A STUDY OF SOFTWARE ENGINEERING METHODOLOGIES & DIFFERENT PRACTICES IN SOFTWARE STARTUPS

[1]Bhargab Jyoti Kalita, [2]Debashis Deb, [3]Nomi Baruah
[1]Student, [2]Student, [3]Assistant Professor
[1]Computer Science & Engineering,
[1]Dibrugarh University Institute of Engineering and Technology, Dibrugarh, India

Abstract : Software startups are newly established companies with no operating history and fast in producing cutting-edge technologies. These companies develop software under highly uncertain conditions, tackling fast growing markets under severe lack of resources. Therefore, software startups present an unique combination of characteristics which pose several challenges to software development activities Startup companies are becoming important suppliers of innovative and software intensive products. The failure rate among startups is high due to lack of resources, immaturity, competitive market and various evolving technologies. The significance of startups for economic development is growing in software business due to their ability to quickly create innovative technologies and their potential to scale to a wide ever changing, ever evolving market. This study aims to structure and analyze the different practices used by software startup companies reported by various practitioners and researchers.

***Index Terms* - Software Development, software startups, systematic study.**

I. INTRODUCTION

A software startup or start-up is started by individual founders or entrepreneurs to search for a repeatable and scalable business model. More specifically, a startup is a newly emerged business venture that aims to develop a viable business model to meet a marketplace need or problem. Founders design startups to effectively develop and validate a scalable business model[1].Recent developments in technologies have created an increasing demand for innovative software products. Startup companies are addressing this need and gain importance as suppliers of software-intensive products and innovation. The inherent nature of software enables small companies to produce and launch software products fast with few resources [2]. The concepts of startups and entrepreneurship are similar. However, entrepreneurship refers all new businesses, including self- employment and businesses that never intend to grow big or become registered, while startups refer to new businesses that intend to grow beyond the solo founder, have employees, and intend to grow large. Start-ups face high uncertainty and do have high rates of failure, but the minority that go on to be successful companies have to potential to become large and influential[1].A wide body of knowledge has been created in recent years through several empirical studies, investigating how companies leverage software engineering. However, research on software development activities in newly created companies is scarce. In the past, very few publications have identified, characterized and mapped work practices in software startups and no structured investigation of the area has been performed [3]. However, most of startup companies fail before realizing any significant achievements. Partially this is due to market factors or financial issues, however the impact of software product engineering and inadequacies in applied engineering practices is not fully explored, and might be a significant contributing factor for the high failure rates[2].Startups are unique, since they develop software in a context where processes can hardly follow a prescriptive methodology. Startups share some characteristics with other contexts such as small companies and web engineering, and present a combination of different factors that make the development environment different from established companies.[3]

I.1 DIFFERENCES OF STARTUPS FROM LARGE SCALE & MULTINATIONAL COMPANIES:

There are various differences between software startup companies & large scale multinational companies.

1)Getting the Job[4]: It's usually easier to get a job in a startup company, whereas in large scale companies, people needs to go through five, six or even ten interviews before they actually get a job.

2)More Responsibilities[5]: In case of Startups, there is a little room to waste money or the precious time on the employees who do not meet the expectations. Every role comes with great expectation. While the same may not be applicable to such extent in Large scale companies.

3)Risk on Job security[6]: Startups are riskier in comparison to MNC's. Job will be safe only if the startup does well. But if the startup runs out of money, then a person may lose the job. Whereas, in MNC's there's little risk of the company going out of money, so there's more job security.

4)Bonding with coworkers[5]: Most startups are with handful of people and thus there are more opportunities for one to get to know the coworkers on a personal level. Whereas, in large companies, an employee gets to know only a handful of people he's in regular touch with due to his work.

5)Salaries: In a startup company, there may be issues on receiving salary because of low share generation. However, in MNC's there must not be any issues regarding regular dispatch of salary.

6)Recognition[5]: Working for a big corporate means an employee's hard work might get unrecognized for the majority of the time especially at a junior level position. While at a startup one will receive instant recognition because everybody, including the company itself, is working in an entry-level position.

