A Systematic Mapping Study On Software Startups Education

Rafael Chanin
PUCRS, School of Technology
Av. Ipiranga 6681, Porto Alegre, Brazil
rafael.chanin@pucrs.br

Afonso Sales
PUCRS, School of Technology
Av. Ipiranga 6681, Porto Alegre, Brazil
afonso.sales@pucrs.br

Leandro Pompermaier
PUCRS, School of Technology
Av. Ipiranga 6681, Porto Alegre, Brazil
leandro.pompermaier@pucrs.br

Rafael Prikladnicki
PUCRS, School of Technology
Av. Ipiranga 6681, Porto Alegre, Brazil
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 publications in this process. These studies were classified into four 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 explored 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 technology education;

KEYWORDS

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

ACM Reference Format:
Rafael Chanin, Afonso Sales, Leandro Pompermaier, and Rafael Prikladnicki. 2018. A Systematic Mapping Study On Software Startups Education. In *EASE'18: 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, June 28–29, 2018, Christchurch, New Zealand*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3210459.3210478

1 INTRODUCTION

In the past years we have seen great advances in the technology that changed the way society interacts [29]. This process has enabled software development companies and even individuals to create scalable business that can reach millions of users [16]. These high-risky technology businesses, that aim to find a sustainable and repeatable business model, are called *startups* [4].

Unfortunately, the majority of the startups do not survive the first two years of their existence [16]. External factors, such as competition and market instability, definitely account for this result. However, internal factors also play a key role in this regard [24]. 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 [17].

From an education standpoint, technology-related undergraduate and graduate programs are adapting themselves in order to fit startup content into their curriculum [11]. The challenge usually lies in providing students with the right set of tools that will help them deal with a startup chaotic environment. In addition, a software startup is not just about software development; it is also about critical thinking, problem solving, and adaptability. Therefore, these abilities also need to be addressed in order to develop students capable of running or working for a startup. There has been a significant amount of academic work concerned in the study of software development processes in a startup context [16–18, 29, 32]. However, research on software development education focused on software startup is scarce.

The goal of this paper is to identify the main academic contributions on software engineering education in the software startup context. In order to do so, we performed a systematic mapping study [6, 35] aimed at (i) understand the state-of-the-art research on software startup development education; (ii) collect best practices and methodologies used on software startup education; and (iii) identify gaps for future studies. From an initial set of 224 papers, we have identified 31 publications worth analyzing from 5 distinct scientific databases. This paper reports on the findings of this study.

The remainder of the paper is structured as follows: Section 2 describes the research methodology. In Section 3 we present the results of the systematic mapping study. Section 4 explores the proposed research questions. Finally, we conclude the paper in Section 5 exploring our final remarks and proposing future works.

EASE'18, June 28–29, 2018, Christchurch, New Zealand

ACM ISBN 978-1-4503-6403-4/18/06…$15.00
https://doi.org/10.1145/3210459.3210478

2 RESEARCH METHODOLOGY

The research question proposed for this study is: *What is the state-of-art in literature in regards to software startup development education?* In order to do so, we carried out this systematic mapping review following the recommendation of the most influential researchers in this area [6, 26, 35].

2.1 Research questions definition

In order to answer the main research question of this study - *What is the state-of-art in literature in regards to software startup development education?* - we broke down this objetive into two subquestions:

- **RQ1**: *Which tools, models, methodologies and frameworks are applied in a software startup education context?*
- **RQ2**: *What are the reported best practices in regards to teaching software startup?*

The purpose of RQ1 is to understand if there are tools, models, methodologies and frameworks that are somehow new to the technology educational context and that have been used specifically to teach software startups processes. With RQ2 we intend to analyze which teaching strategies and best practices are put in place. Since startups have the goal to solve real-world problems [3], there is a possibility that non-traditional teaching approaches could be used.

2.2 Data sources and search strategy

With regard to identifying the publications for this study, we design a search string following the guidelines proposed by Kitchenham and Charters [27]. 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 was:

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)

The databases sources were chosen based on the list proposed by Kitchenham and Charters [27]. 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 sum, the databases searched were: *ACM Digital Library*, *IEEExplore*, *Scopus*, *El Compendex*, and *Science@Direct*. In regards to the selection criteria, we chose publications: (i) that were available online; (ii) that were written in English; (iii) from 1998 to May 2017; (iv) published in Journals, Conferences, Workshops, Symposiums; (v) with a minimum of 4 pages. 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 [38], started to be formed and studied. After executing our search strategy, we came across 224 papers.

