Thesis Project Portfolio

Agile in the Workplace: Personal Technical Experience Under Agile at a Fintech Startup

(Technical Report)

The Social and Personal Implications of the Agile Coding Methodologies on Computer Science Workplaces

(STS Research Paper)

An Undergraduate Thesis

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Aaron Ponnraj

Spring, 2022

Department of Computer Science

Table of Contents

Sociotechnical Synthesis

During my technical experience as 1787fp as a Full Stack Software Engineering Intern, I noticed the discrepancies present in the work methodology used. Coding methodologies are workflow methods used in the computer science work space. As methods transform with the programming industry's progressession, the agile methodology serves as the modern-day workflow, succeeding the plan-driven water methodology. There were social and organizational implications that came with the agile shift. Agile served to meet rising demands by focusing on speed and flexibility. The agile environment may be somewhat unconventional, so it requires employees to change their work habits.

We have encountered a paradox, now working under agile but having been trained under waterfall. Continuing to work under conditions meant for a plan-driven methodology is bound to conflict with agile, as the two differ from each other by a great deal. In my own workplace, I noticed elements of both negatively conflicting with one another, creating a slowed down, somewhat unproductive environment. As an example, the lead engineer had created the project our team was to be working on, and did not provide documentation nor communicate the components of the project, causing re-teaching of concepts and tools to take up a majority of the time as opposed to developing. Not only this, but we must understand how Agile has been reflected in work environments today, and understand just how well it has been implemented in common workspaces. The best way to evaluate this concept is to look at prior exposure within the programming workspace (CAST, 2020); In my technical project, I will reflect on my own experiences in the workspace under an Agile methodology.

During the Summer of 2021, I worked under what was technically an Agile work methodology. The perception of agile and its explicit effects will be cross-examined with my

own experiences, and in turn will create some kind of verification on its actual effectiveness in small workplaces.

Agile in the Workplace: Personal Technical Experience Under Agile at a Fintech Startup

CS 4991 Capstone Report, 2022
Aaron Ponnraj
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
amp7yqb@virignia.edu

Abstract
1787fp is a fintech startup located in McLean, VIrginia, existing under a proprietary application–*1787fp*. This app serves as a financial planning tool targeted towards young adults. I served in a technical internship at the startup, primarily refactoring the app. Here, I utilized my own coding knowledge to work directly with the Xamarin.Forms framework and AWS tools in working in a small engineering team, getting hands-on experience with agile methodology–and the pitfalls with the methodology's implementation. The internship was successful, and moving forward, the application was to be recreated using another framework, adding more features.

1. Introduction

We have implemented many different coding methodologies as the computer science workplace has evolved, causing constant changes and restructuring. The modern methodology, Agile, currently serves as the most important restructure for development. Prior to this, companies primarily used the waterfall approach, which was typically very plan driven and did not account for potential errors. Due to increasing client demands and more requirements for flexibility, there was a necessity to restructure the methodologies in place (Oleksandrova , 2021). Progressively, programming became more incremental and rapid, putting less emphasis on planning and more on adaptability.

2. Related Works

Continuing to work under conditions meant for a plan-driven methodology is bound to conflict with agile, as the two differ from each other by a great deal (Rodov, 2016). It is essential that we acknowledge both methodologies respectively, and try to define conditions of Agile and Waterfall into two distinguished sets. Not only this, we must understand how Agile has been reflected in work environments today, and understand just how well it has been implemented in common workspaces.

Agile is implemented in many programming work settings today; this is true throughout the industry, ranging from big tech companies to start ups like my own. As to gain further context, I need to delve into my personal experience and reflect on the overall productivity I saw as a team. Given that I had simply worked in a startup it was

hard to measure relative productivity and see how much was achieved as there were not any alternate teams to compare against. Going forward, it is assumed that most startups face similar issues as this one. Many other major tech companies implement agile, and are able to do it seamlessly.

The best way to evaluate this concept is to look at prior exposure within the programming workspace (CAST, 2020); In my technical project, I will reflect on my own experiences in the workspace under an Agile methodology. During the Summer of 2021, I served in a technical internship where I worked as a Full-Stack developer under what was technically an Agile work methodology. The perception of agile and its explicit effects will be cross-examined with my own experiences, and in turn will create some kind of verification on its actual effectiveness in small workplaces.