7)Ownership & Leadership[6]: In startup culture, it's generally easy to lead a team or even become the CEO, because of less strength of people. While in MNC's, it's too difficult to lead a company. It may take ten or twenty years of rigorous work to lead the company or still may be not possible at all, because competitors are huge in numbers.

I.2 OBJECTIVE:

The aim of this project is to understand the key idea underlying the most commonly used methodologies in startups. The field of Software engineering uses many types of methodologies and practices whereas only a few of them are applicable in small scale industries or startups. Each methodology or practice has some characteristics which may be common or unique. We are to determine the methodologies and the respective characteristics.

II. DIFFERENT KINDS OF SOFTWARE STARTUP METHODOLOGIES & PRACTICES

II.1Scrum: Scrum is an agile framework for managing knowledge work, with an emphasis on software development. It is designed for teams of three to nine members, who break their work into actions that can be completed within time-boxed iterations, called "sprints", no longer than one month and most commonly two weeks, then track progress and re-plan in 15-minute stand-up method of empiricism. Scrum replaces a programmed algorithmic approach with a heuristic one, with respect for people and self-organized meetings called daily scrums[7]. Scrum implements the scientific method to deal with unpredictability and solving complex problems[8].

II.2 Kanban: Kanban is a lean method to manage and improve work across human systems. This approach aims to manage work by balancing demands with available capacity, and by improving the handling of system-level bottlenecks. Work items are visualized to give participants a view of progress and process, from start to finish - usually via a Kanban Board. Work is pulled as capacity permits, rather than work being pushed into the process when requested[9]. In manufacturing, Kanban starts with the customer's order and follows production downstream. At its simplest, kanban is a card with an inventory number that's attached to a part[10].

II.3 Scrumban: Scrumban is an Agile management methodology describing hybrids of Scrum and Kanban and was originally designed as a way to transition from Scrum to Kanban. Scrumban is a management framework that emerges when teams employ Scrum as their chosen way of working and use the Kanban Method as a lens through which to view, understand and continuously improve how they work[11]. Scrumban emerged to meet the needs of teams wanting minimize the batching of work and adopt a pull-based system. Scrumban provides the structure of Scrum with the flexibility and visualization of Kanban, making it a highly versatile approach to workflow management[12].

II.4 LSD: Lean software development is a translation of lean manufacturing principles and practices to the software development domain. It is emerging with the support of a pro-lean subculture within the agile community. Lean offers a solid conceptual framework, values and principles, as well as good practices, derived from experience, that support agile organizations[13]. Lean software development is a rendering of the larger lean movement to specifically optimize the IT value stream within the application and software development domain. Factors that are used to influence and improve the manufacturing process are translated to achieve the same goals of delivering value to customers through waste elimination and continuous improvement. Though software development differs from the manufacturing context in which lean was born, it draws on many of the same principles[14].

II.5 Agile Software Development: Agile software development is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers/end-users. It advocates adaptive planning, evolutionary development, early delivery, and continual involvement and it encourages rapid and flexible, response to change.[15] A total of 6 out of 30 primary studies in startups have used agile software development methodology. Agile methodology is an umbrella for other many methodologies which also took the attention of several authors. Agile methodology is prescribed as a preferred methodology for startups for the reason of shorter development time, their smaller company size and easiness to manage teams[16]. There is significant anecdotal evidence that adopting agile practices and values improves the agility of software professionals, teams and organizations; however, some empirical studies have found no scientific evidence[15]. Agile methodology also promotes adaptive planning which enables continuous improvement or response to change, & for being solution-focused[16].

II.6 Extreme Programming(XP): Extreme Programming is one of the agile methodology practices by software startup companies. It has a generative set of guidelines consisting of 12 inter-related principles. Some of the principles include 40 hour work week, coding standards, collective code ownership, continuous integration, pair programming & refactoring, etc[16]. XP is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. It advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted[17]. It aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development[18]. Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, code simplicity and clarity, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers[17].

