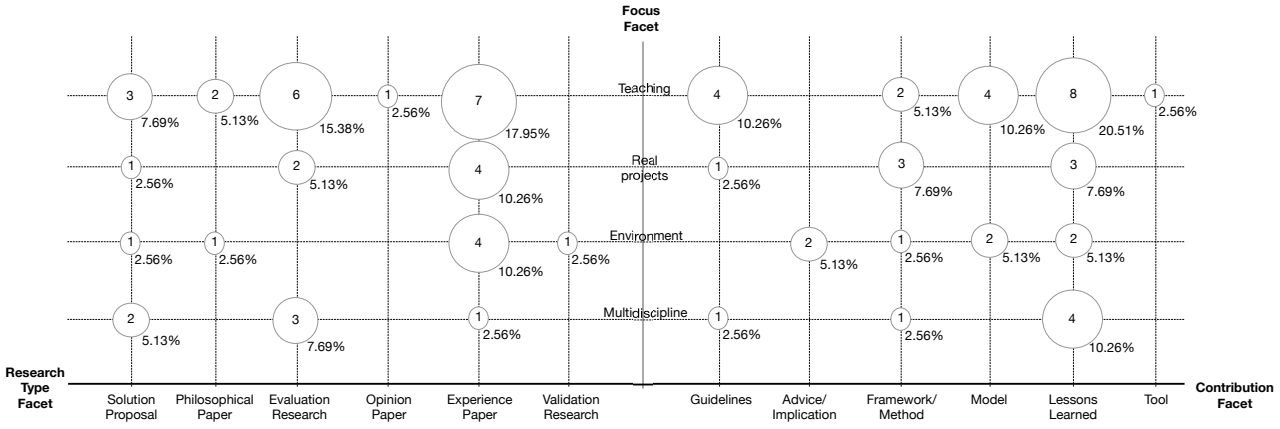

Figure 4.6 – Systematic map by focus, research type and contribution.

In our last bubble plot (Figure 4.7) we combined the contribution facet, with the research method facet and the paper rank facet. The majority of the selected studies (54%) are case studies. Nearly half of the case studies delivers lessons learned. It appears to have an interesting opportunity to explore different research methods in order to evaluate the research questions proposed in this paper. In addition, there is a clear gap on tools and on advice/implication when it comes to the contribution facet.

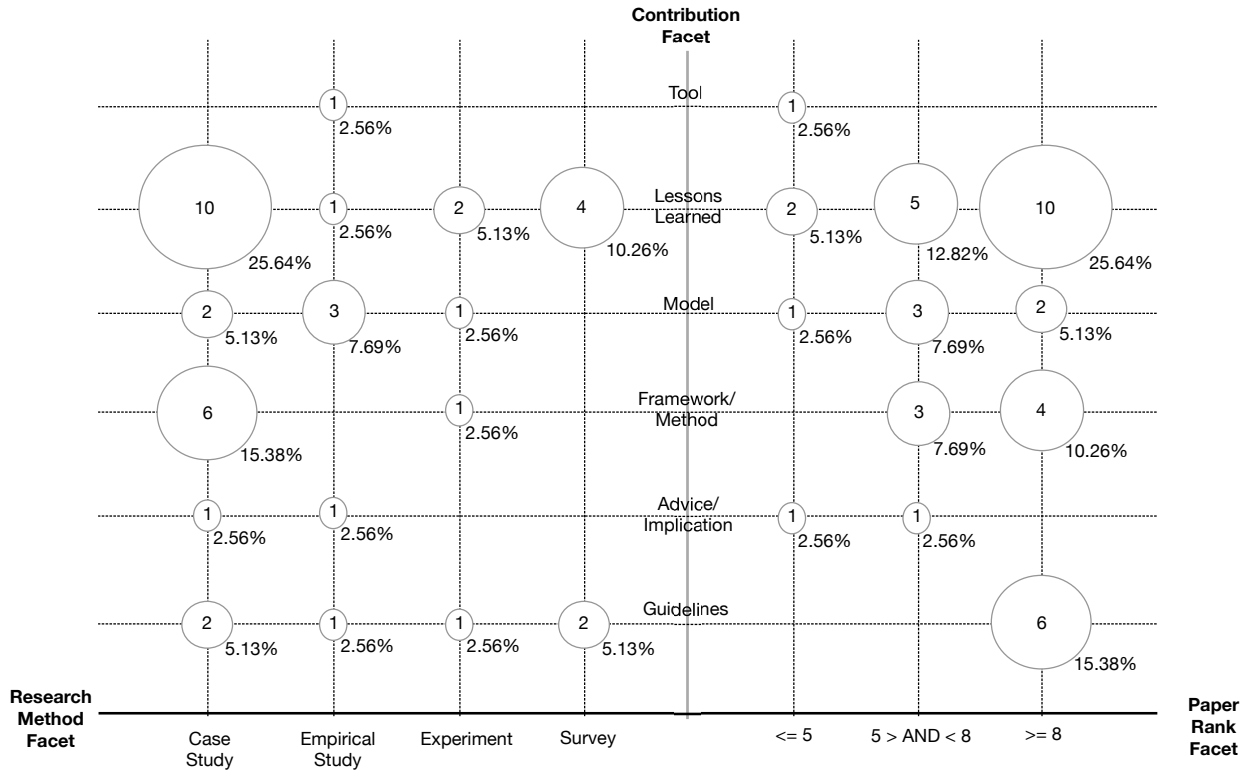

Figure 4.7 – Systematic map by contribution, research method and paper rank.

4.3 Research Questions Analysis

In this section we analyze the two research questions proposed for this study (RQ2 and RQ3). For each question we combined the information developed in Section 4.2 with the insights and learnings from each of the 39 primary study selected.

4.3.1 RQ2. Which tools, models, methodologies and frameworks are applied in a software startup education context for software engineering students?

The overview of this systematic mapping study detailed in Table 4.6, indicates that there is one study [RMEDJ14] contributing with tools, six studies [Bar10, GG17, NDSA16, QSEMD+18, SXT+09, Zha15] focusing on models, and seven studies [Bha14, BRR17b, CDK11, dLNKK16, ITOGI16, PG12, ZA15] exploring methods and methodologies. By combining and summarizing this information, the main contributions to the field are:

- Business Model Canvas: helps students define a vision for their business model. It is specially useful when dealing with technology students, since the canvas goes beyond the product and also focus on the market;

- Customer Development Process: proposed by Blank and Dorf [BD12], this model helps students take actionable steps in order to validate business hypothesis;
- Design Thinking: very useful during the ideation phase, but it is also used further in the process when creative solutions need to be developed;
- Agile: when students start coding, agile is the preferred software development approach. This is no surprise since the software development process should be flexible due to the characteristics of a startup.

In regards to tools, Rioja Del Rio *et al.* [RMEDJ14] suggests the use of the Business Model Canvas (BMC) [OP10] combined with software startup projects developed in the classroom. The argument is that the BMC gives students the opportunity to analyze all aspects of a business model, and not only the software itself. The BMC entails the value proposition of the business, the customer segment, the channels to reach customers, the relationships established with customers, key resources, key activities, key partners, revenue streams and cost structure.

The studies that presented models that are applied in software startup education revealed interesting insights. Génova and González [GG17] claim that there are three stages in a complete engineering education process: instruction (traditional education environment, with exams and projects), training (when students receive a problem and choose the mean to solve it) and mentoring (when students are able to self-propose their own objectives). The authors postulate that "*education is incomplete if the third stage is not reached*". From an education institution perspective, there are several challenges to achieve the third stage. For instance, if students self-propose their goals and objective, how can it be evaluated fairly? Furthermore, the authors question if it is possible to actually teach creativity and self-determination. In sum, this study presents a gap in the software engineering educational process and it clearly states that if software engineering schools do not offer opportunities for students to achieve the third stage, they will not become real engineers; they will be producing "programmed machines". From a startup education perspective, achieving the third stage is crucial. Otherwise educational institution will be mostly graduating workers rather than entrepreneurs.

Zhang [Zha15] proposes a model that combines technology, business and environment. The argument is that technical and business knowledge forms the foundation for the software startup learning process. However, the environment plays a key role in this process. There are several resources that students usually are unaware of, such as networking events, mentoring, funding resources and incubators. By putting all these components together, students have the opportunity to experience the creation and the development of a startup within a meaningful context. However, this is not easy to achieve in practice and some tradeoffs need to be observed. One instructor alone usually cannot deliver all the content. There is a need of at least two instructors (one for the business and the other for the

technical lectures). This means that two different schools (the business school and the IT school) should coordinate activities and efforts. In regards to content, faculty should be careful not to deliver materials that would be uninteresting for a groups of students, *i.e.*, going deep on coding in a classroom full of business students. Going further, faculty should implement strategies to manage conflict that may arise due to students' different backgrounds. Another important factor that should be taken into account is whether the course is full-time or part-time. Teaching students that work during the day and study at night is completely different than teaching students that are 100% focused on their studies.

The model proposed by Barbe [Bar10] intends to connect all aspects of a startup process creation and development. It goes from the basic technical knowledge all the way to business acceleration and funding. The reasoning for creating this model was that businesses can be created by technical founders. However, if they lack business and soft skills, they will either fail or will need to hire people with these skills. Therefore, in this context, students not only learn the technical foundations for developing a software startup, but they are also exposed to the whole startup development process.

Zaina and Álvaro [ZA15] propose a methodology that combines user-centered design [RC08] and Lean Startup [Rie11] in order to foster innovation and entrepreneurial behavior in a software engineering course. The authors argue that computer related courses usually focus only on technical issues, and it does not encourage students to be creative, innovative, and to open businesses. Two case studies were conducted in order to verify the effectiveness of the methodology. Results indicate that students not only learned important business concepts, but also perceived the importance of understanding real customers' needs.

Buffardi *et al.* [BRR17b] argue that it is very hard to emulate real world projects in an academic setting. When students work with "toy" projects, they might learn technical content, but they will not experience real customer pressure, competition, and other "real life" situations. The study also presents an interesting information (from Nurkkala and Brandle [NB11]) regarding the gaps between software engineering students' and industrial software engineers' experiences. They are sixfold: (i) real product versus a project; (ii) long duration versus short duration; (iii) low turnover versus high turnover; (iv) high complexity versus low complexity; (v) needs maintenance versus no maintenance; and (vi) real customers versus no customers.

Therefore, a methodology was proposed in order to minimize these gaps. The idea was to promote collaboration between software engineering and entrepreneurship students (who would act as customers). Even though software engineering students reported that the experience was relevant to them, the whole process just mimics a real project context. It is not ideal, but it gives students a good perception about what it takes to develop a real startup. In this kind of situations, instructors need to evaluate the trade-offs. Depending on the characteristics of the course, it may be too difficult to address real projects.

In Pauca and Guy [PG12], the difference between working in a academic setting versus developing a real project is also highlighted. Software projects developed in classroom are usually trivial, uninteresting or not meaningful to students, according to the authors. When students work with real challenges and problems that can affect society, the engagement goes up significantly. This concept was labeled *socially relevant computing* [BNS08]. Hence, the methodology consists of combining mobile application development, agile methods for project management [SB01], and social relevant projects. Results from a case study indicate that this approach can foster the creation of software startups since several students continued to work on their projects as a business after the end of the course.

Discussion. In sum, there is no single approach to address software startup education. Several strategies have being used in order to teach software startup. Some of them are focused on encouraging creativity, big-picture thinking, and critical thinking, while others focus on method, attention to detail, and in-depth analysis. Since courses have a limited amount of time, faculty need to evaluate the trade-offs associated with each approach.

4.3.2 RQ3. What are the reported best practices in regards to teaching software startup for software engineering students?

We have extracted several practices and lessons learned from the 39 primary studies. In the remainder of this section we discuss them according to the focus facet of the classification schema.

Teaching. From a teaching perspective, several insights and best practices were found. Several authors argue that software startup courses should not have explicit learning goals nor exams. The learning happens as students go through the process (for instance, talking to customers, working in teams, or building an MVP). In this sense, a flipped classroom approach is ideal. Traditional lectures should be used only to deliver basic concepts. Thus, the journey is more important than the endpoint; the goal of the course should be the experience of the software startup development process, and not just a single deliverable at the end.

Students are generally evaluated by writing personal and team reports as well as by presenting the progress of their projects. Therefore it is important to document every step of the process, from ideation to the final deliverable/presentation. If the class is taught by multiple instructors, it is important to establish consistency regarding grading. Prior to the beginning of the course, faculty needs to agree upon assessment instruments and rules.

Regarding teams, four or five members is ideal according to our research. Working with more than five people requires a lot of coordination, whereas having less than four members could result in a poor team composition. Teams should always choose a leader,

who will act as a “team liaison”. Instructors should set up a time to meet with each team individually on a regular basis. The goal is to receive general advice as well as to verify if the method is being followed. It is important to point out that this is not a rule; depending on the project configuration and on the maturity of the team, it is definitely possible to work with more than five people.

Moreover, teams should present their progress to the whole class several times during the semester. By doing so, they can receive and give feedback to each other. Additionally, this approach helps students developing their oral communication skills.

When it come to software development tools and processes, some authors suggest that students should use the same project portals, repositories and software development language. From the faculty point of view, it is very hard to help teams if each of them is using different technologies. Another advantage is that students may help each other if technology is the same across projects. The problem with this approach is that not always a single programming language fits well into all projects; in the real world, people choose the programming language that makes more sense to the context of their application.

Finally, product development is different from innovation development. The former is straight forward, while the latter is chaotic. In order to add structure to the innovation process, tools, such as the *Business Model Canvas* and methodologies, such as *Customer Development* and *Design Thinking* must be introduced. For the project management process, agile was the most recommended approach.

Real Projects. Customer and problem should never be given students. They need to explore these issues through interviews or other research methods. Otherwise learning is limited to software engineering, project management and teamwork. It is possible, however, to connect students to the industry to look for problems worth solving. This approach not only gives students the opportunity to connect with corporate executives and managers, but it also helps in finding real problems. Anyway, it is always important to leave the floor open for students to define their own projects; students should only pick an industry problem if it is exciting enough for the team to work on.

Multidiscipline. Opportunities should aim at cross-discipline collaboration. Software engineering/computer science courses and business courses should be combined and taught together in the same classroom by two or more instructors. Even though this approach requires coordination among faculty since class planning and execution is time-consuming, it is a great opportunity to mix up students with different skills in order to work in multidisciplinary projects. In this situations, faculty should not allow teams with only one skill set. Teams must have software engineering as well as business skills across members. There is a lot of learning when students with different backgrounds share experiences among themselves. Additionally, faculty must be aware that managing students from different backgrounds require patience and ability to solve conflicts. It is recommended to set the ground

rules at the beginning of the course. Moreover, teams should also develop self-governance guidelines that address how they will make decisions and resolve conflicts.

Environment. When possible, faculty should create opportunities for external validations. If students fail in finding customers, faculty should look for partners (such as corporate executives, or startup founders) to give feedback to students. It is not ideal, but at least student have the opportunity to discuss their projects with an experienced person. Usually these partners can be found within the university ecosystem (such as in a technology park), and they can also serve as mentors and advisors.

Discussion. It is very difficult to provide a realistic setting for students in the context of software startup development. It often comes at the expense of practices, processes, and goals. Even when connections with real world problems and people are made, in several cases students do not continue working on the projects once the course is over. However, successful cases were reported. Some projects actually end up being embraced by university incubators. However, this only happened when there was a formal and close connection between the course and the ecosystem.

Another important take away is related to courses ordering and organization. Heintz and Klein [HK14] suggest that software engineering courses should begin by showing students the “big picture”, rather than making them take foundational courses (such as Math) early on. The argument is that students do not get engaged if they do not see the purpose of a given content. Once students understand the whole process, there is a bigger chance they will see value in the “traditional” courses they are taking.

4.4 Systematic Mapping Study Final Remarks

We conducted a systematic mapping study in order to identify and characterize the main academic contributions on software engineering education in the context of software startups. The goal was to understand which tools, frameworks, models, methodologies and best practices are applied in this matter. After performing the research, we classified the studies according to five facets: focus, contribution, research method, research type, and paper rank. The focus facet revealed that studies fell into one of the following categories: real projects, multidiscipline, environment, and teaching. This information is consistent with data found in the literature [GPU+15, GUP+14, NDSA15, PGU+14]; a startup tries to solve real world problems, it needs a multidisciplinary team to succeed, and the environment plays a key role.

Our results indicate that research on software startup education is taken its first steps. Most studies in the area are recent and the majority were published in conferences

and symposiums, and fewer in journals. This is clearly an indication that there is a lot more to be explored.

In regards to RQ2 - Which tools, models, methodologies and frameworks are applied in a software startup education context for software engineering students? - we could identify that there is no consensus regarding tools, models, methodologies and frameworks for teaching software startups. We raise two hypothesis for this matter. The first one is related to adaptability and context. Depending on the focus of the course, a different strategy and a different set of tools and methods are needed. The second hypothesis is that this field is just starting to be explored by the scientific community. Therefore, there could be an opportunity to design a single approach to be used in software startup education.

RQ3 - What are the reported best practices in regards to teaching software startup for software engineering students? - brought interesting insights and take aways from software startup teaching experiences. Even knowing that a startup addresses a real world problem, offering this experience is challenging. The connection between the educational setting and the university ecosystem, such as technology parks and incubators, seems to minimize this gap.

In conclusion, this systematic mapping was a first attempt to better understand how software startup is taught to software engineers in educational institutions. We understand that several opportunities were created and can be explored from the findings we carried out. We intend to examine the identified gaps in order to develop further research on the proposed topic.

The experimental study described in this chapter was presented [CSPP18a] at the 22nd International Conference on Evaluation and Assessment in Software Engineering (EASE) in 2018.

5. STUDENTS' PERCEPTION ON LEAN PRINCIPLES: A MULTI-METHOD STUDY

As already mentioned in this document, some entrepreneurship courses teach emerging methodologies for startup development like Lean Startup [Har15, Nob11, WKA16] and Customer Development [CPF+17]. Fitzgerald and Stol [FS17] argue that these methodologies are consequence of the need of a closer integration between business needs and development. However, despite of consisting interesting concepts and ideas, these methodologies can be difficult to comprehend and to apply in practice [BOBL13], which may hinder software engineering students to grasp business skills by following the courses. To the best of our knowledge, this is a problem that has not been investigated in the literature. Therefore, this study focuses on the following research question:

- **RQ4**: *What are the benefits and challenges of software engineering students learning the Lean Startup approach?*

To answer the research question presented, we performed a multi-method study composed of two parts. Firstly, we studied two different teams of students learning by applying the Lean Startup methodology to develop a new product or service. A set of benefits and challenges in the learning process perceived by the students were identified. Secondly, we conducted a survey to a group of students from a mobile application development course that also taught Lean Startup and Customer Development concepts to further examine some specific challenges discovered in the case study. We analyzed data in both parts using thematic analysis, a technique for identifying, analyzing and reporting patterns (themes) in qualitative data [SCC98]. The results indicate that students understand the importance of collaboration with other courses, such as business development and marketing, and the importance of user involvement in development. However, they tend to simplify concepts, trying to adapt them to what they are familiar with, sometimes losing the main concepts behind the principles. This research protocol is depicted in Appendix A.

5.1 Students' Perception on Lean Principles - Research Design

In order to answer our research question RQ4 - *What are the benefits and challenges of software engineering students learning the Lean Startup approach?* - we followed a multi-method study design composed of two parts: (1) a multiple case study conducted in two sites, and (2) a survey applied to software engineering students in a third site.

5.1.1 Data collection

The first part consisted of a multiple case study [Yin03] based on two cases within different educational configurations:

Case A. Two students in the end of a software engineering course in a Brazilian university employed Lean Startup and agile methodologies in their bachelor thesis project. The group initially consisted of three students but after a few months one student dropped out. The project consisted of building a website to inform students, professors and employees about events occurring in the university campus. It took approximately nine months. Nonetheless, the effort was not equally distributed along the period.

Case B. Four students, one from a Masters and three from a Bachelor program in software engineering from an Italian university participated in the optional Lean Entrepreneurship course. The setup of the course was project-based, following a learning-by-doing style (the course was described in a paper written by Wang *et al.* [WKA16]). Students developed a project based on the business idea of their own: a platform to connect car owners keen to rent their cars to possible renters when they were not using them. This course lasted four months.

We collected data mainly in the end of each project. For Case A, students developed an undergraduate project thesis to be presented for evaluation that described all the tools used as well as the project history. The team described the product development in three iterations. Each iteration description had the following sections: building, diffusion, metrics and learning. We also included the emails exchanged between the advisor and the team. An initial set consisted of 150 emails had a team member as the sender or receiver. A subject analysis revealed that 10 of them were not related to the project. Most of them were about deadlines and meetings arrangements. We analyzed those related to the difficulties encountered by team members.

For Case B, the course ended with a retrospective session conducted by the teacher in which the group depicted the project history. During the retrospective, students were asked to draw the journey during the course on an A0 paper sheet, using whatever format or notion that they deemed appropriate. Then they were asked to explain the drawn diagram. The retrospective session took more than one hour, and in the first nine minutes the team discussed internally what to drew. We recorded, and later transcribed, all conversations and discussions during the retrospective session.

The second part of this study consisted of a survey undertaken in a 2-year iOS application development course in Brazil. The class was composed of 40 students with strong software engineering background and 10 design students specialized in user experience for mobile applications.

During the course, students conducted various software development projects, classified as small tasks, nano and mini-challenges. At the end of the course, students worked on a final software development project (a big-challenge activity) which had a six-month duration. They always worked in groups (between three and six students) composed of developers and at least one designer.

The survey was performed following the guidelines and best practices proposed by Wohlin *et al.* [WRH+12]. Since this study focuses on software engineering students, we did not apply the survey on the design students. Therefore, the survey was sent to 40 individuals.

All students that participated in this survey have already received lessons on Lean Startup and Customer Development and have practiced and applied these concepts in various activities. The survey was intended to further explore the challenges identified in our multiple case study. In order to do so, we decided to present a small scenario that asked students how they could contribute to a project. The scenario is described as follows:

"An entrepreneur friend of yours comes to you with a project idea that he had not yet implemented. He wants to create an app/system that connects elderly people with technology teachers. His idea is to help people to get into the digital world (learn how to use an email, Facebook, Instagram, etc). At first, the business model would be to charge the elderly for the class and keep a commission (between 5% and 10%) and pass the remaining to the teacher. You like the idea and join the project. Before leaving, you received a task: describe, in your opinion, what and how you can help. The idea is to present the next steps and (given your expertise) how you are going to help this project."

