How to Choose Your Agile Methodology

Manage Your Expectations: How Has Agile Lived up to its Promise to Decentralize Power?

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By
Jacob Rice

October 27, 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.

ADVISORS

Kent Wayland, Department of Engineering and Society

Rosanne Vrugtman, Computer Science

Introduction

Problem: *How can Agile development be effectively used when developing a software product?*

When building a software product, the processes by which it is built are nearly as important as the technical details, as the processes define how work is done and by whom the work is done. In the modern day, most software development is done in an Agile environment. Agile is not one specific framework, but rather a class of frameworks that provide "an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches" (Atlassian, n.d.). This is in direct opposition to plan-driven methodologies which depend a lot on developing a plan to develop the software early in the process and sticking to that plan (GeeksForGeeks, 2021), leading to a lack of flexibility which Agile provides. This flexibility and iterative approach that Agile provides have led to its adoption by many modern software development companies. In fact, according to the 15th State of Agile Report, 94% of software development companies report that they are practicing Agile development (Digital.ai, 2021). With this emphasis on Agile development only increasing, it is important for us to understand how Agile processes can be implemented in a way that they help rather than hinder your developers' workflows. Therefore, it is important to understand why and how to select an appropriate agile methodology for your project. Thus, my technical project aims to formulate a set of guidelines for selecting an Agile methodology from one of several main methodologies.

In addition to understanding how to properly select and implement Agile methodologies for your software product, you also need to be able to understand the effects of implementing Agile on the people with whom you work as well as how workflows change as a result of Agile.

This is especially important when thinking about shifting your product away from more traditional development processes (i.e., plan-driven methodologies), as the nature of Agile is such that the roles of everyone on your team are subject to change to conform with the Agile way of doing things. In particular, I focus on how Agile changes the role of the manager on a software development team, as well as the extent to which preexisting power dynamics between developer and manager are changed when developing using Agile.

Technical Project

Problem: *Creating a set of guidelines for selecting an appropriate Agile methodology for your software product.*

When developing a software product, it is important to determine the processes by which the software will be developed. By doing this early in the process, you can resolve several managerial concerns (knowing when you are done with a task and what to do next). Additionally, by creating standardized processes for the development of your software product, you make it easier for multiple people on a team to work together because you minimize the amount you have to learn about the workflows of other team members and how they make decisions as all team members will instead be making decisions based on the processes defined previously. Some of the most common ways of structuring the software development lifecycle in the modern day are what we call Agile. Most importantly, though, Agile processes follow the principles established in the Agile Manifesto, which amount to prioritizing individuals and the customer as well as being responsive to change (Beck et al., 2001).

Because 94% of software engineering companies use some kind of Agile framework when developing software, it is necessary to understand how this 94% is selecting processes and whether those processes are appropriate for their situation. Further, these companies also practice a number of different kinds of Agile Frameworks. The same State of Agile report explicitly denotes the usage of 5 unique frameworks (implementations of Agile principles): Scrum, Kanban, Iterative, Xtreme Programming (XP), and Lean Startup. Further, the report mentions 2 frameworks which, are actually compositions of other frameworks mentioned here that have evolved and become frameworks in their own rights: Scrumban and Scrum/XP hybrids. While all of these methodologies are considered Agile, they all come with their own set of unique practices which may or may not mesh with pre-existing organizational values. Thus, it is important to select an Agile framework for your project with organizational values in mind. In fact, selecting the wrong process for your team may actually result in reversion to more plan-driven development methods (Crouch, 2020), removing the flexibility and other benefits that Agile provides. With all these differing frameworks, it can be difficult to select the appropriate framework for your team. Thus, it becomes necessary to understand when and how each framework should be used so that the most appropriate framework for your use case is selected.

Thus, my technical project will be to develop a set of guidelines for the selection of agile methodologies for developing your software product. To do this, I will first conduct a review of the current knowledge of the process of methodology selection to extract the factors that are commonly mentioned as being important in the selection of an Agile methodology. After doing this, I will inspect a subset of Agile methodologies. Specifically, I will be examining the three most used methodologies according to the 15th State of Agile report: Scrum, Kanban, and Xtreme Programming (XP) (Digital.ai, 2021). I will then read the existing documentation for each of these three frameworks to determine what processes each framework calls for. From here, I will analyze the implementation details of each process to determine the kinds of environments in which they might be useful. Using this knowledge, I plan to compare the advantages and disadvantages of each of the selected methodologies as well as the environments in which I believe they will find the most success. I will also investigate what industries and kinds of software projects each of these types of frameworks are used in. From there, I will be able to construct a set of guidelines, using my analysis and research of the common work and organizational values that each of these frameworks have found success in to determine when one framework might be more valued over another.

