Agile or not? Factors that Motivate Agile Development Method to Emerge and to be Popular

(STS Paper)

A Research Paper Submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia
In Partial Fulfillment of the Requirements of the Degree
Bachelor of Science, School of Engineering

Haotian Liu
Spring, 2020

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____
Haotian Liu

Approved _____ Date _____
Sharon Tsai-hsuan Ku, Department of Engineering and Society

Fable of Tomorrow

Dear tomorrow's news readers,

The year 2020 started with much turmoil. Political instability in the Middle East region raised high concerns among leaders of the globe, while later a novel virus spread in blazing speed, infected and caused death to thousands in many nations. All these were under the shadow of some wrestling between some of the world's biggest superpowers and the subsequent fear of war, starvation and poverty. It was during this time that you may have encountered many, many contradicting opinions, particularly those bumped into your eyes when you were reading news. You were shocked; panicking and shaken by this great hodgepodge of views; you don't know what to believe anymore – news reports and opinions from different sources often seemed desperately polarizing to you. Propaganda, hate speech and selective facts – these are the things you fear that you might have not filtered out but have taken in due to their sometimes perfect camouflage as truthful and sometimes provocative news reports and opinion analysis. You have lost the battle against falsified information and therefore fallen victims of political manipulation and partisan conflict.

Naturally, you would have some built-in human instincts to argue against someone who's culture, values and viewpoints do not align well with yours; if provoked deliberately, these instincts have the potential to quickly turn into motives to discriminate, dominate and even harm. Aided by constraints put on information access, these provocations are well underway in many places around the world. They often lead to bias and hatred towards certain demographics and subsequent communal and regional conflicts. In an era with great information overload, it is truly our generation's job to identify and deter the influence of such provocation.

I want you to be able to freely and efficiently access information from not one, but multiple sources, if not every single one of them. I believe the key to mutual tolerance and understanding is through seeing how things are portrayed, reported and commented differently on from every perspective across the whole spectrum: from The New Yorker to Fox News, from South China Morning Post to Wall Street Journal, from The Jerusalem Post to The Palestine Telegraph, I want you to see the whole picture of a certain news topic. It is only through seeing the whole picture that one can truly comprehend something that may easily trigger polarizing opinions. It is only through fully comprehending a particular topic / event that one can steer clear from being misled and utilized for political agendas. I want to help.

And thanks to the same set of technologies that resulted in this great information overload, help is possible. Large scale and distributed internet infrastructure allow for hundreds of thousands of news entries to be aggregated from a large set of publishers and ingested every minute. With advancements in machine learning, AI and natural language processing, analyzing and annotating this news can be done in near real-time. A platform that allows news readers to see how up-to-date news topics is reported differently across the spectrum is hence possible.

I aspire to build you such a platform. Wish me luck.

Abstract

The past couple decades have witnessed the booming of Agile software development methodology in the industry. As of 2020, almost all major tech companies that offer Software as a Service (SaaS) (Santhoshkumar, 2017) such as Google, Facebook and Twitter have adopted the Agile method in most, if not all their software engineering teams (Feitelson, 2013). Thanks to rapid evolution in both software and internet infrastructure, while the Agile development methodology gained popularity at blazing speed since it was introduced in the Manifesto for Agile Software Development (Beck, 2001) in 2001, the software engineering industry as whole also has seen a paradigm shift in multiple ways, for example, most companies has transitioned their business model from offering Software as a Product (SaaP, or COTS) to a SaaS-based model. This thesis aims to study the relationship between booming of the Agile methodology and these paradigm shifts in the software engineering industry and find out factors that motivated Agile methodology to emerge and to be popular.

Through a case study on DoubleThink, an Agile-driven tech startup that allows today's news readers to discover and identify different viewpoints on trending news topics, this thesis provides a real-time examination of how Agile method function in framing and solving real-world problems in a team of software engineers. This thesis will also use Actor Network theory to conceptualize the relationship between each human actor and non-human actant including software engineers themselves, stakeholders such as investors and product users in a simulated software engineering network, as well as the process of translation that connects the actants and actors. In addition, this thesis will also synthesis both in-person interview and online questionnaire results from a group of real software engineers from companies that may or may not be an Agile-abiding.

Background on Agile Development Method

In plain words, software development methods are nothing but a set of commonly agreed upon rules that software developers, while working in the same organization such as a company or a commercial software engineering / scientific research team tend to follow as guidelines for how to design, document, program, test and main a software deliverable (Atlassian, 2020), and more importantly, how to collaborate with other software engineers. As a part of software development process, software development methodologies have gone through a dramatic evolution over recent decades. Earlier software development methodologies (1980s ~ 2000s), including the Waterfall model and the Incremental model tend to be very plan-driven and focused largely in incremental development (instead of iterative development while compared to the Agile method) (Petersen, 2010). The Agile software development methodology first emerged as a conference memo in the Agile Manifesto published in 2001. Signed by 27 highly regarded software engineers, this document served as a bullet point-style summary of a set of software development guidelines that may precede the plan-driven model. In the coming years, their methodologies continued to gain popularity and be adopted by tech companies and words like "Scrum" and "Daily sprint" (Bass, 2014) became popular across the industry.