The idea behind this approach was to leave the floor open to students to develop their thoughts freely; we believe that if we had mentioned explicitly that we were working on a survey about Lean Startup/Customer Development, students would have been induced to answer accordingly. Students were asked to write down their answers in a text editor and send them to author of this document.

5.1.2 Data analysis

The data analysis conducted in both parts employed mainly thematic analysis, following the scheme proposed by Braun and Clarke [BC06]. Although suggested to psychological studies, this approach has been commonly used in software engineering research [CD11].

In the first part, we labeled relevant pieces of data and initial themes emerged. The labeled excerpts were then grouped together. Through a comparison between them, we made some improvements. In addition, we conducted a cross-case analysis to compare the two cases related to their setup and results according to the themes identified in each individual case. In the second part, we analyzed the survey responses under the themes identified in the case studies.

5.2 Results Analysis

This section presents our results. The first subsection presents findings from the multiple case study: a detailed description of each case and a cross-case analysis. The second subsection displays the survey results.

Multiple-case study

Case A. In this case, students never mentioned a business model for their project. Although Lean Startup can be useful to non-profit organizations, there was no concerns about how to support the project financially. One possibility that students mentioned in emails were that the project could substitute the pre-existent university events page. In this sense, also in the emails, there was a concern from the advisor about why users should switch to the new product. Regarding design and user experience, in the thesis, the text mentioned that suggestions made during first iteration focused on design aspects and still in the third iteration: *"most of the critics [...] were related to the experience provided by the site and that it was not attractive enough"*. Finally, for marketing and distribution, the difficulties were users acquisition and engagement. The first was present in each interaction description and they also wrote: *"with the acquired learning, it was clear the importance of time spent in marketing"*. The latter was present in an email from one of team members: *"engagement: one of the challenges is to make the user to come back and give her interaction ways that make her want to come back every week to see all the events"*.

In the thesis, students made a conceptual error: they used the word "hypothesis" to describe a list of features, such as "create an interface". They confused hypothesis testing with building incrementally. A possible explanation is that the Build-Measure-Learn cycle first step is to formulate hypothesis, so they felt obligated to write a hypothesis while reporting this interaction. Nevertheless, they followed a cyclical process as described in their thesis (Figure 5.1).

It is interesting to show pieces of data where students highlight their learning: *"that demonstrated the importance of keeping development always in touch with the final user"* or *"the importance that the client represents in the process".* Additionally, in their conclusions,

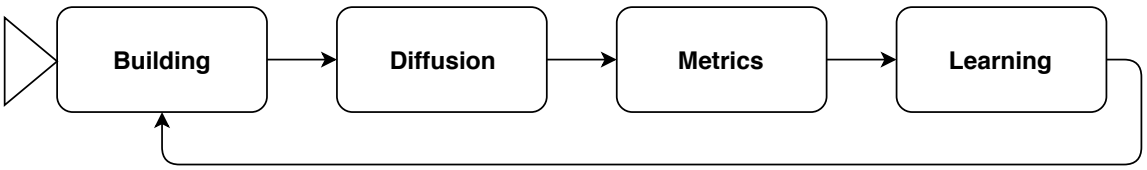

Figure 5.1 – A schematic view of group A's thesis.

they stated that: *"not always the concept about a project or a product idea by developers point of view is, in fact, a need to the target users".*

Case B. From a business perspective, the second group mentioned that the advisor drew attention to the need of a specific insurance to make the product viable. Although such insurance was fundamental to the product viability, the group had already started developing software architecture. Regarding design and UX, they said that they still needed help to understand *"what the consumer wants"*. They also mentioned that they were not able to create an interesting logo that could enhance product adoption.

The tendency to follow a straight path is clear in the picture the second group drew (Figure 5.2). Although classes in the course advocated the Build-Measure-Learn cycle from the Lean Startup methodology [Rie11], the depicted flow still recalls the waterfall process. The group also mentioned in the retrospective while discussing the insurance: *"we started the implementation of the website after the form. We didn't think about it".* During the retrospective, the team also made a conceptual error: when asked what an MVP was, they replied: *"MVP is just the minimum set of features we should put in our platform".* Instead, according to Lean Startup, MVP is the minimum set of experiments one should run to test a hypothesis.

As takeaways from the course, this group mentioned: *"what the consumer wants is more important than you implementing"* and *"if it does not put value to the market, there's no sense to making it".*

Cross-case analysis. Although students had similar team sizes and backgrounds, they followed really different processes. The first group performed a cyclical process trying to formulate hypothesis and to validate them, that is, following a Build-Measure-Learn cycle. They also focused on customer feedback that comprehends a good part of the methodology [GCRB15]. Meanwhile, the second group basically followed a straight plan (having the idea, gathering requirements, designing and testing the idea, development, platform testing and deployment), even after participating in classes about the Lean Startup methodology. In software startups, Giardino *et al.* [GWA14] already saw this plan driven approach as waste and a reason for startups' failure. Meanwhile, Bosch *et al.* [BOBL13] argue that agile and lean practices are better for early stage software startups.

A possible explanation is that in the first group, students was more engaged in learning the methodology or really developing a viable product. Another possibility would be

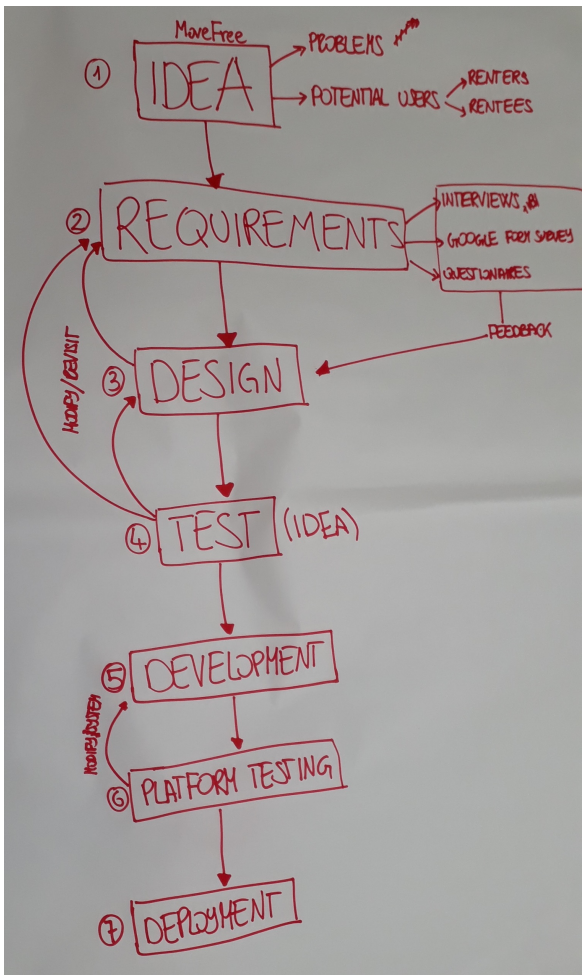

Figure 5.2 – Flow draw by the second group.

that they are better students than the second group. Even if these hypotheses are correct, an element should have been more important to guide a group: the advisor. For the second group, the advisor had an important act telling them about the insurance issue. However, students did not mention anything else about her performance. For the first group, the advisor highlighting the importance of customer feedback and hypothesis testing, guiding the students to focus on them. The second group mentioned another interesting point: *"it's different from the other courses: you're given [a task] and deliver [it]; that is, there's no market connection".*

There were, though, some similarities between the two cases. First, neither teams worried about business issues: Case A students did not come up with a clear business model and Case B students ignored concerns about insurance and continued developing the technical solution. Of course, the academic environment may be the reason of such insouciance about business. Second, both teams recognized the importance of validation and customer feedback.

Thematic analysis identified 9 themes categorized in the two categories: *Benefits* and *Challenges*, as described in Table 5.1.

Category	Theme
Benefits	Business learning
	Design learning
	Feedback importance
	Marketing learning
	Success based on metrics
	Team building learning
	User experience learning
Challenges	Business perspective ignored
	Conceptual errors

Table 5.1 – Themes found in first phase.

The benefits include students learning regarding different areas:

- **Design:** for instance, all suggestions in the first iteration of Case A concerned design issues;
- **Team building:** like a student in Case B mentioned: *"people from different backgrounds can give different opinions"*;
- **Marketing and distribution:** as Case A thesis states: *"with the acquired knowledge it is clear the importance of time dedicated to distribution"*;
- **Business:** Case A thesis states: *"not always the idea of determined project or idea from the perspective of developers is, in fact, a necessity of targeted users".*

Besides that, several excerpts show the importance of user feedback during projects development. For instance, in Case A's students wrote: *"Although the alerts page visits had a peak after launching, the feature was abandoned by users, generating almost no visits. This shows that possibly it wouldn't be interest to invest in its development."*

The challenges comprise the avoidance to tackle business issues as mentioned earlier, the tendency to follow a straight plan focusing on technical challenges and the difficulty to understand some concepts, like MVP. In this regard, in Case A, students used hypothesis as a synonym of prototype. Besides that, team members from Case B defined MVP as *"just the minimum set of features we should put in our platform"* instead of the correct concept of validation.

5.2.1 Survey results

From the 40 students surveyed, 37 answered (92,5% response rate). In the large majority (28 times), students focused on the *validation* of the idea, highlighting the **business learning** the course enabled. In their answers, students generally described how they would perform validation. The most common technique was *interview*, mentioned 11 times. In a student words: *"To interview the target audience and to understand what is its real needs and difficulties."*. Other tools mentioned were prototype (6), concierge (2) and talking to close people, like relatives (2). However, several students (6) did not detail how they would perform the validation they proposed. In summary, there is no learning cycle, in Lean Startup terms, in this process. Even though some learning happens during validation, this approach resembles a traditional software/business requirement process. These answers also illustrate conceptual errors students made. Figure 5.3 summarizes this idea: students understand validation as a step to get to the implementation, and not as a learning process.

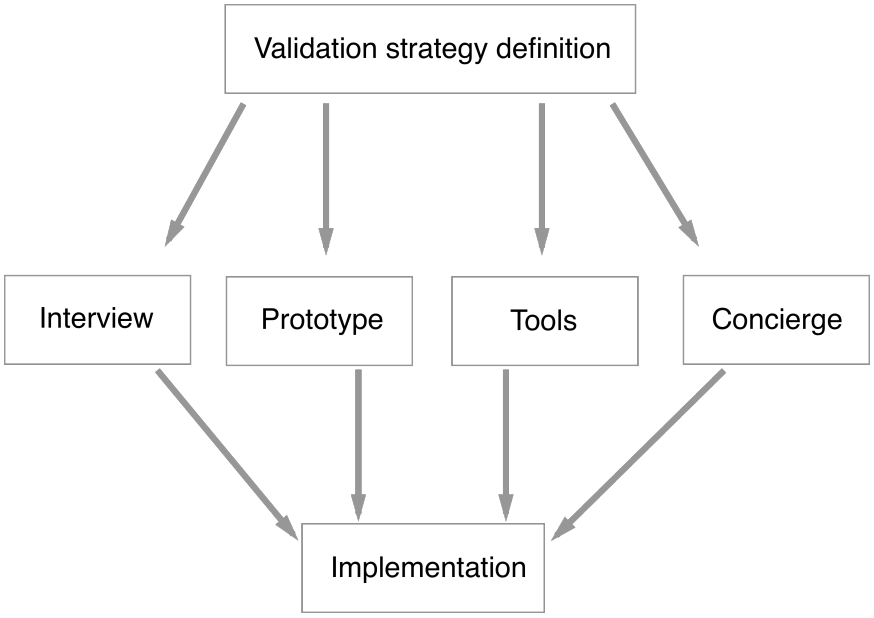

Figure 5.3 – Students' perceptions on validation.

As another example of conceptual errors, five respondents criticized the idea without proposing to validate it. One student said that he would not participate in the project: *"I think the idea will not work, because doing an app to connect people that do not know how to use technology will not be effective to reach them, because they will never succeed to use this app without previous knowledge. The app will not reach its target audience. I suggest to pivot the idea."*. It is understandable to have personal opinions about an idea. However, the Lean Startup process makes it clear that one should look for facts, and not opinions. This outcome showed us how personal opinions and beliefs have a significant influence on startup projects.

Five times students focused on promote themselves to the fictional job and three of them would not even test the idea. For instance, *"I could contribute by developing the project, helping with brainstorm dynamics, prototypes ideas, etc. I could also help to put the platform website up, the meeting schedule service and an iOS app, organizing and managing the team if necessary."*.

Three students highlighted the importance of listening to user feedback. One student mentioned that *"it is highly important, as soon as we develop some features, to test with the audience in order to receive feedback so possible changes could be made"*.

Marketing and distribution concerns were mentioned 3 times. For instance, one student said: *"Besides that, I would do ads in games that elderly people use to play [some mobile games] and advertise the platform there"*.

Students mentioned concerns about user experience 6 times. One student described: *"To facilitate the app user flow and to avoid that the user loses herself, I would leave the minimum of options available in the screens. Besides that, I would put some gamification elements to make the user experience more pleasant"*.

5.3 Discussion

Our results suggest that a Lean Startup or Customer Development course indeed makes software engineering students aware of business related concerns of a software product like marketing. In our case studies, both groups tried to validate their business assumptions about users, although to different levels of success. Moreover, Case A also faced several issues about user interface and distribution. One reason that Team B have not have confronted them may be the shorter period spent in the project. Students' statements about learning also corroborate this observation.

Nevertheless, students face difficulties to avoid a straight plan focusing on technical challenges. They had already absorbed the plan-driven idea of "getting things done". From the first phase, two facts support this finding: from Case A, the observation that the group tended to focus on developing the solution and, from Case B, the flow students drew. In the second phase, such an issue happens when students described technical plans on what to be done next, ignoring a validation stage. Even when they suggested a validation stage, they do not think it in a cyclical manner: adding a validation as an initial stage to be performed before implementation.

In addition, students still struggle to understand concepts such as MVP and validation, generally using them as synonyms of simpler and already known concepts: MVPs as prototypes and validations as interviews. For example, students in Case B defined incorrectly an MVP as a product with a small set of features, ignoring the concept of business hy-

pothesis. In Case A, students wrongly used the term “hypothesis” to designate a prototype. For validation, Case B students and many survey respondents described the employment of interviews without further details.

Based on our findings, we suggest that software engineering students to be immersed in the concept of software business and customer relationships earlier in their formation process. More practical courses with real users or customers could be a solution. Such courses would be in line with the move from product to services seen in software products where “companies must transition from working with planned releases with detailed requirement specifications to continuously experimenting with customers” [Bos16]. Since Case A’s results were slightly better than Case B (students performed a cyclical process focused on customer feedback), some indications could be gathered from Case A which suggest how to improve students’ business capabilities. In this case, the advisor encouraged students to work in cycles and real users interacted with the platform. The problems they had with imprecise concepts and the lack of concern about the project’s business sustainability should be the focus for improvement in the future courses.

5.3.1 Threats to validity

Runeson and Höst [RH08] describe a common scheme to assess threats to validity when reporting a case study composed by four aspects: construct validity, internal validity, external validity and reliability.

Construct validity reflects “to what extent the operational measures that are studied really represent what the researcher have in mind” [RH08]. The use of multiple information sources for both case studies reduces this issue as suggested by Yin [Yin03].

Internal validity is related to causal relationships and represents the possibility of other factors not taken into account also explain a consequence. Although this study is exploratory, we used pieces of data to support each claim in data analysis and explored alternative explanations like students drivers. Triangulation of data from different sources within each case study also improved internal validity [LSS05]. Another threat was that one of the researchers was the first group mentor; it could have biased the data analysis. The use of three student groups in three different scenarios in two countries improved **external validity**.

The use of thematic analysis improved the **reliability** which goal is “to minimize errors and biases in the study”, that is, if another researcher performs the same study in the future, she reaches the same results [Yin03]. Besides that, this study describes all steps performed in data collection and analysis.

5.4 Students' Perception on Lean Principles Final Remarks

This study focused on the challenges and benefits of startup methodologies courses for software engineering students. We performed a multiple-case study and a survey with students in three different contexts from two countries. Our results indicate that students understood the importance of business concepts and developed several soft skills, but still struggle to understand key aspects of the Lean Startup methodology, usually over-simplifying them, and experience a tendency to follow a straight path focused on technical success.

Based on the difficulties found, we proposed some insights that could be used to create or improve business and innovation related courses taught to computer science and similar areas. Lecturers should be concerned with important concepts, like MVP and validation, and perform a deeper inspection to check students' understanding of these concepts. They should also organize their practical courses to encourage students to follow a cyclical path avoiding the tendency to pursue a straight path.

Our results stimulate further studies on introducing business and innovation education into software engineering courses. For instance, it would be interesting to verify whether introducing these concepts in the beginning of the course, when students are less biased, could bring better learning results.

The experimental study described in this chapter was presented at the 50th ACM Technical Symposium on Computer Science Education (SIGCSE) in 2019 [MCW+19].

6. SURVEY ON SOFTWARE STARTUP EDUCATION

The goal of this experiment is to understand how software startup is taught by lecturers/professor across the world. So far, we do not know how courses are carried out aside from papers describing individual experiences. Therefore, this study focused on understanding how software startup courses are taught in universities, specially in regards to the multidisciplinary context of a software startup. In regards to the purpose of this thesis, this study focuses on the following research question:

- **RQ3**: *What are the reported best practices in regards to teaching software startup for software engineering students?*

6.1 Survey on Software Startup Education - Research Method

In order to study the current state of practice of software startup education in universities, we devised a qualitative, largely open-ended survey. The goal of the survey was to understand in detail how software startups are currently taught in universities world-wide. In creating the survey, papers discussing software startup courses in universities, alongside our own teaching experiences in the same area, were used to ensure that the questions covered all aspects of such courses, ranging from duration to group size where applicable. Though some questions were given multiple choice answer options, most of the survey consisted of open-ended questions. Open-ended questions were utilized to gather data as rich as possible with a survey while still consuming less resources from the responder than a qualitative interview would have. Similarly, a survey was selected as the method of data collection over interviews due to the nature of the phenomenon being studied. Though interviews would no doubt have achieved the same goal, we considered the resource-intensiveness of interviews to be a problem when interviewing other scholars. Furthermore, university education as an area of study and course-based university teaching is a well-understood phenomenon that can arguably be comprehensively covered with pre-determined questions.

The survey contained questions about both the course and the teacher(s). Aside from the way software startups are being taught, we were also interested in understanding which disciplines were concerned with them the most. In addition to focusing on teaching methods, the questions also covered the basic course information: course length, course name, which discipline the course is a part of, whether the course is mandatory or optional and other such generic university course information. Aside from asking how the course is held, we also aimed to find out some of the reasoning behind the choices by asking some why-based questions. The survey protocol can be found in Appendix B.

The survey was sent out to individuals involved in teaching software startups in universities. Aside from contacting such individuals we knew beforehand, we searched for such courses online and contacted the teachers.

6.2 Results

In this section we present the results based on the 15 responses gathered during this process. Table 6.1 presents the countries involved as well as the number of instructors per country. Perhaps due to the nature of software startups, all of the responses have described courses either involving a high degree of practical work or focusing entirely on practical project work on a hypothetical or real software startup. As software startups operate under a lack of resources, have little to no operating history, and typically consist of inexperienced (developers or otherwise) individuals [UAW+16], it is indeed possible and even rather simple to replicate or simulate experience in a university course setting, just as it is possible to have the students attempt to found a real-world software startup in the process. Indeed, all courses were described to be practice-oriented courses involving teamwork.

Table 6.1 – Instructors per country.

# Country	# of instructors
Brazil	3
Germany	2
USA	2
Denmark	1
Finland	2
Italy	2
Netherlands	1
Slovakia	1
Sweden	1

We also asked instructors if they had previously worked on a startup. This information is somewhat important since this experience can bring more value to the classroom. Roughly half of them (7 out of 15) had already experienced the creation of a startup.

In regards to the multidisciplinary context of software startups, these organizations are software companies operating in terms of academic disciplines, in an area combining business and information technology. This was also reflected in the responses. Twelve of the fifteen courses were open to either a combination of IT and business students, or all students regardless of their major. Furthermore, all of the courses described in the surveys involved team-based work between students, and largely encouraged multidisciplinary teams including both business and IT students, as well as others if applicable. Student team

sizes in the courses were varied but the common consensus was that at least three students would ideally be in a team as “2 is not a team, it is a pair”, as one of the responses remarked. Conversely, five students per team was generally considered to be a soft upper limit, with multiple responses arguing that more than five students in a team would be likely to create problems in work distribution among the team.

Whereas all of the courses involved practical work, the nature of it was varied between responses. Some courses were more focused on software engineering with a secondary focus on the entrepreneurship aspect, whereas other courses were more focused on the entrepreneurship and innovation aspect with a secondary, if any, focus on practical software engineering. In two cases, the student teams would work on external commissions from real-world customers, although the trend seemed to be that the students were expected to develop their own ideas. These ideas, then, were worked on during the courses, and while they were never required to become real software startups, the students were typically encouraged to do so. In some cases, the students had indeed gone on to create successful real-world startups based on their ideas from the courses. As we can see in Figure 6.1, only 3 of the 15 courses have spawned successful real world startups.

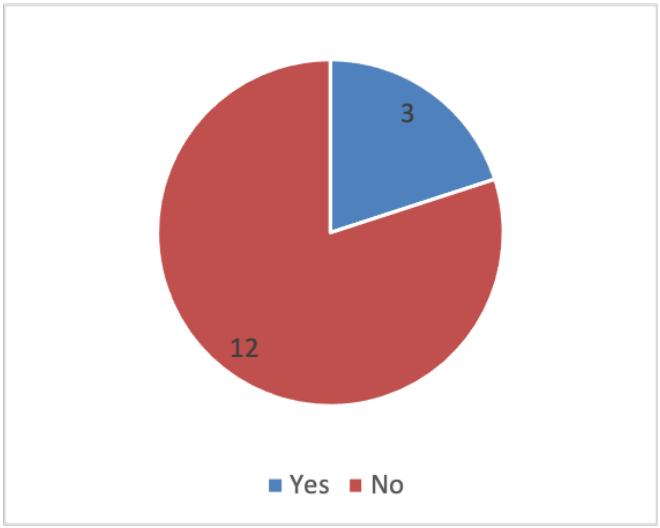

Figure 6.1 – Courses that have spawned real startups.