3. Process Design

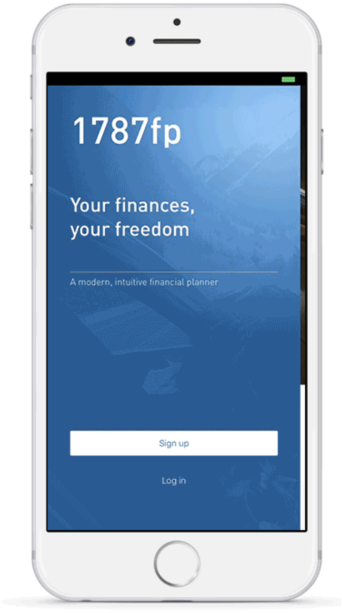


Figure 1: 1787fp application user interface (1787fp, 2021)

In this experience, I took part in a local fintech startup, 1787fp, as a Full Stack Software Engineer Intern. The internship I participated in lasted between June 2021 and August 2021, serving in a remote setting. The startup existed under a proprietary app; this app was created as a financial planning tool to be used by young adults as a means of tracking finances, managing investments, and setting financial goals. Some features included tracking credit and creating asset savings. The app was created using a Xamarin Forms framework, existing on iOS and Android. The app also relied on AWS User Pools as its form of user authentication.

I worked on a team with one other intern, essentially debugging and performing work on the company's proprietary app. Working under an agile methodology, my team performed sprints every two weeks. These sprints would entail a check-in to the company CEO, where my team would receive feedback and tickets to follow. In the course of this internship, I worked directly with the lead engineer, working primarily on the front end. In particular, I worked on bugs pertaining to user authentication and interactive screen elements (buttons and text boxes). My team and I also used GitHub in sharing and pushing codebase changes.

4. Results

The first ticket my team received was an issue pertaining to user login; upon entering credentials, a user would be unable to move on to the next screen upon clicking the "Login" or "Sign Up" buttons. The app was

engineered by a particular software engineer, who served as our lead engineer. Though created extensively, the app lacked documentation, and communication was limited with the lead engineer. This made it difficult to develop efficiently, as the team was completely dispersed.

By the end of the internship term, my team and I had refactored the app and further prepared it for its public release. We fixed the login feature by directly working with AWS User Pools, and we had modified the front end elements for some features of the app including budgeting goals and investment portfolio.

In this workplace, I noticed elements of agile and waterfall negatively conflicting with one another, creating a slowed down, and somewhat unproductive environment. The lead engineer had created the project our team was to be working on, and did not provide documentation nor communicate the components of the project, causing reteaching of concepts and tools instead of developing to take up a majority of our time. This seemed like an example of failed agile development, evident by the lack of efficiency.

5. Conclusion

Completion of the internship allowed for further development and refactoring of the app in height of its public release. By the end of the technical internship, the app's progress had moved accordingly, only occasionally being affected by Borno's tasks regarding the logistics of the 1787fp startup itself. Specifically, the issue regarding the user authentication, a ticket that took quite some time, ended up resolved. Even so, the workflow of the startup went unexpectedly, where communication was limited and tasks were assigned disproportionately within the team of interns. There was also a great learning curve in the internship, given the framework and tools associated with the app were fresh and were not taught under my university's curriculum.

6. Future Work

To follow my internship term, the app has a pending feature; the credit report. This feature will likely be complex to implement, relying on API's to link users to their respective banks from the 1787fp app. Along with this, the app is still intended to be modified under the Google Flutter API. However at this time, it seems that there has been little development and it is unknown to me whether or not these intended features have been fully implemented into the application.

Moving forward, I hope to apply this experience in future opportunities, using this internship as a backbone of my relevant work history.

7. Acknowledgements

Jean Borno, CEO of 1787fp allowed for this technical experience to occur, and has ownership of the 1787fp app. 1787fp offered the true startup experience, which can also be accounted for by my team's lead engineer and coexisting intern alike.

References

CAST. (2020). The Lean Development Methodology: Decrease Costs, Effort, and Waste. Default. Retrieved March 4, 2022, from https://www.castsoftware.com/glossary/lean-development.

(2021). Home. 1787FP. (n.d.). Retrieved April 6, 2022, from https://www.1787fp.co/

Oleksandrova, O. (2021, September 9). Infographic: A brief history of software development methodologies. Intetics. Retrieved March 3, 2022, from https://intetics.com/blog/a-brief-history-of-software-development-methodologies/.

Rodov, A. (2016, May 13). Blending Agile and Waterfall The Keys to a Successful Implementation. Project Management Institute. Retrieved February 4, 2022, from https://www.pmi.org/learning/library/blending-agile-waterfall-successful-integration-10213.

