Leveraging Agile Methodologies to Produce Quantifiable Success

(Technical Paper)

Analyzing Agile Methodologies and their Adoption Within the Workplace

(STS Paper)

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

Ryan Pope

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

Signature _____ Date _____
Ryan Pope

Approved _____ Date _____
Rosanne Vrugtman, Department of Computer Science

Approved _____ Date _____
Sean Ferguson, Department of Engineering and Society

Introduction

In many modern workplaces, employees are required to communicate across departments and professions, and are often not able to communicate in-person (due to the ever-increasing availability of remote work). Thus, interrelationships developed within the workplace can be strengthened through an emphasis on mastery of communication skills. Employees in the workplace are expected to participate in team-based environments while simultaneously optimizing both technological achievement and social wellbeing.

Existing course material, such as in CS3240 Software Development Methods at UVA, tends to take an emphasis on workflow methodologies, such as Scrum and Waterfall, and these are often implemented in a software-focused workplace. However, there are certainly some shortcomings of these methodologies that cannot be predicted by simple theory -- taking a retrospective look is critical to understanding why some forms of agility are adopted over others. Additionally, in looking at existing agile environments, we can begin to understand how progress and thought propagate (or break down) as a result of the agile environment. LaFuente and Prata (2019) mention that agile communication can often be modeled using objects -- a key part of the Scrum process is its object-oriented nature, with sprints, tasks, and 'ceremonies' building towards a team's goal. However, this obsession with efficiency also has some shortcomings, chiefly the possibility of introspection towards the process itself -- progress is not only found in the completion of tasks, but also in the ability to understand and analyze the tasks being completed.

Technical Topic

During my entry-level internship at Yext, a DC-based search engine optimization and consulting software company, I led a summer-long project to migrate Yext's consulting site

building process. I leveraged the scripting efficiency of Go to create a comprehensive repository (repo) migration tool, allowing existing company repos to integrate seamlessly with Yext's continuous integration (CI) tools. This innovation saved significant amounts of time in the site development process used company-wide. Working on this project also helped me to develop numerous soft skills, especially communication and project management. I frequently interacted with teams on both sides of the company and was able to achieve a serious impact through positive and frequent communication.

The course material from UVA's CS3240 (Software Development Methods) and CS4750 (Database Systems) prepared me very well for this environment. One of the most impactful concepts I learned in CS3240 was that a good work experience is all about communication. Being knowledgeable about the scrum methodology allowed me to easily adjust to the sprint-oriented environment at Yext and prepared me to set timely goals to plan my week of work tickets. Additionally, as much of my backend code at Yext was interfaced with an SQL database, my experience in CS4750 was especially useful, and I was often able to get a jump-start on database-related tasks.

STS Topic

While 'iterative and incremental' software development methods have existed since as early as 1957, the advent of the 'agile' methodology is more recent, in response to a demand for more "lightweight" software development processes (as opposed to the traditional, heavyweight "waterfall" approach). Agile methodologies are workplace methodologies focused around iterative development as opposed to long-form project development and aim to tackle the challenge of consistently and rapidly aligning with ever-changing customer needs and goals.

There are several methodologies under the ‘agile’ classification, but they all share the four core motivations from Williams and Cockburn’s “agile manifesto” (2003):

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Because so many differing agile methodologies exist, it is critical to examine the reasons for many companies choosing some over others, as well as reasons for the success of these methodologies. Hanslo, Mnkandla, and Vahed (2019) discuss several factors for the adoption of Scrum methodology, which has a focus on anticipating volatility in the software development process, by emphasizing the importance of team adaptability over meticulous planning. In their study, the researchers were able to compile a list of 14 factors influencing Scrum adoption, including experience, specialization, and resource management. These factors were used as the independent variables, and Scrum adoption as the dependent variable, in a statistical analysis of the results of an online questionnaire about Scrum processes.

As a result of their study, Hanslo, Mnkandla, and Vahed found that several of these factors had a significant impact on the adoption of the Scrum methodology and that these factors could be organized into four groups: individual factors, team factors, technology factors, and organization factors. The study found that an individual’s escalation of commitment, as well as their experience, had positive impacts on a team’s decision to adopt Scrum, while a higher level of specialization led to a negative impact on that decision. In terms of the overall team, sprint management abilities and teamwork were key in Scrum adoption, while over-engineering had a

negative impact. Additionally, the study found that the more complex the team's technology, the less compatible the team was with a Scrum methodology.