A clear line between a mock-up startup and a real startup in the courses described in the responses was not generally drawn. Even though the startups were not all intended to be real-world startups, or to become ones at a later point in time, all teams were expected to validate their ideas in some way, verifying that they would satisfy a real need. This typically meant carrying out surveys and interviewing potential customers, or even creating actual landing pages and social media profiles for the course startup. This was also the approach used for other work on the startups: for instance, in one of the courses everyone would pitch to real investors at a course end event, even if they had no plans of actually continuing to work on the idea after the course. In this fashion, teams that wanted to create a real startup based on the idea were free to do so without needing to take any actual steps, and the ones

that were there purely for educational purposes nonetheless created a mock-up startup as if they had been working on a real one. Only one of the courses was described to be purely educational. According to the data provided by this survey, we concluded that 10 of the courses aimed at developing real startups (see Figure 6.2).

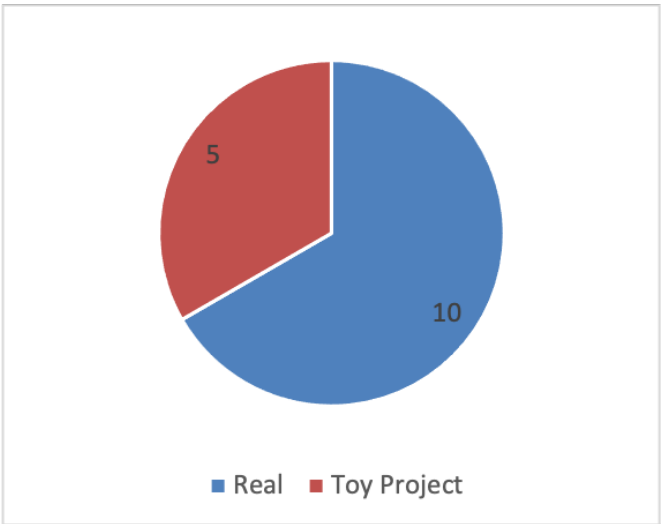

Figure 6.2 – Real X Toy Projects.

Past these similarities, however, the way the courses were carried out on the level of smaller details was highly varied. For example, in terms of deliverables or gradable tasks, some courses would require the students develop a working piece of software whereas other courses would focus more on honing the idea and then pitching the idea as the final result of the course. Figure 6.3 shows how many courses required that students develop a real piece of software, whereas Figure 6.4 presents how many courses required hat students develop a prototype. It is interesting to note that 4 courses do not required neither a real piece of software nor a prototype to be developed.

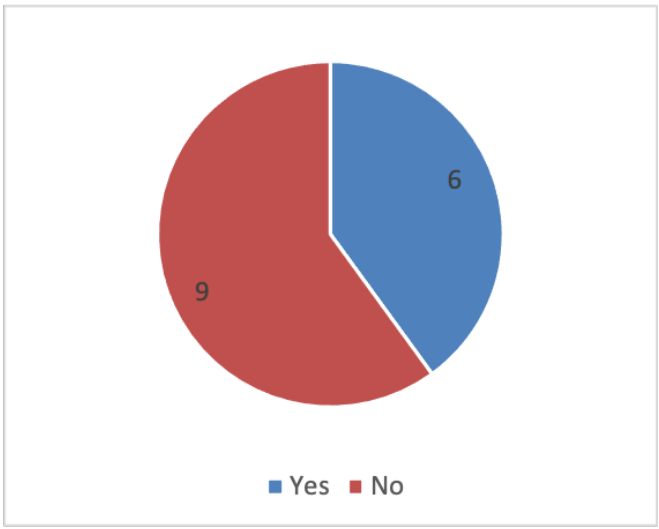

Figure 6.3 – Courses requiring software to be developed.

In the cases where software development was to be carried out, agile methods, mostly *ScrumBut*, were typically followed, but on the other hand programming language and platform were typically not pre-determined. Seeing as the idea being carried out largely determines how it could (or should) be done, this is understandable unless the course is more focused on teaching, for instance, mobile application programming for Android while simultaneously teaching startup entrepreneurship. The way the students were supervised during the course also highly depended on the required deliverables of the course.

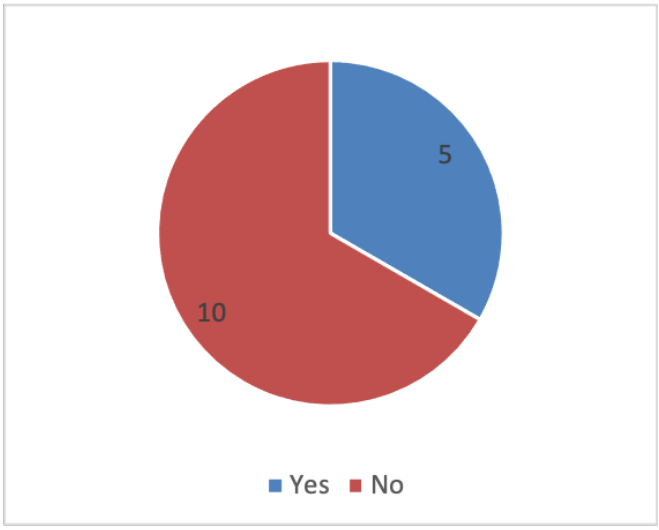

Figure 6.4 – Courses requiring a prototype to be developed.

Though the courses focused on practical work, they featured weekly or otherwise regular lectures. Aside from teaching relevant theories such as the Lean Startup Methodology [Rie11], the lectures were typically used to support the practical work more closely as well. Past the Lean Startup methodology, little consensus existed on which methods or theories to teach. In fact, the learning goals for the courses were notably varied, which serves to highlight the differences in the foci of the courses. Learning goals listed in the responses included:

- Strategies to test out business hypotheses;
- Practical programming skills;
- Project management skills;
- Helping students discover which aspects of entrepreneurship they like the most personally;
- Innovative business practices;
- Being a startup practitioner;
- Agile software development methods;

- Team skills;
- Using practitioner tools such as GitHub;
- Entrepreneurship.

Based on the number of responses, we have outlined some of the general trends in the way software startups are taught in universities. The more general trends in the way software startups are taught can already be seen in the data in order to provide a tentative answer to our research question. However, due to our sample size it is not realistic to draw any further conclusion. The combination of this study with others from this thesis can help us defining and proposing our framework.

6.3 Survey on Software Startup Education Final Remarks

To summarize our findings, the described courses focus on carrying out practical work, either in the form of software engineering, creating a startup idea and developing it further, or both. The courses generally involve creating a mock-up startup in student teams and, at minimum, coming up with an idea and developing it into a business plan. No clear line is usually drawn between mock-up startups and real startups in that the student teams are expected to carry out the same tasks regardless of their own goal with their course startup or startup idea. The courses often encourage students to create a real startup with their idea but do not require them to do so.

In regard to the multidisciplinary nature of a software startup, some of the courses focus primarily on one aspect of software startups such as software engineering and practical programming. These courses are typically only open to students of that subject such as software engineering. However, most courses seem to either involve students from different disciplines, typically from business and IT ones, in order to create multidisciplinary teams. Such multidisciplinary courses seem to be more common than those focused aimed at only students of software engineering, for instance.

Another important point is that startup-related concepts are seen as an integral part of entrepreneurship by now. Notably, the courses were not necessarily referred to as startup courses of any kind. In fact, only three out of the ten responses so far had the construct *startup* in the course title. The course titles were more associated with innovation, entrepreneurship, and software engineering practice.

Finally, it is very likely that courses described in academic papers present non-conventional educational ideas rather than tried-and-true methods for teaching. There is no reason to write a paper about a lecture-based course on software startups that ends in an exam about a book and the course content. Thus, in contacting the authors of various

papers in relation to our survey, the data has become biased in this fashion. It is unlikely that all or even most courses on software startups would be so focused on practice, even though it would appear that the amount of practice-focused courses in the area could be higher than usual.

The experimental study described in this chapter was presented [CKK+18] at the Software-intensive Business Workshop on Start-ups, Platforms and Ecosystems (SiBW) in 2018.

7. TEACHING SOFTWARE STARTUPS USING CHALLENGE BASED LEARNING: A CASE STUDY

The goal of this study is to apply Challenge Based Learning (CBL) in an undergraduate entrepreneurship course which focuses on software startups in order to identify strength and weaknesses of such methodology. By running this experiment we expect to further understand how CBL could be adapted in order to address software startup related concepts. The research protocol for this study is depicted in Appendix C. In regards to the purpose of this thesis, this study focuses on the following research question:

- **RQ2**: *Which tools, models, methodologies and frameworks are applied in a software startup education context for software engineering students?*

7.1 Methodology

The methodology was applied in an undergraduate software startup entrepreneurship course with 30 students. Meetings were held twice a week for one hour and 40 minutes for a period of four months. Table 7.1 summarizes the course schedule.

Figure 7.1 presents a graphical representation of the CBL methodology drawn by one of the students in the class. It is important to point out that we have chosen CBL instead of other active learning methodology because CBL focuses on real-world problems that are connected to students. This is the context in which a startup is developed.

7.1.1 Step 1: Engage

In the first month, students defined the *Big Idea* and completed the engagement stage. During this period, through lectures and activities, the CBL methodology was presented and students began the Engage phase. The identification and definition of the *Big Ideas* started by students bringing several themes of interest to the classroom. In this way, they could find common interests and formed teams according to those ideas. The result of this process was the formation of five teams. Once *Big Ideas* and teams were defined and formed, students explored their *Essential Question* in order to get to a *Challenge* to be solved. At the end of this stage, students not only delivered their results, but they also carried out the first reflection on their learning process.

Table 7.1 – Course schedule.

Phase	Class	Topic	Goal
Engage	1, 2	Course introduction	Present the course format, objetives and goals
	3, 4	Introduction to the CBL methodology	Understand the purpose of the CBL and explore its phases (Engage, Investigate, and Act)
	5, 6	Definition of Big Ideas and Essential Questions	Brainstorm on Big Ideas; team formation; Big Idea and Essential Questions definition
	7, 8	Initial research	Challenge definition. Teams begin to organize their research process. Guiding Questions, Activities, and Resources definition
Investigate	9, 10	Market research	Teams should begin their field research according to the guiding questions raised
	11, 12	Field research analysis	Data analysis; teams should evaluate their preliminary results
	13, 14	Planning and prototyping	Finalize investigation; define solution; begin working on requirements for the given solution
Act	15, 16	Solution development I	Prototype construction and development
	17, 18	Solution development II	Prototype construction and development
	19, 20	Investments	Understand the resources needed to develop the solution
	21, 22	Solution evaluation	Validate solution with potencial users and customers
	23, 24	Pitch	Understand the process of delivering an effective pitch presentation
	25, 26	Fundraising and Partnerships	Understand the process of raising capital for a startup
	27, 28	Project Presentation	Present the project to a group of investors

7.1.2 Step 2: Investigate

From the second month of the course, students carried out the research phase by defining *Guiding Questions, Activities and Resources*. During this period, students conducted a review of the literature on their topic in addition to working on a field research with potential users of the product or service. Moreover, students were encouraged to validate the relevance of their projects, as well as to make changes to their research project. Students could even go back to the Engage phase in order to modify their challenge. At the end of this stage, students delivered their research findings (as a report) and also their second reflection on the learning process.

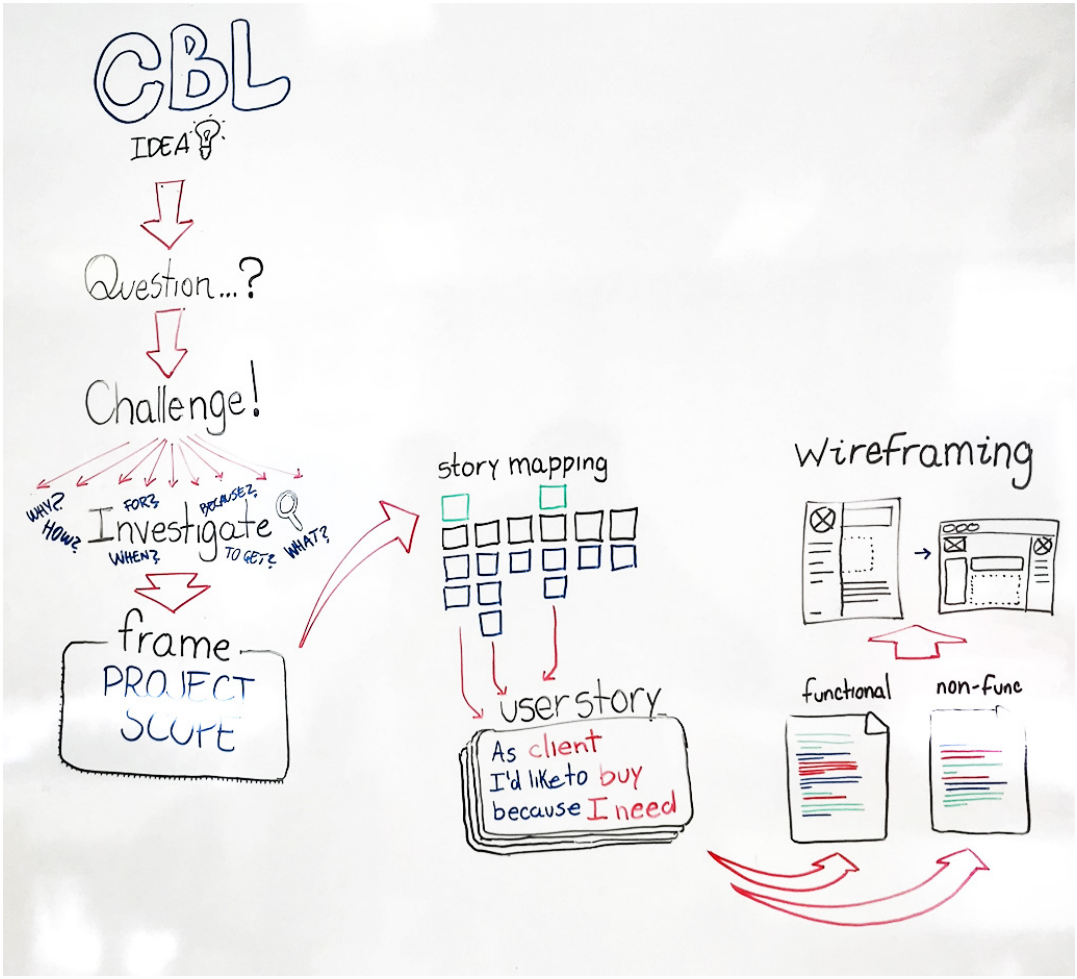

Figure 7.1 – CBL framework (Author: Course Instructor).

7.1.3 Step 3: Act

After the research phase, students developed their solution for the proposed challenge based on their research findings. During this stage, students evaluated and validated their final project with potential users and customers in order to verify whether the solution meet their needs. At the end of the semester, students presented the process they went through as a pitch presentation. The deliverables of this stage were a report summarizing the whole CBL process - from ideation to the final solution. In addition, the final reflection was carried out in order to explore students' learnings and also to verify whether the CBL methodology was relevant to them.

7.2 Data Collection

Data was collected from students' deliverables (reports and presentations) as well as from reflections. Reflections were conducted on the following prompt: "*what did you learn the most in this phase? What did you struggle with? Is there anything that could be done differently?*". Reflections were done individually and privately. This encouraged students to be more open to share their thoughts. In addition to this information, at the end of the course a questionnaire was applied so we could collect and explore more data regarding the development of the projects. The results of this process is explored in Section 7.3. Finally, we also collected a reflection from the teacher, so we could understand his perception not only in regards to the CBL methodology, but also on students' involvement and engagement.

7.3 Evaluation

We present in this section some research findings as well as the teacher and students' perception about the use of CBL in the classroom environment for startup development.

7.3.1 Students' Engagement

The first question we asked students was related to their engagement. The goal was to understand how connected they were with class activities, project and also with their peers. Figure 7.2 presents the results. As it can be observed, 25 out of the 30 students reported to be highly engaged. This outcome is outstanding, since it is very rare to keep students excited throughout the whole process.

Table 7.2 presents the most important attributes that motivated students in regards to the whole course. This data was collected through an open-ended question. Students could say as many attributes as they wanted. In total, we had collected 44 attributes. It is important to point out that we grouped similar answers together according to their meanings and contexts. As we can see in Table 7.2, self-interest was the most cited reason for students' motivation. In addition, we can observe the importance of being connected to the Big Idea and to the team.

We also wanted to know the reasons why students were engaged to their projects. Figure 7.3 presents the outcomes. Making a difference in people's lives seems the be the most important factor for students to work on their projects. It is interesting that none of them pointed out the opportunity of making money. This seems to be a characteristic of the new

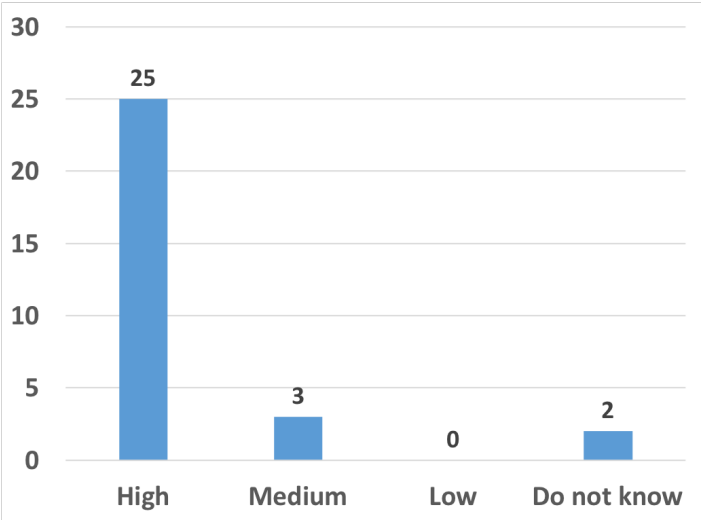

Figure 7.2 – Students' engagement.

Table 7.2 – Attributes that motivate students.

# times	Attribute
10	Self-interest
6	Connection with the Big Idea
6	High involvement with the team
5	Team motivation
4	Good communication with the peers
3	Impact generation
3	Learning experience
2	Solution for you or people close to you
1	High number of users
1	Confidence
1	Recognition by the team
1	Startup development
1	Working in teams

generation; money will always be important, but being engaged and connected to a given project is more relevant.

Teamwork was also a point of interest to us. Hence, we asked students about team engagement. Twenty one students reported that all participants were equally involved in their project's activities and that they all collaborated to better find the solution to the proposed challenge. Seven students understood that their teams were not as excited as they should be, and only two students mentioned that teamwork was below expectation. It is interesting to notice the different perceptions among students. Even though the majority of them were comfortable with their peers, a few students felt that they could have done better than they did.

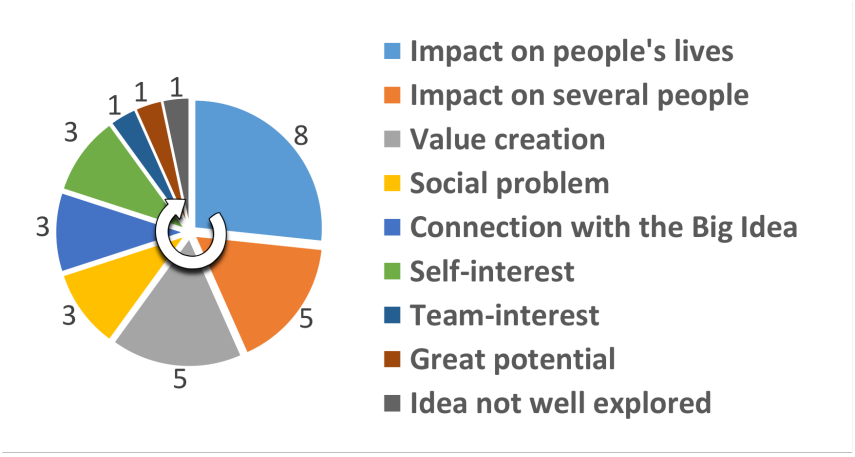

Figure 7.3 – Reasons that engaged students in their project.

Another important aspect that we were interested in was how students were open to new ideas and thoughts. Since CBL fosters and stimulates engagement, we wanted to know how this process worked and how students felt about it. Almost all students (25 of them) stated that their ideas and thoughts were heard and accepted by their peers. Four students reported that their ideas were partially accepted by others, and only one student had the perception that his ideas were never taken into account throughout the process.

In order to verify whether the three CBL phases (Engage, Investigate, and Act) were important to students, we asked students if the framework helped them going further into the process of creating their startups. All students agreed that CBL was important specially because of the Engage phase. Defining an essential question and a challenge helped teams understand their goals. Once the goal is clear, it is much easier to investigate on it.

7.3.2 Students' Difficulties

We also asked students about their difficulties and problems throughout the process specially in regards to the development of a startup. We divided our question into two moments. The first one was related to the Investigate phase (Figure 7.4) and the second one to the Act phase (Figure 7.5).

It is clear that availability was a key problem during the investigation phase. We have a few hypothesis that could be further analyzed. Since this is a software engineering undergraduate course, maybe students are not very comfortable with doing actual research (specially field research, when they have to talk to real users). This is not a common activity for a software engineering student. Another possibility is that students were not able to organize themselves to work on this activity.

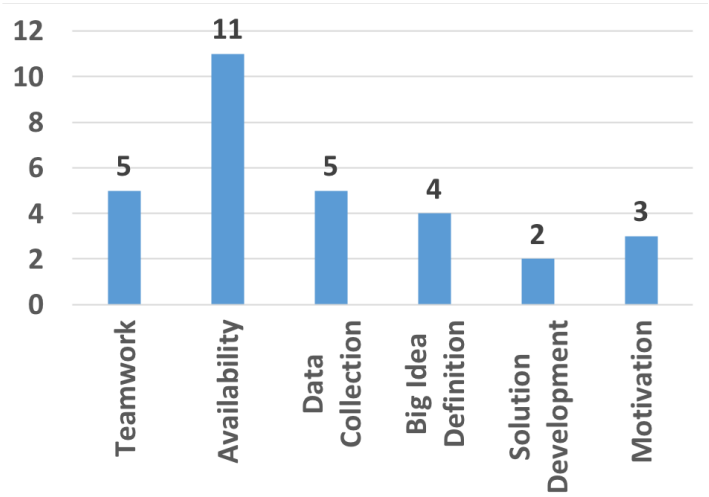

Figure 7.4 – Difficulties during the Investigate phase.