III. RELATED RESEARCH

1) Jain & Nair[19] proposed "Agile Adoption in Tech Startups- Our Findings". In this paper, the researchers tried to explain that in small scale startups, the needs & goals may be entirely different from large organizations. They tried to present a comparison of existing process models, their advantages & limitations in context to small startups & thereby suggesting respective improvements to the rigorous Agile methodology which is currently followed. They have presented the findings in the form of a case study where they have adopted the suggested methodology.[19]

2) Buffardi, Robb, & Rahn[20] proposed "Learning Agile with Tech Startup Software Engineering Projects". In this paper, the researchers explain how agile software development methods align with lean startup practices, & also studies on the software engineering realism and formative assessment of students' surveyed experiences. The study found several similar student outcomes to other project models; however, it also identified limitations in the pilot with corresponding recommendations for future implementations.[20]

3) Bosch, Olsson, Björk and Ljungblad[21] proposed "The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups". In this research, they proposed operational support for software startup companies. They proposed 'Early Stage Software Startup Development Model'(ESSSDM). The model offers novel support for practitioners for investigating multiple product ideas and also to abandon any idea if needed.[21]

4) Bidewell and Sapsford[22] proposed "The (re)evolution of the lean startup methodology". In this research, they examine the validity & application of Lean startup. Here, the successful application of the lean startup methodology is brought into question, highlighting areas of the model that may need further development. They compared lean theories to strategic management theory and the external market environment and highlighted the lack of a competition orientation as an area in need of investigation.[22]

5) Saeed and Tingling[23] proposed "Extreme Programming beyond Adoption: A Longitudinal Case Study of a Software Start-up". In this research, they described the adoption of Agile methods and Extreme Programming by a software startup and found that all XP principles were not adopted equally and were subject to temporal conditions. Small releases, continuous integration and refactoring were most vigorously advanced by management and adopted by developers.[23]

6) Becker[24] proposed "Using extreme Programming in a Student Environment: A Case Study". In this case study, they tried to analyze whether or not it is possible to teach extreme Programming at a university by means of a course that presents a mixture of theory and practice within extreme programming. In this context, a case study was carried out to determine which of the practices of extreme Programming are more appropriate to university projects. The case study indicates that it is worth investing the effort to teach extreme Programming to students to enable them to apply extreme Programming or at least some of its practices in future business and university projects.[24]

7) Taylor, Greer, Coleman, McDaid and Keenan[25] proposed "Preparing Small Software Companies for Tailored Agile Method Adoption: Minimally Intrusive Risk Assessment". This study describes a minimally intrusive assessment approach for small software companies preparing for agile method adoption and tailoring in the light of key risks. The contribution of this study is that small software companies have an alternative to 'mere experimentation' with agile methods and can take reasoned steps towards their adoption and tailoring.[25]

8) Rola[26] proposed "Kanban For Small Software Projects".The paper describes the Kanban system, its origins and recent adaptations to the area of software engineering. It discusses the philosophy behind the system and rationale for using its individual components. The author proposes a research methodology consisting of qualitative methods including systematic self- observation, direct observation and a comprehensive analysis of existing archival case studies.[26]

9) Puonti[27] proposed "Tailored Project Management Framework from SCRUM and Lean Practices: Case Study of Two Colombian Companies". In this study a tailoring method is created based on literature and aided by the case study method to find which project management methodologies tailored together are most suitable to address the project management problems faced at the two case companies.[27]

10) Ahmad, Markkula, Oivo and Kuvaja[28] proposed "Usage of Kanban in Software Companies: An empirical study on motivation, benefits and challenges". In this paper, the authors tried to provide up-to-date knowledge of the current state of Kanban usage in software companies, regarding the motivation for using it as well as the benefits obtained and challenges faced in its adoption. They also investigated how the challenges identified in the study can be addressed.[28]

11) Banijamali, Dawadi, Ahmad, Similä, Oivo and Liukkunen[29] proposed "An Empirical Study on the Impact of Scrumban on Geographically Distributed Software Development". In this research, the authors tried to describe an empirical study conducted within two Software Factory settings in Finland and Italy to investigate how Scrumban can impact coordination in geographically distributed software development. It also explains that technical and security issues in the coordination of distributed projects may demand for solutions other than Scrumban[29].