The Social and Personal Implications of the Agile Coding Methodologies on Computer Science Workplaces

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 for the Degree

Bachelor of Science, School of Engineering

Aaron Ponnraj

Spring, 2022

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 _____

Aaron Ponnraj

Approved _____ Date _____

Hannah Rogers, Department of Engineering and Society

Abstract

Agile is the primary workplace methodology presently used within the programming industry; agile relies on cyclical, incremental development, prioritizing speed and efficiency, due to the increased demand of projects, contrasting greatly to its predecessor, the waterfall model–a methodology that relies greatly on planning and finite measurement. This creates inexplicit difficulties for workplaces implementing agile. This thesis examines the agile methodology in its implementation to the computer science workplace. Investigation requires us to look into the general agile experience provided by experienced managers and engineers alongside my own personal technical experience. We will study interactions among actors in an overarching project. These actors can be interpreted as components of a project, pertaining to outputs, demands, and employee experience. Agile, counterintuitively, has considerably been difficult to manage by experienced employees, as adjustments actually shifted work environments considerably. With this also came geographical shifts and the inability to properly measure results. Though there is no direct solution to answer these challenges, it is crucial to point out the inadequacies of the current methodology and limit difficulties it may produce.

Introduction

As the computer science field has progressed, workflow methods known as coding methodologies developed alongside. As methodologies progressed alongside the programming industry, the agile methodology eventually became the industry's dominant work methodology. Agile's introduction greatly contrasted to the prior waterfall methodology (a plan-driven approach) radically shifting the workplace. Patel (2016) says it is due to the nature of agile, and how it drastically flips qualities within a workplace. As demands for software products increased, projects needed to be expedited. These demands were not satisfied under the waterfall methodology; agile served as a solution, where it focused on speed and incremental development (Wells, 2013). The agile environment may be somewhat unconventional, given its little emphasis on necessary components–project deliverables and documentation. Consequently, workplaces encounter difficulties when integrating agile. Components, or actors, in this overarching network are completely restructured as a result of switching to agile, and their interactions with one another shift greatly, requiring further analysis.

The shift to agile was monumental in the programming industry. Agile considerably contrasts its predecessors, creating the need for substantial restructuring. This caused a cultural change that has extended programming to more individuals worldwide (Charron, 2005) . Workers were able to collaborate at a larger scale while also maintaining connections to one another. Also, teams were able to swap individuals as they pleased without need for complex restructuring (Patel, 2021); this also gave flexibility to those who wanted to move workplaces. Implications of agile on development environments can be seen in the development of jobs over time. The methodology deserves room for research in order to better understand its implications.

Scope

Agile is assumed to be the preferred methodology in the present programming industry, given the complexity and growing amount of demands for projects. The agile implementation has been seen working positively in most cases, however the same could not be said in the cases of start-ups and companies initially implementing agile into their own environment. The aim of this report is to highlight the issues present for workplaces within the agile space, and offer solutions to ongoing challenges concerning the adoption of agile in a workplace. Not only that, we should mostly consider the implications that result in the switch to agile, rather than the use of it alone. We would not highlight environments that have already integrated agile, but rather those that are undergoing a transition to an agile work methodology. This allows us to understand Agile's implementation as it stands, and whether or not difficulties exist.

Actor Network Theory

This paper will explore the implications of the switch to agile in the workplace and how it modified and continues to affect employee circumstances. I will explore the issue of agile implementation using an ANT framework, a methodological approach that examines actors within a given network at a completely objective level. These actors, human or nonhuman, are examined in their interactions within a network, and the assessment of this network should reveal its true nature (Jones, 2009). We will draw directly on Latour's interpretation of ANT from *On actor-network theory: A few clarifications*. He presents the notion of ANT from an engineering perspective; here actors are not limited to human entities, but they can be viewed as circulating objects present with the flow of a network. Latour describes a network as an entity in which

actors interact with one another, correlating to the concept of a single project in our analysis. We will use this interpretation of ANT in analyzing agile's effects on the computer science industry.

This analysis will dive into the programming workplace and the ways in which it is affected by the shift from waterfall to agile. Actors will be analyzed on how they function in an overarching network. Investigating the programming workplace under the ANT framework requires us to consider an actor-network that can be translated to the scope of our analysis; the network is identified as project development, where a project corresponds to a team's overarching task for an allotted amount of time. Actors present within project development include nonhuman, measurable outputs within the workplace, demands within a project, and cultural components of the employee experience.

In this thesis, I will inspect the actors present within this conflict to better understand it and potentially find some kind of solution to this issue, drawing upon experts of the agile methodology. These experts include those who have managed under agile, alongside those who introduced the methodology to the workplace. Drawing upon conclusions from credible sources will allow us to recognize agile's effects on a project's development, and understand how components of a project interact with one another. I will also look into my own experience in an agile workplace, and how it compares to that of experts. STS knowledge will aid in this reflection; specifically, I will use the actor network theory framework in order to narrow the scope of agile's effects to cultural, organizational, and technical levels of the programming industry. Discovering these actors present in the agile workspace will allow me to have a basis in understanding in which aspects agile has influenced work environments. From this research, I hope to put the implications of agile methodologies in layman terms, and have a complete understanding of said implications.

