Investigating the Effect of Of Actors Within the Scrum Methodology

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 Requirement for the Degree
Bachelor of Science in Computer Science, School of Engineering

By
James Nathan Barnette

Spring 2024

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.

Nathan Barnette

ADVISOR

Joshua Earle, Department of Engineering and Society

Introduction

Recent surveys reveal that an impressive 87% of Agile practitioners now use Scrum—a notable jump from just 58% in 2021 (Institute Project Management, 2022). Agile is a software development methodology characterized by iterative development, flexibility, collaboration, and responsiveness to change. Scrum is a rapidly growing workflow methodology within Agile, which can be a powerful tool in project management. Its widespread adoption underscores its inherent business value. Scrum's appeal comes from its unique adaptability, its minimal requirements for upfront research, and its maximization of work not done - minimizing wasted time. Project managers value the iterative nature of Scrum and its emphasis on constant communication, making teams nimble and able to work together quickly. At its core, Scrum emphasizes tools that are highly valued in today's fast-paced development environments.

Even with its many benefits, Scrum is far from a perfect solution to productivity issues in the workplace. Its short deadline cycle often leads to employee burnout, its lack of upfront research leads to unpredictability in final delivery times, and it is generally more demanding to clients and developers alike when compared to popular non agile methodologies such as the Waterfall method. This lack of initial research also leads to less thorough documentation, often causing issues for developers over long term project development. These drawbacks are substantial and suggest a need for a deeper investigation into Scrum's application and execution (Agile, 2022).

Actor-Network Theory (ANT) can be applied to the Scrum methodology to uncover factors which might influence team performance. ANT is a methodological approach which examines actors within a network at an objective level. This approach examines the interactions between these actors, human or non-human, to reveal the true functionality of the network. This

paper hopes to identify the most critical points which might determine the success or failure of a Scrum team, offering insights that may help shape the future of software development practices.

Background on Scrum and Agile Methodologies

Agile methodology is a project management workflow which endeavors to create a nimble and adaptable strategy for accomplishing large tasks. Agile methodologies emphasize collaborative approaches between all project stakeholders, including developers, project managers, and customers. This communicative approach leads to a shared understanding of requirements and objectives. Agile methodologies are also distinguished by their iterable and incremental natures. They break large, monumental problems into bite sized pieces of work that are in more manageable increments. These consumable increments are completed and able to be delivered to clients much more often than in other methodologies, such as the Waterfall model. The waterfall method is a sequential software development approach where progress flows steadily downward through distinct phases, with each phase dependent on the completion of the previous one (Waterfall, 2024).

Agile development is an iterative and flexible approach to software development that emphasizes collaboration, adaptability, and customer satisfaction. Scrum, originating from the game of rugby where opposing teams huddle together during a “scrum” to restart the game, is one of the most popular frameworks within the agile methodology. The Scrum process is designed to help teams deliver high-quality software by breaking down complex projects into smaller, more manageable tasks. All of these tasks are compiled into a product backlog, which a product manager combs through to determine an individual sprint’s work. Scrum mainly differs from other Agile methodologies with its time boxed sprint infrastructure. A sprint is a

time-boxed iteration during which a specific set of tasks or work items are supposed to be completed. They typically last between two to four weeks. Each sprint has its own planning period, daily standups, and a retrospective review period (Sutherland, 2014). Scrum is also characterized by distinct and clearly defined roles, artifacts, and ceremonies which guide and facilitate the development process. The main roles in Scrum are the Scrum Master, Product Owner, and Development Team. The main artifacts are the Product Backlog, Sprint Backlog, and Product Increment. The ceremonies include a planning period, a daily check-in meeting often called standup, sprint review, and sprint retrospective. The structure of Scrum, which emphasizes continuous feedback and collaboration, make it an attractive choice as an Agile solution to workflow management (Sutherland, 2014).

