Software Engineering Education: Converging with the Startup Industry

Nitish M. Devadiga
*Datarista Inc., Providence, RI, USA
*Carnegie Mellon University, Pittsburgh, PA, USA
ndevadiga@datarista.com, ndevadig@alumni.cmu.edu

***Abstract*— Startups are agents of change that bring in innovations and find solutions to problems at various scales. An all-rounded engineering team is a key driver for the ability to execute the entrepreneurial ambition, from building a minimum viable product to later stages of product vision. Software engineering education provides students with the knowledge to transition to mature companies with defined structure in place successfully. However, the fluidity, risk, time-sensitivity, and uncertainty of startups demand a dynamic and agile set of skills to rapidly identify, conceptualize and deliver features as per market needs. This requires the adoption of latest development trends in software processes, engineering and DevOps practices with automation to iterate fast with low governance and the ability to take on multiple roles. This paper presents a study of the dynamics and engineering at startups and compares it with the current curriculum of software engineering.**

***Index Terms*— Software Engineering Education, Startups, Startup Engineering, Innovation and Education, Software Engineering Degree Programs**

I. Introduction

Startups advance the state of technology with innovations to solve a wide array of problems in diverse domains and are a key driver for economic development. In the year 2016 $71 billion was invested in startups by VC's, and in the first quarter of 2017, $16.5 billion is invested in more than 1,800 startups [5]. More than one-third of these deals are in the software sector. Startups incorporated in the past decade or so like Facebook, Dropbox, Airbnb, Netflix have evolved into successful businesses and are setting the standards of the new way of software development in the industry. They are a catalyst to the evolution of next set of startup companies. Many early stage startups with their entrepreneurial drive and focus are building innovative products in a rapidly evolving and uncertain environment.

Engineering in startups is unique in comparison to mature organizations and late stage startups.

Sutton defines startups as creative and flexible in nature and reluctant to introduce processes or bureaucratic measures, which may result in ineffective practices [6]. Eric Ries, the creator of the Lean Startup movement, defines a startup as a newly formed company, the purpose of which is to develop new, usually innovative products or services in uncertain circumstances [22]. This paper presents a study of the dynamics and engineering at startups and compares it with the current curriculum of software engineering. This study took place over a period of two years from 2015 to 2017. Data was collected on software processes, team dynamics, development, deployment, infrastructure and hiring requirements. This data was collected from interviews with startup organizations**, meetups and job interviews. The characteristics of startups were analyzed and compared with the current state of software engineering education and identified that the current curriculum does not adequately support the requirements of the startup industry.

A. Background

With the rise of business incubators, startup schools, university entrepreneurship programs there is a constant drive to build solutions targeting diverse domains. This has caused a high demand for software engineers who form the foundation of these startups by understanding the vision, the problem space and building the product based on market needs. Startups work on a tight schedule especially in the initial phases (Seed, Series A) as they need to balance company funding, market demand, and development costs. The engineering team is the core foundation of a startup; it lays the groundwork for sales and marketing and reaching the desired consumers within expected time frame. The development of the product has a direct impact on whether it meets the customer's requirements and whether the product will ship in time. A startup requires an able development team which can grasp the vision, understand the market demands and builds a scalable product that meets the functional and systemic requirements. As startups rely on venture capitalist funding it is low on human resources to execute the task though must get to the market quickly, as time-to-market is a key metric for success at startups.

Software engineering as a discipline over the past 45 years with its curricula based on software engineering book of

*The views expressed herein are of the author alone and not necessarily the views of Carnegie Mellon University or Datarista Inc. or any of its affiliates. The information presented herein is only for informational and educational purposes. This work was done by the author while working at The MathWorks and then continued with it at Datarista Inc.

**Including the organizations author worked at, as mentioned above.

Nitish Devadiga is a graduate of Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: ndevadig@alumni.cmu.edu) and currently works for Datarista Inc. Providence, RI 02903 USA (e-mail: ndevadiga@datarista.com).

2377-570X/17 $31.00 © 2017 IEEE
DOI 10.1109/CSEET.2017.38

knowledge (SWEBOK) [10] has provided students with the necessary software engineering skills to transition and excel in the software industry smoothly. Software engineering education today provides students with valuable industry skills like project management, software estimation, software architecture, etc. which are a great asset in organizations which follow the formal software development structure [1]. Established organizations have multiple teams to perform specific jobs, and their release process is well defined, such that software engineering graduates fit right in as it charts well with the education they have received.