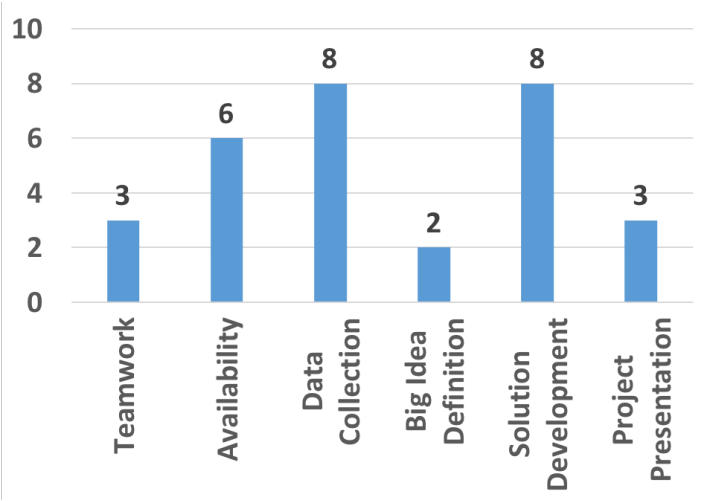

Figure 7.5 – Difficulties during the Act phase.

In regard to the difficulties during the Act phase, the solution development was a critical point. We further found out that students felts that they did not have enough time during the semester to work on their solution. Some students even suggested that this class should be two-semester long. In this way, they would have more time to develop their solutions. Regardless of this perception, the professor stated that the students' solutions were appropriate for the scope of the course.

Finally, we wanted to understand how students dealt with when they needed to pivot their idea. In the CBL context, a pivot means that the current step must the revisited, or it is necessary to go back and redesign their previous step. For example, if a team is on the investigation phase and they find out that the challenge is not worth solving, they need to return to the Engage phase in order to adjust it according to their findings.

Figure 7.6 shows that only nine students understood the need to modify one or more aspects of the project during the investigation phase. This means that they either had

to adapt their strategy during investigation or they had to go back to the Engage phase to adjust their challenge.

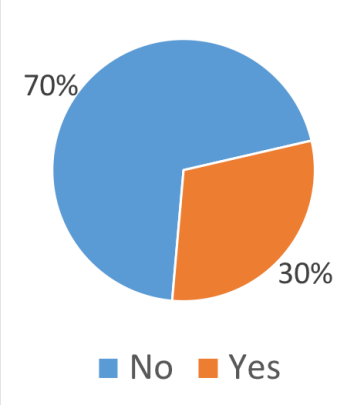

Figure 7.6 – Percentage of students who pivoted during the investigate phase.

During the Act phase, however, more students felt the need to adjust part of the process (see Figure 7.7). This happened probably due to the fact that when teams begin working on their solutions, it becomes clearer whether or not they are closer to solve the challenge. Therefore, more adjustments might be needed in order to nail on the solution.

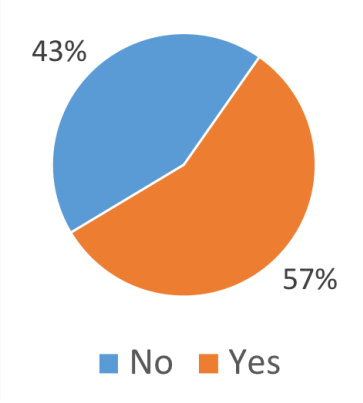

Figure 7.7 – Percentage of students who pivoted during the act phase.

Regarding assessment, this course had no written exam. Students were evaluated throughout the whole process by presenting their evolution on a weekly basis. This accounted for 60% of their grades. The remaining 40% was related to the final pitch presentation. It is important to point out this assessment configuration has nothing to do with the CBL process. The goal of this approach is to focus on the journey and not only at the end result.

7.3.3 Students' and Teachers' Perceptions

In addition to the questionnaire, students and faculty perceptions were also collected through reflections. The purpose of this analysis was to identify positive and negative aspects as well as improvement points in regards to the CBL methodology.

From faculty stand point the application of this methodology was very interesting, because it brought to the classroom the construction of a business project based on a real opportunity. According to the professor, the methodology allowed students to work on a solution based on a consistent research process. Since students were connected to the challenge they were trying to solve, the level of engagement grew as teams learned more about the context their were working on. Even though teamwork is not a trivial process, by focusing on users' needs, team members were able to solve conflicts more easily, since they were excited about delivering value to people they care about. A negative aspect would be the lack of time to work on the Act phase. When students got the work on their solutions, the semester was almost over. The professor suggested breaking down this course into two semesters, so students could investigate in the first semester, and focus an entire semester to work on their solutions.

In regards to the students' perceptions on the CBL methodology, what called our attention was the lack of engagement in the beginning of the semester. Most students admitted that their focus was on passing the course. They were not concern about what they would be learning. However, the Engage phase was key to change this perception. Since students were working on something they cared about, engagement went up. One student said: "*I never like when teachers ask us to research on a given topic. However, when it is something that I have an interest on, the game changes; it becomes more fun. I did not even realized I was actually studying.*"

One interesting aspect about the Investigate phase is that some students felt frustrated about their findings, since they could not validate their hypothesis and had to restructure their challenge. However, this was a big learning point: the process of developing a business is cyclical. There is a need to iterate with customers and users in order to understand exactly what they want. The professor helped students dealing with this frustration by explaining that it would be way worse if they had build a whole solution that would be useless to users. In fact, these students saved time.

Students also agreed that there was not too much time to work on the solution. One team mentioned that they would continue working on the project after the course was over. However, this was an important take away from this course: one semester might not be enough to fully experience a software business creation.

7.4 Teaching Software Startups Using CBL Final Remarks

The use of CBL as a learning methodology for startups development have provided many positive dimensions to students, including the possibility of collaborative work, the reflection process for their learning process, the active searching for technical knowledge, as well as the personal involvement to solve a real problem.

Unlike other learning methodologies e.g.: *Problem or Project Based Learning*, the CBL foundations are characterized by the active and experiential learning, where students can actively acquire knowledge through work on open-ended problems. The possibility and flexibility of allowing students to choose their own projects and problems to be solved are great differential that increase students' engagement throughout the learning process.

During one semester students needed to define which problem and customer would be tested in order to run a set of experiments and to validate whether their problem was relevant for possible customers, collect customers' feedback, and to develop a solution for the problem, and test whether the business could gain traction. This process in a short period time was one of the big challenges faced for those who want to create a startup, specially for undergraduate students with no experience in dealing with real users.

Collecting information from real users is always a challenging and difficult task. Many students have reported that interaction with users, as well as the constant discovery of new requirements in the searching for the solution of the problem have created some barriers to develop an efficient solution of the problem. One possible solution for this situation could be the application of Lean Startup strategies into the methodology, where customer interaction must occur in short cycles of time during the solution development, maximizing the learning process and minimizing the risks of a useless solution.

The experimental study described in this chapter was presented at the XXXIII Brazilian Symposium on Software Engineering (SBES) in 2019 [DSC+19].

8. INITIAL PROPOSAL - CHALLENGE BASED STARTUP LEARNING

As mentioned in Section 2.3, the combination of CBL and Scrum was very effective in regards to teaching students how to develop mobile applications from a technical point of view. However, the framework fails to address the challenge from the point of view of the people affected by it. Linus Torvalds, the creator of Linux, once said that "*any program is only as good as it is useful*". Thus, it is crucial that students begin the process with this mindset. The goal of a startup, as already mentioned in this work, is to create a sustainable and scalable business. Therefore, knowing how to build a mobile application (or any software application for that matter) is not enough; the system has to be useful for a given group of people.

In this sense, the proposed framework is a result of applying CBL, Lean Startup, Customer Development and software development strategies, for two years in an undergraduate digital entrepreneurship course. By doing so, learners not only have the opportunity to develop the necessary technical skills, but they can also connect themselves with potential users/customers in order to gain real world experience. It is important to point out that the main goal of this framework is not to teach how to develop a successful startup (even though this would be great for students and the community), but to teach the software startup development process.

Figure 8.1 presents the Challenge Based Startup Learning framework overview. The process starts in the *engage* phase, where students define their big idea, essential question and the challenge, using design thinking and brainstorming processes. This phase is straightforward and it follows the regular CBL model. It is critical for students to build engagement and excitement. If students are not really connected to the challenge, they should revisit this step; any startup begins from a founder's vision about an idea. If there is not enough excitement about it, students may quickly abandon the project, as it happens in real life [GUP+14].

Once this first step is completed, learners can move to the sprints phase. Sprint length as well as the number of sprints should be defined according to the time and the context of the course. In regards to the length, we suggest a minimum of two-week and a maximum of four-week period. As it can be observed in the framework, each sprint combines activities from the *investigation* and *act* phases. At the beginning of this process, naturally, students need to focus more on investigating the proposed challenge as well as the potential stakeholder affected by it. In order to do so, Customer Development and Lean Startup strategies can be put in place. As students gather the necessary information and begin to validate their hypothesis, they may plan further sprints focusing more on the execution (*act* phase), *i.e.*, delivering software. However, if students fail in this process and are not able

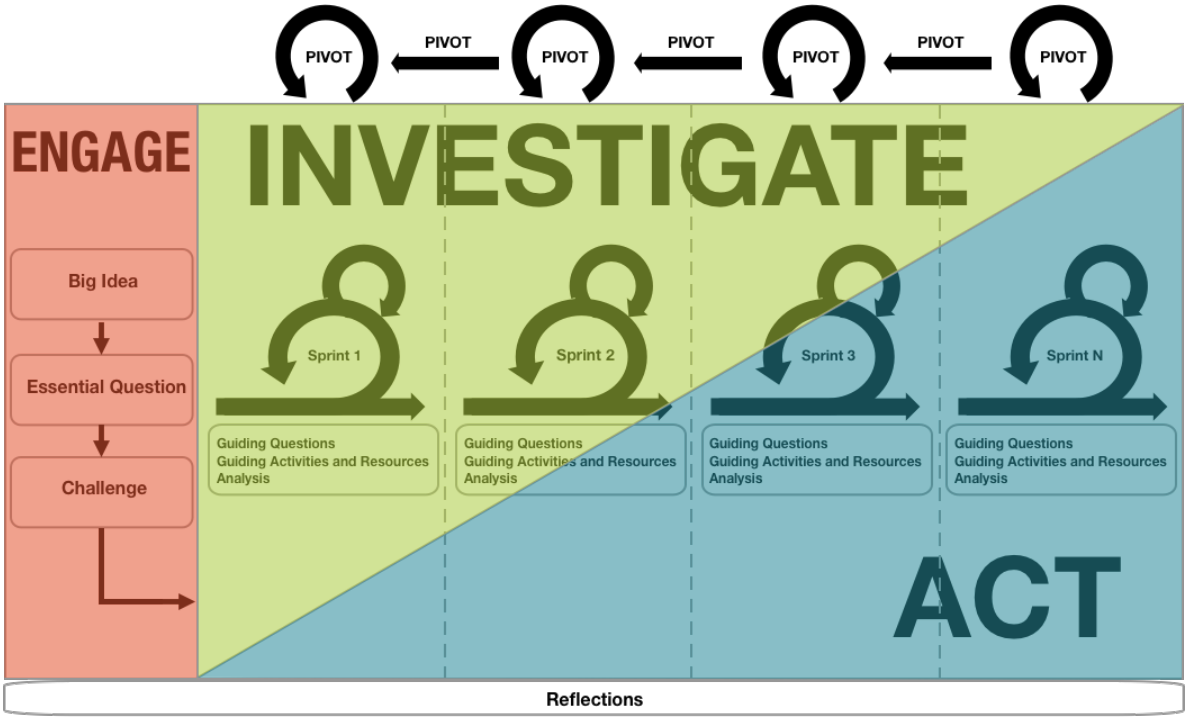

Figure 8.1 – Challenge Based Startup Learning framework.

to validate their assumptions, they should *pivot* [BD12] and remain in the same step until information is validated. It is also possible to move back in the framework in case students understand that more investigation is needed.

In the first sprint, we suggest running interviews in order to gather information regarding the challenge. The goal is to validate whether a group of people carry the same problem. To begin with, potential users/customers need to be defined. Using the concept of personas [Coo99] can help students achieve this task (other strategies to define user/customers are also valid; this is just a suggestion). Once the persona is developed, *guiding questions* and *guiding activities* are designed. The general rules for running interviews are [SRS13]:

- *No leading question*: asking “Do you prefer *A* or *B*?” may lead an interviewee to say *A*, when in fact the person prefers *C*;
- *Ask open-ended questions*: interviewers may find interesting information when asking for stories;
- *No pitching*: it is a moment about learning, not about selling;
- *Ask questions about the past*: people are awful at predicting the future. Past behavior are best predictor of future behavior;

Even though it is not statistically relevant, talking to 20 people should be enough to understand the process and to look for a pattern. If the problem is not validated or no

pattern is found, the team should pivot and redesign the interview process. Since we are talking about a learning environment, it is not wrong to move forward in the framework even if assumptions are not validated as long as both students and teachers are aware of this situation.

For the next sprint, we suggest a value proposition testing. Developing a landing page with a call to action process, structuring a social media page, or creating a video are some of the strategies to be put in place. The goal is to learn whether potential users/customers can be attracted, and if they are somehow interested in the value proposition proposed. A big challenge in this step is to find the right channels to promote the action. Once more, if students fail in this experiment, they should pivot and rethink about the strategy.

Creating and maintaining audience is defined as the next step. Designing a blog or a social media channel are two great ways to do it. The goal is to create content that will attract and engage potential users/customers. By succeeding in this step students will have real people to test and to try out the proposed solution.

It is important to point out that all these Lean Startup and Customer Development strategies can and should be continuously executed. For instance, if a blog is designed, content should be created on a regular basis in order to keep people interested on the topic.

At this point, students should have enough information to create the first prototype or the first version of the solution. Sprints should be organized in such a way that, at the end of them, there is always something to show to potential users. Even if the solution presents bugs or it is not ideally designed, it is better to collect users' feedback as soon as possible than waiting until the software is "perfect" [Rie11].

As it can be observed in the framework, reflections occur throughout the whole process. At the end of each sprint, it is important to stop for a few moments to reflect over the experience. This material should be used by teachers to improve the process as well as to help students overcome problems and difficulties. This process is done by either recording a video or by writing a small essay.

Lecturers must be aware that students might struggle in finding a topic to work on. If that is the case, it is interesting to have a list of big ideas to present to students. In addition, working with pivots can be risky due to time restrictions of a semester. If students need to pivot in one of the proposed steps, we suggest two alternatives: (i) give students more time to work on that step and reduce the time at the end of the semester for the solution development; (ii) explain to students that they should have pivoted if this was a real project. However, they need to move on to the next phase so they can experience the whole startup creation process.

Regarding assessment, the focus is always on the process rather than the final result. In other words, even if students fail in collecting enough responses from interviews,

for instance, they should be graded based on how well the interview was designed and conducted, and not by the number of responses.

8.1 Framework Application and Preliminary Results

The proposed framework was applied in an undergraduate digital entrepreneurship course that lasted one semester. Table 8.1 presents the schedule overview of the course. At the end of each activity, students also deliver a reflection document. The semester started by presenting the basic concepts (Challenge Based Learning, Lean Startup, Customer Development and Scrum). Right after this moment, the class was divided into five teams of three to four students and groups began to work on their *Big Ideas*, *Essential Question*, and *Challenge*. The result of this process is presented in Table 8.2.

Week	**Activity**	**Deliverable**
1	Basic concepts	Exercises
2	Engage phase	Big Idea, Essential Question, and Challenge
3–4	Sprints	Interview results
5–6	Sprints	Value proposition testing results
7–8	Sprints	Content creation
9–12	Sprints	Low-fidelity prototype
13–16	Sprints	High-fidelity prototype

Table 8.1 – Schedule overview.

Once the challenge was defined, teams moved to the sprints phase. Sprint length was set to two weeks so we could have seven sprints in total. In the first sprint, teams designed their strategy to gather information through interviews and observations. Some groups developed online surveys, while others decided to personally talk to potential users. A key learning point was that even though online surveys can bring more data, personal interviews bring more knowledge, since students can interact with people and go to different directions depending on the answers they get. Even though all teams concluded that they needed more data to support their findings, we decided to move to the next experiment so they could learn and experience other techniques.

In the following sprint, teams were required to design a landing page presenting their value proposition with a call to action process (all teams created a form to collect information, such as name and email). Then, teams needed to announce and promote their page in order to test acceptance. Results are presented in Table 8.3. The data in the table shows the number of page views each project obtained and the correspondent conversion

Big Idea	Essencial Question	Challenge
Tourism	What people look for when visiting another country?	Deliver a great experience for people visiting St. Barth.
Charity	What makes people engage in charity events?	Make donation easier for everyone.
Finance	How does the use of cash impact the life of students?	Make payments easier.
International Culture	How does people interact when visiting another country?	Make connections that matter.
Entertainment	What people look for when going out?	Deliver the best venue option according to your taste.

Table 8.2 – Engage phase.

rate (percentage of people that filled out and submitted information through the form). All teams were very disappointed with the results. However, the biggest learning happened here; students realized it is not easy to get someone's attention.

Big Idea	Page Views	Conversion
Tourism	422	2.4%
Charity	276	0.36%
Finance	312	2.6%
International Culture	188	1.6%
Entertainment	205	4.4%

Table 8.3 – Call to action results.

The next sprint was focused on content creation. Teams had to decide on developing either a blog or a Facebook page in order to develop a community around their idea. Although this activity began in the third sprint, teams were asked to generate content until the end of the semester.

The goal of the last four sprints was to develop a working version of the proposed solution in order to collect feedback from users. The backlog for each project was organized is such a way that by the end of the fifth sprint an MVP had to be available in production (a low-fidelity prototype). Three teams (the tourism, the charity and entertainment ones) were able to meet this deadline. The other two groups were only able to deliver their first version of the project on the sixth sprint. Bad backlog management was the cause of such delay. It was no coincidence that the groups that manage to deliver earlier were able to get real users feedback and improved their solutions, while the other two groups did not have time to do so.

8.2 Survey

In order to evaluate the framework, an individual survey was conducted at the end of the *Engage* phase and after sprints one, two, three, five and seven. The main goal was to verify whether students understood the process of creating a real startup. The sample population was the total number of students in the class (18 students), and questions were designed to cover students' perceptions of the whole process as well as of each deliverable (interview process, landing page, content creation, and the solution). Questions were asked on a six–point Likert scale (totally disagree, disagree, somewhat disagree, somewhat agree, agree, and totally agree). The choice for an even-numbered scale was to avoid neutral positions. In addition, we also analyzed the data from students' reflections. At the end of each sprint students were asked to document their perceptions and thoughts.

In regards to the *Engage* phase, 83% of the students totally agreed and 17% agreed that the CBL process helped them finding an engaging challenge to work on. From a startup point of view, this is very important since founders need to share the same vision and need to be equally engaged in the project.

After the first sprint, students were asked a question about the interview process and another one about their motivation so far. In this case, 94% of the students totally agreed that the interview process was very helpful in understanding people's perception over a given problem. Regarding their motivation, 55% agreed and 45% somewhat agreed that they were satisfied and motivated with the results so far. Further analysis indicated that some students were frustrated about the difficulties in getting responses. This point was later explored in the classroom to show students how hard it is to grab people's attention; and running a startup is all about that.

The landing page development activity survey presented interesting results. All students totally agreed that performing a value proposition testing was a great way to play with conversions. Even though conversion rates were below their expectations, motivation and engagement went up at the end of the third sprint.

Regarding content creation, one student somewhat disagreed, ten students somewhat agreed, and seven students agreed that this activity was helpful for the startup process development. At this point, students could not see the value of creating a community (that would be further testing and evaluating their systems).

At the end of the fifth sprint, as already mentioned, three teams managed to have a working version of their proposed solution, while two teams could not meet the deadline. Interestingly, 95% of students totally agreed and 5% agreed that having gone through a few experiments before building the software gave them good insights and information to better define the product backlog as well as to correctly prioritize tasks. All students were highly engaged and motivated by the process.

Finally, the overall perception of the Challenge Based Startup Learning framework was that it gave students a great experience on what it takes to run a real startup. All students totally agreed on that statement. Besides, some students mentioned that it was very engaging to work on projects that can really make a difference in people's lives.

8.3 Challenge Based Startup Learning Final Remarks

In this chapter we presented the Challenge Based Startup Learning, a framework that combines Challenge Based Learning, Lean Startup, Customer Development and software development techniques. Our preliminary results indicate that the framework can help students experience the process of creating a real startup.

This is the initial proposal of the framework. The intention is to introduce the first insights and thoughts regarding our proposition. Further investigation and research still needs to be done in order to test other variables. We are aware that there is not enough data to support the validity of the framework yet.

The experimental study described in this chapter was presented [CSPP18b] at the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITICSE) in 2018.

9. EVALUATION AND EVOLUTION OF THE FRAMEWORK

In order to evaluate and analyze the proposed framework, we decided to run an expert panel. According to Beecham *et al.* [BHB+05], an expert panel is as exploratory study that focuses on analyzing a model, process, method, practice or technique in order to look for strengths, weaknesses, and improvement points. Experts' knowledge and experience bring a lot of value since they can come up with new ideas and thoughts, and they can help researchers avoiding taking the wrong directions [BHB+05].

The selected experts should have previous knowledge of some of the topics related to the research under evaluation. The information gathered from these panels are useful to evolve and to validate models [SC01]. Shepperd and Cartwright [SC01] also mention that an expert panel is a recognized way of performing an initial evaluation of a model. In addition, expert panels are appropriate when evaluating complex or technical contexts that may require a very specific knowledge [Slo03].

The expert panel process was undertaking following the recommendations given by Slocum [Slo03]. We have set three main goals for this research:

- Gather the view of experts about the current Challenge Based Startup Learning framework [CSPP18b];
- Collect suggestions and improvement points for the framework; and
- Propose an evolution of the framework, based on the experts recommendations.

We have selected 14 experts to evaluate the current Challenge Based Startup Learning framework. According to Beecham *et al.* [BHB+05], there is no problem working with a small sample of experts. The goal is not to find statistical explanations, but rather to gain expert feedback on a given context. These experts were chosen due to their previous experience working with Challenge Based Learning, and startups.

9.1 Interview Protocol

The interview was performed face-to-face with each of the experts. At the beginning of each conversation the current model (as presented in Fig. 8.1) was presented and described in details. The paper describing the model [CSPP18b] was also used in order to present an example of a case study.

After the presentation, the following questions were asked for each of the experts:

1. Your location (city/country/institution);

2. Academic experience (in years);
3. Industry experience (in years);
4. Challenge Based Learning experience (in years);
5. Challenge Based Learning knowledge (0 - 5);
6. Lean Startup experience (in years);
7. Lean Startup knowledge (0 - 5);
8. Positive aspects of the proposed framework;
9. Negative aspects of the proposed framework;
10. Improvement opportunities;
11. Suggestions/comments.