STS Research Problem

Question: *To what extent does the introduction of Agile change the overall control structure of software development companies?*

In software projects, it is often necessary to monitor the progress of a software product, as well as its adherence to budgetary and time constraints. In plan-driven environments, this is simple: everything is planned by the manager months or even years in advance, and there are comprehensive guidelines in place to ensure compliance with these standards. However, in Agile environments, this changes. Requirements can come and go, and planning is done iteratively rather than all at once before the project gets off the ground. This fundamental difference in development style, necessitates a change in management practices for managers to be able to perform these activities in an Agile environment (Aguanno, 2004). Indeed, sometimes in Agile organizations, developers may even take up duties that traditionally belong to the manager like task delegation (Maruping et al., 2009). By allowing this flattening of hierarchies, developers are better able to use their domain-specific knowledge and their motivation is increased (Puranam, 2022).

In order to reap the benefits of flatter hierarchies, you must use processes which allow such flat hierarchies to exist in the first place. In particular, Agile in theory creates necessarily flat hierarchies, if not at the organizational level, then at least at the team level (Sochova, 2020). By placing the emphasis on the team rather than the individual, Agile implies that there is no distinction of power levels between people on a team, and all team members share all responsibilities of the software development process. Thus, the power of the traditional manager is split in Agile environments (De Smet, 2018). This demonstrates the ability Agile has to deconstruct traditional divisions of power and labor by delegating tasks that are traditionally associated with individuals in positions of power to the team. Thus, Agile sets itself up to break away from traditional management and control schemes. With up to 94% of software development companies reporting the use of some kind of Agile in the workplace and even more transitioning to Agile (Digital.ai, 2021), it becomes important to understand whether or not Agile truly delivers on its potential to redistribute power in software development companies so that companies can take advantage of the benefits that come by allowing their developers more freedom.

To help me understand the changing hierarchies within the system of developers and managers, I will use Actor Network Theory (ANT) to map out the pre-existing relationships between developers, managers, and the technologies and practices they interact with on a day-to-day basis, using these networks to synthesize hierarchies. By mapping out these complex relationships, we can understand the interplay between the myriad pieces of the larger system. Further, by using ANT to model this network of relationships, we also have a point of reference to understand how these relationships change as Agile is introduced into the system. By considering the developers and managers as actors in the system, we can also understand how their actions influence the implementation of Agile in their organization.

Literature Review

It has been shown that, in creative projects, such as video game development, the introduction of Agile methodologies creates a flatter hierarchy in theory, but, in practice, a “soft” hierarchy can form (Hodgson & Briand, 2013). That is to say, team leaders and others in managerial positions seem to exhibit some degree of soft control over decisions made when creating the project. In addition to the soft control exhibited by team leaders and other managers in the flatter hierarchy of the project, the influence of higher management still exists, further reinforcing the notion of a hierarchy in the workplace. Thus, in these cases, Agile may fall short of its potential to reduce workplace hierarchy. While game development is a subset of software development companies, these results may not necessarily translate to software development because of the more specialized structure of a game development company and the modifications that need to be made to Agile processes to adapt for use in this environment. However, it is still important to understand how hierarchy manifests in Agile organizations to see if Agile truly does flatten hierarchies in software development companies.

In order to understand the extent to which hierarchy changes in an organization, we need some way to model hierarchy and understand how powers are distributed within the organizatio. To this end, control theory provides us with a way to understand divisions of power and how responsibilities flow in an organization. The primary purpose of many of the theories presented here is to reduce information asymmetry, which occurs when parties have different knowledge (Connelly et al., 2011). The predominant theory of control employed by many businesses using more traditional management methods is agency theory. In agency theory, principals (managers) contract agents (in our case, developers) to do some work for them, and this work is then monitored by the principal to ensure that the agent does not diverge from the interest of the principal (Jensen and Meckling, 1976). This theory, then, aligns itself most closely with plan-driven methodologies in our case. That is, the manager acts as a delegator of tasks and monitor of work being done to ensure that targets and deadlines are met according to well-defined plans that they lay out. On the other hand, other theories that more closely align themselves with Agile methods are signaling theory and stewardship theory.