Particular topics in software engineering education today have drifted from the software development practices adopted in technology-oriented companies [2]. Students who joined established organizations did not find much difficulty in transitioning to the development workflow as these organizations have well set up training plans and invest heavily in getting new engineers up to speed in their engineering practices. Also, the organizations have well-documented processes for employees to get up to speed and help themselves. On the contrary in startups, the environment is dynamic, unpredictable, and even chaotic. As Bach defines a startup as "a bunch of energetic and committed people without defined development processes" [13]. Startups don't have the time and resources to invest in training activities like established organizations [6] and thus requires people who can start contributing immediately and have the knowledge and experience to comprehend the existing systems and methodologies [23]. In this paper, we identified certain gaps in the current software engineering curricula which do not equip the students with enough knowledge and skills to adopt and perform as engineers who can be part of the founding team at startups.

B. *Software engineering students*

Students pursuing software engineering degrees have varied backgrounds with most students having a background in computer science. Experience levels of students vary anywhere from 0 to 6+ years, students at the start of this range tend to have worked as software engineers or interns, and students at the end of this interval have worked as project or product managers. Students after graduation mostly join as software engineers in established technology organizations [14]. These organizations have a more computer science oriented interview process and don't emphasize much on software engineering skills as that is taken care of by their employee on-ramp and training activities [2]. Another important point to note is that students that join as software engineers in these organizations don't have much authority in the project goals and directions as those are done by the senior engineering and project management teams.

II. RESEARCH METHODOLOGY

The goal of this study is to understand the relevance and impact of software engineering in the startup industry. It was realized that a significant segment of the technology industry, i.e., startups are not adequately studied in software engineering programs such that the students are not well equipped to operate optimally at startups. A similar observation was observed by Sutton [6] that "software startups represent a segment that has been mostly neglected in process studies" [24], [25], [18]. This premise led to the study of the engineering principles and expectations of startups today and provide that feedback to academia.

To learn more about engineering at startups, the author collected information from organizations the author worked at, interviewed employees at startups, established organizations, technology meetups [17], startup events, and analyzed results from job interviews. The author interviewed senior software engineers, software engineers, CTO's in particular organizations and product managers. The interviews were conducted conversationally to let the interviewee speak at length about their experiences and opinions, and were questioned with a predefined set of topics.

A. *Data collection*

The author conducted all the interviews; the meetups attended were mainly in the Boston area. Notes were taken during the meeting, and post the interview, organized and consolidated. For notes that were not clear, participants were asked follow-up questions to understand their intent better. Meetups on subjects that were relevant to the engineering development were attended and discussed with the attendees. During candidate job interviews for engineering roles, notes were taken on the knowledge that the student gained from their graduate degree and how it aligned with the industry.

Interview questions were based on the following topics:

Description of business: About the firm and what problem the company was solving and who their customers are, this was mainly to understand the business and its domain better.

Funding: Is the company funded by VC's and how many rounds of financing the business has completed, the goal was to understand the allocation of funds for different tasks and their current burn rate. Tried to gain insight into the how volatile and fast is the work environment.

Team: Asked about the team and the experience levels of employees in the engineering group. Tried to gain insight into their team dynamics and work interactions. Most startups were in their early phases; employee count was below 20 in most cases. Gathered information on how many of those employees were part of the engineering team and their impact and contribution to the product.

Development cycle: Asked about their adopted software processes and how strictly they were adhering to their textbook versions, relevancy and application of software estimation, project management and other topics mentioned in SWEBOK [2].

Engineering practices: Asked what engineering practices the team employed and skills desired from potential future employees. Use of programming patterns, system design and architecture, coding practices, tools the development team uses, the transition from architecture to code, and testing.

Code management: Asked how the team managed code repository and whether they followed any particular workflow

for distributed development, code reviews and how new employees ramp up with the system.

Build and Deployment process: Asked about their build system, runtime infrastructure, its maintenance and their build to deployment process and the people responsible for this activity.

Engineer roles: Asked about the roles of engineers and gained insight into the experience levels of engineers who acted in these functions.

Documentation: Asked about documentation of design and architecture of the product, its alignment with SWEBOK, its maintenance, and new employee on-ramp documentation.

Customer support: Asked about customer support, if there was any dedicated support team and triaging to engineering.

Cost management: Asked how development and infrastructure costs were managed, primary owner for such tasks, level of understanding required to understand infrastructure cloud computing costs.

B. Data consolidation and theory validation

Collected data notes were analyzed, tagged into different topic categories and extracted their intrinsic characteristics.