2.3 Screening of Papers and Keywording

The screening process started by excluding duplicates, which accounted for 62 items, leaving us with 162 papers. After that, we followed the exclusion criteria defined in Section 2.2. This process eliminated 74 items, leaving us with 88 papers. 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. At the end, our screening process led to 31 publications to be fully analyzed.

According to Petersen *et al.* [35], "*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* ". The keywording process starts by reading abstracts of the publications 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. The resulting classification scheme is presented and discussed in Section 2.5.

2.4 Data extraction and mapping

The data extraction and mapping was performed by using two tools: a spreadsheet, and the software *Mendeley*[1]. This application helps users manage papers for research purpose. After reading the papers, we added the following categorization to our spreadsheet: (i) *Focus Facet*: the categories created during the classification scheme process; (ii) *Contribution Facet*: type of contribution, based on Shaw [43] and Paternoster et. al. [32] works; (iii) *Research Method*: method used on the research (case study, survey, *etc.*); (iv) *Research Type*: type of research (adapted from Wieringa et. al. [46]); (v) *Paper Quality*: a grade (0 to 10), based on the work from Salleh et. al. [40].

2.5 Classification Scheme

Five categories were defined in the process of data extraction and mapping: focus (teaching, real projects, multidiscipline, and environment), contribution (advice/implication, framework/method, guidelines, lessons learned, model, and tools), research method (case study, empirical study, experimental study, and survey), research type (evaluation research, experience paper, opinion paper, philosophical paper, solution proposal, and validation research), and paper quality (see Table 1).

Table 1: Paper Quality Criteria.

Criteria	Question
References	Is the study well referred? (1 point)
Paper Goal	Is the goal clearly stated? (1 point)
Sample Observation	Is the research 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)
RQ1	Does que paper answer RQ1? (2 points)
RQ2	Does que paper answer RQ2? (2 points)

3 RESULTS OF THE SYSTEMATIC MAPPING

From an initial set of 224 papers identified through the search strategy, we selected 31 publications. The systematic map overview is presented in Table 2.

We followed the guidelines from Petersen et. al. [35] and used bubble plots to combine and compare the different facets that were

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

Table 2: Systematic map overview.

1st Author (year)	Research method	Research type	Contribution	Focus	Paper Quality
Génova (2016) [15]	Empirical Study	Philosophical Paper	Model	Teaching	9.5
Järvi (2015) [22]	Case Study	Evaluation Research	Lessons Learned	Teaching	9.5
Schilling (2010) [42]	Case Study	Evaluation Research	Guidelines	Teaching	9.5
Zaina (2015) [47]	Case Study	Evaluation Research	Framework/Method	Teaching	9.0
Currie (2011) [10]	Case Study	Experience Paper	Framework/Method	Real Projects	9.0
Chesney (2014) [9]	Case Study	Evaluation Research	Guidelines	Real Projects	9.0
Izurieta (2016) [21]	Case Study	Experience Paper	Framework/Method	Real Projects	9.0
Zhang (2015) [48]	Experiment	Solution Proposal	Model	Teaching	8.5
de Lange (2016) [12]	Case Study	Validation Research	Framework/Method	Environment	8.5
Joseph (2006) [23]	Case Study	Evaluation Research	Lessons Learned	Multidiscipline	8.5
Daimi (2008) [11]	Empirical Study	Solution Proposal	Guidelines	Teaching	8.5
Chenoweth (2008) [8]	Survey	Experience Paper	Guidelines	Teaching	8.0
Ribeiro (2016) [37]	Case Study	Experience Paper	Lessons Learned	Environment	8.0
McMahon (2014) [28]	Case Study	Experience Paper	Lessons Learned	Teaching	8.0
Vitolo (2016) [45]	Experiment	Evaluation Research	Lessons Learned	Multidiscipline	8.0
Kaltenecker (2013) [25]	Empirical Study	Philosophical Paper	Lessons Learned	Teaching	8.0
Buffardi (2017) [7]	Case Study	Solution Proposal	Framework/Method	Multidiscipline	7.5
Porter (2015) [36]	Case Study	Solution Proposal	Lessons Learned	Multidiscipline	7.5
Bharadwaj (2014) [2]	Experiment	Experience Paper	Framework/Method	Teaching	7.5
Nguyen-Duc (2016) [30]	Case Study	Evaluation Research	Model	Teaching	7.5
Breytenbach (2013) [5]	Case Study	Experience Paper	Lessons Learned	Environment	7.5
Pauca (2012) [33]	Case Study	Solution Proposal	Framework/Method	Real Projects	6.5
Heintz(2014) [20]	Case Study	Experience Paper	Lessons Learned	Teaching	6.5
Ford (2004) [14]	Case Study	Experience Paper	Lessons Learned	Multidiscipline	6.5
Barbe (2010) [1]	Empirical Study	Experience Paper	Model	Environment	6.5
Gross (2000) [19]	Survey	Evaluation Research	Lessons Learned	Teaching	6.5
Sun (2009) [44]	Empirical Study	Philosophical Paper	Model	Environment	5.0
Engelsma (2014) [13]	Case Study	Experience Paper	Lessons Learned	Real Projects	4.5
Sarraipa (2016) [41]	Case Study	Solution Proposal	Advice/Implication	Environment	4.5
Rioja Del Rio (2014) [39]	Empirical Study	Opinion Paper	Tool	Teaching	4.5
Pauli (2008) [34]	Case Study	Experience Paper	Lessons Learned	Real Projects	2.0