The adoption of Scrum has been widespread, especially in recent years. This surge in popularity shows Scrum's effectiveness in addressing the demands of modern software development, where requirements can change unpredictably, and speed to market is critical. The framework's adaptability, coupled with its emphasis on minimizing upfront research, enables teams to pivot quickly when necessary. Moreover, Scrum's iterative nature allows for continuous improvement and fosters effective communication within teams, ensuring that all members are aligned with the project's goals and progress (Sutherland, 2014).

Actor Network Theory

Actor-Network Theory (ANT) offers a framework to analyze the complexities inherent in technological and social systems. Developed by scholars Bruno Latour and Michel Callon, among others, ANT postulates that the relationships between human and non-human entities (actors) within networks form the function and nature of the network they are in. Each actor, human or non-human, has power and knowledge within these structures. Latour also notes the

importance of translation within ANT, which involves the alignment of interests, goals, and meanings among actors (Latour, 2005).

Scrum team members' interactions with various elements within the network can be observed to see how they align or translate their intentions. Applying ANT can help us understand the negotiation processes within the team and the dynamics of these teams as a whole. By viewing Scrum teams as networks of actors, ANT examines how these entities interact, negotiate, and align their interests. This lens provides reasoning for the underlying factors that affect team performance and project outcomes, potentially including things such as the role of communication tools, project management software, and organizational policies (Latour, 2005; Callon, 1984). ANT emphasizes the process of translation, where actors translate their interests into terms understandable to others within the network, so that every actor can work together in harmony to accomplish a synchronized goal. The concept of translation applies particularly well to Scrum, where diverse team members need to work closely together to meet common goals. Understanding these translation processes can shed light on the challenges and opportunities for enhancing collaboration and efficiency within Agile teams (Callon, 1984).

Various actors within the Scrum framework influence software development practices, which can be examined by applying ANT to Scrum. By dissecting these actor-networks, strategies can be identified for optimizing Scrum practices - both by addressing its limitations and leveraging its strengths to improve software development outcomes.

Challenges in Scrum

Each tech shop has their own way of doing things, but estimating the amount of work that is pulled into a specific sprint has the same general process. Product managers need to constantly

balance the total predicted effort necessary to complete all tasks in a sprint with the current bandwidth of its development team. They do this in two ways. Historically, a team of developers expected to work on a task will attempt to quantify the effort required to finish it via a process called “pointing.” Pointing is a procedure in which every team member is educated on the plan to solve a given objective. When every member has a clear understanding of the work required to complete a task, they blindly vote on a fibonacci scale how much effort it will take to finish. This is a delicate, finicky, process. A process further hampered by the fact that, much like “utils” in basic economic theory, this effort has no inherent unit. A lack of standardization in the one source of numerical data which might help load sprints optimally becomes frustrating when one’s sole job is to plan and map out development cycles. The only useful level of standardized data comes when looking at a baseline of a team or developers past pointed work. Project managers use this logic and base the total amount of points allowed in a sprint off of a team’s past performances, while accounting for unique circumstances (Sutherland, 2014).

While pointing, or estimating a task’s effort, has long been at the heart of sprint planning, there is no empirical evidence to suggest any improvement in a developer’s predictive abilities over any time frame (L. Cao, 2022). Assuming that there will be improvement, or overly relying on points as a metric, can be problematic for sprint planning. Furthermore, basing the success of a sprint off of a broken metric can be problematic for the health of a Scrum team as a whole. Sprint teams often face many external pressures from upper management or product managers to be on a strict timeline, and when unable to meet a poorly estimated level of work output these pressures can rise. Human judgment clearly isn’t the best resource in this matter, because of a lack of knowledge, predictive ability, or even possibly because team members may be attempting

to temper expectations. Whatever the reason, these challenges speak to broader issues in software development that many deal with on a regular basis (Broza, 2012).

Analysis

Why Scrum?

Prior to the takeover of Scrum in the modern day development process, the preeminent workflow methodology used was called the Waterfall method. The waterfall methodology typically operates in five key linear stages, which must be completed in their entirety before moving on to the next stage. The first stage is the requirements stage, in which the complete scope of the project is defined - from business constraints to user needs. The second stage is the design stage, where the implementation and execution plan is meticulously formed. The third stage is the implementation stage, during which developers complete the project as designed. The fourth stage is verification, during which tests are written and the final product is checked to meet the requirements. The fifth and final stage is maintenance on the final product (Waterfall, 2024).