1) Business characteristics

Vision and strategy: The image of the company defines the entrepreneurial ambition and/or potential scalable product or service idea for a large enough target market. Setting the vision along with an initial strategy was the key attribute for all organizations. In startups, this was extremely important as it had a direct impact on fundraising, especially in the early stages. This attribute was also deemed essential for hiring as founders and potential hires needed to believe in the collective vision of the company. At established organizations, though this was important, not much emphasis was given as they seemed to have high employee count and mainly needed people to do well at their job.

Time-to-market: This defines the amount of time the product will take to be released to their customer base. The timing of product release was an important attribute for startups. It had a direct impact on customer acquisition and its revenue. At established organizations, this was important however they already had a mature well-received product out in the market and had enough resources/income to sustain delays.

Limited resources: This topic defines human resources and funds that are available. It was brought up multiple times while interviewing participants at startups as startups thrive on VC funding and in most cases, don't have a positive case cash flow. The burn rate of startups vs. its growth defines how long the startups will last.

2) Engineering characteristics

Team: This topic understands the dynamics of the team, their interactions and the tools teams used for team management. Identified the roles engineers of the founding team managed and how they balanced time-to-market with funding. It was a key attribute observed in early-stage startups. The impact and influence of engineers in product development cycle and product releases were analyzed and compared with established companies. The difference in the engineering culture and product contribution were noted. Identified the software development process adopted by teams for managing the product lifecycle. Software processes at established organizations are categorized into a light-weight agile process (like Kanban) whereas in startups the software processes among founding teams were more of casual discussions and high trust, and in mid-stage startups, it was limited to quick standups and assignments of tasks using software's like Jira.

Application development: This topic was widely discussed during the interview with participants. We identified key components of application development:

a) Development velocity – This varied from startups to established organizations. At startups, product release was led by the engineering teams, and in established organizations, there were separate teams to perform these tasks. Startups adopted short release cycles with rapid prototyping and based on the customers at times selected ad-hoc development.

b) Clean code – This topic reflected the strong emphasis on writing scalable code. This was more prevalent among senior engineers and was also considered a key growth attribute.

c) Testing – This reflects the investment in testing and test infrastructure by the engineering team, testing was mainly done by the engineers themselves and QA was done at later stages post deployment to the test system.

d) Build process – This captures the build process and systems that are used by organizations.

e) Code Reviews – In most cases code reviews are done for all changes to the system, from bug fixes to feature developments. Strict code review was an accepted practice among most organizations at various scales.

f) Distributed development – Developers work concurrently, and this requires systems that would support seamless concurrent development from build to deployment. Developers make changes to the systems several times a day and ensure that the system can scale as more developers contribute to the system.

Roles: Engineers part of the founding team at startups took up multiple positions due to limited resources in the organization. Engineers did market research and contributed to new product ideas, designed system architecture and took up DevOps activities to manage infrastructure code and deployment. At established companies, engineers mostly worked within their verticals.

Documentation: Explored the extent of documentation done by engineers in startups and established organizations. Identified what pieces of the system were documented and what was deemed as important and not important.

Infrastructure and DevOps: Organizations now adopt infrastructure as code practices and any change to the infrastructure is done through code. DevOps engineers manage these tasks and ensure that the system can scale based on customer demands. This role also entitles understanding of cloud computing architecture and building redundant systems.

Hiring: It was noted that job interviews at startups were more diverse where the candidates were interviewed on varied topics like programming, algorithms, system design, infrastructure, code practices, familiarity with tools, software

architecture, project management, etc. At established organizations, the interview process was much more streamlined with algorithms, design and other computer science topics based on job profile.

III. Findings

Engineers at startups take up major responsibilities and roles [1] due to limited resources and in some cases, lack of expertise. At established technology organizations engineers were required to have a good systems level understanding to take up on the internet scale challenges, however, engineers mostly worked within their verticals. As startups want to bring the product to market as soon as possible, applications were built iteratively with rapid prototyping [15], as the resource that they are mostly lack of is time.

Most developers had a local environment setup where they could run the product locally in their system and test it. Developers write test cases for the feature they are building and test it locally in their system. Post-development engineers run their changes in a remote build system agent, and the code is tested in a fresh setup of the infrastructure. Startups use commercial offerings to avoid spending the time building such systems and maintain focus on their core product, reducing the amount of infrastructure they need to maintain. At major companies, they have their homegrown version of such systems, and engineers gradually learn this on their job [19].

Most companies follow the distributed development model like Git Flow [11] with distributed version control systems like Git and its online hosting repository service GitHub.