The idea behind asking for the expert's knowledge on Challenge Based Learning and Lean Startup on a scale from 0 to 5 was to clearly differentiate experience from knowledge. One may have studied Lean Startup for years, for instance, but may have never applied the methodology in a real startup.

In addition, it is important to point out that all experts involved in this process had at least one year of Challenge Based Learning experience. One might wonder how someone who had never seen Challenge Based Learning before would react to this framework. However, we decided that, for this study, we would only take into consideration experts that have worked with Challenge based Learning.

9.2 Experts Demographics

The details regarding the experts background and experience is presented in Table 9.1. The labels presented in the table are depicted as follows:

- AE: Academic experience (in years):
- IE: Industry experience (in years);
- CBLE: Challenge Based Learning experience (in years);
- CBLK: Challenge Based Learning knowledge (scale from 0 to 5);
- LSE: Lean Startup experience (in years);

Table 9.1 – Experts' background.

Country	AE	IE	CBLE	CBLK	LSE	LSK
Brazil	10	8	1	3	5	3
Brazil	2	3	2	4	3	4
Brazil	11	2	5	5	3	4
Brazil	12	10	2	4	3	4
Brazil	4	6	4	5	2	2
Brazil	5	4	5	4	2	3
Brazil	17	9	4	5	1	2
Brazil	1	8	1	4	3	4
Brazil	4	25	4	4	0	4
Brazil	16	15	5	5	0	1
Brazil	5	4	3	5	2	3
Indonesia	3	17	1	3	0	0
Italy	3	25	3	5	3	3
Italy	3	5	3	5	1	1

- LSK: Lean Startup knowledge (scale from 0 to 5).

There is a lot of diversity across all data presented in Table 9.1. The average academic experience of the group is 7 years, although it varies from 1 to 17 years. When it comes to industry experience, the average is 10 years. Regarding CBL experience, no expert knew CBL for more than 5 years, indicating that it is fairly new concept to them. The same happened to Lean Startup experience, although in this case three experts reported having no experience working with this methodology.

9.3 Expert Panel Results

Following the interview protocol described in Section 9.1 we gathered positive and negative aspects related to the proposed framework, as well as improvement opportunities and other suggestions by using an open coding strategy. The following sections depict the most relevant and important points related to each of these information.

9.3.1 Positive Aspects

To begin with, seven experts mentioned that it is interesting to see how all these processes and methodologies can fit well together. When it comes to the development of innovative projects using CBL, they could see how Lean Startup can really support the process. Since most of the time students do not have experience working with startups, the framework can give them a real feeling of what it takes to develop a startup. In sum, it is a good attempt to build bridges across learning methods that have a lot in common.

Four experts focused their positive feedback on the interview process suggested by the framework. They argued that this was a great idea, since software engineering students, in general, do not have any experience interviewing other people. One of the experts says "*students go to the streets with no background about how to talk to people in order to pull, and not to push information*". Another expert mentioned that "*the framework reflects the natural way to make validation. I liked the interviews*".

Regarding the *build-measure-learn* process, all experts emphasized that the framework gives a lot of room for experimentation, failure and learning. Interestingly, this is exactly what a startup is all about. In this sense, it seems the framework can help students understand in practice the process a startup go through. One expert mentioned that the framework "*provides structure during the investigation/act phase, which is often blurry because CBL is not explicitly tuned to designing a specific product or service*".

Finally, one expert pointed out that this framework can help instructors into motivating students to develop their own startups. Since it is easy to comprehend and to replicate, the framework can also be used as a way to bring more students into the entrepreneurship world.

9.3.2 Negative Aspects

One of the main issue that almost all experts pointed out as a negative aspect is that, depending on the type of project students are working on, it may take time to run experiments. In these situations, the framework should propose or suggest a way to avoid this problem. In addition, if the course is too short (for instance, two months or less), the framework might not work for the same reason: time constraints.

Six experts mentioned that the framework focuses more on agile development (scrum) rather than on Lean Startup. By looking at Fig. 8.1, aside from the *pivots*, in fact there is not any other reference to the Lean Startup methodology.

Regarding the reflections, a few experts did not quite understand its role throughout process. Moreover, it was also not clear to them how much guidance is embedded in the

framework in terms of basic questions anyone should make. For instance, should students come up with guiding questions and activities or this structure would be provided by the instructor?

Half of the experts believed that there could be a risk working with this framework when students do not have prior CBL, Scrum and Lean Startup knowledge. They argued that either students should know them, or these concepts need to be presented and explored in advance.

An interesting point that one expert mentioned is that the framework is limited to four sprints. Even though this is actually not true (instructors can run as many sprints as needed), the framework - as presented today - might lead to this conclusion.

9.3.3 Improvement Opportunities

Several experts mentioned that the framework should guide students into the process. They agree that the proposed framework is a good start, but students might feel lost when they actually need to work on the activities. Suggestions on this issue were related to proposing at least a few guiding questions and activities for each sprint, but students can and should come up with more questions and activities. However, the main ones (such as “who is my customer?”) have to be explicitly presented.

In regards to the *Engage* phase, some experts asked how students get to a challenge. Even though this process is suppose to be similar to the “regular” CBL methodology, experts suggested that tools and methods could be offered to help students into this process. For instance, they could learn how to run a brainstorming session in order to come up with as many ideas as possible, combine them and later agree upon a single one.

When it comes to Lean Startup processes and tools, experts suggested to explicitly present which tools or methods could be used in each part of the process. For instance, if students are validating a value proposition, the framework could suggest the use of a landing page.

One interesting point that was raised by most experts was related to content creation. In the current framework, content creation is performed in one of the sprints. Experts believed that this activity should begin as soon as possible and it should never stop. The sooner students begin to create a relationship with an audience, the sooner they have a chance to test out a hypothesis with real people.

Regarding pivoting, one expert observed that the framework does not allow pivoting during the *Engage* phase. Students need to know that they can change a *challenge*, an *essential question*, or even a *big idea*.

Even though the framework clearly shows that reflections happens throughout the whole process, some experts pointed out that it would be better if there was an explicit reflection moment after the end of each sprint. In this case, students would know that before moving to the next sprint, they have to reflect on the work performed in the previous sprint.

Finally, there was an interesting discussion on defining key achievements or learning goals for each sprint. In other words, how students know they can move to the next sprint? Is it a matter of hypothesis validation, achieving a set of milestones, or both?

9.3.4 Key Findings

After gathering all information from the expert panel, the authors analyzed the data in order to agree upon the main take aways. The goal was to further propose changes and improvements to the current Challenge Based Startup Learning framework.

We came across seven improvement points we believed that could be incorporated or adapted to the current framework in order to better deliver value to students:

1. Make it clear that the framework is not limit to a given number of sprints. Even though the framework depicted in Fig. 8.1 present the last sprint as “N”, indicating that there can be more sprints, we understood this information can be more visible.

2. Guiding questions, activities and resources for each sprint (including the *Engage* phase) should be provided by instructors. Students can and should come up with other guiding questions and activities. However, it is important for them to have at least a few essential questions already defined, specially because in most cases students do not have any background on software startup development.

3. Lean Startup processes and tools could be explicit explored throughout the process. As pointed out by the experts, the framework does not refer to any Lean Startup tool or process aside from the *pivoting*.

4. Content creation can begin as soon as the challenge is defined. The idea is that students will have more chances to interact with real users if they start connecting with them as early as possible. In addition, this process should persevere until the end of the project/course.

5. Allow pivoting during the *Engage* phase. In fact, it is possible to revisit the *Engage* phase in the regular CBL methodology. The idea is to make it clear that students can change their main assumptions.

6. Reflections should happen at the end of each sprint. After running any experiment, it is a good idea to make students reflect on their work. By providing this opportunity at the end of each sprint, students can reflect more often on their learning process.

7. Defined key achievements and/or goals for each sprint. It is important that students know what is expected from them at the end of each sprint. By having clear goals students can create a vision on the path they have to take.

It is worth mentioning that the order of these items is random; we did not intend to prioritize them in any manner. The goal is to make changes in the framework in such a way we can accommodate all seven aforementioned improvements.

9.4 Proposed Framework

The updated overview version of the Challenge Based Startup Learning framework is presented in Fig. 9.1. We tried to translate most of the suggestions from the experts directly into one image. However, we could find a way to include the details regarding guiding questions, activities and resources, and sprint goals.

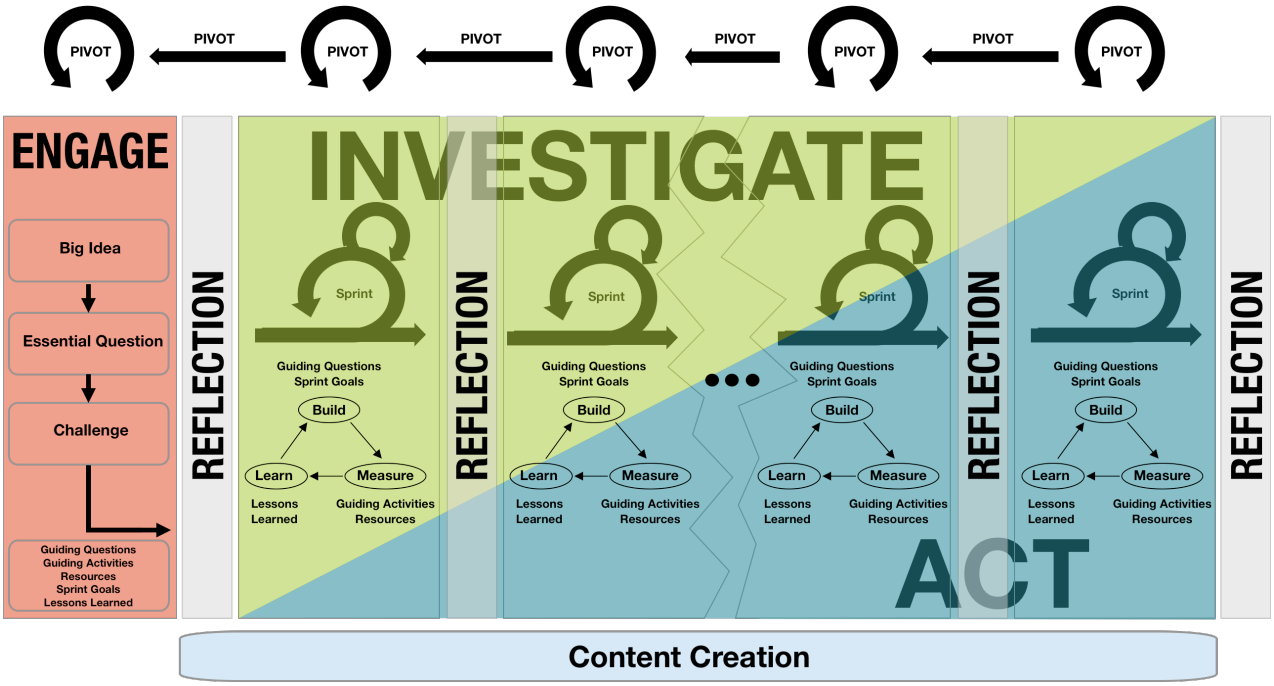

Figure 9.1 – Proposed Challenge Based Startup Learning framework.

Therefore, we decided to add extra layers to the framework in order to better guide students into the process. It is important to mention that all the suggested guiding questions, guiding activities, resources, and sprint goals are just a starting point; they were defined by the authors based on their own experience and the guidelines proposed by Blank [Bla13].

However, students and instructors should add their own ideas according to the context of the class and of the project being developed.

The first layer, presented in Fig. 9.2, entailed the *Engage* phase. The goal is to define a big idea, an essential question and a challenge. In order to do so, it is crucial for students to discuss their passions and the problems they want to solve.

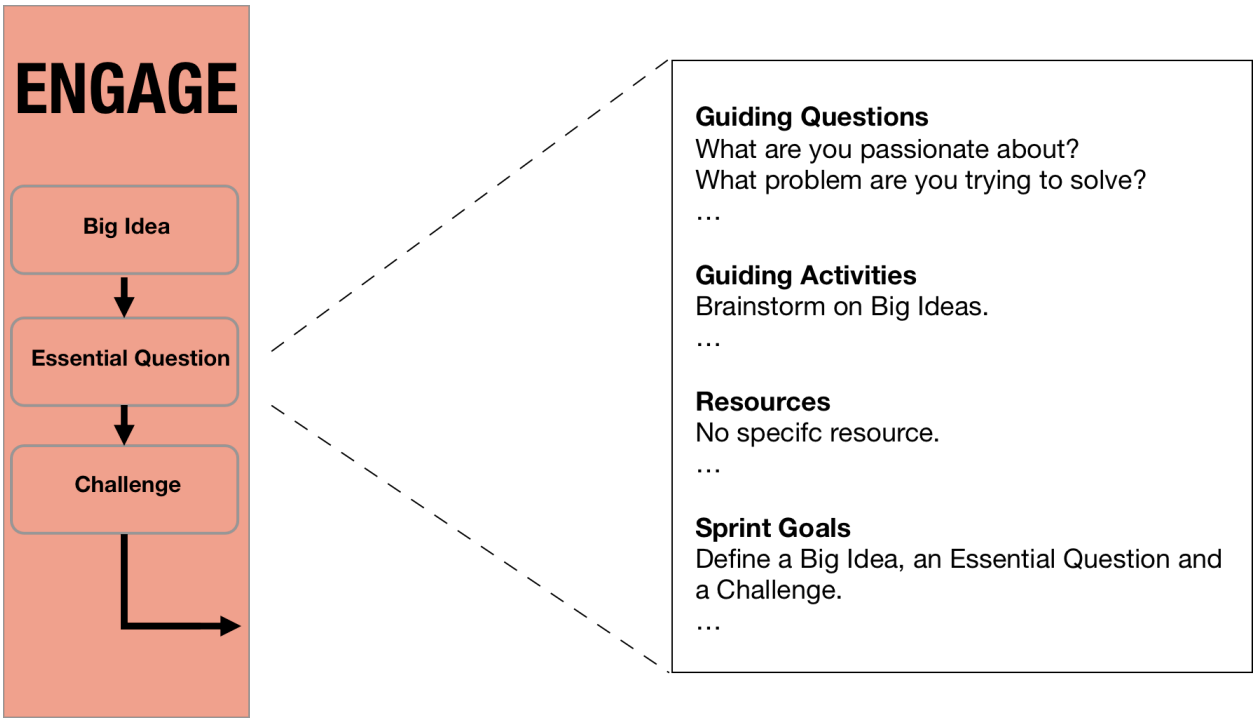

Figure 9.2 – Engage phase.

Once students have the topic they will be working on as well as the challenge, they can begin working on content creation (see Fig. 9.3). This is an ongoing process (see Fig. 9.1), and students should be aware that the sooner they are able to generate content and engage potential users/customers, the better the chances of having real people interacting with them.

Fig. 9.4 entails examples of guiding questions, activities, and resources for the interview sprint. Students should reflect at least on who their customer is, what problems do they have, how they are dealing with these problems today, and where they can find these customers. This is an important step when developing a startup since one of the main reasons they fail, according to Steve Blank, is because founders focus on product development rather than on Customer Development [Bla13].

When it comes to value proposition testing (see Fig. 9.5), students must focus on the benefits they would like to deliver to their customers. Developing a landing page can be an effective way to test it. In addition, students must be aware that it is important to measure customers' interest somehow. This measurement can be performed by asking them to fill out a form or just by providing their emails. In fact, any kind of currency a customer provides is a way to measure their interest.

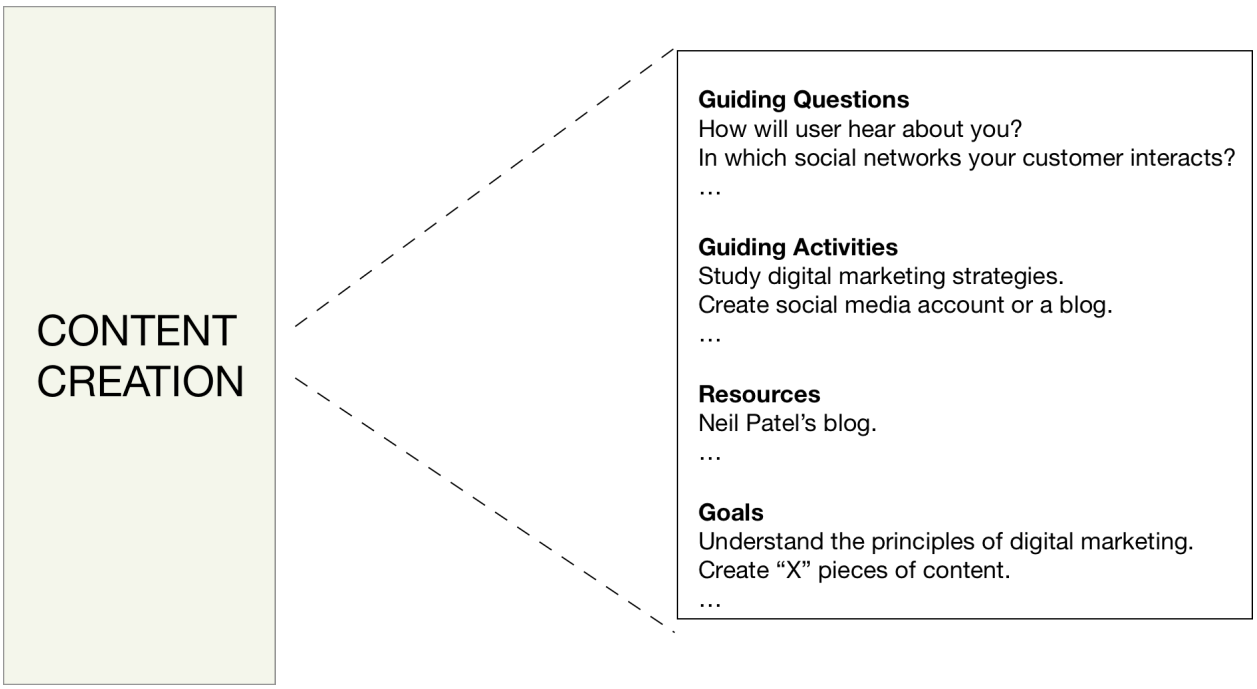

Figure 9.3 – Content creation.

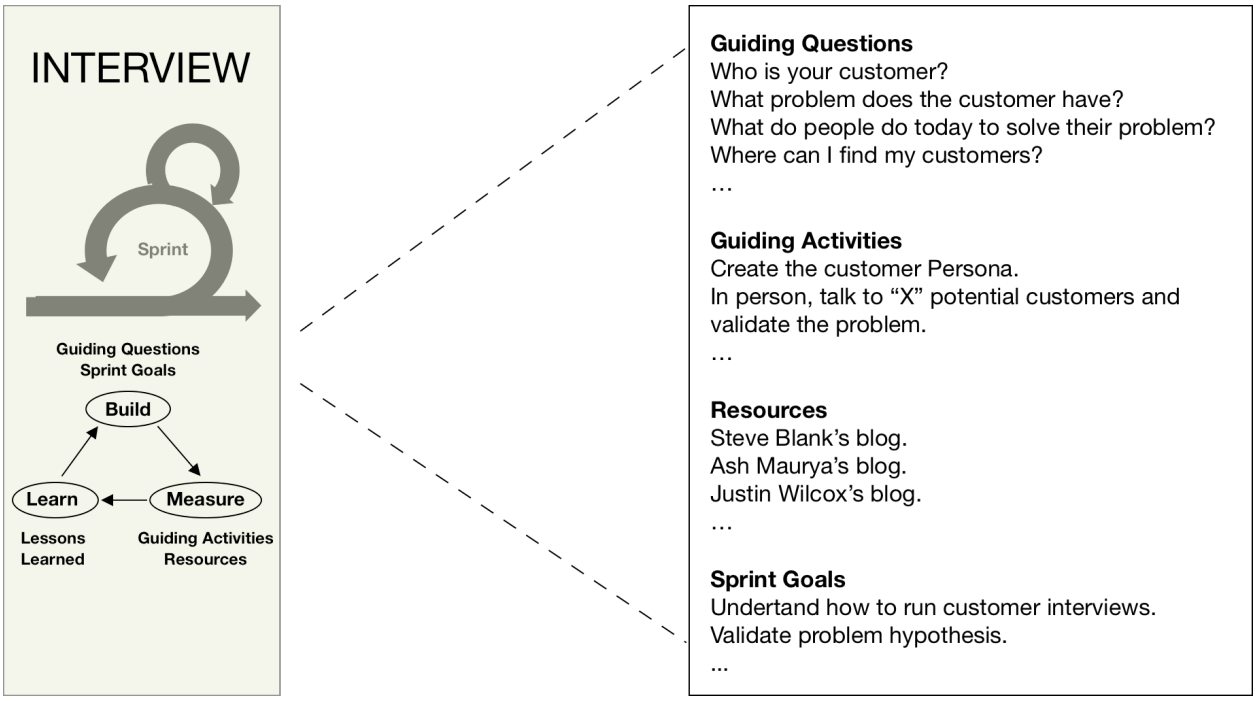

Figure 9.4 – Customer interview.