Rationale

Agile greatly contrasts its predecessor–Waterfall leading to difficulties when employees undergo its implementation. If the situation is not resolved and we are not able to properly grasp agile methodology in its entirety, then we can expect workflow difficulties, causing a decline in the software industry. We must acknowledge both methodologies respectively, and define conditions of the methodologies into two distinguished sets. We can share this knowledge with those it is relevant to in order for employers to properly implement such techniques in their workplaces.

I too realized the adversities of agile in my own workplace; I performed as a Full Stack Software Engineer at a local startup called 1787fp, which directed its project development under the agile methodology. Under this methodology, tasks were not completed efficiently. This is a constant theme in most work environments; this resulting lack of knowledge of Agile will create tension in the workplace and negatively affect those within it. Rodov (2016) says "it is challenging to accommodate changes [to agile]." Agile intends to increase productivity by providing flexibility, receiving requirements and making changes to a product. However, this positive effect is present solely in workplaces who have implemented agile fully. We may gain insight on agile's implementation upon analyzing my own experiences, alongside the general experience.

Analysis

Continuous Workplace Changes

Switching to an agile methodology holds several inherent disadvantages, as a shift as monumental as agile greatly flips the workplace from a tangible perspective. Traditionally,

workplaces do follow a waterfall program; even growing up students are taught under a strict regime of planning as opposed to openness and flexibility. Planned paths are ingrained in our work habits, making the shift to agile, a plan that relies on eliminating planning overall, rather difficult. Author Rachaelle Lynn offers an interesting article on these challenges, offering the five key disadvantages of the agile shift:

1. Poor resource planning
2. Limited documentation
3. Fragmented output
4. No finite end
5. Difficult measurement

Projects under the agile methodology are intended to exist for a non-finite amount of time, making it difficult to anticipate the amount of resources necessary to do so. This includes the lack of documentation provided for a project. Agile does not rely on producing documentation throughout development. Given these projects are built on a continuous, cyclical schedule, creating some kind of initial is an irrelevant task, as the content of the project may be modified completely by its completion. Given the incremental nature of agile, projects have no particular result. Agile is not measured by the process but rather the results of its increments; this creates a fragmented output. Teams work on each component in different cycles, preventing the resulting output from being a cohesive unit (Lynn, 2021). Demands also change throughout development, making the relative success of one increment alternatively become unsuccessful. The combination of these inadequacies of planning under agile create a project that does not exist under a defined scope. Based on these findings, actors in the overarching network are consequential in their interactions; continuous demands result in agile's incremental nature, in turn limiting outputs, documentation, and project scope. Agile inherently produces negative

interactions among a project's components, evident by the relationships of components described by Lynn.

Agile Experience of 1787fp

Agile was used in my own technical experience. During summer 2021, I served as a Full Stack Software Engineer Intern at a startup 1787fp, working on a small team directly under the company's boss. In this workplace, incremental development, or sprints, were used in keeping check of tasks throughout an ongoing project's development. An overall goal existed for the project, but the tasks of each sprint changed sporadically; this was mainly due to other inadequacies in the workplace. The project had already existed under an independent contractor, where it had been developed in its entirety, more closely following the waterfall methodology. The project existed with little documentation, given the cyclical nature of the agile methodology. This also resulted in a rather shaky foundation, relying heavily on the employee to quickly learn and evaluate the project with little resources to do so. Subsequently, the project's development became, requiring constant explanations and notes from this prior contractor. These factors hindered the project's development as a whole, where each sprint's goals extended to the increments to follow.

This follows the general agile experience, where limited documentation and resources contribute to an overall hindered project. Agile, intended to be quick and adaptable, counterintuitively becomes ineffective when improperly implemented into the workplace. Similarly to the general agile experience, actors negatively impacted this network as a whole. Although, this is due to the inadequate implementation of components, where sprints could not keep up with issues produced as the result of limited resource planning.

Worldwide Development

The agile methodology has the ability to work with "geographically dispersed teams" as a way to complete projects at a global scale. This also allows workers to accept differences in work culture based on project environments. In differing cultures, four key ideas are considered: establishing the best technology for all involved, identifying communicative hindering factors, designing a solid framework for development, and updating expectations routinely (Ventresca, 2020). Project scale has increased immensely, as agile allows for a great deal of flexibility. Ventresca writes, "The agile environment exaggerates the need to be flexible, accept change openly and recognize that real-time interactions are the cornerstone of success." This creates difficulties in ensuring consistency in the workplace. As a paradox, Ventresca also mentions that any project under the Agile Methodology needs clear expectations and ground rules. Flexibility and groundedness conflict with one another, which fails to encourage adjustment.

Outputs are diminished as a result of agile, deterring the workplace as a result. Agile lacks qualities of consistency and measurable results, both which are crucial to the success of a project. Reliant on adaptation, agile requires its employees to remain subject to change, making project development considerably inefficient for those experienced under the waterfall methodology. Demands, altering in each increment of development, result in a fragmented project as a whole, discounting a project's documentation and schedule. This impacts the employee experience as a whole, given the emphasis on adaptability and flexity. Employees who previously worked under the strict waterfall find it difficult to adapt to more fluid agile methodology. Actors within the frame of project development shift dramatically with the switch to agile, creating a workspace with limited resources.