Contrasting to plan-driven development methodologies, Agile methodology focuses on rapid, iterative instead of planned, incubate development (Dingsøyr, 2010). Goals and milestones aren't defined nearly as clear in an Agile-driven development process as they are in a plan-driven process; instead, each developer works in tight collaboration to bring the "next" minimally viable product (the MVP) (Duc, 2016) and keep improving from it and get to the next iteration of software build / release / update. Incidentally, an Agile-abiding team typically has a shorter release / update cycle when compared to a plan-driven team. In addition, in an Agile system it is

often easier to make changes in code bases / integrating new requests as well. Agile methodology does not prioritize working software over comprehensive documentation but also puts customer collaboration over contract negotiation (Beck, 2001).

In spite of its rapid adoption growth in the past few decades, Agile method doesn't reach every organization: in software development teams where safety, security and stability are considered an absolute top priority, such as government, bank and oil / gas industry, plan-driven method still holds its place. Incidentally, Agile methodology is mostly adopted by consumer software companies such as social media (Facebook), ride-sharing applications (Uber) and consumer tech (Apple).

STS Framework and Research Methods

The focal question of this thesis attempts to ask what are the factors that motivates Agile development methodology to be emerge and to be popular. In order to provide a comprehensive answer to this question, this thesis will use the concept of translation in Actor Network Theory as a means to illustrate the relationship between each actant in this software engineering network. The process of translation in Actor Network Theory refers to the collective understanding from all actants and actors of the network that such a network is worth building and defending, as doing such will benefit most, if not all the network's participants (Callon, 1986). Actor Network Theory is particularly well-suited for analyzing the software development realm because it focus on the connection between each actant / actor and does recognize the position and value for non-human actants in the network (Murdoch, 1997): in the simulated software engineering network, such non-human actant, including the software composed by software engineers in the form of compiled codebase, often plays an important role in synthesizing relationship between network actors.

Primary dataset for this research project is collected through both in-person interviews and online questionnaire with a group of real software engineers who may or may not work at a company that follows the Agile principles.

Research Data Analysis

Some primary findings from studying interviewee responses from the two in-person interviews already conducted suggests that while Agile is becoming a norm in today's consumer-facing software industry, there are companies that still does not employ an Agile methodology. These companies which does not practice Agile tend to be further away from the definition of "tech" companies: bank (Capital One), healthcare provider (Kelly Healthcare) and defense contractors (Northrop Grumman). A common reason that seemed to prevent these companies to switch to agile in its software engineering teams seems to be the fact that these companies are largely enterprise-facing and have very high stakes in its operations: any bug / feature loss has a potential to result in large amounts of damage to property and even human life. Consequently, stakeholders in these companies choose to prioritize accuracy, reliability and safety over speed, up-to-dateness and featurefulness. To achieve these priorities, plan-driven methodology is often deemed as a better fit. Interviewee A, a senior software engineer at Kelly Health Care, shared his insights on why his team in the company still follows plan-driven methods:

> *I think it (plan-driven methodology) fits me perfectly. I work partially remote (MTW on-site and the other times from my home office) so a well-defined team schedule is much better than a some walk-in meeting / discussion. Plus I am pretty much the team leader now so I have a say on the team's dev method. Also since we are in healthcare our work is much higher stake than say like social media and other things (people may die if we make a mistake!) so*

accuracy is at a much higher priority than speed. That's why we don't lean too much towards Agile as of right now…

Another interviewee B, who works as an E3 level engineer (entry-level for college new graduates) (levels.fyi, 2020) at the NLP (Natural Language Processing) team at Facebook, argues that Agile method is at the core of Facebook spirit and it benefits him as a software developer in general:

I like the "move fast and break things" culture at Facebook. Abiding to the Agile method that my team uses also improves my interpersonal skills a lot since there is a lot of talking between employees. Plus I got to get up early every morning since there is an early morning standup in my team. Technical wise it kinda forces me to do things / pick up a new tech stack fast cuz no one will be waiting on you for a long time and if they do your performance score may drop…