12) Iqbal and Abbasi[30] proposed "Requirement Engineering Process in Agile Software Development: Review". In this research, the authors tried to discuss various approaches of requirement engineering and the ways to perform traditional requirement engineering in agile software development. The benefits and challenges in addition to the improvement ideas in distributed environments are also discussed[30].

IV. RESULTS

Table IV.1: Characteristics & Methodologies of Software Startups

CHARACTERISTICS	SCRUM	KANBAN	SCRUMBAN	LSD	AGILE SOFTWARE DEVELOPMENT	XP
Limiting Work in progress	No limit on different stages within a sprint.	WIP's limited directly per work flow state.	WIP's limited directly per work flow state.	Limited as per necessity	No limit on different stages within a sprint.	Limited as per necessity
Hindrance & Obstruction	Addresses immediately	Addresses immediately and team shall immediately solve the obstruction	Addresses immediately and team shall immediately solve the obstruction	Addresses immediately	Addresses immediately	Addresses immediately
Roles	Product owners,scrum masters, team	As needed roles, of prescribed usually.	Team plus as needed roles.	Team plus as needed roles.	Team plus as needed roles.	Team plus as needed roles.
Changes to work scope	Should wait until the next sprint.	Added as needed.	Added as needed.	Added as needed.	Added as needed.	Added as needed.
Backlogs	Sprint Backlog, Product backlog	Backlog with Limits	Backlog with limits. Just time in cards.	Backlog with only the most important user stories	Sprint Backlog, Product backlog	Backlog with limits.
Teams	Must be cross functional	Cross functional teams optional, specialized team allowed.	Can be cross functional or specialized.	Highly specialized teams not required.	Recommend cross functional	Can be cross functional .Teams work in a strict priority order.
Work Flow Visualization	Partial Workflow Visualization	Full Visualization	Full Visualization	Partial workflow	Partial workflow.	Partial workflow
Estimations	Yes(in story points or days)	No,similar sized work items.	No, similar sized work items optional.	Yes (in story points)	Yes(in story points or hours)	Yes.
Board	Simple board burn down chart	Board mapped on the process	Board mapped on the process	Simple board.	Board mapped on the process	Board for planning and progressive tracking. Boards are flexible.
Iterations	Yes(spiral)	Iterations optional.	Not mandatory, could have sprints	Yes(big design up front).	Yes(short time boxes)	Yes(iterations that are one or two long)

CHARACTERISTICS	SCRUM	KANBAN	SCRUMBAN	LSD	AGILE	XP
Prioritization	Backlog grooming done by product owner.	Prioritization is optional.	There shall be a prioritized product backlog.	Product development prioritization(using 2*2 matrix)	There shall be a prioritized product backlog.	Xp priorities things by customer. FCFS maintain.
Ceremonies	Daily scrum, sprint planning, review, retrospective.	None required; Dynamic planning.	Scrum related ceremonies is needed. Depends on iteration decision, daily scrum	Depends on iteration decision.	Daily stand-ups,sprints,regular reviews and retrospectives.	Planning games,small releases,metaphors,simple design,testing etc.
Adaptive	Yes	Yes	Yes	Yes	Yes	Yes
Adding required additional roles	Applicable	Applicable	Applicable	Applicable	Applicable	Applicable
Identifying significant problem on daily basis	Partially applicable	Applicable	Partially Applicable.	Partially applicable	Partially applicable	Applicable
Values based methodology	Yes	Yes	Yes	Yes	Yes	Yes
Working simultaneously on multiple products	Applicable	Applicable	Applicable	Applicable	Applicable	Applicable
Lean & Agile	Yes	Yes	Yes	Yes	Yes	Yes
Continuous & Empirical process	Yes	Yes	Yes	Yes	Yes	Yes
Focus on minimal marketable features	Applicable	Applicable	Applicable	Applicable	Applicable	Applicable