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 publications in a given intersection.

Figure 1 presents the first bubble plot designed for this study. It combines the paper quality facet, with the research type facet and the focus facet. Most studies were considered as medium and as high quality. 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.

In Figure 2 we combined 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.

4 RESEARCH QUESTIONS ANALYSIS

In this section we analyze the two research questions proposed for this study. For each question we combined the information developed in Section 3 with the insights and learnings from each of the 31 publications selected.

4.1 (RQ1) Which tools, models, methodologies and frameworks are applied in a software startup education context?

The overview of this systematic mapping study, detailed in Table 2, indicates that there is one study [39] contributing with tools, five studies [1, 15, 30, 44, 48] focusing on models, and seven studies [2, 7, 10, 12, 21, 33, 47] exploring methods and methodologies. By combining and summarizing this information, the main contributions to the field are: (i) *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; (ii) *Customer Development Process* - proposed by Blank and Dorf [4], this model helps students take actionable steps in order to validate business hypothesis; (iii) *Design Thinking* - very useful during the ideation phase, but it is also used further in the process when creative solutions need to be developed; (iv) *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. [39] suggest the use of the Business Model Canvas (BMC) [31] 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 studies that presented software startup education models revealed interesting insights. Génova and González [15] 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,

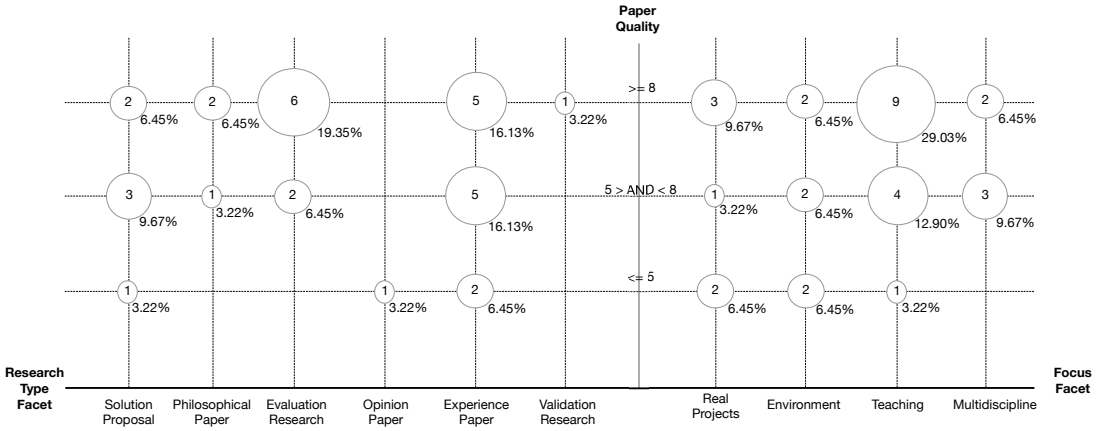

Figure 1: Systematic map by paper quality, research type and focus.