The prototyping sprint(s), depicted in Fig. 9.6, is the moment where students start thinking about their solution. Since they already gathered a lot of information from running interviews, testing their value proposition, and interacting with potencial customers (through content creation), it is now time to develop prototypes in order to get more feedback from customers. If there is enough time, it is interesting that students develop both a low fidelity

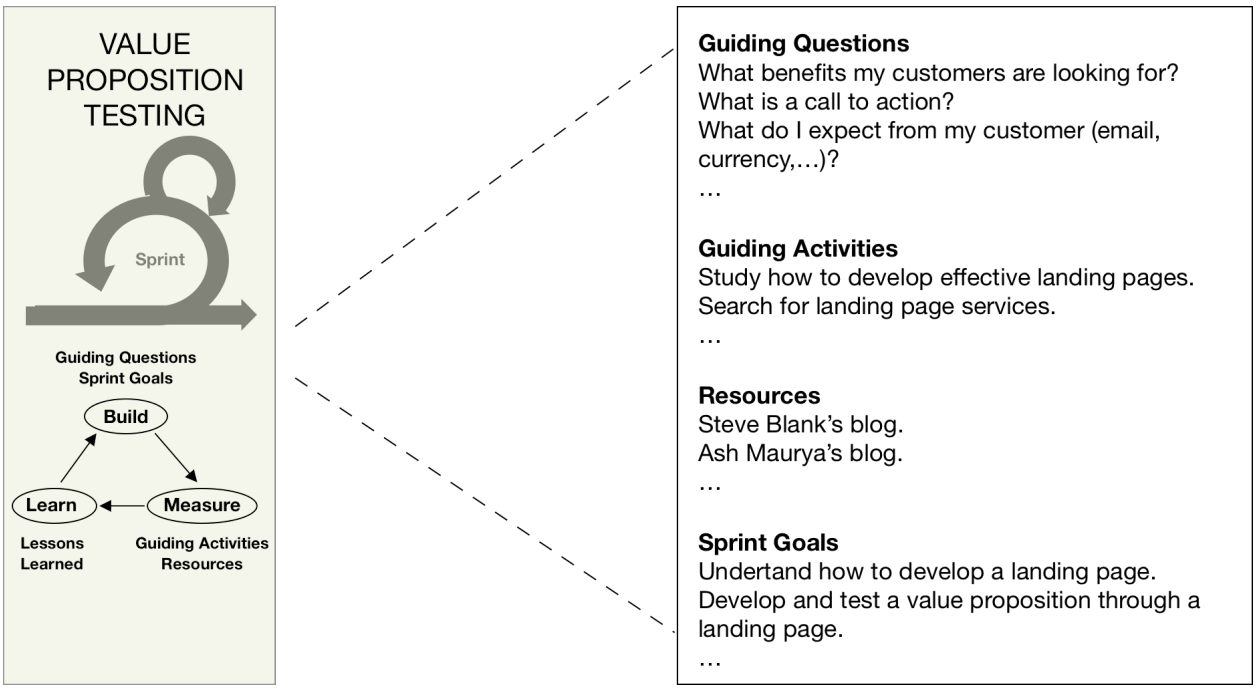

Figure 9.5 – Value proposition testing.

prototype as well as a high fidelity one. By doing so, students will have the opportunity to experience the evolution of the prototype development process.

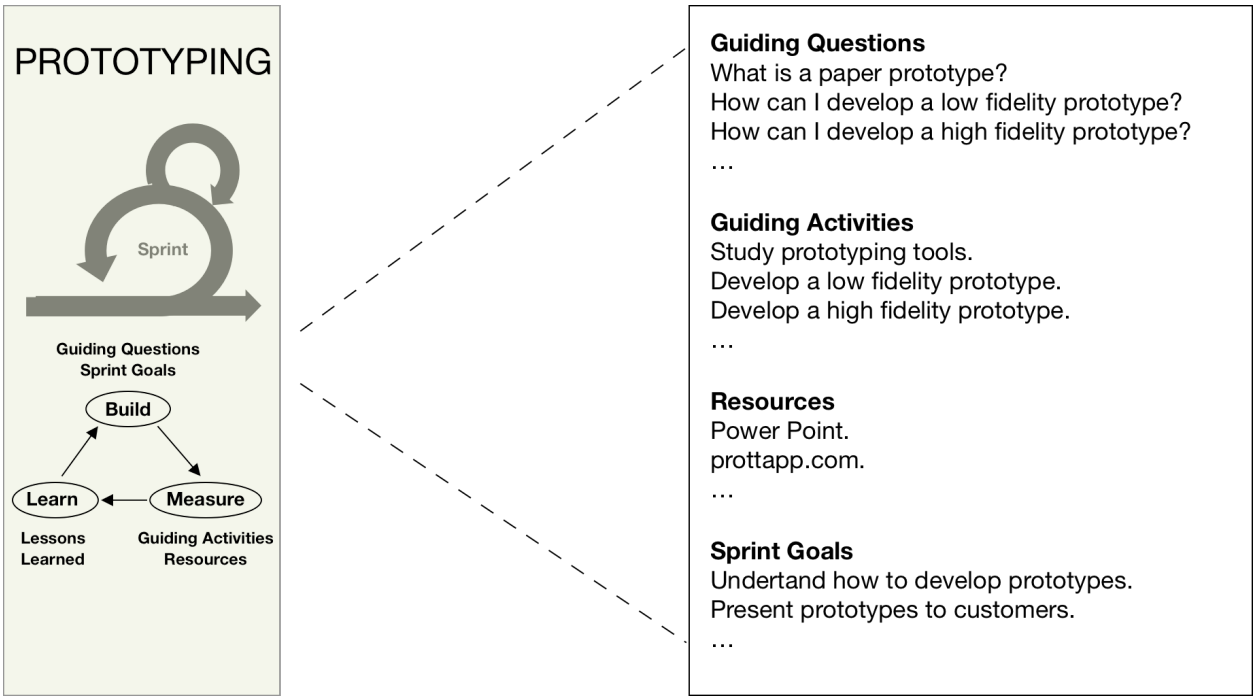

Figure 9.6 – Prototyping.

Finally, when there is an opportunity to actually develop a piece of software based on all knowledge acquired during the process, students can move to the development sprints (see Fig. 9.7). This is the moment where students can not only practice their software development skills, but also their knowledge on software startup processes.

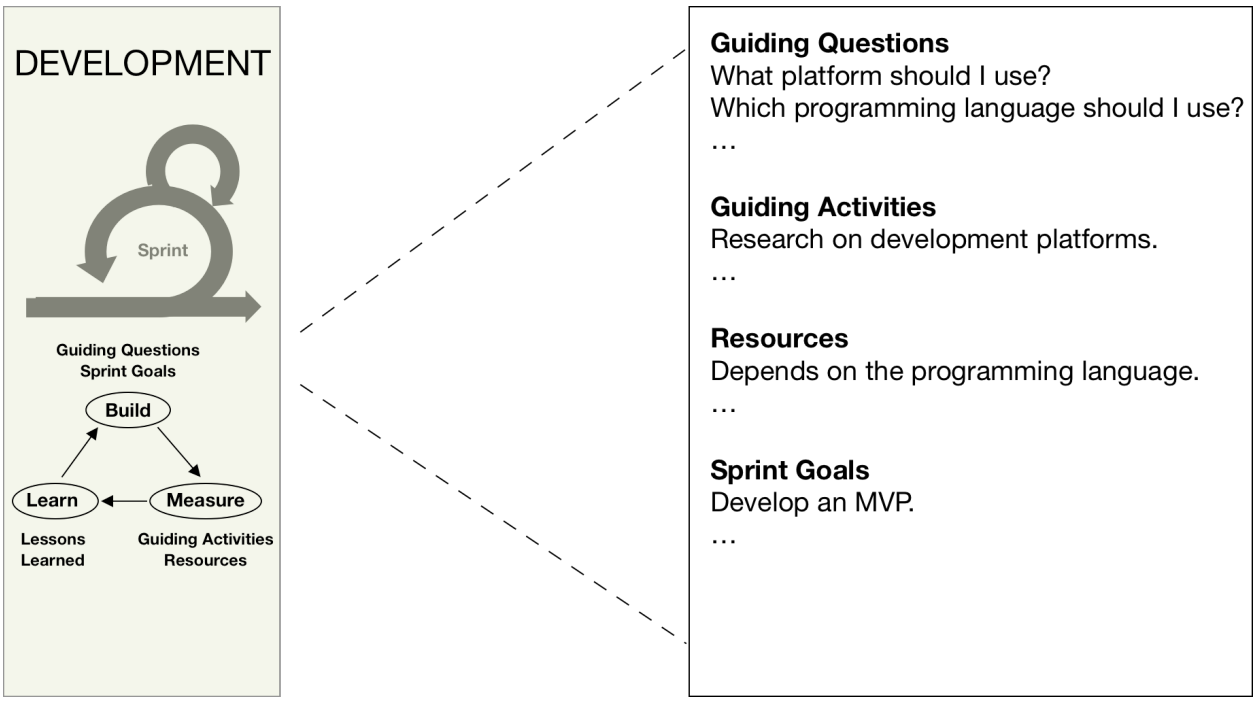

Figure 9.7 – Development.

It is important that instructors guide students into developing software based on the learnings acquired during the previous sprints. As Ash Maurya mentioned in one of his studies [Mau12], it is important to avoid the innovator's bias for the solution. In other words, students (and entrepreneurs) should fall in love with the problem, not the solution.

As a final remark, it is important to mention that all these phases (Engage, Content Creation, Interview, Value Proposition Testing, Prototyping, and Development) are flexible in terms of timeframe; they should all be adapted according to the course context. For instance, if there is no pre-requisite for the course (for instance, programming), maybe instructors should take away the development sprints. In other words, the Challenge Based Startup Learning framework can be seen as building blocks.

Another important point is related to the reflections. As students move forward (or even backwards) into the framework, it is vital that they stop for a moment to think through their learning process. This is a key component of the CBL framework since it helps both students and instructors into adjusting the process on the fly. Additionally, reflections deepen the relationship among students as well as between students and instructors [NCT16].

9.5 Expert Panel Final Remarks and Recommendations for SE Programs

In this chapter we presented an evaluation and evolution of the Challenge Based Startup Learning framework performed through an expert panel. The panel was formed by 14 people from different backgrounds and locations, and they all have previously worked

with the Challenge Based Learning methodology. The process was very interesting since it resulted in great contribution for the future development of the framework.

It is worth mentioning that the framework should be adapted to the context of the educational setting; instructors must be sensitive to the course objectives and goals and use the framework accordingly. For instance, if the instructor would like to use the framework in a software development course, it is important to give students the opportunity to code their solution. On the other hand, if the course has no pre-requisite, it might be a good idea to stop at the prototyping phase (since students may not be coders).

Additionally, when it comes to startup development it is important to embrace the culture of failure and the “love for the problem” mindset. We believe that this framework can help students into understanding that a software startup journey differs from other software development contexts. There are several unknown variables that needed to be addressed along the way in order to increase the chances of success.

In Section 2.3.1 we presented details on how the Brazilian Computer Association suggests the implementation of the software engineering curriculum regarding entrepreneurship [AZN+17]. Moreover, we analyzed 7 software engineering programs in Brazil in order to verify how these guidelines were put into practice. We concluded that even though the document provides great insights and guidance, it does not bring clear examples on how universities can effectively implement those guidelines. After studying seven software engineering programs we could observe and confirm that, in fact, there is no pattern when it comes to providing entrepreneurship and software startups concepts to students.

In this context, we come up with a set of recommendations on how to implement the proposed framework comprising the expected competencies software engineering students must develop according to the software engineering curriculum regarding entrepreneurship [AZN+17]:

1. Courses length: 4-credits (or 60 hours) is the minimum timeframe recommended. Since students must go through the process of developing a real startup, they need time not only to develop the experiments, but also to gather feedback from real users or customers. Therefore, the more time they have to work on their projects, the greater the experience becomes. In a 2-credit course students can go through the content, but there is not enough time to practice the process.

2. Multidiscipline: Whenever possible, courses should be open to students from other fields (such as business, marketing, and design). This brings more value to the process, due to the multidisciplinary nature of software startups.

3. Identify business opportunities: By going through the Engage phase and the Investigation phase (running customers interview, for instance) proposed by the framework, students can not only comprehend the process of identification of business opportunities, but they can also apply this concept in practice. Thus, it is crucial that instructors

provide the opportunity for students to interact with real users so students can validate (or invalidate) the opportunities they envisioned.

4. Create business models, transforming ideas into products or services: the use of tools such as de Business Model Canvas [OP10] or the Lean Canvas [Mau12] help students defining a vision for their business model. These tools are very useful when dealing with technology students (such as software engineering students), since they goes beyond the product and also focus on market aspects.

5. Plan innovative projects: Planning an innovative project (such as a startup) is about learning. The Lean Startup process [Rie11] prescribes that all processes should be geared to accelerate the feedback loop (Build-measure-learn). As it can be observed in the proposed framework, every step is designed to go through these steps. If students invalidate a given hypothesis of their business model, they must pivot in order to create a new experiment. Moreover, after each sprint, students must reflect on their learning. This step is extremely important since it gives students the opportunity to evaluate whether they are absorbing the content.

6. Fundraising for innovative projects: Although the proposed framework does not specifically touch on this topic, it is simple for instructors to cover this content in one class. Topics that can be presented to students: angel investment, incubators and accelerators, and crowdfunding.

7. Manage innovative businesses: the suggested approach is to combine the Customer Development process [BD12] with agile methodologies. As already presented in the proposed framework, it is important that students learn that both product and customers must be developed in parallel. Content creation and value proposition testing are examples of Customer Development approaches, whereas prototyping and the product development itself are examples of product development approaches.

The experimental study described in this chapter was presented at the 10th International Conference on Software Business (ICSOB) in 2019 [CSPP19].

10. CONCLUSIONS

This chapter concludes the research undertaken in this thesis and it also presents directions for future work. In Section 10.1 we review the research objectives of this work. Section 10.2 summarizes the contributions of this thesis. In Section 10.3 we presents suggestions for future work. Finally, in Section 10.4 we list all publications that were published or accepted for publication since the beginning of this research.

10.1 Review of Research Questions and Objectives

The primary research objective of this thesis was to develop a framework to help faculty and the academic community to better deliver software startup content to software engineering students. This goal was derived from the main research questions of this work: *How can we teach software startup related concepts to software engineering students?*.

In order to answer the main research question of this study, we broke down this objetive into the following questions:

- **RQ1**: *What is the state-of-art in literature in regards to software startup education for software engineering students?*
- **RQ2**: *Which tools, models, methodologies and frameworks are applied in a software startup education context for software engineering students?*
- **RQ3**: *What are the reported best practices in regards to teaching software startup for software engineering students?*
- **RQ4**: *What are the benefits and challenges of software engineering students learning the Lean Startup approach?*

Moreover, the following activities were proposed in order to answers the research questions:

- Study software startup related concepts in order to understand them in depth;
- Conduct a systematic mapping study on software startup education for software engineering students;
- Understand how software engineering students perceive software startup concepts;
- Understand how instructors around the world teach software startup to software engineering students;

- Propose a framework/model to teach software startup to software engineering students;
- Evaluate and refine the proposed framework;
- Propose an implementation of the framework in software engineering programs in Brazil;
- Document and report results to the scientific community.

The literature review presented in Chapter 2 gave the author of this thesis the opportunity to study Software Startups, Lean Startup, Customer Development, as well as, methodologies and approaches to better deliver entrepreneurial concepts in a classroom environment. Thus, this step was crucial to the following steps of this research.

The following activity was to conduct a systematic mapping study on software startup education in order to understand the state-of-the-art is this topic, and alto to collect best practices and methodologies used on software startup education for software engineering students. This study was presented in Chapter 4. We have found 39 primary studies and found very interesting insights, such as the difficulty to provide a real world experience to students, and the need to apply an active learning methodology in order to give students the opportunity to try out the process of developing a new startup. This study addressed **RQ1**, **RQ2**, and **RQ3**.

In order to understand how software engineering students perceive software startup concepts, we performed a multi-method study comprised by a survey and by two case studies (see Chapter 5). This study was very interesting since it has indicated that students understood the importance of business concepts, but they struggle to grasp key aspects of the Lean Startup methodology; they tend to over-simplifying them. In addition, students look for a straight path rather than actually going for the build-measure-learn process. This study addressed **RQ4**.

In regards to understanding how instructors around the world teach software startup to software engineering students, we conducted a survey with 15 instructors/professors that teach such courses. The details of this study is depicted in Chapter 6. We have found out that in most cases students are not required to create a real startup; they can develop mock-ups or “toy” projects. Another interesting insight was related to the multidisciplinary aspect of software startups. When courses are opened to students from other fields other than software engineering, the outcome tend to be better. This study addressed **RQ3**.

In order to bring more data to answer **RQ2**, we conducted a case study (presented in Chapter 7), in which we applied the Challenge Based Learning methodology in a software startup course. After this process, there was enough information to propose a first version of the framework that would help instructors when teaching software startup to software engineering students. This framework was called *Challenge Based Startup Learning*, since it is

a combination of the CBL methodology along with Lean Startup and Customer Development concepts. The details of this initial proposal is depicted in Chapter 8.

Finally, the proposed framework was evaluated and refined by running a Expert Panel. This process was very important since it brought great insights from people with a lot of experience in the area. Furthermore, a set of recommendations were designed for those who would like to try out the framework in their academic environment. This whole process is presented in Chapter 9.

By addressing all 4 research questions in the aforementioned studies, we can conclude that the main research question of this thesis - *How can we teach software startup related concepts to software engineering students?* - was also explored and addressed. The creation of the framework, the **Challenge Based Startup Learning**, along with the recommendations on how to implementing it in a software engineering curriculum can help instructors when teaching software startup related concepts to their students.

It is important to mention that the *Challenge Based Startup Learning* framework can be adapted to other educational settings. Even though the framework was developed focusing on software engineering students, faculty can adapt it according to the context. For instance, if a designer instructor wants to use the framework in his/her class, it seems obvious that students will not code their final solution. The last sprints could focus on improving their prototypes. Moreover, this framework can also be adapted for graduate courses; it is just a matter of adapting the framework accordingly.

In addition, one should look at this framework as a guidance for the software startup teaching process. Its flexibility allows faculty to work on a single course with it, but it is also possible to combine courses, and even to develop a whole program based on this framework.

Finally, all studies presented in this thesis were documented and presented to the scientific community. Other studies that were not a part of this work were also undertaken throughout this process. A list of all publications are presented further in this chapter in Section 10.4.

10.2 Contributions of this Work

New startups are created every day; some of them are known for reaching billions of people across the globe. Sadly, most startups do not succeed [GPU+15]. Even though there are many factors that could lead to the failure of a startup, bad software engineering practices is pointed out as a key reason [Col05, GPU+15].

Most universities across the world have already recognized the importance of providing entrepreneurial skills to engineering students [PML+15]. Being technically competent is a must, but it is not enough. Knowing how to develop, market, and sell products and ser-

vices is essencial to survive in the reality we live in. Several institutions are already providing programs and courses focused on entrepreneurship in order to fulfill this need [CCD13].

This work proposed the creation of a framework that can help academic institutions to teach software startup related concepts to their students. In this context, the goal was to provide faculty with the necessary tools to help students to grasp software startup principles and experience the creation and development of a software startup.

Moreover, this thesis contributes to reinforce the importance of multidiscipline in software engineering. A software is only useful when it solves a real problem, or when it improves a given process. Therefore, students must understand that they need to go beyond technical aspects of software engineering, and embracing the whole context.

From a scientific perspective, we hope to foster the discussion as well as the evolution of this topic. As it could be observed in this work (specially in the systematic mapping review), research on software startup education is taken its first steps. There is a good opportunity to further explore the topic.

It is very important to mention the connections and contributions made through the Software Startup Global Research Network (SSRN)[1]. This research group is composed by over 50 people across the globe. In 2017 we were able to bring Prof. Pekka Abrahamsson to Porto Alegre as a visiting professor to teach a software startup graduate course in our Computer Science program. Our partnership with the SSRN has contributed with 9 papers to the scientific community. I would also like to make a special reference to Dr. XiaoFeng Wang and Jorge Melegati, from Free University of Bozen-Bolzano, for having worked closely with us since we joined the network.

10.3 Future Work

This thesis presented several contributions in regards to software startup education. The main contribution, of course, is the Challenge Based Startup Learning framework, which can serve as a guide to enrich and to develop courses that teach student how to create software startups. In this sense, we see a couple of directions for future work.

The first one is the application of the current version of the framework in an academic setting for further evaluation and evolution. Due to the 2020 pandemic and the social distancing required to stay safe, it was not possible to test and to evaluate the framework. When classes went online, the whole context changed and we understood it was risky to try out the framework with all these new variables. In other words, results would have been significantly influenced by this unusual situation.

[1] https://softwarestartups.org/

The second proposal is to study the use of other teaching methodologies combined with software startup concepts. In this thesis, we chose Challenge Based Learning due to the explicit connection that it makes with technology and real world problems (aside from being an active teaching methodology). However, there might be situations in which other methodologies work better. There are other techniques and methodologies, such as Design Thinking or Game-Based Learning, that could be evaluated and may be used depending on the context.

10.4 Publications

Throughout my journey as a PhD student, I worked on several studies that were either published of accepted for publication.

This first set of studies were partially or entirely related to this thesis.

1. Chanin, R.; Melegati, J.; Detoni, M.; Wang, X.; Prikladnicki, R.; Sales, A. “An Analysis of Students’ Perception Towards User Involvement in a Software Engineering Undergraduate Curriculum”. In: Proceedings of the 12th International Conference on Computer Supported Education, 2020, pp. 325–332.

2. Chanin, R.; Sales, A.; Prikladnicki, R. “Software Startup Education: A Transition From Theory to Practice”. In: Fundamentals of Software Startups - Essential Engineering and Business Aspects, Springer, 2020, chap. 13, pp. 217–234.

3. Chanin, R.; Melegati, J.; Sales, A.; Detoni, M.; Wang, X.; Prikladnicki, R. “Incorporating Real Projects Into a Software Engineering Undergraduate Curriculum”. In: Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings, 2019, pp. 250–251.

4. Chanin, R.; Sales, A.; Pompermaier, L. B.; Prikladnicki, R. “Improving a Startup Learning Framework Through an Expert Panel”. In: Proceedings of the 10th International Conference on Software Business, 2019, pp. 306–320.

5. Nascimento, N.; Santos, A. R.; Sales, A.; Chanin, R. “An Investigation of Influencing Factors When Teaching on Active Learning Environments”. In: Proceedings of the 33rd Brazilian Symposium on Software Engineering, 2019, pp. 517–522.

6. Detoni, M.; Sales, A.; Chanin, R.; Villwock, L. H.; Santos, A. R. “Using Challenge Based Learning to Create an Engaging Classroom Environment to Teach Software Startups”. In: Proceedings of the 33rd Brazilian Symposium on Software Engineering, 2019, pp. 547–552.

7. Melegati, J.; Chanin, R.; Wang, X.; Sales, A.; Prikladnicki, R. “Perceived Benefits and Challenges of Learning Startup Methodologies for Software Engineering Students”. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 2019, pp. 204–210.

8. Chanin, R.; Sales, A.; Pompermaier, L.; Prikladnicki, R. “A Systematic Mapping Study on Software Startups Education”. In: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering, 2018, pp. 163–168.

9. Chanin, R.; Sales, A.; Santos, A.; Pompermaier, L.; Prikladnicki, R. “A Collaborative Approach to Teaching Software Startups: Findings From a Study Using Challenge Based Learning”. In: Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering, 2018, pp. 9–12.

10. Chanin, R.; Sales, A.; Pompermaier, L. B.; Prikladnicki, R. “Startup Software Development Education: a Systematic Mapping Study”. In: Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, 2018, pp. 143–144.

11. Chanin, R.; Sales, A.; Pompermaier, L.; Prikladnicki, R. “Challenge Based Startup Learning: A Framework to Teach Software Startup”. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, 2018, pp. 266–271.