A Solution to Agile's Problems

These challenges can be navigated and mitigated through the use of a lean methodology; The lean methodology emphasizes reducing as much waste as possible in a product's development. This generally entails disregarding extraneous and large up front specifications that would waste time, so that work is done efficiently and goal oriented. Anything that "reduces delivered business value," by wasting effort or time, is considered waste (C.A.S.T. n.d.). This development can be thought of as a more agile approach to agile, where there is flexibility to make decisions at the last minute.

The evolution of agile serves as a useful tool to notice the progression of agile in the workplace. Its dynamic has shifted significantly over time, and continues to evolve as demands change. There have been many attempts to adjust to agile even in a dynamic workplace. David Miller offers an interesting piece on a team's approach to adjusting to an agile methodology. The first of these changes is a management shift. Consider an agile change management style, which breaks down development into three levels: project, release and iteration (Miller, n.d.). This management style results in a reduction to organizational change management. Although, a workplace lacking organizational change has an increased risk overall. Creation of a completely new management style while eliminating a tested one creates a vast shift in the workplace.

Counterargument

An argument is present within the general byproducts found when evaluating the implications of agile; the consensus is that agile allows for flexibility and freedom for the worker, but switches to agile cause some kind of shift on a worker's preferences. Workers must reevaluate their own values; they must also readjust to a more modern mindset of production

causing a great deal of restructuring in these, what seemed to be linear, environments. Along with this, agile is still not used to its fullest, due to an employer attachment to waterfall. It is also unclear how agile and waterfall continue to interact with each other in the modern day, considering the two oppose each other greatly. Switches to agile do increase demand from user and shift requirements more than ever before (Patel, 2021), and a fair amount of employees who have used planned methodologies will require retraining and employers must restructure; in other words, actors within an agile space may interact as intended given proper restructuring to the workplace. Agile and plan driven methodologies converse each other greatly, but elements from both are still used coincidentally (Rodov, 2016). In order to turn a new leaf and properly implement agile, we must eliminate prior thoughts of waterfall. Failing to do so would create some unprecedented slow downs in work, given that the mentalities of the two would negatively conflict with one another.

When agile is initially implemented to an environment, issues remain as a temporary limitation due to the methodology change; the industry experiences this presently, and it continues to remain a concern. There does not seem to be a direct solution to make this transition smooth, but there are ways to aid the process. Employer's may invest more resources in training employees and emphasizing a change of thinking (Ray, 2017). Issues in this transition often stem from employee's clinging to legacy approaches, utilizing traditional workflows in project development. Given proper instruction, members of the workplace can transition to agile while limiting any difficulties.

Conclusion

Using an ANT analysis, it is evident that agile contains several flaws in its attempts to serve as the successor to the waterfall methodology. Actors present within the project development network have changed considerably from a methodological shift to agile, creating issues in resource planning and efficiency. Implications have extended beyond a tangible sense, and have also gone to affect employees in terms of their efficiency and mindsets. Acknowledging these flaws present in the 'agile transition' we face will allow employers to correct their misconceptions and properly implement agile as it was meant to be. Though there are positive consequences of utilizing the agile methodology, the problem remains in the fact that the workplace shift in implementing agile results in an overall decline. Moving forward, employers should better recognize these side-effects that come with moving towards an agile methodology, better preparing for this change and considering recommendations like those offered by David Miller. These include, but are not limited to, shifting management and employee mindsets in terms of planning and evaluating a project's life span.

References

CAST. (n.d.). *The Lean Development Methodology: Decrease Costs, Effort, and Waste*. Default. Retrieved October 4, 2021, from https://www.castsoftware.com/glossary/lean-development.

Charron, R. (n.d.). *Effects of agile practices on social factors*. ResearchGate. Retrieved October 5, 2021, from https://www.researchgate.net/publication/220631372_Effects_of_agile_practices_on_social_factors.

Infographic: A brief history of software development methodologies. Intetics. (2021, September 9). Retrieved October 7, 2021, from https://intetics.com/blog/a-brief-history-of-software-development-methodologies/.

Jones, O. (n.d.). *Actor network theory*. Actor Network Theory - an overview | ScienceDirect Topics. Retrieved May 1, 2022, from https://www.sciencedirect.com/topics/earth-and-planetary-sciences/actor-network-theory

Latour, B. (1996). On actor-network theory: A few clarifications. Soziale Welt, 47(4), 369–381. http://www.jstor.org/stable/40878163

Livingston, K. (2019, May 31). Why agile fails in some companies – is there any solution? Future of work. Retrieved February 15, 2022, from https://geekbot.com/future/why-agile-fails-in-some-companies-is-there-any-solution/#:~:text=One%20of%20the%20biggest%20causes,Agile%20processes%20won't%20work.