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 as it 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". According to the authors, achieving the third stage is crucial. Otherwise educational institution will be mostly graduating workers rather than entrepreneurs.

Zhang [48] 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 (business and 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. Going further, faculty should implement strategies to manage conflict that may arise due to students' different backgrounds.

Buffardi et. al. [7] 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 life" situations. Therefore, a methodology was proposed in order to minimize this gap. 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.

4.1.1 Discussion. 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.

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

We have extracted several practices and lessons learned from the 31 publications. In the remainder of this section we discuss them according to the focus facet of the classification schema.

4.2.1 Teaching. From a teaching perspective, several insights and best practices were found. An interesting point is that 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 MVP). In this sense, a flipped classroom approach is ideal. Traditional lectures should be used only to deliver basic concepts. Therefore, the journey is more important than the endpoint. In other words, 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 the case studies analyzed. 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. Moreover, teams should present their progress to the whole class

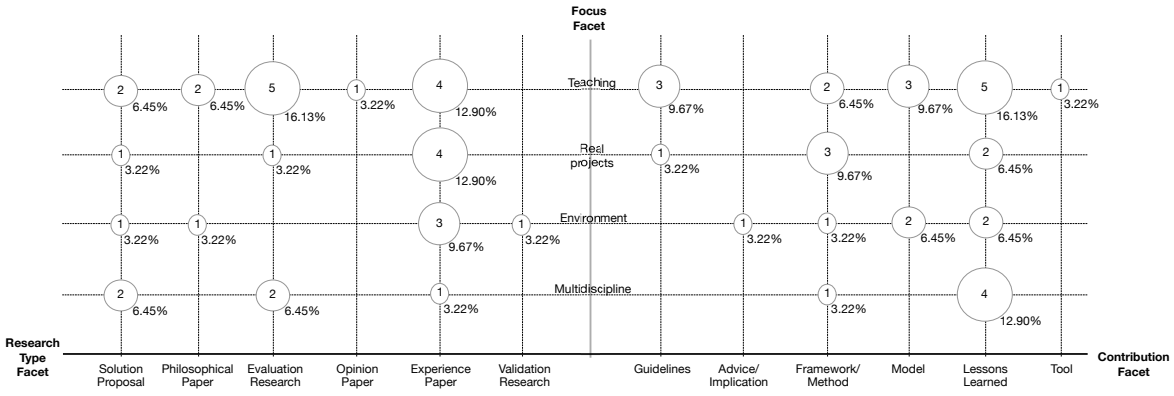

Figure 2: Systematic map by focus, research type and contribution.

several times during the semester. By doing so, they can receive and give feedback to each other. This approach helps students developing their oral communication skills. When it come to software development tools and processes, students should use the same software development language. From the faculty point of view, it is hard to help teams if they use different technologies.

4.2.2 Real Projects. Problems and potential customers 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. 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.

4.2.3 Multidiscipline. Most studies stated that 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 encourage multidisciplinary team formation. 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.

4.2.4 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.

4.2.5 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. But, successful cases were reported. Some projects actually end up being embraced by university incubators.

5 FINAL REMARKS

In this paper we conducted a systematic mapping study in order to identify 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 quality. The focus facet revealed that studies fell into one of the following categories: real projects, multidiscipline, environment, and teaching.

In regards to the first research question - *Which tools, models, methodologies and frameworks are applied in a software startup education context?* - 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.

The second question - *What are the reported best practices in regards to teaching software startup?* - revealed that offering real world experience to students remains a challenge. 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.

ACKNOWLEDGMENTS

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

REFERENCES

[1] D.F. Barbe. 2010. A Model of Cross Disciplinary Education, Technology Transfer and Teaching Non-Technical Skills for Engineers. In *Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments*. IEEE Computer Society, Dublin, Ireland, 1–32.

[2] A.K. Bharadwaj. 2014. An evaluation of teaching theoretical graduate engineering courses adapting different techniques. In *IEEE International Conference on MOOC, Innovation and Technology in Education*. IEEE CS, Patiala, India, 84–88.