This thesis also relates to the author's own personal experience of leading a small agile team as the "scrum master" to build DoubleThink News, a tech startup that allows today's news readers to discover and identify different viewpoints about trending news topics (DoubleThink News, 2020). Since its inception, this startup idea has gained a somewhat considerable amount of attention from some investors, both inside and outside of the University of Virginia. By following a set of Agile-based development guidelines, the team was able to quickly incorporate new news sources, identify, acknowledge and adapt new user / investor opinions to offer a better news feed service in a fast rolling basis. By minimizing each "dev sprint" (sub-projects that fulfills or improves upon a set of requirements) (Zhang, 2010) and having frequent sync-ups including standup meetings twice a week, the team saved a significant amount of time of writing comprehensive tasks lists and making detailed schedules about every sub-task as required by

most plan-driven methodologies. In no more than 2 months' time, the DoubleThink News project grew from a bare idea to an MVP that incorporates over 150 news sources in 20 + countries with a sophisticated yet robust news analysis backend, while it continues to gain user traction and attract new users. Fast-paced development in its early stage ensures that DoubleThink News is able to meet a series of funding deadlines that requires an MVP and continue to be supported both financially and technically and hence grow. For a small tech startup like DoubleThink News, all these wouldn't be possible without embracing the Agile principles.

Discussion

In the software engineering network there are several human actors and some non-human actants. The human actors include relevant members of the engineering sector of a company which are commonly software engineers, project managers and sale and software end users and company stakeholders such as VC investors, board members and other person or entity who has a certain amount of company shares. These actors can be classified into different interest groups: software engineers and project managers, while building the product, often seek the matching compensation in form of cash, RSU (restricted stock unit) or other benefits such as health insurance and vacation for their hard work; software end users seek an optimal balance between user experience and money spent on purchasing / using software; company stakeholders seek to maximize the rate of return on their investment (ROI). The primary actant of this network consists of the software code base itself (that typically compiles into a runnable package sold as a service). It is considered actant because it has the ability to impose significant impacts on other actors within this network. For example, buggy software (whose code base is prone to bugs and can be unstable) will significantly hinder end user experience and in some cases even result in loss in revenue and even cause physical damage. In this case, multiple actors are affected

negatively: company investors will suffer due to damaged company reputation due to its unstable and buggy software service, software engineers and product managers will be subject to the risk of being fired (for writing buggy code) and the company will suffer from potential loss in its revenue. It is therefore important to include the code base as a non-human actant while simulating the software engineering network.

The process of designing, writing and eventually shipping software or host it as a service on the internet can be considered as type 4 translation process as defined by Brun Latour, where different actors in the network associate without losing their identity and interest points (Latour 1997). Engineers, user and investors revolve around the software itself in order to maximize their own interest gains (each laid out as above) and hence form an actor-network; their will and intent to maximize their own interests thus translates into producing the best quality software, a common goal for all human actors in the network. It is also worth noting that user's perception of good quality software has also shifted dramatically over the years. Prior to Web 2.0 (Tim, 2005), good quality software typically referred to software that are bug-free and resource efficient, as most software available were meant to run on client's machines. With the industry-wide shift towards SaaS-based development and internet infrastructure evolution including the evolution into of 4G and 5G internet, more and more started to prioritize speed, user experience, responsiveness and availability. Today, "good quality" software is almost synonymous with fast, elegant and highly available software.

Establishing that the common goal for all actors in the software engineering actor network is to produce the best quality software lays a solid foundation for understanding why the Agile method emerged and became popular.

During the paradigm shift from Software as a Product (SaaP) to Software as a Service (SaaS) in software engineering industry, Agile method gained popularity because it facilitates the production of good quality software better than other traditional methodologies (Saltan, 2018). In a SaaS model, users buy subscription to the usage of a software and the software itself is typically hosted centrally (runs on provider's servers) on the cloud when compared to a SaaP model, where users buy instances of a software itself and run it on their machines. Some prime examples of SaaP is Microsoft Office and professional software such as Adobe Photoshop. In contrast to SaaP, SaaS includes cloud-based service such as Google Doc, Uber and Airbnb. SaaP and SaaS are subjected to very different user expectations in service speed / quality. For example, when encountering a problem / bug in Microsoft Office, a user only has the choice to file a service ticket or to a post on Microsoft's Office support forum; once the bug is fixed, he will have to re-download the whole Office distribution and re-install it. This process typically takes days, if not weeks. Almost certainly no one will wait days or weeks for problems occurred in a SaaS, such as Uber or Robinhood to be fixed, as these services are usually time-sensitive and requires constant high availability. Hence, in a SaaS model, development teams are required to respond to service problem and user requirement changes much faster than in a SaaP model.

Agile method gives software developers the competitive edge to be able to respond to any production-related issues and / or changes in customer requirements in a faster, leaner fashion in every layer of software engineering. For example, a developer in an Agile team may not be required to write as much code documentation as in a plan-driven team (Agile principle prioritizes working software over comprehensive documentation); a project manager in an Agile team may not need to report to his manager about every service ticket raised by customers (Agile principle prioritizes individuals and interactions over processes and tools). Consequently, an

Agile team can move faster and leaner with less administrative overhead / responsibilities blocking the way, thereby achieving shorter update cycles and release iterations. This may help explain why the Agile methodology gained such popularity after it was formally introduced in 2001.