The quality of documentation improved with the maturity of the organization. However, in startups as well as in established organizations the documentation for in-house staff did not use formal notations.

Code review process was considered extremely important in all technology organizations. Back and forth on code review helps the team in keeping the code clean and avoiding bugs. The review process also ensures that the developer has added enough test cases to test the feature. This ensures that if in future some other engineer made changes to that subsystem they will be able to test their change without breaking the system.

Devops was another emerging topic with the rise of high internet scale products and services. Most startups organizations don't own any local infrastructure and run their applications on Amazon Web Services, or Microsoft Azure or other similar compute platforms. These cloud computing platforms provide high redundancy and availability for their compute instances with datacenters across the globe, however, to set up such an infrastructure engineer require sound systems level knowledge. These systems based on their offering have complex billing structures, senior engineers need to understand this as it's difficult for non-technical staff to grasp processor, storage, system level expenses.

IV. Comparison to software engineering curriculum

Software engineering curriculum as taught in universities [9] [10] provide students with varied backgrounds the ability to understand the development of software from its initial phase to product deployment. However, there is great emphasis on strict adherence to the software processes (traditional or agile), software estimation, analysis of software artifacts, etc. This knowledge though useful is slowly being phased out from technology oriented fast paced internet scale companies [2].

Using the above engineering characteristics and comparing it to the core courses students study [9] in software engineering programs, it was identified that,

1) Students spend a significant amount of time studying, practicing software engineering processes, when they work on projects prime importance is given to the process they adopted, time tracking, and deciding if the process they used worked for the project or not. This is observed in the software engineering curriculum at Carnegie Mellon [9], the end of semester presentation emphasizes this.

2) Software Architecture knowledge area is extremely relevant to the industry, especially in startups as the students who join startups play a significant role in defining the system architecture. Students learn architectural patterns, system modeling and its systemic properties, managing tradeoffs and formally documenting architecture models with different views. However, there is not much focus on application design and code structure to write clean, reusable and scalable code to build those architectural components. Software engineers with their software architect hats on should have the ability to stub out well-designed code structure using latest design and programming paradigms. An exemplary demonstration of this can be observed in the seminal paper on MapReduce by Google [26].

3) Code review is another development step that is not incorporated in student projects. Being able to review code and provide valuable feedback is an important attribute of a senior engineer, as it requires the knowledge and ability to digest and produce clean code. This is also a reason for students being hired as entry-level software engineers at startups and other organizations [14].

4) Software testing knowledge area teaches formal testing methods with static analysis random checkers, etc., which equips students in understanding how to test software programs, however it does not address testing of distributed systems, web services, A/B testing, testing patterns as observed in testing pyramid [20] and testing infrastructure resiliency. With the rise of micro-services architecture and single page applications, testing of the user interface and backend systems are essential.

5) DevOps and infrastructure-as-code are subjects that are not taught enough in most software engineering schools. With the rise of cloud computing and infrastructure-as-a-service, students need a practical understanding of cloud platforms. Services like EC2 (compute instances), S3 (object storage), SQS (queue service) on AWS are so widely used today that, its assumed web engineers understand it. Students need to know these infrastructure components to think and design scalable solutions to today's big data problems. This is

important for startups especially as their entire software stack is deployed in cloud platforms [8].

6) Cloud computing services with their distributed property and startups with its lack of resources and rapid development model requires a certain level of automation for management and deployment of software to cloud services. There are infrastructure architecture design patterns like immutable infrastructure etc. built with orchestration tools like terraform, chef, puppet. Understanding of such topics is essential when students join the industry, at large organizations such topics are handled during their developer on-ramp activities.

7) Software management knowledge area focuses on the project schedule, budget, time reporting, and project quality, these topics are not very relevant to the current practices in the software industry and are mostly only used by consulting organizations. Topics like process quality, project area monitoring, CMMI development are hardly used in the technology software industry.

V. Comparison to established organization processes

Established organizations have developed in-house training and on-ramp processes for software engineers to understand their development culture and software process. Graduates of software engineering that take up jobs in these companies join as software engineers [14]. At the size of these organizations, most software engineers don't have much say in the project management or architectural structure of these project as seen in startups. These engineers mostly tend to rely on their computer science knowledge for their day to day job activities. Unlike startups, they get to use the knowledge they gained in software engineering as they move up in the organization and not immediately after joining.

VI. Conclusion

Prominent in evolutionary learning theory is the idea of search, i.e., a problem-solving process in which organizations recombine, relocate, and manipulate existing knowledge in order to create new knowledge - Levinthal, 1997; Katila, 2002 [7].