CHARACTERISTICS	SCRUM	KANBAN	SCRUMBAN	LSD	AGILE	XP
Pull scheduling	Applicable	Applicable	Applicable	Applicable	Applicable	Applicable
Focuses on flow management & waste reduction	Applicable	Completely Applicable	Applicable	Applicable	Applicable	Applicable
Creating a visual task management flow	Applicable	Completely Applicable	Applicable	Applicable	Applicable	Applicable
Maintaining continuous flow of work to quickly deliver value to customers	Yes	Yes	Yes	Yes	Yes	Yes
Promoting regular feedback, experiment based evolution	Yes	Yes	Yes	Yes	Yes	Yes

Metrics	Use velocity as default metrics for planning & process improvement	Use lead time as default metrics for planning & process improvement	Velocity is optional, use lead time as default metrics for planning & process improvement	Lead time,cycle time,velocity,open/close rates.	Lead time,cycle time,velocity,open/close rates.	Lead time,velocity.
Less talk, more action	Applicable	Applicable	Applicable	Partially Applicable	Partially Applicable	`Applicable
Cost effective & time-saving	Yes	Yes	Yes	Yes	Yes	Yes
Used to decrease time in development process	Yes	Yes	Yes	Yes	Yes	Yes
Figure out the requirements	Yes	Yes	Yes	Yes	Yes	Yes
Focused on Development tools	Yes	Yes	Yes	Yes	Yes	Yes
Delivery	Fast	Fast	Fast	Fast	Fast	Fast
Easy to integrate with current 'IT' system	Yes	Yes	Yes	Yes	Yes	Yes
Team size	5-9	Not specified	Small; up to 10 people	Not specified	Small to large teams	2-10
Project size	All	Small		All	Small	Small
Iteration length (week)	4	2	Longer cycles.	Not specified	4-8	2
Meeting face to face	Yes (for collocated teams)	Yes	Yes(Dependent)	Yes	Yes	Yes (communicate face to face daily)
Documentation	Basic	Better	Basic	Not defined	Basic	Basic
Daily meeting	Yes	No	Continuous work on requirements and reduce idle time of team members	Not defined	Yes	Yes
Virtual team support	Yes	Yes	Yes	-	Yes	Yes
People oriented	Yes	No	Yes	No	Yes	Yes
Process oriented	No	Yes	Yes	Yes	No	No
High risk mitigation	Yes	No	Depends	No	Yes	No
Medium risk mitigation	No	Yes	Depends	Yes	No	Yes
Information sharing through documents.	Not defined	Not defined	Not defined	Not defined	Not defined	Not defined
Teamwork	Collaborative as needed by task	Based on pull approach swarming to	Based on pull approach swarming to achieve team	Must have a vision of lean supply chain & logistics management	Based on pull approach	Simple,yet effective environment making highly

		achieve team goals	goals			productive team

CONCLUSION

In this study, we have seen that there are certain similarities as well as dissimilarities between various software development methodologies & practices. Some methodology follows a practice entirely, while the other may not follow the particular practice or even follow that partially. Startups may not strictly follow certain methodologies & principles because of pressure of time & limitation of required resources. Startups usually use some features in the existing working methodology & then add new practices from other methods. It may be even difficult for a startup to adopt a certain methodology in a well-structured way. The software development method & practices are dependent on the experience & maturity level of startups, available resources and size of the team.