Lynn, R. (2021, May 21). What are the disadvantages of Agile? Planview. Retrieved February 15, 2022, from https://www.planview.com/resources/articles/disadvantages-agile/

Miller, D. (n.d.). How to incorporate change management into agile projects. How to Incorporate Change Management into Agile Projects. Retrieved February 15, 2022, from https://blog.changefirst.com/how-to-incorporate-change-management-into-agile-projects

Miller, G. (n.d.). Learning. Project Management Institute. Retrieved February 15, 2022, from https://www.pmi.org/learning/library/agile-problems-challenges-failures-5869

Patel, M. (2021, June 18). *How agile methodology transforms the organizational framework*. dzone.com. Retrieved October 5, 2021, from https://dzone.com/articles/how-agile-methodology-transforms-the-organizationa.

Ray, M. (2017, March 20). 8 ways to ease the transition from waterfall to Agile. SmartBear.com. Retrieved April 23, 2022, from https://smartbear.com/blog/agile-vs-waterfall/

Rodov, A. (2016, May 13). *Blending Agile and Waterfall The Keys to a Successful Implementation*. Project Management Institute. Retrieved October 4, 2021, from https://www.pmi.org/learning/library/blending-agile-waterfall-successful-integration-10213.

Senior Marketing Manager, R. L. (2019, November 21). *History of agile*. Planview. Retrieved October 7, 2021, from https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/history-of-agile/.

Silva, T. S. D., Silveira, M. S., Maurer, F., & Silveira, F. F. (2018, May 1). *The evolution of Agile UXD*. Information and Software Technology. Retrieved October 7, 2021, from https://www.sciencedirect.com/science/article/abs/pii/S0950584917301490.

Ventresca, P. (2020, October 6). Agile Methods and Cultural Impacts. Advanced Management Services. Retrieved October 5, 2021, from https://amsconsulting.com/articles/agile-methods-and-cultural-impacts/.

Waterfall. Smartsheet. (2021). Retrieved November 1, 2021, from https://www.smartsheet.com/content-center/best-practices/project-management/project-management-guide/waterfall-methodology.

Wells, D. (n.d.). *Extreme programming: A gentle introduction.* Extreme Programming: A Gentle Introduction. Retrieved October 4, 2021, from http://www.extremeprogramming.org/.

Wolpers, S. (2021, November 21). Agile failure patterns in organizations. Age. Retrieved February 15, 2022, from https://age-of-product.com/agile-failure-patterns-in-organizations/

Comparing the General Agile Experience to a Personal Technical Experience Under Agile

Investigating the Implications of Coding Methodologies on the Workplace

A Thesis Prospectus Submitted 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 in Computer Science, School of Engineering

Aaron Ponnraj

Fall 2021

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.

ADVISORS

Kathryn A. Neeley, Department of Engineering and Society

Daniel Graham, Department of Computer Science

Introduction: What is Agile and Where Did it Come From

Coding methodologies are workflow methods used in the computer science work space. With methods evolving over time, agile methodology eventually became considered the 'modern' methodology and now dominates the space of programming. There were social and organizational implications that came along with the agile shift. Patel (2016) says it is due to the nature of agile, and how it drastically flips qualities within a workplace. As demands for software products increased, projects needed to be expedited, which could not be answered under the prior methodology in place; agile solved this problem, where it focused on speed and incremental development (Wells, 2013). The agile environment may be somewhat unconventional, given its little regard to components thought to be necessary (documentation and set project deliverables). As a result, it takes all its members time to learn exactly how every component functions and operates.

Agile had a predecessor–the Waterfall model–in which it greatly differs from; this meant it would take time to retrain older employed individuals, possibly decreasing the overall workflow in an environment and slowing down productivity as a whole (Ventresca, 2020). If the situation is not resolved and we are not able to properly grasp agile methodology in its entirety, then we can expect delays in workflow, hurting the software industry overall. We can share this knowledge with those it is relevant to in order for employers to properly implement such techniques in their workplaces.

I realized the unfavorable result of agile in my own workplace, where tasks were not completed efficiently. This is a constant theme in most work environments; this resulting lack of knowledge of Agile will create tension in the workplace and negatively affect those within it. Rodov (2016) says "it is challenging to accommodate changes [to agile]." Agile is supposed to

increase productivity by allowing workers to be more flexible in terms of receiving requirements and making changes to a product. The negative implications of agile are evident, as suggested by my own experience, making it worth further exploring in its entirety. I will look into my own experiences in agile, and gain insight on its implementation in the workplace. Along with this, I will explore the implications present from the shift to agile, and how it affects employees indirectly.