[3] S. Blank. 2013. *The Four Steps to the Epiphany: Successful Strategies for Products That Win*. K&S Ranch, Incorporated.

[4] 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.

[5] J. Breytenbach, C. de Villiers, and G. Hearn. 2013. Directing the South African Ict Labour Force Towards Growth Sectors: A case for non-institutional scarce skills transition and reskilling courses. In *AIS Special Interest Group for Education: International Academy for Information Management - AIS SIG-ED IAIM 2013 International Conference on Informatics Education and Research Conference*. Association for Information Systems, Milan, Italy.

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

[7] K. Buffardi, C. Robb, and D. Rahn. 2017. Tech startups: realistic software engineering projects with interdisciplinary collaboration. *Journal of Computing Sciences in Colleges* 32, 4 (2017), 93–98.

[8] S. Chenoweth. 2008. Undergraduate software engineering students in startup businesses. In *Proceedings of the 21st Conference on Software Engineering Education and Training (CSEET'08)*. IEEE Computer Society, Charleston, SC, USA., 118–125.

[9] D.R. Chesney. 2014. Social context, singular focus. In *Proceedings of 2014 IEEE Frontiers in Education Conference (FIE)*. IEEE CS, Madrid, Spain, 1–6.

[10] E.H. Currie, S. Doboli, and G.L. Kamberova. 2011. Developing the next generation of entrepreneurs. In *Proceedings of 2011 IEEE Frontiers in Education Conference (FIE)*. IEEE Computer Society, Rapid City, SD, USA, S2B 1–6.

[11] K. Daimi and N. Rayess. 2008. 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)*. IEEE Computer Society, Las Vegas, NV, USA, 332–338.

[12] P. de Lange, P. Nicolaescu, R. Klamma, and I. Koren. 2016. DevOpsUse for Rapid Training of Agile Practices Within Undergraduate and Startup Communities. In *European Conf. on Tech. Enhanced Learning*. Springer, Lyon, France, 570–574.

[13] J.R. Engelsma. 2014. Best practices for industry-sponsored CS capstone courses. *Journal of Computing Sciences in Colleges* 30, 1 (2014), 18–28.

[14] R.M. Ford, J.G. Goodrich, and R.S. Weissbach. 2004. A multidisciplinary business and engineering course in product development and entrepreneurship. In *Proceedings of the 34th Annual Frontiers in Education (FIE 2004)*. IEEE Computer Society, Savannah, GA, USA, T2E/5–T2E10.

[15] G. Génova and M.R. González. 2016. Educational Encounters of the Third Kind. *Science and Engineering Ethics* 1 (2016), 1–10.

[16] 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.

[17] 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.

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

[19] W.A. Gross. 2000. An approach to teaching entrepreneurship to engineers. In *Proceedings of the 2000 IEEE Engineering Management Society (EMS 2000)*. Albuquerque, NM, USA, 648–652.

[20] 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.

[21] C. Izurieta, M. Trenk, M. O'Bleness, and S. Gunderson-Izurieta. 2016. The Effectiveness of Software Development Instruction through the Software Factory Method for High School Students. In *123rd Annual Conference in Engineering and Education (ASEE'16)*. New Orleans, USA, 26–29.

[22] A. Järvi, V. Taajamaa, and S. Hyrynsalmi. 2015. *Lean Software Startup – An Experience Report from an Entrepreneurial Software Business Course*. Springer International Publishing, Braga, Portugal, 230–244.

[23] A. Joseph. 2006. Interdisciplinarity, financial software product development, and entrepreneurship in an urban university. *American Society for Engineering Education* 11, 1 (2006), 812.1–812.13.

[24] M. Kajko-Mattsson and N. Nikitina. 2008. From Knowing Nothing to Knowing a Little: Experiences Gained from Process Improvement in a Start-Up Company. In *Int. Conf. on Computer Science and Soft. Eng. (CSSE 2008)*. Wuhan, China, 617–621.

[25] N. Kaltenecker, C. Hoerndlein, and T. Hess. 2013. The Drivers of Entrepreneurial Intentions - An Empirical Study among Information Systems and Computer Science Students. In *AMCIS 2013*. Chicago, IL, USA, 1–8.

[26] B. Kitchenham, D. Budgen, and O.P. Brereton. 2011. Using mapping studies as the basis for further research — A participant-observer case study. *Information and Software Technology* 53, 6 (2011), 638–651.