Software engineering as a discipline would play a vital role in the growth of engineering at startups. Software engineering education equips students with the knowledge to analyze requirements, design and architect software, and build scalable software systems. It provides education to reduce uncertainty and detect patterns to bring order to chaos. In Carnegie Mellon, there are more than 24-degree programs in the school of computer science itself like masters in computer science, masters in machine learning, etc. and these programs are ideal for students who wish to pursue in-depth specialization in those fields [16].

However, in startups, we observed there is a high value for graduate students who have the ability to manage tradeoffs, understand different aspects of software developments and can combine software management with software development. Such talents provide startups with immense value in building a strong foundation for the company.

Startups are agents of change that bring in innovations and cutting-edge software products with broad impact on the market. Software engineering and software development form the core foundation of startups. Despite their high failure rate [21] proliferation of startups is not matched by a scientific book of knowledge [3]. To be able to contribute to software development strategies of startups with scientific and engineering approaches, the first step is to comprehend the startups existing behavior. In this paper, we study multiple attributes of startups and then identify gaps in the current state of software engineering education, which can be used a starting point to understand and build a more relevant industry-focused curriculum for students such that they can be the catalyst for development in the startups of tomorrow.

References

[1] S. Chenoweth, "Undergraduate Software Engineering Students in Startup Businesses," *2008 21st Conference on Software Engineering Education and Training*, 2008.

[2] M. Bass, "Software Engineering Education in the New World: What Needs to Change?" 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), 2016.

[3] C. Giardino, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, "Software Development in Startup Companies: The Greenfield Startup Model," *IEEE Transactions on Software Engineering*, vol. 42, no. 6, pp. 585–604, Jan. 2016.

[4] Kauffman Index Startup Activity, 2016.

[5] Pitchbook-NVCA venture monitor 1Q, 2017.

[6] S. M. Sutton, "The role of process in software start-up," IEEE Softw., vol. 17, no. 4, pp. 33–39, Aug. 2000.

[7] R. Katila, E. L. Chen, and H. Piezunka, "All the right moves: How entrepreneurial firms compete effectively," *Strategic Entrepreneurship Journal*, vol. 6, no. 2, pp. 116–132, 2012.

[8] https://aws.amazon.com/solutions/case-studies/

[9] http://mse.isri.cmu.edu/software-engineering/index.html

[10] P. Bourque and R.E. Fairley, eds., Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, 2014;

[11] https://datasift.github.io/gitflow/IntroducingGitFlow.html

[12] L. Zhu, L. Bass, and G. Champlin-Scharff, "DevOps and Its Practices," *IEEE Software*, vol. 33, no. 3, pp. 32–34, 2016.

[13] J. Bach, "Microdynamics of process evolution," IEEE Comput., vol. 31, no. 2, pp. 111–113, Feb. 1998.

[14] http://www.cmu.edu/career/aboutus/salaries_and_destinations/2016.html

[15] N. M. Devadiga, "Tailoring Architecture Centric Design Method with Rapid Prototyping", 2017; arXiv:1706.01602.

[16] https://www.cs.cmu.edu/masters-programs

[17] https://www.meetup.com

[18] G. Coleman and R. O'Connor, "Using grounded theory to understand software process improvement: A study of Irish software product companies," Inf. Softw. Technol., vol. 49, no. 6, pp. 654– 667, 2007.

[19] Potvin, R., & Levenberg, J. (2016). Why Google stores billions of lines of code in a single repository. *Communications of the ACM, 59*(7), 78-87. doi:10.1145/2854146

[20] https://testing.googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html

[21] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, "Software development in startup compa- nies: A systematic mapping study," Inf. Softw. Technol., vol. 56, no. 10, pp. 1200–1218, Oct. 2014.

[22] Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. New York: Crown Business, 2014.

[23] D. Yoffie, "Building a company on Internet time: Lessons from netscape," California Manage. Rev., vol. 4, no. 3, 1999.

[24] G. Coleman and R. O'Connor, "An investigation into software development process formation in software start-ups," J. Enterprise Inf. Manage., vol. 21, no. 6, pp. 633–648, 2008.

[25] G. Coleman and R. O'Connor, "Investigating software process in practice: A grounded theory perspective," J. Syst. Softw., vol. 81, no. 5, pp. 772–784, May 2008.

[26] J. Dean and S. Ghemawat, "MapReduce," Communications of the ACM, vol. 51, no. 1, p. 107, Jan. 2008.