Technical Topic: Agile Implementation From Global and Personal Perspectives

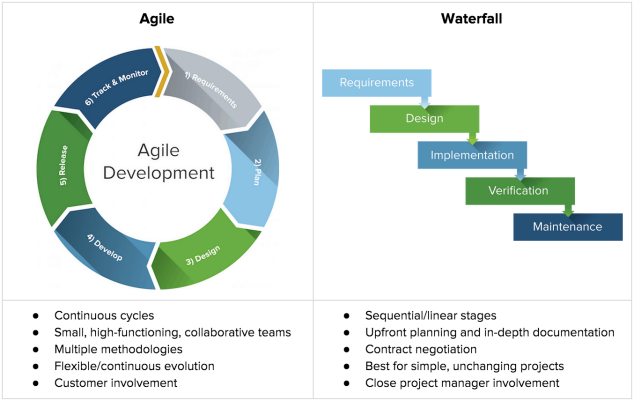
Figure 1

Software development life cycle (Sami, 2020)

Note. Visualization of the software development life cycle, showcasing the natural progress of project development in coding

Figure 2

Agile and Waterfall Visualization ("Waterfall" Smartsheet, 2021)



Note. A comparison between the agile and waterfall methodologies through visualization and description

The software development typically has set stages–requirements, design, implementation, testing, and maintenance or evolution (Sami, 2020). We have implemented many different coding methodologies as the computer science workplace has evolved, causing constant changes and restructuring. The modern methodology we call Agile currently serves as the most important restructure for development. Prior to this, companies primarily used the waterfall approach, which was typically very plan driven and did not account for potential errors. Due to increasing client demands and more requirements for flexibility, there was a necessity to restructure the methodologies in place (Oleksandrova , 2021). The new methodology put in place, Agile, corrected these modern requirements. As context, waterfall was the primary methodology previously, focusing on a plan-driven approach. A plan-driven methodology was intended to be linear, making it less flexible to last minute changes or cyclical maintenance ("Waterfall" Smartsheet, 2021); in other words, it was too narrow to satisfy the increasing demand of programming. Over time, the relevance of computer science increased, and tools used in the workplace became heavily reliant on software. The variety of software demands and the quantity in which they were requested exponentially increased. Progressively, programming became more incremental and rapid, putting less emphasis on planning and more on adaptability (Inetics, 2021). This idea of an adaptive approach is the foundation of Agile and why it is considered the epitome of development.

Agile is implemented in many programming work settings today; this is true throughout the industry, ranging from big tech companies to start ups like my own. As to gain further context, I need to delve into my personal experience and reflect on the overall productivity I saw as a team. Given that I had simply worked in a startup it was hard to measure relative

productivity and see how much was achieved as there were not any alternate teams to compare against. Going forward, it is assumed that most startups face similar issues as this one.Many other major tech companies implement agile, and are able to do it seamlessly. Gaining a better understanding in my own experience, along with those of others, is the first step in better understanding the technical scope of this problem. Lack of productivity is the precedent consequence of continuing with a lack of understanding of the methodology. We want to avoid any mishap at all costs, as Agile was meant to correct the wrongdoings of its predecessor–Waterfall.

We have been unable to implement agile overall, now working under a blend of waterfall and agile. Continuing to work under conditions of a plan-driven methodology is bound to conflict with those of agile, as the two significantly differ (Rodov, 2016). It is essential that we acknowledge both methodologies respectively, and try to define conditions of Agile and Waterfall into two distinguished sets. In my own workplace, I noticed elements of both negatively conflicting with one another, creating a slowed down, somewhat unproductive environment. As an example, the lead engineer had created the project our team was to be working on, and did not provide documentation nor communicate the components of the project, causing reteaching of concepts and tools to take up a majority of the time as opposed to developing. Not only this, we must understand how Agile has been reflected in work environments today, and understand just how well it has been implemented in common workspaces. The best way to evaluate this concept is to look at prior exposure within the programming workspace (CAST, 2020); In my technical project , I will reflect on my own experiences in the workspace under an Agile methodology. During the Summer of 2021, I served as a technical internship for 1787fp where I worked as a Full-Stack developer under an Agile

work methodology. The perception of agile and its explicit effects will be cross-examined with my own experiences, and in turn will create some kind of verification on its actual effectiveness in small workplaces.

STS Topic: Exploring the Implications of Agile in the Workplace

The shift to Agile was monumental and opened the scope of the programming world. Agile and its predecessors differ greatly from each other, causing an overall restructure in organizational matters. This caused a cultural change that has extended programming to more individuals worldwide (Charron, 2005) . People were able to work and collaborate at a global scale while also maintaining connections to one another. Also, teams were able to swap individuals as they pleased without need for complex restructuring (Patel, 2021); this also gave flexibility to those who wanted to move workplaces. Implications of agile on development environments can clearly be seen in the development of jobs over time. As an example, due to the COVID-19 pandemic, computer science workplaces were mostly unaffected in terms of actual work. This is thanks to the adaptable nature of an agile workplace. Agile deserves room for research in understanding the implications of its forthcoming.