12. Chanin, R.; Santos, A.; Nascimento, N.; Sales, A.; Pompermaier, L.; Prikladnicki, R. “Integrating Challenge Based Learning Into a Smart Learning Environment: Findings From a Mobile Application Development Course”. In: Proceedings of the 30th International Conference on Software Engineering and Knowledge Engineering, 2018, pp. 704–706.

13. Chanin, R.; Khanna, D.; Kemell, K.; Xiaofeng, W.; Sales, A.; Prikladnicki, R.; Abrahamsson, P. “Software Startup Education Around the World: A Preliminary Analysis”. In: Proceedings of the 1st International Workshop on Software-intensive Business: Start-ups, Ecosystems and Platforms, 2018, pp. 219–229.

This second set of studies were were not related to this thesis. Nevertheless, I have contributed to each one of them by gathering data, running experiments, or by writing/reviewing them.

1. Cardoso, T. E.; Santos, A. R.; Chanin, R.; Sales, A. “Communication Practices in Continuous Software Development”. In: Proceedings of the 11th International Conference on Software Business, 2020, pp. accepted for publication.

2. Nascimento, N.; Santos, A. R.; Sales, A.; Chanin, R. "Behavior-Driven Development: An Expert Panel to Evaluate Benefits and Challenges". In Proceedings of the 34th Brazilian Symposium on Software Engineering, 2020, pp. accepted for publication.

3. OliveiraJr, E.; Leal, G.; Valente, M.; Morandini, M.; Prikladnicki, R.; Pompermaier, L.; Chanin, R.; Caldeira, C.; Machado, L.; de Souza, C. "Surveying the Impacts of COVID-19 on the Perceived Productivity of Brazilian Software Developers". In Proceedings of the 34th Brazilian Symposium on Software Engineering, 2020, pp. accepted for publication.

4. Melegati, J.; Chanin, R.; Sales, A.; Prikladnicki, R.; Wang, X. “MVP and Experimentation in Software Startups: a Qualitative Survey”. In: Proceedings of the 46th Euromicro Conference on Software Engineering and Advanced Applications, 2020, pp. 322–325.

5. Kemell, K.; Elonen, A.; Suoranta, M.; Nguyen-Duc, A.; Garbajosa, J.; Chanin, R.; Melegati, J.; Rafiq, U.; Aldaeej, A.; Assyne, N.; Sales, A.; Hyrynsalmi, S.; Risku, J.; Edison, H.; Abrahamsson, P. “Business model canvas should pay more attention to the software startup team”. In: Proceedings of the 46th Euromicro Conference on Software Engineering and Advanced Applications, 2020, pp. 342–345.

6. Nascimento, N.; Santos, A. R.; Sales, A.; Chanin, R. “Behavior-Driven Development: A Case Study on its Impacts on Agile Development Teams”. Proceedings of the 13th International Workshop on Cooperative and Human Aspects of Software Engineering, 2020, pp. 109–116.

7. Melegati, J.; Chanin, R.; Sales, A.; Prikladnicki, R. “Towards Specific Software Engineering Practices for Early-Stage Startups”. In: Proceedings of the 21st Agile Processes in Software Engineering and Extreme Programming - Workshops, 2020, pp. 18–22.

8. Chanin, R.; Pompermaier, L. B.; Sales, A.; Prikladnicki, R. “Collaborative Practices for Software Requirements Gathering in Software Startups”. In: Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering, 2019, pp. 31–32.

9. Cardoso, T. E.; Santos, A. R.; Chanin, R.; Sales, A. “Change Management Practices for Continuous Delivery - A Systematic Literature Mapping”. In: Proceedings of the 10th International Conference on Software Business, 2019, pp. 175–182.

10. Pompermaier, L. B.; Chanin, R.; Sales, A.; Prikladnicki, R. “MVP Development Process for Software Startups”. In: Proceedings of the 10th International Conference on Software Business, 2019, pp. 409–412.

11. Melegati, J.; Chanin, R.; Wang, X.; Sales, A.; Prikladnicki, R. “Enablers and Inhibitors of Experimentation in Early-Stage Software Startups”. In: Proceedings of the 20th International Conference on Product-Focused Software Process Improvement, 2019, pp. 554–569.

12. Bajwa, S.; Wang, X.; Duc, A.; Chanin, R.; Prikladnicki, R.and Pompermaier, L.; Abrahamsson, P. “Start-ups must be ready to pivot”, *IEEE Software*, vol. 34–3, 2017, pp. 18–22.

13. Chanin, R.; Pompermaier, L.; Fraga, K.; Sales, A.; Prikladnicki, R. “Applying Customer Development for Software Requirements in a Startup Development Program”. In: Proceedings of the 1st International Workshop on Software Engineering for Startups, 2017, pp. 2–5.

14. Pompermaier, L.; Chanin, R.; Sales, A.; Fraga, K.; Prikladnicki, R. “An Empirical Study on Software Engineering and Software Startups: Findings From Cases in an Innovation Ecosystem”. In: Proceedings of the 29th International Conference on Software Engineering and Knowledge Engineering, 2017, pp. 48–51.

REFERENCES

[AM17] Adorjan, A.; Matturro, G. "'24 Hours of Innovation' - A Report on Students' and Teachers' Perspectives as a Way to Foster Entrepreneurship Competences in Engineering". In: Proceedings of the 1st IEEE World Conference on Engineering Education, 2017, pp. 43–46.

[AZN+17] Araujo, R.; Zorzo, A.; Nunes, D.; Matos, E.; Steinmacher, I.; Leite, J.; Correia, R.; Martins, S. "Referenciais de Formação para os Cursos de Graduação em Computação 2017". Sociedade Brasileira de Computação, 2017, 153p.

[Bab90] Babbie, R. "Survey Research Methods". Wadsworth Publishing Company, 1990, 395p.

[Bar10] Barbe, D. "A Model of Cross Disciplinary Education, Technology Transfer and Teaching Non-Technical Skills for Engineers". In: Proceedings of the 1st IEEE Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments, 2010, pp. 1–32.

[BC06] Braun, V.; Clarke, V. "Using Thematic Analysis in Psychology", *Qualitative Research in Psychology*, vol. 3–2, Jan 2006, pp. 77–101.

[BCC16] Batova, T.; Clark, D.; Card, D. "Challenges of Lean Customer Discovery as Invention". In: Proceedings of the 6th IEEE International Professional Communication Conference, 2016, pp. 1–5.

[BD12] Blank, S.; Dorf, B. "The Startup Owner's Manual: The Step-by-step Guide for Building a Great Company". K&S Ranch, Incorporated, 2012, 571p.

[BdVH13] Breytenbach, J.; de Villiers, C.; Hearn, G. "Directing the South African ICT Labour Force Towards Growth Sectors: A case for Non-Institutional Scarce Skills Transition and Reskilling Courses". In: Proceedings of the 14th International Conference on Information Systems Education and Research, 2013, pp. 1–14.

[BF17] Boutell, M. R.; Fisher, D. S. "Entrepreneurial minded learning in app development courses". In: Proceedings of the 47th Annual Frontiers In Education Conference, 2017, pp. 1–8.

[Bha14] Bharadwaj, A. "An Evaluation of Teaching Theoretical Graduate Engineering Courses Adapting Different Techniques". In: Proceedings of the 2nd IEEE International Conference on MOOC, Innovation and Technology in Education, 2014, pp. 84–88.

[BHB+05] Beecham, S.; Hall, T.; Britton, C.; Cottee, M.; Rainer, A. “Using an Expert Panel to Validate a Requirements Process Improvement Model”, *Journal of Systems and Software*, vol. 76–3, Jun 2005, pp. 251–275.

[Bla13] Blank, S. “The Four Steps to the Epiphany: Successful Strategies for Products That Win”. K&S Ranch, Incorporated, 2013, 370p.

[BNRM17] Binder, F. V.; Nichols, M.; Reinehr, S.; Malucelli, A. “Challenge Based Learning Applied to Mobile Software Development Teaching”. In: Proceedings of the 30th IEEE Conference on Software Engineering Education and Training, 2017, pp. 57–64.

[BNS08] Buckley, M.; Nordlinger, J.; Subramanian, D. “Socially relevant computing”, *ACM Special Interest Group on Computer Science Education Bulletin*, vol. 40–1, Mar 2008, pp. 347–351.

[BOBL13] Bosch, J.; Olsson, H. H.; Björk, J.; Ljungblad, J. “The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups”. In: Proceedings of the 4th International Conference on Lean Enterprise Software and Systems, 2013, pp. 1–15.

[Bos16] Bosch, J. “Speed, Data, and Ecosystems: The Future of Software Engineering”, *IEEE Software*, vol. 33–1, Jan 2016, pp. 82–88.

[BRR17a] Buffardi, K.; Robb, C.; Rahn, D. “Learning Agile with Tech Startup Software Engineering Projects”. In: Proceedings of the 22nd Annual Conference on Innovation and Technology in Computer Science Education, 2017, pp. 28–33.

[BRR17b] Buffardi, K.; Robb, C.; Rahn, D. “Tech Startups: Realistic Software Engineering Projects With Interdisciplinary Collaboration”, *Journal of Computing Sciences in Colleges*, vol. 32–4, Apr 2017, pp. 93–98.

[BTBK08] Budgen, D.; Turner, M.; Brereton, P.; Kitchenham, B. “Using Mapping Studies in Software Engineering”. In: Proceedings of the 20th Annual Workshop of the Psychology of Programming Interest Group, 2008, pp. 195–204.

[BW06] Berglund, H.; Wennberg, K. “Creativity Among Entrepreneurship Students: Comparing Engineering and Business Education”, *International Journal of Continuing Engineering Education and Life Long Learning*, vol. 16–5, Jan 2006, pp. 366–379.

[CCD13] Case, S.; Coleman, M. S.; Deshpande, G. “The Innovative and Entrepreneurial University: Higher Education, Innovation and

Entrepreneurship in Focus", Technical Report, US Department of Commerce, Economic Development Administration, 2013, 102p.

[CD11] Cruzes, D. S.; Dyba, T. "Recommended Steps for Thematic Synthesis in Software Engineering". In: Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, 2011, pp. 275–284.

[CDK11] Currie, E.; Doboli, S.; Kamberova, G. "Developing the next generation of entrepreneurs". In: Proceedings of the 41st Annual Frontiers In Education Conference, 2011, pp. 1–6.

[Che08] Chenoweth, S. "Undergraduate Software Engineering Students in Startup Businesses". In: Proceedings of the 21st IEEE Conference on Software Engineering Education and Training, 2008, pp. 118–125.

[Che14] Chesney, D. "Social Context, Singular Focus". In: Proceedings of the 44th Annual Frontiers In Education Conference, 2014, pp. 1–6.

[CKK+18] Chanin, R.; Khanna, D.; Kemell, K.; Xiaofeng, W.; Sales, A.; Prikladnicki, R.; Abrahamsson, P. "Software Startup Education Around the World: A Preliminary Analysis". In: Proceedings of the 1st International Workshop on Software-intensive Business: Start-ups, Ecosystems and Platforms, 2018, pp. 219–229.

[Col05] Coleman, G. "An Empirical Study of Software Process in Practice". In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005, pp. 1–6.

[Coo99] Cooper, A. "The Inmates Are Running the Asylum". Macmillan Publishing Co., Inc., 1999, 261p.

[CPF+17] Chanin, R.; Pompermaier, L.; Fraga, K.; Sales, A.; Prikladnicki, R. "Applying Customer Development for Software Requirements in a Startup Development Program". In: Proceedings of the 1st IEEE/ACM International Workshop on Software Engineering for Startups, 2017, pp. 2–5.

[Cro02] Crowne, M. "Why Software Product Startups Fail And What to do About it. Evolution of Software Product Development in Startup Companies", *IEEE International Engineering Management Conference*, vol. 1, Aug 2002, pp. 338–343.

[CSPP18a] Chanin, R.; Sales, A.; Pompermaier, L.; Prikladnicki, R. "A Systematic Mapping Study On Software Startups Education". In: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering, 2018, pp. 163–168.

[CSPP18b] Chanin, R.; Sales, A.; Pompermaier, L.; Prikladnicki, R. “Challenge Based Startup Learning: A Framework to Teach Software Startup”. In: Proceedings of the 23nd Annual Conference on Innovation and Technology in Computer Science Education, 2018, pp. 266–271.

[CSPP19] Chanin, R.; Sales, A.; Pompermaier, L. B.; Prikladnicki, R. “Improving a Startup Learning Framework Through an Expert Panel”. In: Proceedings of the 10th International Conference on Software Business, 2019, pp. 306–320.

[CSS+18] Chanin, R.; Sales, A.; Santos, A.; Pompermaier, L.; Prikladnicki, R. “A Collaborative Approach to Teaching Software Startups: Findings from a Study Using Challenge Based Learning”. In: Proceedings of the 11th Workshop on Cooperative and Human Aspects of Software Engineering, 2018, pp. 9–12.

[dCA13] da Cruz, E. F. Z.; Alvaro, A. “Introduction of Entrepreneurship And Innovation Subjects in a Computer Science Course in Brazil”. In: Proceedings of the 43rd Annual Frontiers In Education Conference, 2013, pp. 1881–1887.

[Dev17] Devadiga, N. M. “Software Engineering Education: Converging with the Startup Industry”. In: Proceedings of the 30th IEEE Conference on Software Engineering Education and Training, 2017, pp. 192–196.

[DKI+10] Doboli, S.; Kamberova, G. L.; Impagliazzo, J.; Fu, X.; Currie, E. “A Model of Entrepreneurship Education for Computer Science And Computer Engineering Students”. In: Proceedings of the 40th Annual Frontiers In Education Conference, 2010, pp. 1–6.

[dLNKK16] de Lange, P.; Nicolaescu, P.; Klamma, R.; Koren, I. “DevOpsUse for Rapid Training of Agile Practices Within Undergraduate and Startup Communities”. In: Proceedings of the 11th European Conference on Technology Enhanced Learning, 2016, pp. 570–574.

[DR08] Daimi, K.; Rayess, N. “The Role of Software Entrepreneurship in Computer Science Curriculum”. In: Proceedings of the 38th Annual Frontiers In Education Conference, 2008, pp. 332–338.

[DSC+19] Detoni, M.; Sales, A.; Chanin, R.; Villwock, L. H.; Santos, A. R. “Using Challenge Based Learning to Create an Engaging Classroom Environment to Teach Software Startups”. In: Proceedings of the 33rd Brazilian Symposium on Software Engineering, 2019, pp. 547–552.

[DSVDBS05] Dochy, F.; Segers, M.; Van Den Bossche, P.; Struyven, K. “Students’ Perceptions of a Problem-Based Learning Environment”, *Learning Environments Research*, vol. 8–1, Jan 2005, pp. 41–66.

[Eng14] Engelsma, J. “Best Practices for Industry-Sponsored CS Capstone Courses”, *Journal of Computing Sciences in Colleges*, vol. 30–1, Oct 2014, pp. 18–28.

[ESRTCBS09] Edwards, M.; Sánchez-Ruiz, L. M.; Tovar-Caro, E.; Ballester-Sarrias, E. “Engineering Students’ Perceptions of Innovation And Entrepreneurship Competences”. In: Proceedings of the 39th Annual Frontiers In Education Conference, 2009, pp. 1–5.

[ESSD08] Easterbrook, S.; Singer, J.; Storey, M.-A.; Damian, D. “Selecting Empirical Methods for Software Engineering Research”. Springer London, 2008, chap. 11, pp. 285–311.

[FB17] Frederiksen, D.; Brem, A. “How Do Entrepreneurs Think They Create Value? A Scientific Reflection of Eric Ries’ Lean Startup Approach”, *International Entrepreneurship and Management Journal*, vol. 13–1, Oct 2017, pp. 169–189.

[FF09] Fetaji, M.; Fetaji, B. “Analyses of Mobile Learning Software Solution in Education Using the Task Based Learning Approach”. In: Proceedings of the 31st International Conference on Information Technology Interfaces, 2009, pp. 373–378.

[FGW04] Ford, R.; Goodrich, J.; Weissbach, R. “A Multidisciplinary Business And Engineering Course in Product Development And Entrepreneurship”. In: Proceedings of the 34th Annual Frontiers In Education Conference, 2004, pp. 1–6.

[FHL+17] Fagerholm, F.; Hellas, A.; Luukkainen, M.; Kyllönen, K.; Yaman, S.; Mäenpää, H. “Patterns for designing and implementing an environment for software start-up education”. In: Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications, 2017, pp. 133–140.

[FS17] Fitzgerald, B.; Stol, K. “Continuous Software Engineering: A Roadmap and Agenda”, *Journal of Systems and Software*, vol. 123, Jan 2017, pp. 176–189.

[FW03] Fernandez, E.; Williamson, D. “Using Project-Based Learning to Teach Object Oriented Application Development”. In: Proceedings of the 4th Conference on Information Technology Curriculum, 2003, pp. 37–40.

[GCRB15] Ghezzi, A.; Cavallaro, A.; Rangone, A.; Balocco, R. “A Comparative Study on the Impact of Business Model Design & Lean Startup Approach versus Traditional Business Plan on Mobile Startups Performance”. In: Proceedings of the 17th International Conference on Enterprise Information Systems, 2015, pp. 196–203.

[GG17] Génova, G.; González, M. “Educational Encounters of the Third Kind”, *Science and Engineering Ethics*, vol. 23–6, Dec 2017, pp. 1791–1800.

[GHK97] Gorman, G.; Hanlon, D.; King, W. “Some Research Perspectives on Entrepreneurship Education, Enterprise Education and Education for Small Business Management: A Ten-Year Literature Review”, *International Small Business Journal*, vol. 15–3, Apr 1997, pp. 56–77.

[GPU+15] Giardino, C.; Paternoster, N.; Unterkalmsteiner, M.; Gorschek, T.; Abrahamsson, P. “Software Development in Startup Companies: The Greenfield Startup Model”, *IEEE Transactions on Software Engineering*, vol. 42–6, Dec 2015, pp. 585–604.

[Gro00] Gross, W. “An Approach to Teaching Entrepreneurship to Engineers”. In: Proceedings of the 6th IEEE Engineering Management Society, 2000, pp. 648–652.

[GSX15] Gary, K. A.; Sohoni, S.; Xavier, S. “Pre-Conference Workshop: Agile Teaching And Learning”. In: Proceedings of the 35th Annual Frontiers In Education Conference, 2015, pp. 1–2.

[GUP+14] Giardino, C.; Unterkalmsteiner, M.; Paternoster, N.; Gorschek, T.; Abrahamsson, P. “What Do We Know About Software Development in Startups?”, *IEEE Software*, vol. 31–5, Sep 2014, pp. 28–32.

[GWA14] Giardino, C.; Wang, X.; Abrahamsson, P. “Why Early-Stage Software Startups Fail: A Behavioral Framework”. In: Proceedings of the 5th International Conference on Software Business, 2014, pp. 27–41.

[Har15] Harms, R. “Self-Regulated Learning, Team Learning And Project Performance in Entrepreneurship Education: Learning in a Lean Startup Environment”, *Technological Forecasting and Social Change*, vol. 100, Nov 2015, pp. 21–28.

[HHO03] Hilmola, O.; Helo, P.; Ojala, L. “The Value of Product Development Lead Time in Software Startup”, *System Dynamics Review*, vol. 19–1, Jan 2003, pp. 75–82.

[HK14] Heintz, F.; Klein, K. I. “The Design of Sweden’s First 5-year Computer Science And Software Engineering Program”. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education, 2014, pp. 199–204.

[İB13] İnel, D.; Balım, A. “Concept Cartoons Assisted Problem Based Learning Method in Science and Technology Teaching and Students’ Views”, *Procedia - Social and Behavioral Sciences*, vol. 93, Oct 2013, pp. 376–380.

[Inc08] Inc., A. “Apple Classrooms of Tomorrow - Today”, Technical Report, Apple Inc., 2008, 41p.

[Inc09] Inc., A. “Challenge Based Learning: A Classroom Guide”, Technical Report, Apple Inc., 2009, 40p.

[IPSL09] Ikonen, A.; Piironen, A.; Saurén, K.; Lankinen, P. “Cdio Concept in Challenge Based Learning”. In: Proceedings of the 4th Workshop on Embedded Systems Education, 2009, pp. 27–32.

[ITOGI16] Izurieta, C.; Trenk, M.; O’Bleness, M.; Gunderson-Izurieta, S. “The Effectiveness of Software Development Instruction through the Software Factory Method for High School Students”. In: Proceedings of the 123rd American Society for Engineering and Education Annual Conference and Exposition, 2016, pp. 26–29.

[JA11] Johnson, L.; Adams, S. “Challenge Based Learning: The Report from the Implementation Project”, Technical Report, The New Media Consortium, 2011, 39p.

[Jos06] Joseph, A. “Interdisciplinarity, Financial Software Product Development, And Entrepreneurship in an Urban University”, *American Society for Engineering Education*, vol. 11–1, Jan 2006, pp. 812.1–812.13.

[JTH15] Järvi, A.; Taajamaa, V.; Hyrynsalmi, S. “Lean Software Startup – An Experience Report from an Entrepreneurial Software Business Course”. In: Proceedings of the 6th International Conference on Software Business, 2015, pp. 230–244.

[KAP+06] Kontio, J.; Ahokas, M.; Poyry, P.; Warsta, J.; Makela, M. M.; Tyrvainen, P. “Software Business Education for Software Engineers: Towards an Integrated Curriculum”. In: Proceedings of the 19th Conference on Software Engineering Education and Training Workshops, 2006, pp. 4–7.

[KBB11] Kitchenham, B.; Budgen, D.; Brereton, O. “Using Mapping Studies as the Basis Ror Further Research - A Participant-Observer Case Study”, *Information and Software Technology*, vol. 53–6, Jun 2011, pp. 638–651.

[KC07] Kitchenham, B.; Charters, S. “Guidelines For Performing Systematic Literature Reviews in Software Engineering”, Technical Report, Keele University and Durham University Joint Report, 2007, 57p.

[KCM+14] Kon, F.; Cukier, D.; Melo, C.; Hazzan, O.; Yuklea, H. “A Panorama of the Israeli Software Startup Ecosystem”, Technical Report, SSRN, 2014, 28p.

[KHH13] Kaltenecker, N.; Hoerndlein, C.; Hess, T. “The Drivers of Entrepreneurial Intentions - An Empirical Study Among Information Systems And Computer Science Students”. In: Proceedings of the 19th Americas Conference of Information Systems, 2013, pp. 1–8.

