# Software Startup Education Around the World: A Preliminary Analysis*

Rafael Chanin[1], Dron Khanna[2], Kai-Kristian Kemell[3], Wang Xiaofeng[2], Afonso Sales[1], Rafael Prikladnicki[1], and Pekka Abrahamsson[3]

[1] PUCRS, Porto Alegre, Brazil
{rafael.chanin, afonso.sales, rafaelp}@pucrs.br
[2] Free University of Bozen-Bolzano, Bolzano, Italy
{dron.khanna, xiaofeng.wang}@unibz.it
[3] University of Jyväskylä, Jyväskylä, Finland
{kai-kristian.o.kemell,pekka.abrahamsson}@jyu.fi

**Abstract.** New software startups are born everyday around the world. Nonetheless, failure is the fate of most of them. The community already knows that several facts, such as market competition or lack of resources, can impact the destiny of a startup. However, little has been explored in term of the impact of software startup education on the success of failure of startups. In this sense, this study presents the initial steps that we are taking to understand how software startups are taught around the world. To do so, we design a qualitative survey aimed at software startup educators at Universities. Our goal is to gather enough information so we can help the academic community in improving their own courses. So far, we have gathered 10 responses from lecturers across the globe. This paper describes these findings.

**Keywords:** Software Startup Education · Software Startup · Entrepreneurship.



## 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 [11]. Companies such as Google, Netflix, and WhatsApp are examples of organizations that were born under these conditions.

These technology endeavours, that are developed under high uncertainty, are called *startups* [2]. Most startups follow the *lean startup* methodology [19], which combines short software development cycles with constant interaction with users. The goal is to reduce risk by focusing on constant learning [4]. 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

* This work is partially funded by FAPERGS (17/2551-0001/205-4).

on understanding what customers need, what they expect, and how much they are willing to pay for a solution [6].

From an education perspective, several software engineering/computer science courses have focused on entrepreneurship in the last years ([8, 12, 15]). In addition, several technology-related courses are adapting their curriculum in order to include startup/entrepreneurship content [9].

One of the biggest challenges reported on these studies is the lack of a realistic environment for student to work on their startups [18]. 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 sense, the goal of this paper 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 focuses on the following research questions:

- **RQ1: Aside from conventional lecture-based courses ending in an exam, how are software startups taught in universities?**
- **RQ2: How do these courses deal with the multidisciplinary nature of a software startup (business, technology, design)?**

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the research method used in this research. In Section 4 we show our preliminary results. Finally, we draw our conclusions in Section 5.

## 2 Background

In this Section we define and explain in details what a software startup is. In addition, we depict how software startup education is being explored by the academic community.

### 2.1 Software Startups

Though software startups have recently had a large economic impact across the globe, no clear consensus on what exactly a software startup is exists [22]. 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 [17, 21]. They largely rely on external funding especially early on in their lifecycles, and have little to no prior operating history [1, 21]. 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 [1]. Software startups specifically, however, operate in particularly volatile markets, using current

top-of-the-line technologies to engineer innovative products and services [13]. This, combined with the scarcity of resources, leads to software startups generally operating under highly uncertain conditions [20].

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 [1]. 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 [1], 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, we consider 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 [7], with some estimates citing numbers as high as 90%. 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 [22]. 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 increasingly impactful at an international economic level [22], 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 [21, 14]. 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 ma-

terial 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 (for instance, Slush [4]) attract famous practitioner speakers and large, successful startups are 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.

More specifically in relation to software startups, software startups have been shown to develop software differently from SMEs and large corporations [17]. Software startups often use varying agile methods and practices, or even develop software purely ad hoc [17]. 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 find themselves accumulating technical debt. After all, in the case of failure, which is the fate of most software startups [22], that technical debt will never have to be addressed.

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. In the following sub-section, we discuss the current state of practice on software startup-related education based on literature.

### 2.2 Software Startup Education

Three of the authors of this papers have worked on a systematic mapping review on software startup education [5]. The goal of this work was to identify the main academic contributions on software engineering education in the software startup context by understanding the state-of-the-art research on software startup development education, and by collecting best practices and methodologies used on software startup education. After running the systematic mapping process, the researchers ended up fully exploring 31 publications. In this section we summarize the main contributions from this work.

