Poster: Startup Software Development Education: A Systematic Mapping Study

Rafael Chanin, Afonso Sales, Leandro Pompermaier, Rafael Prikladnicki
Pontifical Catholic University of Rio Grande do Sul, School of Technology
Av. Ipiranga 6681, Porto Alegre, Brazil
{rafael.chanin,afonso.sales,leandro.pompermaier,rafaelp}@pucrs.br

ABSTRACT

This study aims to characterize the state-of-the-art of the software startup education by analyzing and identifying best practices, opportunities and gaps on this field. To do so, we conducted a systematic mapping study in order to analyze and evaluate studies on software startup education. As a result, we found 31 primary studies in this process. These studies were classified into for categories: real projects, multidiscipline, environment and teaching. We concluded that research on software startup education is still scarce. Furthermore, there are several gaps and opportunities to be explore in future works. One of them is the difficulty in providing a real world experience in a educational setting. Successful cases reported combine three major components: real world projects, the right environment and a multidisciplinary context.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; • **Information systems** → *Mobile information processing systems*;

KEYWORDS

Startup Education, Startup, Software Development Education, Software Engineering Education, Entrepreneurship Education.

ACM Reference Format:
Rafael Chanin, Afonso Sales, Leandro Pompermaier, Rafael Prikladnicki. 2018. Poster: Startup Software Development Education: A Systematic Mapping Study. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3195055

1 INTRODUCTION

Everyday new startups are born around the world. Unfortunately, the majority of them do not survive the first two years of its existence [3]. The academy has been producing a significant amount of papers concerning the study of software development process in the context of a startup [3–5, 7, 8]. However, research on software development education focused on software startup is scarce. Therefore, this paper aims to identify the main academic contributions on software engineering education in the software startup context. Hence, we perform a systematic mapping study to (i) understanding the state-of-the-art research on software startup development education; (ii) collecting best practices and methodologies used on software startup education; and (iii) identifying gaps for future studies. From an initial set of 224 papers, we have identified 31 primary studies worth analyzing.

The research question proposed is: *What is the state-of-art in literature in regards to software startup development education?* In order to answer the main research question, we broke down this objetive into two subquestions: (RQ1) *Which software engineering tools, models, methodologies and frameworks are applied in a software startup education context?* and (RQ2) *What are the reported best practices in regards to teaching software startup?*

2 RESULTS OF THE SYSTEMATIC MAPPING

This systematic mapping followed the recommendation of the most influential researchers in this area [2, 9]. In this section we present the findings from the two research questions proposed. For each question we combined the information gathered from the systematic mapping process with insights and learnings from each of the 31 primary study selected.

2.1 (RQ1) Which software engineering tools, model and/or frameworks are applied in a software startup education context?

The overview of this systematic mapping study indicates that there is one study contributing with tools, five studies focusing on models, and seven studies 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 [1], 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.

• **User-Centered Design**: encourage students to be creative and, most importantly, to understand real customers' needs.

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195055

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 needs to evaluate the tradeoffs associated with each approach.

2.2 (RQ2) What are the reported best practices in regards to teaching software startup?

We have extracted several practices and lessons learned from the 31 primary studies. They were categorized as follows:

• **Teaching**: the learning happens as students go through the process (talking to customers, working in teams, building MVP), and not through exams. In this sense, a flipped classroom approach is ideal.

• **Real Projects**: customer and problem should never be given to students. They need to explore these issues through interviews or other research methods. Otherwise learning is limited to software engineering, project management and teamwork.

• **Multidiscipline**: opportunities should aim at cross-discipline collaboration. Software engineering, computer science and business courses should be combined and taught together in the same classroom by two or more instructors.

• **Environment**: when possible, faculty should create opportunities for external validation. This can be achieved by connecting with the university ecosystem (such as a technology park).

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 [6] 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.

3 FINAL REMARKS

We conducted a systematic mapping study in order to identify and characterize the main academic contributions on software engineering education in the software startup context. The main 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 quality. The focus facet revealed that studies fall into one of the following categories: real projects, multidiscipline, environment, and teaching. This information is consistent with data found in the literature [3, 4, 7, 8]; a startup tries to solve real world problems, it needs a multidisciplinary team to succeed, and the environment plays a key role (to find partners and investors, for instance).

Our results indicate that research on software startup education is taken its first steps. Most of the studies are recent and the majority were published in conferences/symposiums, and fewer in journals. This is clearly an indication that there is a lot more to be explored.

In regards to the research questions proposed, we could identify a gap on tools and frameworks. We raised two hypothesis for this matter. First, there are no tools of frameworks designed to help the software engineering teaching process in the context of software startups. Second, there are tools and frameworks already in use, but they have not been studied and reported by the scientific community. In both scenarios there is an opportunity to create and develop further research. However, if the first hypothesis is correct there is a great opportunity to proposed new techniques that can be applied and tested in educational settings to be further published to the scientific community.

Another related research gap founded is that there is not a clear agreement in the scientific community regarding best practices to be adopted to teach software startups for software engineering students. As mentioned in the beginning of this section, working in real world projects with multidisciplinary teams and the right environment seems to be the right combination. However, methods and approaches vary among studies. We believe there is an opportunity for the creation of a more cohesive way of dealing with software startups in a education environment.

In conclusion, this systematic mapping was a first attempt to better understand how software startups are 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.

ACKNOWLEDGMENTS

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

REFERENCES

[1] S. Blank and B. Dorf. 2012. *The Startup Owner's Manual: The Step-by-step Guide for Building a Great Company.* K&S Ranch, Incorporated.

[2] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham. 2008. Using Mapping Studies in Software Engineering. In *Proc. of the 20th Annual Meeting of the Psychology of Prog. Interest Group (PPIG 2008).* Lancaster, UK, 195–204.

[3] C. Giardino, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson. 2016. Software Development in Startup Companies: The Greenfield Startup Model. *IEEE Transactions on Software Engineering* 42, 6 (2016), 585–604.

[4] C. Giardino, M. Unterkalmsteiner, N. Paternoster, T. Gorschek, and P. Abrahamsson. 2014. What Do We Know about Software Development in Startups? *IEEE Software* 31, 5 (2014), 28–32.

[5] C. Giardino, X. Wang, and P. Abrahamsson. 2014. *Why Early-Stage Software Startups Fail: A Behavioral Framework.* Springer, Paphos, Cyprus, 27–41.

[6] F. Heintz and K. I.E Klein. 2014. 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.* Atlanta, GA, USA, 199–204.

[7] A. Nguyen-Duc, P. Seppänen, and P. Abrahamsson. 2015. Hunter-gatherer Cycle: A Conceptual Model of the Evolution of Software Startups. In *Proc. of the 2015 Int. Conf. on Software and System Process (ICSSP 2015).* Tallinn, Estonia, 199–203.

[8] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson. 2014. Software Development in Startup Companies: A Systematic Mapping Study. *Information and Software Technology* 56, 10 (2014), 1200–1218.

[9] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. 2008. Systematic Mapping Studies in Software Engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE).* Bari, Italy, 68–77.