REFERENCES

1) www.wikipedia.org/wiki/Startup_company [accessed on 22.10.2018].
2) www.researchgate.net [accessed on 25.10.2018].
3) Paternoster, N. , Giardino, C. , Unterkalmsteiner, M. , Gorschek, T. and Abrahamsson, P. 2014. Software Development in Startup Companies: A Systematic Mapping Study; Electronic Research Archive of Blekinge Institute of Technology. Journal of Information and Software Technology, 56(10): 1.Introduction.
4) www.glassdoor.com [accessed on 28.10.2018].
5) www.inc42.com [accessed on 02.11.2018].
6) www.alexlod.com [accessed on 05.11.2018].
7) https://en.wikipedia.org/wiki/Scrum_(software_development) [accessed on 10.11.2018].
8) https://www.scrum.org/resources/what-is-scrum [accessed on 11.11.2018].
9) https://en.wikipedia.org/wiki/Kanban [accessed on 15.11.2018].
10) https://whatis.techtarget.com/definition/kanban [accessed on 20.11.2018].
11) https://en.wikipedia.org/wiki/Scrumban [accessed on 22.11.2018].
12) https://leankit.com/learn/agile/what-is-scrumban/ [accessed on 22.11.2018].
13) https://en.wikipedia.org/wiki/Lean_software_development [accessed on 23.11.2018].
14) https://dzone.com/refcardz/getting-started-lean-software?chapter=1 [accessed on 23.11.2018].
15) https://en.wikipedia.org/wiki/Agile_software_development [accessed on 24.11.2018].
16) Tegegne, E. 2018. Software Development Methodologies and Practices in Startups- Systematic Literature Review. Master's Thesis, Faculty of Information Technology and Electrical Engineering/M3S, University of Oulu: 29-31.
17) https://en.wikipedia.org/wiki/Extreme_programming[accessed on 25.11.2018].
18) https://www.agilealliance.org/glossary/xp/ [accessed on 25.11.2018].
19) Jain, S. and Nair, R. 2015. Agile Adoption in Tech Startups-Our Findings. Jaipur International Journal of Converging Technologies and Management (IJCTM), 1(2): Abstract.
20) Buffardi, K. , Robb, C. , Rahn, D. 2017. Learning Agile with Tech Startup Software Engineering Projects. Journal of Software Engineering and Design, 3(5):Abstract, 1.Introduction.
21) Bosch, J. ,Olsson, H. , Björk, J. , Ljungblad, J. 2013. The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups. B.Fitzgerald et al.(Eds.): LESS2013, LNBIP 167, DOI: 10.1007/978-3-642-44930-7_1:Abstract, 1.Introduction.
22) Bidewell,P. and Sapsford, L. 2015. The (re)evolution of the lean startup methodology. Master of Science Thesis in Entrepreneurship(ENTN19,2015), Sten K. Johnson centre for Entrepreneurship: Abstract.
23) Saeed, A. and Tingling, P. 2013. Extreme Programming beyond Adoption: A Longitudinal Case Study of a
Software Startup. International Journal of Business and Management Invention, 2(7):Abstract.
24) Becker, C. 2010. Using extreme Programming in a Student Environment: A Case Study. Master's Project(E2010:05), Karlstad University: 2.
25) Taylor, P. , Greer, D. , Coleman, G. , McDaid, K. and Keenan, F. 2007. Preparing Small Software Companies for Tailored Agile Method Adoption:Minimally Intrusive Risk Assessment. Journal of Software Process Involvement and Practice, 13(5):Abstract.
26) Rola, L. 2011. Kanban For Small Software Projects. Master's Thesis, Faculty of Engineering and Physical Sciences, University of Manchester:1.Introduction.
27) Puonti, P. 2017. Tailored Project Management Framework from SCRUM and Lean Practices: Case Study of Two Colombian Companies. Master's Thesis, Faculty of Faculty of Engineering and Science, Aalborg University: Abstract,1.Introduction.
28) Ahmad, M. , Oivo, M. , Markkula, J. and Kuvaja, P. 2014. Usage of Kanban in Software Companies: An empirical study on motivation, benefits and challenges. 9th International Conference on Software Engineering Advances, NICE, France:Abstract, Introduction.
29) Banijamali, A. , Dawadi, R. , Ahmad, M., Similä, J. , Oivo, M. and Liukkunen, K. 2016. An Empirical Study on the Impact of Scrumban on Geographically Distributed Software Development. 4th International Conference on Model-driven Engineering and Software development, Rome, Italy:Abstract, 1.Introduction.
30) Iqbal, R. and Abbasi, A. 2014. Requirement Engineering Process in Agile Software Development: Review. Research Journal of Computer and Information Technology Sciences,2(5): Abstract.