In signaling theory, the side with private information will “signal” that private information to the other concerned party (Spence, 1973). In management, this results in the employee relaying unknown information to the manager. This puts the onus on the employee rather than the manager to convey unknown information to reduce information asymmetry. Additionally, in stewardship theory, managers act not out of individual interest, but rather out of the interest of those they manage because the interests of the manager align with the interests of the managed (Davis et. al, 1997). In effect, this means that the manager aims to cooperate with, rather than control, the people they manage, which fits well with Agile’s potential to decentralize power. By utilizing these different theories of control, we can understand the power dynamics of a given organization and, thus, the hierarchy within it.

Methodology

I plan to conduct interviews with five to ten individuals who work in a software development company that has recently or is currently undergoing the transition from more traditional, plan-driven methodologies to Agile practices. I plan to interview both developers and managers to understand both sides of the control relationship. During the interviews, I will ask questions about the changes that developers and managers have experienced during this transitional period and how the new environment compares to the old, specifically regarding organizational power dynamics. This will allow me to see the differences in culture caused by the transition to Agile and how the people involved interpret the differences that arise from the transition away from plan-driven processes. By conducting these interviews, I hope to be able to understand how the power dynamic plays out and shifts over the course of the transition to Agile. After understanding the overall control structure of the organization, I can learn to what extent the shift to Agile has contributed to a shift in the type of controls in place in the organization.

Conclusion

By conducting research on how the power dynamics in a software development company change through the introduction of Agile methodologies, I hope to learn more about how the processes we adopt can shape the social power structures which influence how and why software is developed in a certain way. Further, by conducting my technical research, I hope to understand how Agile processes are selected as well as provide the beginnings of a framework for the selection of Agile processes for a software product. By inspecting these two different aspects of Agile development and how they can influence the success of Agile for your product, we can learn ways to more effectively implement Agile.

Future technical work may involve elaboration on the guidelines developed, as well as the development of a more comprehensive framework for understanding the structure of your software organization and the selection of Agile processes therefor. Additionally, rather than considering methodologies as a whole, it may be useful to think of a methodology as being a collection of individual practices which can then be composed to create a more “custom” Agile framework. Future non-technical work may involve the exploration of this question outside of the English-speaking world and the degree to which Agile changes power structures in software development companies which exist outside of the context of the United States of America.

References

Aguanno, K. (Ed). (2004). Managing Agile Projects is Different. In Managing Agile Projects. (pp. 69-74). Multi-media Publications Inc.

Atlassian. (n.d.). *What is Agile?* https://www.atlassian.com/agile

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). *Manifesto for Agile Software Development.* Manifesto for Agile Software Development. https://agilemanifesto.org/

Davis, J., Schoorman, F., & Donaldson, L. (1997). Toward a Stewardship Theory of Management. *The Academy of Management Review*, *22*(1), 20–47. https://doi.org/10.2307/259223

De Smet, A. (2018). The agile manager. *The McKinsey Quarterly*, *3*, 76-81.

Digital.ai. (2021). *State of Agile Report*. (Report No. 15). Digital.ai. https://digital.ai/resource-center/analyst-reports/state-of-agile-report/

GeeksForGeeks. (2021). *Overview of Plan-driven Development*. GeeksForGeeks. https://www.geeksforgeeks.org/overview-of-plan-driven-development-pdd/

Hodgson, D., & Briand, L. (2013). Controlling the uncontrollable: 'Agile' teams and illusions of autonomy in creative work. *Work, Employment and Society*, *27*(2), 308–325. https://doi.org/10.1177/0950017012460315

Jensen, M., & Meckling, W. (1976). Theory of the firm: Managerial behavior, agency costs and ownership structure. *Journal of Financial Economics*, *3*(4), 305–360. https://doi.org/10.1016/0304-405X(76)90026-X

Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009). A Control Theory Perspective on Agile Methodology Use and Changing User Requirements. *Information Systems Research*, *20*(3), 377–399.

Puranam, P. (2022). Should Employees Be Allowed to Choose What They Want to Do? *Insead Knowledge.* https://knowledge.insead.edu/leadership-organisations/should-employees-be-allowed-choose-what-they-want-do

Sochova, S. (2020). Hierarchy. *Agile and Scrum Blog.* https://agile-scrum.com/2020/04/01/hierarchy/

Spence, M. (1973). Job Market Signaling. *The Quarterly Journal of Economics*, *87*(3), 355–374. https://doi.org/10.2307/1882010