One key takeaway can be derived from these results -- the only factors that were found to have a significant negative impact on Scrum adoption were when team environments could not be framed as traditional software engineering -- having a microscopic focus on parts of the development process, such as having highly specialized employees, over-engineering the product, or utilizing complex technology, made Scrum a poor choice for certain teams. The reasoning for this can be drawn back to the original inspiration for Agile methodologies -- adaptability. When teams are not able to be adapted to work within volatile environments, the entire idea of an Agile methodology falls apart.

Because the purpose of Scrum is to ensure the possibility of consistent, simultaneous progress towards project goals, it is antithetical to this purpose to have a team that is not easily adaptable to new scenarios. For example, if an engineer is only able to work on a small subsection of the overall project (the engineer is highly specialized), and the coming weeks' sprinted work is focused on a different subsection, significant bottlenecks can occur, and goals may not be met because of the lack of adaptability. These bottlenecks can be avoided, however, as examined by LaFuente and Prata's (2019) study into the object-oriented nature of a Scrum environment. In the environment analyzed by the study, weekly sprint boards are laid out using Post-It notes on a whiteboard, and so tasks can be accurately estimated for a team to divide and conquer. Teams often also have a "burndown chart", plotted as 'remaining effort' vs 'day of sprint', thus showing teams how many tasks should be completed per day to 'burn down' the rest of the sprint. This reveals a key factor of Agile methodologies -- making use of objects, in this case, Post-It notes, to break down goals into bite-sized tasks and track the completion of these

tasks. LaFuente and Prata describe this as a ‘post-itly’ way of thinking -- involving a compulsion towards progress, as instead of the team having their eyes on satisfying one major goal, the goal instead evolves into moving as many objects across the sprint board as possible.

On the other hand, LaFuente and Prata also make note that Scrum has several shortcomings in practice. Because of the obsession with ‘post-itly’ tasks, there is very rarely time for research, reflection, or theorization about either the tasks or the overarching goals. The authors make an interesting proposal -- because of society’s ‘super valuation’ of efficiency, methodologies like Scrum often incorrectly conflate slowness with deceleration. Because of this, ‘processes of record, tracing-out, and immersion’ are not able to coexist with those of traditional development progress. If teams saw slowness as a way to ‘enrich the present’s experiential and conceptual density’, rather than a way to delay future goals, progress could be made on the tasks, while simultaneously leveraging slowness, to magnify a team’s understanding of their abilities, the project goals, and also the process itself.

While agile methodologies are not appropriate for every workplace and can lead to a number of shortcomings within the development process, their focus on adaptability and incremental progress leaves them widely applicable in software development as well as elsewhere. Being able to distribute tasks among a team using an object-oriented, ‘post-itly’ approach not only encourages progress on goals, but also furthers the abilities of the team members, and reinforces equal contribution within teams. If teams are to take this object-oriented approach and incorporate the embracement of ‘slowness’, progress towards the future can exist in tandem with the intensification of the present.

Next Steps

Many of the existing studies regarding Scrum development processes, including the study conducted by Hanslo, Mnkandla, and Vahed (2019) are quite restricted, not only in terms of their sample size but also in their limited perspectives into the places where these methodologies are implemented. In the conclusion of Hanslo, Mnkandla and Vahed's paper, the authors mention that future research could consider an organization's success rate at the adoption of Scrum methodologies, based on the organization's current practices. If this study were to be conducted, its researchers should aim to gain insight into the specific implementations of the processes by each company -- whether or not the company embraces 'slowness', and if the more 'post-itly' approaches tend to lead to more quantifiable success.

Additionally, in performing a critical analysis of agile adoption in workplaces, researchers should include several important independent variables: project development styles, level of teamwork, organization characteristics, and styles of customer interaction. Furthermore, the profitability and/or efficiency of the organization at the adoption of scrum is an important factor to consider, as it may reveal that organizations that can achieve a high level of efficiency without an agile workplace may be negatively affected by its introduction. This research could provide critical perspectives into what environment styles teams flourish most in, as well as how members of these teams best learn and grow as individuals within agile environments.

References

Hanslo, R., Mnkandla, E., & Vahed, A. (2019). Factors that contribute significantly to scrum adoption. *Proceedings of the 2019 Federated Conference on Computer Science and Information Systems*. https://doi.org/10.15439/2019f220

LaFuente, I., & Prata, W. (2019). (fr)agile objects: Thinking Scrum through post-it notes. *Ethnographic Praxis in Industry Conference Proceedings*, *2019*(1), 237–253.

https://doi.org/10.1111/1559-8918.2019.01283

Williams, L., & Cockburn, A. (2003). Agile software development: It's about feedback and change. *Computer*, *36*(6), 39–43. https://doi.org/10.1109/mc.2003.1204373