The Waterfall model offers many advantages compared to Agile development. If executed correctly, Waterfall requires a lot of upfront work which pays off in the end. Because of the comprehensive research involved in the planning stage, it is much easier to estimate the amount of time it will take to accomplish an individual task and even the project itself. Projects using this methodology are easy to understand and easy to get caught up to speed because of lengthy documentation created upfront. This documentation makes it much easier to gain effective contribution from less experienced employees than in agile. The crux of the waterfall

model is the planning stage. If detailed enough, it can make a project manageable and somewhat easily implementable (Waterfall, 2024).

While Waterfall has clear upsides and is still a viable method of project management today, it has lessened in popularity due to its stringent workflow and hardset rules. It is incredibly difficult to change directions during the implementation phase of the waterfall methodology, and in the business world today clients change their mind about deliverables constantly. Additionally, business needs are ever changing and companies, especially small to mid-sized companies, want to be nimble and have the ability to pivot work output on a whim based on market needs. Waterfall's biggest pain point is its inability to take client feedback and tailor deliverables accordingly. This adaptability is invaluable when competing for customer satisfaction, and is why many shops have switched to using agile and Scrum methodologies (Waterfall, 2024; Agile 2022).

The Scrum Framework: An ANT Perspective

Applying Actor Network Theory to Scrum provides key insights into the dynamics of software development projects. ANT gives a framework to analyze the relationships and interactions between both human and non-human elements in a network. The relationships between these actors molds the feel and functionality of a given network (Latour, 2005; Callon, 1984).

To apply ANT to Scrum, the human and non-human actors within the network must first be defined. Human actors within this network include the Scrum Master, Product Owner, and Development Team. The Scrum Master acts like a team lead, almost taking the role of a sherpa, guiding the team through a productive Scrum process while adhering to the rules and procedures

which define Scrum. The Product Owner generally represents the interests of the company or stakeholder querying work from the development team. They have a clear idea of what the final product should look like and ensure that plans made during the beginning of a sprint do not misalign with the minimum requirements of the project as a whole. The development team consists of professionals working together to accomplish the tasks created during sprint planning. Non-human actors in Scrum include the Scrum artifacts, which are mainly the product backlog, sprint backlog, and product increments. Other tools such as project management software and communication platforms also fall into this category. Another noted weak point of Scrum, documentation, would also be a non-human actor in this instance.

The interactions between the human and non-human entities in this network make the Scrum process a powerful tool when used correctly. There is a constant back and forth between these actors - Scrum places an emphasis on consistent, open communication and this network shows evidence of such discourse in action. For example, the Product Owner (a human actor) regularly updates the product backlog (a non-human actor) to define the coming work ahead for the development team. These project requirements and tasks found in the backlog influence the work and priorities of the development team (human actors). Similarly, the development team may use many non-human resources which facilitate workflow management and communication. Examples of such technologies in the workplace include communication tools such as Slack and workflow or software management tools such as Bitbucket, Jira, and Trello.

Latour notes the importance of translation and alignment within ANT, which involves the alignment of interests, goals, and meanings among actors. In Scrum teams, understanding how team members and various elements align or translate their intentions can give insight into the effectiveness of a team. An alignment between the Product Owner's prioritization of tasks and

the Development Team's understanding and execution of these tasks dictates the health and success of a sprint. On the opposite side of things, if communication or workflow tools do not work as intended or aren't fully integrated, the whole actor network becomes unstable. Similarly, the network would suffer if the Scrum Master does not effectively play the role of obligatory passage point by failing to define clear tasks or failing to mediate interactions between actors. Such misalignments, or "breakdowns" in ANT terminology, can lead to challenges and inefficiencies within the Scrum process.