[27] B. Kitchenham and S. Charters. 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Technical Report EBSE 2007-001. Keele University and Durham University Joint Report, Keele and Durham, UK.

[28] E. McMahon. 2014. From Product Development to Innovation. In *Proceedings of ASEM 2014*. Virginia Beach, USA, 118–127.

[29] 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.

[30] A. Nguyen-Duc, S.M.A. Shah, and P. Ambrahamsson. 2016. Towards an early stage software startups evolution model. In *Proc. of the 42th Euromicro Conf. on Soft. Eng. and Advanced Applications (SEAA 2016)*. Limassol, Cyprus, 120–127.

[31] A. Osterwalder and Y. Pigneur. 2010. *Business model generation: a handbook for visionaries, game changers, and challengers*. John Wiley & Sons.

[32] 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.

[33] V.P. Pauca and R.T. Guy. 2012. Mobile Apps for the Greater Good: A Socially Relevant Approach to Software Engineering. In *Proc. of the 43rd ACM Tech. Symposium on Comp. Sci. Education (SIGCSE '12)*. Raleigh, NC, USA, 535–540.

[34] J.W. Pauli, T.E. Lawrence, and B.F. Brown. 2008. Development of a new software product from a classroom project. In *Proc. of the 5th Int. Conf. on Information Technology: New Generations (ITNG 2008)*. Las Vegas, NV, USA, 97–100.

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

[36] J. Porter, J. Morgan, R. Lester, A. Steele, J. Vanegas, and R. Hill. 2015. A course in innovative product design: A collaboration between architecture, business, and engineering. In *Proceedings of IEEE FIE 2015*. IEEE CS, El Paso, TX, USA, 1–5.

[37] C.M.F.A. Ribeiro, F.A. Aleixo, and M.A. Freire. 2016. 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 (PROFES 2016)*. Trondheim, Norway, 636–639.

[38] E. Ries. 2011. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business.

[39] C. Rioja Del Rio, A. Morgado-Estevez, and J.J. Dominguez-Jimenez. 2014. Entrepreneurship and lean manufacturing for software engineering. In *SEFI Annual Conference 2014*. Birmingham, UK.

[40] N. Salleh, E. Mendes, and J. Grundy. 2011. Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. *IEEE Transactions on Software Engineering* 37, 4 (2011), 509–525.

[41] J. Sarraipa, F. Ferreira, E. Marcelino-Jesus, A. Artifice, C. Lima, and M. Kaddar. 2016. Technological Innovations tackling Students dropout. In *Proc. of the 7th Int. Conf. on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*. ACM, Vila Real, Portugal, 112–118.

[42] J. Schilling and R. Klamma. 2010. The difficult bridge between university and industry: a case study in computer science teaching. *Assessment & Evaluation in Higher Education* 35, 4 (2010), 367–380.

[43] M. Shaw. 2003. Writing Good Software Engineering Research Papers: Minitutorial. In *Proceedings of the 25th International Conference on Software Engineering (ICSE '03)*. IEEE Computer Society, Portland, OR, USA, 726–736.

[44] D. Sun, J. Xue, X. Tan, P. Liu, Z. Sun, and J. Yao. 2009. Model Analysis of Talents' Abilities and Qualities for Information-Based Entrepreneurship. In *Proceedings of ICISE 2009*. Nanjing, China, 2968–2971.

[45] T.M. Vitolo, K.E. Hersch, and B.J. Brinkman. 2016. Making the connection: Successful cross campus collaboration among students. In *Proc. of 2016 IEEE FIE*. IEEE Computer Society, Erie, PA, USA, 1–7.

[46] R. Wieringa, N. Maiden, N. Mead, and C. Rolland. 2005. Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion. *Requirements Engineering* 11, 1 (2005), 102–107.

[47] L.A.M. Zaina and A. Alvaro. 2015. A Design Methodology for User-centered Innovation in the Software Development Area. *Journal of Systems and Software* 110, C (2015), 155–177.

[48] S. Zhang. 2015. A Technology-Business-Environment Model for Effective Internet Entrepreneurship Education. In *Proc. of the 12th Int. Conf. on Information Technology-New Generations (ITNG 2015)*. Las Vegas, NV, USA, 632–637.