Conclusion

By studying the translation process in a simulated software engineering network, interviewing real software engineers of a diverse background and incorporating the author's own personal experience of leading a highly agile team as the Scrum Master, it is reasonable to draw the conclusion that the emergence and popularity of Agile methodology can be attributed to the industry-wide shift from SaaP to SaaS and subsequent customer demand of fast, elegant and up-to-date software. However, adoption of Agile principles is also clearly dependent on a company's software product priorities.

References

Feitelson, Dror G., et al (2013). Development and Deployment at Facebook. IEEE Internet Computing, vol. 17, no. 4, pp. 8-17, July-Aug. 2013, doi: 10.1109/MIC.2013.25.
Atlassian (2020). What is Software Development? Retrieved May 7, 2020, from https://www.atlassian.com/software-development

Petersen, Kai, et al (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices. Retrieved May 5, 2020, from https://link.springer.com/article/10.1007/s10664-010-9136-6

Beck, Kent, et al (2001). Manifesto for Agile Software Development. Retrieved October 28, 2019, from http://agilemanifesto.org/

Dingsøyr, Torgeir, et al (2012). A decade of agile methodologies: Towards explaining agile software development. Retrieved October 29, 2019, from https://reader.elsevier.com/reader/sd/pii/S0164121212000532?token=63097B9B21147C8122D7BF0C3150BBF348BE88254B4BDF9251A7969F65E4FCDE10FDBDEE293AAFD983B6F38593728B3A

Callon, Michel (1986). Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St Brieuc Bay. John Law (ed.), Power, Action and Belief: A New Sociology of Knowledge (London: Routledge & Kegan Paul).

Dingsøyr, Torgeir, et al (2010). Agile Software Development: An Introduction and Overview. Retrieved October 28, 2019, from https://www.researchgate.net/publication/234274661_Agile_Software_Development_An_Introduction_and_Overview

Duc et al (2016). Minimum Viable Product or Multiple Facet Product? The Role of MVP in Software Startups. Retrieved October 29, 2019, from https://link.springer.com/chapter/10.1007/978-3-319-33515-5_10

Bass, Julian (2014). Scrum Master Activities: Process Tailoring in Large Enterprise Projects. Retrieved October 29, 2019, from https://ieeexplore.ieee.org/abstract/document/6915249

Murdoch, J. (1997). Inhuman/Nonhuman/Human: Actor-Network Theory and the Prospects for a Nondualistic and Symmetrical Perspective on Nature and Society. Environment and Planning D: Society and Space, 15(6), 731–756. https://doi.org/10.1068/d150731

levels.fyi (2020). About Facebook. Retrieved May 6, 2020, from https://www.levels.fyi/salary/Facebook/SE/E3/

Saltan, Andrey (2018). Engineering and Business Aspects of SaaS Model Adoption: Insights from a Mapping Study. Retrieved March 9, 2020, from https://tutcris.tut.fi/portal/files/17801463/SiBW2018.pdf#page=126

DoubleThink News (2020). DoubleThink News | About us. Retrieved May 5, 2020, from https://doublethinks.com/about

Y. Zhang et al (2010). Agile Model-Driven Development in Practice. IEEE Software, vol. 28, no. 2, pp. 84-91, March-April 2011, doi: 10.1109/MS.2010.85.

Santhoshkumar, P. (2017). Changes in Necessities Trade after Migrating to the SaaS Model. Retrieved March 9, 2020, from https://www.irjet.net/archives/V4/i11/IRJET-V4I11233.pdf

Ma, Dan (2007). The Business Model of "Software-As-A-Service". Retrieved March 6, 2020, from https://ieeexplore.ieee.org/abstract/document/4278733

O'Reilly, Tim (2005). What is Web 2.0? Design patterns and business models for the next generation of software. Retrieved March 6, 2020, from https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html

Latour, B. (1997). On actor-network theory: A few clarifications. Retrieved March 6, 2020, from http://www.bruno-latour.fr/sites/default/files/P-67%20ACTOR-NETWORK.pdf

Appendix: in-person interview questions

In-person interview questions are selected as a subset from a list of interview questions as follow:

1. How do you like your company's Agile method? How does it sharpen / impede you technical / interpersonal skills?
2. How do you like your company's (traditional) software development method? How does it sharpen / impede your technical / interpersonal skills?
3. How does the software development method of your company / team affect your initial decision to join the company / team? Does it impact your decision at all?
4. How do you think your company's current approach can be improved? What's not to like about it?
5. How do your colleagues like about your team / company's software development methodology?
6. What is your ideal software development workflow?
7. What would you say to your manager during an one-on-one about the team / company's current software development method?