The authors broke down the research into two research questions. The first one was related to tools, models, methodologies, and frameworks used in a software startup education context, whereas the second research question focused on best practices and lessons learned.

[4] https://www.slush.org

Regarding the first question, the authors found that the main components used to teach software startups are:

- *Business Model Canvas* - the Business Model Canvas [16] helps students understand startups in its entirety. Since technology students tend to focus more on the product and not on other important aspects of a startup (such as the market), the Business Model Canvas provides a way open students' mind in order for them to envision the big picture;
- *Customer Development Process* - The Customer Development Process, proposed by Blank and Dorf [2], helps students validate their business hypothesis. By telling entrepreneurs to "*get out of the building*", Blank and Dorf are saying that the validation process goes beyond product development; it is necessary to get closer to real potential customers in order to understand their needs;
- *Design Thinking* - The Design Thinking process is mostly used for ideation, specially when students need to come up with an idea to work on, or when they need creative solutions to move one with their projects; *Agile* - Whenever students need to create a real software project, Agile is the most used methodology. Since Agile methods, such as Scrum, provides flexibility and are receptive to project/product changes, it fits well on a software startup context.

In addition, some studies brought interesting insights for those who involved in software startup courses. For instance, Génova and González [10] claim that students must go through three stages in order to achieve a complete software startup education: (i) *instruction*, (ii) *training*, and (iii) *mentoring*. The first stage is related to tradicional educational settings, when students are able to learn content and are assessed by exams. The second stage is when students are able to choose their own way to solve a problem. Finally, the third stage is when students are able to self-propose their own goals and objectives.

In another study, Buffardi *et al.* [3] argue that working with mock-up projects is not effective, since students do not experience real life challenges. On the other hand, it is hard to emulate or to work with real world projects in an academic setting. The proposed solution was to create a multidisciplinary course with both software engineering and business students. In this situation, business students act as customers. Even though this is not an ideal scenario, at least provides a good overview of a software startup development process.

The conclusion regarding this first research question is that there are several different approaches being used. Since courses have different goals, objetives, and resources, each one ended up having a different focus. For instance, some courses just aim at inspiring students to further pursue an entrepreneurial career, while others focus more on technical aspects of a software startup.

In regards to the second research questions, we can break down the learnings into four categories:

- Teaching: The journey is more important that the endpoint. In this sense, lecturers should assess students' progress. Exams are not a good option since concepts are easy in theory, but very hard to be applied in practice;

- Real Projects: Whenever possible, courses should be connected to the market. When students work with real projects, their engagement and excitement rises. However, it is not always possible to do so. If that is the case, faculty should provide means for students to at least mimic a real world scenario;
- Multidiscipline: Coordinating and combing courses from different colleges (in this cases, technology and business) is always challenging. However, good experiences have been reported. Students learn more when dealing with peers with different backgrounds;
- Environment: The course should not be limited to the classroom. Connecting with the university ecosystems (such as technology parks) and even with other stakeholders always enriches the learning experience.

To sum up, several initiatives have been put in place in order to address software startup education. However, since it is a new topic not only for the academic community, but also to the industry, there is a lot or room for further research and development.

## 3 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 in its entirety can be found in the following link: *https://bepidpoa.typeform.com/to/kuh8bK*.

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. However, this survey is still on-going, and we are in search of more responses from those involved with teaching software startups, as we will discuss further in the following sections.

## 4 Preliminary Results

Though the survey is still on-going, and we wish to gather more responses before presenting further analysis on the subject, in this section we present preliminary results based on the 10 responses gathered thus far. Perhaps due to the nature of software startups, all of the responses so far 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 [22], 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.

In relation to our second research question (RQ2), software startups are software companies operating in terms of academic disciplines, in an area combining business and information technology. This was also reflected in the responses. Eight of the ten 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.

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.

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. 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.

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 [19], 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 so far, we have outlined some of the general trends in the way software startups are taught in universities. However, this research is still on-going, and based on our current set of data we are as of yet unable to provide conclusive answers to our research questions. 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, but our sample size is still too low to go into further detail.