[KMN08] Kajko-Mattsson, M.; Nikitina, N. “From Knowing Nothing to Knowing a Little: Experiences Gained From Process Improvement in a Start-Up Company”. In: Proceedings of the 1st International Conference on Computer Science and Software Engineering, 2008, pp. 617–621.

[Ko17] Ko, A. J. “A Three-Year Participant Observation of Software Startup Software Evolution”. In: Proceedings of the 39th International Conference on Software Engineering, 2017, pp. 3–12.

[KPP95] Kitchenham, B.; Pickard, L.; Pfleeger, S. “Case Studies for Method and Tool Evaluation”, *IEEE Software*, vol. 12–4, Jul 1995, pp. 52–62.

[Li07] Li, S.-C. “The Role of Value Proposition and Value Co-Production in New Internet Startups: How New Venture e-Businesses Achieve Competitive Advantage”. In: Proceedings of the 7th Portland International Center for Management of Engineering and Technology Conference, 2007, pp. 1126–1132.

[LSS05] Lethbridge, T.; Sim, S.; Singer, J. “Studying Software Engineers: Data Collection Techniques for Software Field Studies”, *Empirical Software Engineering*, vol. 10–3, Jul 2005, pp. 311–341.

[LT16] Lenarduzzi, V.; Taibi, D. “Mvp explained: A systematic mapping study on the definitions of minimal viable product”. In: Proceedings of the 42th Euromicro Conference on Software Engineering and Advanced Applications, 2016, pp. 112–119.

[Mau12] Maurya, A. “Running Lean: Iterate From Plan A to a Plan That Works”. O’Reilly Media, Inc., 2012, 207p.

[MBT06] Mafra, S. N.; Barcelos, R. F.; Travassos, G. H. “Aplicando uma Metodologia Baseada em Evidência na Definição de Novas Tecnologias de Software”. In: Proceedings of the 20th Brazilian Symposium on Software Engineering, 2006, pp. 239–254.

[McM14] McMahon, E. “From Product Development to Innovation”. In: Proceedings of the 35th American Society for Engineering Management International Annual Conference, 2014, pp. 118–127.

[MCW+19] Melegati, J.; Chanin, R.; Wang, X.; Sales, A.; Prikladnicki, R. “Perceived Benefits and Challenges of Learning Startup Methodologies for Software Engineering Students”. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 2019, pp. 204–210.

[MG09] Marcangelo, C.; Gibbon, C. “Problem based learning evaluation toolkit”, Technical Report, Higher Education Academy Health Sciences and Practice Subject Centre, 2009, 6p.

[NB11] Nurkkala, T.; Brandle, S. “Software Studio: Teaching Professional Software Engineering”. In: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, 2011, pp. 153–158.

[NC08] Nichols, M.; Cator, K. “Challenge Based Learning White Paper”, Technical Report, Apple Inc., 2008, 23p.

[NCT16] Nichols, M.; Cator, K.; Torres, M. “Challenge Based Learning Guide”. Digital Promise, 2016, 59p.

[NDSA15] Nguyen-Duc, A.; Seppänen, P.; Abrahamsson, P. “Hunter-gatherer Cycle: A Conceptual Model of the Evolution of Software Startups”. In: Proceedings of the 9th International Conference on Software and System Process, 2015, pp. 199–203.

[NDSA16] Nguyen-Duc, A.; Shah, S.; Ambrahamsson, P. “Towards an Early Stage Software Startups Evolution Model”. In: Proceedings of the 42nd Euromicro Conference on Software Engineering and Advanced Applications, 2016, pp. 120–127.

[Nob11] Nobel, C. “Teaching a ’Lean Startup’ Strategy”. URL https://https://hbswk.hbs.edu/item/teaching-a-lean-startup-strategy, Accessed: 2020-10-10.

[NYN16] Nunes, D. J.; Yamaguti, M.; Nunes, I. “Refinamento de Competências do Egresso do Curso de Engenharia de Software”. In: Proceedings of the 9th Education in Software Engineering Forum, 2016, pp. 143–155.

[OP10] Osterwalder, A.; Pigneur, Y. “Business Model Generation: A Handbook For Visionaries, Game Changers, And Challengers”. John Wiley & Sons, 2010, 278p.

[Par14] Pardo, A. “Problem-Based Learning Combined With Project-Based Learning: A Pilot Application in Digital Signal Processing”. In: Proceedings of the 11th Technologies Applied to Electronics Teaching Conference, 2014, pp. 1–5.

[PFMM08] Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. “Systematic Mapping Studies in Software Engineering”. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, 2008, pp. 68–77.

[PG12] Pauca, V.; Guy, R. “Mobile Apps for the Greater Good: A Socially Relevant Approach to Software Engineering”. In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, 2012, pp. 535–540.

[PGU+14] Paternoster, N.; Giardino, C.; Unterkalmsteiner, M.; Gorschek, T.; Abrahamsson, P. “Software Development in Startup Companies: A Systematic Mapping Study”, *Information and Software Technology*, vol. 56–10, Oct 2014, pp. 1200–1218.

[PK01] Pfleeger, S.; Kitchenham, B. “Principles of Survey Research: Part 1: Turning Lemons into Lemonade”, *ACM Special Interest Group on Software Engineering Software Engineering Notes*, vol. 26–6, Nov 2001, pp. 16–18.

[PLB08] Pauli, J.; Lawrence, T.; Brown, B. “Development of a New Software Product From a Classroom Project”. In: Proceedings of the 5th International Conference on Information Technology: New Generations, 2008, pp. 97–100.

[PML+15] Porter, J.; Morgan, J.; Lester, R.; Steele, A.; Vanegas, J.; Hill, R. “A Course in Innovative Product Design: A Collaboration Between Architecture, Business, And Engineering”. In: Proceedings of the 45th Annual Frontiers In Education Conference, 2015, pp. 1–5.

[PPV00] Perry, D.; Porter, A.; Votta, L. “Empirical Studies of Software Engineering: A Roadmap”. In: Proceedings of the 22nd International Conference on Software Engineering, 2000, pp. 345–355.

[QSEMD+18] Quezada-Sarmiento, P. A.; Enciso, L.; Mayorga-Diaz, M. P.; Mengual-Ándres, S.; Hernandez, W.; Vivanco-Ochoa, J. V.; Carrión, P. V. “Promoting Innovation And Entrepreneurship Skills in Professionals in

Software Engineering Training: An Approach to the Academy And Bodies of Knowledge Context". In: Proceedings of the 9th IEEE Global Engineering Education Conference, 2018, pp. 796–799.

[RAF16] Ribeiro, C.; Aleixo, F.; Freire, M. "Driving Academic Spin-off by Software Development Process: A Case Study in Federal Institute of Rio Grande do Norte-Brazil". In: Proceedings of the 17th International Conference on Product-Focused Software Process Improvement, 2016, pp. 636–639.

[RC08] Rubin, J.; Chisnell, D. "Handbook of Usability Testing: How to Plan, Design And Conduct Effective Tests". John Wiley & Sons, 2008, 348p.

[RH08] Runeson, P.; Höst, M. "Guidelines for Conducting And Reporting Case Study Research in Software Engineering", *Empirical Software Engineering*, vol. 14–2, Dec 2008, pp. 131–164.

[Rie11] Ries, E. "The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses". Crown Business, 2011, 320p.

[RMEDJ14] Rio, C. R. D.; Morgado-Estevez, A.; Dominguez-Jimenez, J. "Entrepreneurship And Lean Manufacturing For Software Engineering". In: Proceedings of the 42nd European Society for Engineering Education Annual Conference, 2014, pp. 1–8.

[Sal17] Salas, R. P. "Teaching Entrepreneurship in Computer Science: Lessons Learned". In: Proceedings of the 47th Annual Frontiers In Education Conference, 2017, pp. 1–7.

[San16] dos Santos, A. R. "Um Método de Aprendizagem Baseada em Desafios: Um Estudo de Caso em Ambientes de Desenvolvimento de Aplicativos", Ph.D. Thesis, PUCRS, 2016, 179p.

[SB01] Schwaber, K.; Beedle, M. "Agile Software Development with Scrum". Prentice Hall PTR, 2001, 158p.

[SC01] Shepperd, M.; Cartwright, M. "Predicting With Sparse Data", *IEEE Transactions on Software Engineering*, vol. 27–11, Jan 2001, pp. 987–998.

[SCC98] Strauss, A.; Corbin, J.; Corbin, J. "Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory". SAGE Publications, 1998, 312p.

[SCT01] Shull, F.; Carver, J.; Travassos, G. H. "An Empirical Methodology for Introducing Software Processes", *ACM Special Interest Group on Software*

Engineering Software Engineering Notes, vol. 26–5, Sep. 2001, pp. 288–296.

[SDJ07] Sjøberg, D. I. K.; Dybå, T.; Jørgensen, M. “The Future of Empirical Methods in Software Engineering Research”. In: Proceedings of the 1st Future of Software Engineering Workshop, 2007, pp. 358–378.

[SFMJ+16] Sarraipa, J.; Ferreira, F.; Marcelino-Jesus, E.; Artifice, A.; Lima, C.; Kaddar, M. “Technological Innovations Tackling Students Dropout”. In: Proceedings of the 7th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion, 2016, pp. 112–118.

[Sha00] Shaw, M. “Software Engineering Education”. In: Proceedings of the 22nd International Conference on Software Engineering, 2000, pp. 371–380.

[Sha03] Shaw, M. “Writing Good Software Engineering Research Papers: Minitutorial”. In: Proceedings of the 25th International Conference on Software Engineering, 2003, pp. 726–736.

[SHH+05] Sjoeberg, D.; Hannay, J.; Hansen, O.; Kampenes, V.; Karahasanovic, A.; Liborg, N.; Rekdal, A. “A Survey of Controlled Experiments in Software Engineering”, *IEEE Transactions on Software Engineering*, vol. 31–9, Oct 2005, pp. 733–753.

[SK10] Schilling, J.; Klamma, R. “The Difficult Bridge Between University And Industry: A Case Study in Computer Science Teaching”, *Assessment & Evaluation in Higher Education*, vol. 35–4, Jul 2010, pp. 367–380.

[Slo03] Slocum, N. “Participatory Methods Toolkit: A practitioner’s manual”. Belgian Advertising, 2003, 167p.

[SMG11] Salleh, N.; Mendes, E.; Grundy, J. “Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review”, *IEEE Transactions on Software Engineering*, vol. 37–4, Jul 2011, pp. 509–525.

[SRS13] Scheinrock, J.; Richter-Sand, M. “The Agile Startup: Quick and Dirty Lessons Every Entrepreneur Should Know”. John Wiley & Sons, 2013, 349p.

[SSFK18] Santos, A.; Sales, A.; Fernandes, P.; Kroll, J. “Challenge-Based Learning: a Brazilian Case Study”. In: Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, 2018, pp. 155–156.

[SSFN15] Santos, A.; Sales, A.; Fernandes, P.; Nichols, M. “Combining Challenge-Based Learning and Scrum Framework for Mobile Application Development”. In: Proceedings of the 20th Annual Conference on Innovation and Technology in Computer Science Education, 2015, pp. 189–194.

[Sut00] Sutton, S. M. “The Role of Process in Software Start-up”, *IEEE Software*, vol. 17–4, Jul 2000, pp. 33–39.

[SXT+09] Sun, D.; Xue, J.; Tan, X.; Liu, P.; Sun, Z.; Yao, J. “Model Analysis of Talents’ Abilities and Qualities for Information-Based Entrepreneurship”. In: Proceedings of the 1st International Conference on Information Science and Engineering, 2009, pp. 2968–2971.

[Tho00] Thomas, J. “A Review of Research on Project-Based Learning”, Technical Report, Autodesk Foundation, 2000, 49p.

[UAW+16] Unterkalmsteiner, M.; Abrahamsson, P.; Wang, X.; Nguyen-Duc, A.; Shah, S.; Bajwa, S.; Baltes, G.; Conboy, K.; Cullina, E.; Dennehy, D.; et al.. “Software Startups - A Research Agenda”, *e-Informatica Software Engineering Journal*, vol. 10–1, Oct 2016, pp. 89–124.

[VHB16] Vitolo, T.; Hersch, K.; Brinkman, B. “Making the Connection: Successful Cross Campus Collaboration Among Students”. In: Proceedings of the 46th Annual Frontiers In Education Conference, 2016, pp. 1–7.

[WKA16] Wang, X.; Khanna, D.; Abrahamsson, P. “Teaching Lean Startup at University: An Experience Report”. In: Proceedings of the 2nd International Workshop on Software Startups, 2016, pp. 1–4.

[WMMR06] Wieringa, R.; Maiden, N.; Mead, N.; Rolland, C. “Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion”, *Requirements Engineering*, vol. 11–1, Mar 2006, pp. 102–107.

[WRH+12] Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.; Regnell, B.; Wesslén, A. “Experimentation in Software Engineering”. Springer Science & Business Media, 2012, 236p.

[Yin03] Yin, R. “Case Study Research: Design and Methods”. SAGE Publications, 2003, 181p.

[ZA15] Zaina, L.; Alvaro, A. “A Design Methodology for User-centered Innovation in the Software Development Area”, *Journal of Systems and Software*, vol. 110–C, Dec 2015, pp. 155–177.

[Zha15] Zhang, S. “A Technology-Business-Environment Model for Effective Internet Entrepreneurship Education”. In: Proceedings of the 12th International Conference on Information Technology-New Generations, 2015, pp. 632–637.

APPENDIX A – STUDENTS' PERCEPTION ON LEAN PRINCIPLES RESEARCH PROTOCOL

Study Protocol

Goal

Understand how students perceive and grasp software startups concepts, such as the Lean Startup Methodology.

Study Design and Methodology

We have designed a multi-mathod study composed by two case studies and one survey. Data was analyzed using thematic analysis.

Research Question

- What are the benefits and challenges of software engineering students learning the Lean Startup approach?

Participants

- Two students in the end of a Software Engineering course in a Brazilian university employed Lean Startup and agile methodologies in their bachelor thesis project - Case Study A.
- Four students, one from a Masters and three from a Bachelor program in Software Engineering from an Italian university participated in the optional Lean Entrepreneurship course - Case Study B.
- 40 students from a iOS application development course in Brazil - Survey.

Data Collection

- For case study A, we analyzed the content of the thesis, as well as the emails exchanged between the students and their advisor.
- For case study B, data was collected through a retrospective session. Students were required to draw the project history as well as to describe positive and negative aspects regarding the process.

- For the survey, students were presented a small scenario and they had to describe how they could contribute to the project. This scenario was already presented in Section 5.1.

Data Analysis

The data analysis conducted in both parts employed mainly thematic analysis, following the scheme proposed by Braun and Clarke [BC06]. Although suggested to psychological studies, this approach has been commonly used in Software Engineering research [CD11].

APPENDIX B – SOFTWARE STARTUP EDUCATION RESEARCH PROTOCOL

Study Protocol

Goal

Understand how software startup is taught by lecturers/professor across the world.

Study Design and Methodology

We have designed a qualitative, largely open-ended survey. The goal of this survey was to understand in detail how software startups are currently taught in universities world-wide.

Research Questions

- Which teaching methods are used to teach software startups?
- Do students need to develop a real piece of software?
- Is the course somehow connected with the IT department?

Participants

Individuals involved in teaching software startups in universities.

Data Collection

Aside from contacting such individuals we knew beforehand, we searched for such courses online and contacted the teachers.

Data Analysis

First, We fully analyzed each response individually. Then, we compared and combined responses in order to find patters and interesting insights. Since we designed a lengthy survey, we were more interested in quality rather than quantity.

Survey Presentation and Questions

Hello, we are a part of SSRN (Software Startup Global Research Network) and we are running a research on Software Startup Education. **If you teach startup development related courses, your contribution will be really valuable to us**. This survey is comprised of less than 50 questions and it won't take more than 15 minutes of your time.

Thank you,

Rafael Chanin, Dron Khanna, Kai-Kristian Kemell, Wang Xiaofeng, Rafael Prikladnicki, Afonso Sales, Pekka Abrahamsson

1. How old are you?
2. What is your Gender?

   Male

   Female

   Prefer not to Answer
3. What is your highest degree? (For instance, PhD, MsC, BsC.)
4. Where do you teach and what is your job title? (Please, specify the university name and your title - professor, tutor, etc.)
5. What is your background field?

   Software Engineering

   Information Systems

   Business/Economics

   Other
6. Have you ever been involved in a startup yourself? If yes, please tell us briefly about this experience.
7. How many time has your startup course been held so far?
8. What is the name of the course? If possible, can you provide a link to the course description online?
9. How does the course posit itself in the curriculum of your department or the university? Which subject is it a part of? (Faculty of Computer Science, Engineering, Design, Economics, etc.)

10. Who can participate in the course? Is the course open to all students or limited to students of some majors, or even open to students from other universities (through some kind of collaboration between Universities)?

11. Is the course mandatory or optional, and for whom?

12. Do the students work on a real startup during the course, or is the course focused exclusively on theory?

13. Briefly describe the course (length, goals, how many hours per week, average number of students, etc).

14. What are the main learning goals of the course?

15. Does the course have other goals, such as fostering student entrepreneurship? (For instance, connecting students with other technology parks, investors, etc.)

16. How are student graded (project, exams, etc)? (If students work on a startup during the course, how is the practical startup work graded?)

17. Do you follow a formal methodology to teach? If yes, what it is?

18. What tools are used during the course? (For instance, Business Model Canvas, Validation Board, or any software tool.)

19. Are the student startup teams multidisciplinary (e.g., one developer, one designer, one business student per team) or can they comprise of any combination of students?

20. How large are the student teams ideally?

21. Is there a lower limit for team size? If yes, what is it and what it the reason for that limit?

22. Is there an upper limit for team size? If yes, what is it and what it the reason for that limit?

23. How many teachers/instructors are involved in the course? Please briefly describe their roles.

24. Are there outside practitioners involved as mentors or visiting lecturers? If yes, please describe their role in the course.

25. How is the progress of the student startup teams tracked during the course?

26. Why have you chosen to teach the class this way? Have you tried something in the past that did not work, for instance?

27. Based on your experiences with the course or course feedback, what have you learned? What has worked and what has not? Do you plan on changing the course in the future based on the feedback?

28. Describe the ideation process. Do students come up with their own ideas or are they given ideas?

29. Are student startups created for educational purposes only or are they potentially real startups? Can students who are already involved in a startup participate in the course using that startup to do so?

30. How do the students validate their ideas? What processes or tools do they use?

31. Is pivoting a part of the process? If yes, how do the students know when to pivot?

32. How do the students in the course generally market their idea?

33. What startup-related deliverables are required to pass the course? What is required of the startup during the course?

34. Are ideas pitched during the course? If yes, in what setting (e.g., course event, practitioner event)?

35. Has the course spawned any successful real world startups? If yes, please briefly describe these startups.

36. Is it common for students to keep working on their startup after the course ends? Is there any help they can receive from the people involved in the course if they wish to do so?

37. Are students expected to spend money on their startups? Are they provided with funds or are they expected to use their own funds?

38. Is the course somehow connected with the Computer Science/Software Engineering/IT department? If yes, how so?

39. Is there any specific Computer Science/Software Engineering pre-requisite to take the course? If yes, what are they?

40. Is there specific Computer Science/Software Engineering content taught in the course? If yes, can you specify them?

41. Do students need to develop a real piece of software?

    If so...

    What software development methodology is used?

What language/framework is used?

What platform (e.g. Android, iOS, web) is used, or are the students free to choose one?

How are software requirements gathered?

Do they set up an analytics to measure conversions? If not, do they measure it some other way?

Does the course include teaching software development practices, or are students expected to know all the required skills and tools beforehand?

42. Do students need to develop a prototype using existing tools (such as WIX, or WordPress)?

    If so...

    Which tools do the students use?

    Do they set up an analytics to measure conversions in their prototype? If not, do they measure it some other way?

    Are there any limitations in using these tools or students can get around in terms of the evolution of the solution?

43. Any other comments or information you believe it is important to provide?

APPENDIX C – TEACHING SOFTWARE STARTUPS USING CBL RESEARCH PROTOCOL

Study Protocol

Goal

Identify strength and weaknesses when using Challenge Based Learning in a software startup course.

Study Design and Methodology

We have designed a case study in which we observed students during their activities.

Research Questions

- Does CBL help engaging students when working on software startup projects?
- What are the strength and weaknesses of such methodology when teaching software startups?

Participants

- 30 students from an undergraduate software startup entrepreneurship course.
- The instructor of the course.

Data Collection

Data was collected from students' deliverables (reports and presentations) as well as from reflections (which is a component of the CBL framework). Reflections were done individually and privately by students and the instructor. At the end of the course a questionnaire was also applied to students.

Data Analysis

The data analysis conducted employed mainly thematic analysis, following the scheme proposed by Braun and Clarke [BC06].

Final Questionnaire

1. You have an idea for a startup. How would you plan the development and validation processes?
2. Which steps would you propose in order to develop this project?
3. In your opinion, what are the critical steps involved in this process?
4. I believe that it is mandatory to explore the problem before start the development of the product.

   Totally agree

   Partially agree

   Indifferent

   Partially disagree

   Totally disagree

5. I believe that the final product should be developed along with the definition of the problem.

   Totally agree

   Partially agree

   Indifferent

   Partially disagree

   Totally disagree

6. I believe that the project development should be done in a sequencial order, such as, problem definition, validation, and product development.

   Totally agree

   Partially agree

   Indifferent

   Partially disagree

   Totally disagree

7. I believe that is mandatory to review the previous steps during product development, as new demands and needs appear.

   Totally agree

   Partially agree

Indifferent

Partially disagree

Totally disagree

8. I believe that potencial users should participate in the development of the product.

   Totally agree

   Partially agree

   Indifferent

   Partially disagree

   Totally disagree

9. I believe that all steps of a product development should happen simultaneously, such as problem definition, validation, and product development.

   Totally agree

   Partially agree

   Indifferent

   Partially disagree

   Totally disagree

10. I believe that a startup should be development with an interdisciplinary team.

    Totally agree

    Partially agree

    Indifferent

    Partially disagree

    Totally disagree

11. I believe that a successful project is the one with a lot of customers.

    Totally agree

    Partially agree

    Indifferent

    Partially disagree

    Totally disagree

12. I believe that before the development of a product it is mandatory to analyze customers' needs.

    Totally agree

Partially agree

Indifferent

Partially disagree

Totally disagree

13. I believe that a successful project has more quality in terms of software than its competitors.

    Totally agree

    Partially agree

    Indifferent

    Partially disagree

    Totally disagree

14. I believe that a successful project is the one with a relevant problem to be solved.

    Totally agree

    Partially agree

    Indifferent

    Partially disagree

    Totally disagree

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br