An argument is present within the general byproducts found when evaluating the implications of agile; switches to agile cause some kind of shift on a worker's preferences. Workers must reevaluate their own values; they must also readjust to a more modern mindset of production causing a great deal of restructuring in these, what seemed to be linear, environments. Along with this, agile is still not used to its fullest, due to an employer attachment to waterfall. It is also unclear how agile and waterfall continue to interact with each other in the modern day, considering the two oppose each other greatly. Switches to agile do increase demand from user

and shift requirements more than ever before (Patel, 2021), and a good amount of employees who have used planned methodologies will require retraining and employers must restructure. Agile and plan driven methodologies converse each other greatly, but elements from both are still used coincidentally (Rodov, 2016). In order to turn a new leaf and properly implement agile, we must eliminate prior thoughts of waterfall. Failing to do so would create some unprecedented slow downs in work, given that the mentalities of the two would negatively conflict with one another.

In my research project, I will reflect on the influence on the workplace due to the shift to agile, looking back on previous employers and their experiences, and understanding how they had to deal with adversity in allowing employees to be more flexible in building products at a large scale. Not only that, I will inspect the actors present within this conflict to better understand it and potentially find some kind of solution to this issue. STS knowledge will aid in this reflection; specifically, I plan to use the ideas of actor network theory in order to narrow the scope of agile to cultural, organizational, and technical levels. Discovering the actors present in the agile workspace will allow me to have a basis in understanding in which aspects agile has influenced work environments. From this research, I hope to put the implications of agile methodologies in layman terms, and have a complete understanding of said implications.

Conclusion: Tying it Together

Through the technical portion of my project, I will properly reflect on my own experiences in the computer science workspace, under an agile methodology, and cross-examine it with the general agile experience. I will offer an assessment of agile in the workplace, based on my own experiences, as a deliverable. When gauging the STS topic, I will examine the

implications of the shift to agile, and the influence that the new environment has had on the actors present within the computer science field. These implications will focus on those that exist within the workplace, and those that are more external, more corresponding with employee emotions and mindsets. Creating and understanding these deliverables I will focus on through my project will allow me to define the problems present within the agile workspace. Acknowledging the realities of the 'agile transition' will allow employers to correct their misconceptions and properly implement agile as it was meant to be. I hope to understand why the computer science industry is struggling to fully shift to agile in order to find a solution to progress the field overall.

References

CAST. (2020). *The Lean Development Methodology: Decrease Costs, Effort, and Waste*. Default. Retrieved October 4, 2021, from https://www.castsoftware.com/glossary/lean-development.

Charron, R. (2005, July). *Effects of agile practices on social factors*. ResearchGate. Retrieved October 5, 2021, from https://www.researchgate.net/publication/220631372_Effects_of_agile_practices_on_social_factors.

Oleksandrova, O. (2021, September 9). Infographic: A brief history of software development methodologies. Intetics. Retrieved October 7, 2021, from https://intetics.com/blog/a-brief-history-of-software-development-methodologies/.

Patel, M. (2021, June 18). *How agile methodology transforms the organizational framework*. dzone.com. Retrieved October 5, 2021, from https://dzone.com/articles/how-agile-methodology-transforms-the-organizationa.

Rodov, A. (2016, May 13). *Blending Agile and Waterfall The Keys to a Successful Implementation*. Project Management Institute. Retrieved October 4, 2021, from https://www.pmi.org/learning/library/blending-agile-waterfall-successful-integration-10213.

Senior Marketing Manager, R. L. (2019, November 21). *History of agile*. Planview. Retrieved October 7, 2021, from https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/history-of-agile/.

Silva, T. S. D., Silveira, M. S., Maurer, F., & Silveira, F. F. (2018, May 1). *The evolution of Agile UXD*. Information and Software Technology. Retrieved October 7, 2021, from https://www.sciencedirect.com/science/article/abs/pii/S0950584917301490.

Ventresca, P. (2020, October 6). Agile Methods and Cultural Impacts. Advanced Management Services. Retrieved October 5, 2021, from https://amsconsulting.com/articles/agile-methods-and-cultural-impacts/.

Waterfall. Smartsheet. (2021). Retrieved November 1, 2021, from https://www.smartsheet.com/content-center/best-practices/project-management/project-management-guide/waterfall-methodology.

Wells, D. (2013). *Extreme programming: A gentle introduction.* Extreme Programming: A Gentle Introduction. Retrieved October 4, 2021, from http://www.extremeprogramming.org/.