## 5 Final Remarks

To summarize our findings, regarding que first research questions - *Aside from conventional lecture-based courses ending in an exam, how are software startups taught in universities?* - the 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 second research question - *How do these courses deal with the multidisciplinary nature of a software startup (business, technology, design)?* - 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. In our opinion, 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.

## References


1. Blank, S.: The Four Steps to the Epiphany: Successful Strategies for Products That Win. K&S Ranch, Incorporated (2013)
2. Blank, S., Dorf, B.: The Startup Owner's Manual: The Step-by-step Guide for Building a Great Company. K&S Ranch, Incorporated (2012)
3. Buffardi, K., Robb, C., Rahn, D.: Tech startups: realistic software engineering projects with interdisciplinary collaboration. Journal of Computing Sciences in Colleges **32**(4), 93–98 (2017)
4. 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. pp. 9–12. CHASE '18, ACM, New York, NY, USA (2018)
5. Chanin, R., Sales, A., Pompermaier, L.B., Prikladnicki, R.: A systematic mapping study on software startups education. In: EASE. pp. 163–168 (2018)
6. Coleman, G.: An empirical study of software process in practice. In: System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on. pp. 315c–315c. IEEE (2005)
7. 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 **1**, 338–343 (2002). https://doi.org/10.1109/IEMC.2002.1038454
8. da Cruz, E.F.Z., Alvaro, A.: Introduction of entrepreneurship and innovation subjects in a computer science course in Brazil. In: 2013 IEEE Frontiers in Education Conference (FIE). pp. 1881–1887 (2013). https://doi.org/10.1109/FIE.2013.6685162
9. Daimi, K., Rayess, N.: The Role of Software Entrepreneurship in Computer Science Curriculum. In: Proceedings of the 2008 International Conference on Frontiers in Education: Computer Science & Computer Engineering (FECS 2008). pp. 332–338. IEEE Computer Society, Las Vegas, NV, USA (July 2008)
10. Génova, G., González, M.: Educational Encounters of the Third Kind. Science and Engineering Ethics **1**, 1–10 (2016)
11. Giardino, C., Paternoster, N., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software Development in Startup Companies: The Greenfield Startup Model. IEEE Transactions on Software Engineering **42**(6), 585–604 (2016)
12. Harms, R.: Self-regulated learning, team learning and project performance in entrepreneurship education: Learning in a lean startup environment. Technological Forecasting and Social Change **100**, 21–28 (2015). https://doi.org/10.1016/j.techfore.2015.02.007
13. Hilmola, O.P., Helo, P., Ojala, L.: The value of product development lead time in software startup. System Dynamics Review **19**(1), 75–82 (2003)
14. Kon, F., Cukier, D., Melo, C., Hazzan, O., Yuklea, H.: A panorama of the israeli software startup ecosystem. Available at SSRN 2441157 (2014)

15. Kontio, J., Ahokas, M., Poyry, P., Warsta, J., Makela, M., Tyrvainen, P.: Software Business Education for Software Engineers: Towards an Integrated Curriculum. 19th Conference on Software Engineering Education and Training Workshops (CSEETW'06) pp. 4–7 (2006). https://doi.org/10.1109/CSEETW.2006.15
16. Osterwalder, A., Pigneur, Y.: Business model generation: a handbook for visionaries, game changers, and challengers. John Wiley & Sons (2010)
17. Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software Development in Startup Companies: A Systematic Mapping Study. Information and Software Technology **56**(10), 1200–1218 (2014)
18. 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: 2015 IEEE Frontiers in Education Conference (FIE). pp. 1–5. IEEE (2015)
19. Ries, E.: The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. Crown Business (2011)
20. Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business (2011)
21. Sutton, S.M.: The role of process in software start-up. IEEE Software **17**(4), 33–39 (Jul 2000). https://doi.org/10.1109/52.854066
22. Unterkalmsteiner, M., Abrahamsson, P., Wang, X., Nguyen-Duc, A., Shah, S., Bajwa, S.S., Baltes, G.H., Conboy, K., Cullina, E., Dennehy, D., et al.: Software startups–a research agenda. e-Informatica Software Engineering Journal **10**(1) (2016)