The retroactive period during a sprint is allotted for human actors to reflect on the work done during a sprint and allows for discussion about how to best structure the network. Human actors communicate during this period and can renegotiate or realign their interactions with both human non-human actors if they feel necessary. Such a period works very well within an ANT framework and serves to minimize breakdowns.

Discussion

Scrum is a methodology which allows for continuous incremental improvement for a team whose main objective is to meet project requirements as quickly as possible. In a setting where these project requirements may change at any time, its adaptability makes it the most popular choice for project management. However, Scrum is not without many pitfalls. To get the most out of Scrum, every human actor involved needs to actively attempt to minimize breakdowns and misalignments between actors. To do so effectively requires buy-in from team members. Team members need to be educated about Scrum enough to recognize where breakdowns occur and know how to effectively discuss strategies to limit them with other team members. If team members understand the purpose that sprint practices and ceremonies, such as

daily standup, serve they will be more likely to take them seriously and foster purposeful discussion and solutions. As with the alignment of goals between the Product Owner and Development team, the more knowledgeable everyone is about what they are doing and how they are doing it, the easier everyone will find their work to be. Open communication is the key to sharing this knowledge between actors and the key to Scrum success.

One interesting thing I would like to note is the level of knowledge required to effectively participate in any Agile environment. When preparing to write this paper I read a few other theses written by students who complained about the effectiveness of Scrum, advocating for waterfall methodology in its entirety. Specifically, Aaron Ponnraj says that while using Scrum, "tasks were not completed efficiently." Ponnraj claims Agile made the onboarding process extremely difficult and confusing, especially because of a lack of documentation (Ponnraj, 2022). While Ponraj makes a valid claim that I actually agree with, he fails to take into consideration the perspective of a more tenured employee than an intern in his position. Because of how knowledgeable the actors have to be in a Scrum to reduce breakdowns, interns and other generally inexperienced employees should be expected to struggle with the methodology when first exposed to it. Bluntly, Scrum is not intended for beginners. Every contributor within Scrum should have a working knowledge of the technologies they are using and how to use them. With a workforce of more inexperienced individuals, a less adaptable, more straightforward approach such as Waterfall could and should be used instead.

The focus Scrum places on individual sprint needs rather than on the overarching picture of the health of the codebase causes a lack of documentation and an accumulation of technical debt, or the accumulated cost of shortcuts or compromises in software development. Both of these issues contribute to the need for experienced employees as discussed above. However, one

way to mitigate these problems can be found through the addition of something called a kaizen into sprint planning. A kaizen is a Japanese business philosophy of continuous improvement of working practices. During the retroactive period of a sprint, teams are encouraged to reflect on what should be improved in the codebase. They come to a consensus on what to improve and pledge to work on said issue in the following sprint. In a given sprint, kaizens are generally timeboxed into small incremental improvements of the codebase intended to help improve the development team's ability to implement new features and maintain current software (Sutherland, 2014).

Conclusion

After careful examination, it is important for both human and non-human actors to work in conjunction with each other to determine the success of a sprint. Product owners must ensure their goals stay closely aligned with that of development teams, workflow tools must be fully integrated and functional, and all human actors in the network must be well versed in Scrum methodology. Translation in an ANT framework closely relates to the retrospective period in a sprint, aligning disparate actors' goals and efforts. This retrospective period improves progress and collaboration within Scrum teams as a result.

ANT analysis does provide many potential points of failure in a methodology like Scrum. Even so, with a well informed group of human actors, Scrum offers unparalleled flexibility and deliverability as a workflow methodology. The key to such success is continual communication and iteration, things already built into the Scrum framework. Further research into this subject could look into the impact of specific optimizations on Scrum team performance. It is important to continually adapt and refine Scrum processes to meet ever changing needs in the software

development landscape. Agile methodologies, informed by sociotechnical theories like ANT, have the ability to drive innovation and efficiency in software development and the world as a whole.

References

Agile methodology: Advantages and disadvantages. Agile Methodology Explained | U of M CCAPS. (2022, February 11). https://ccaps.umn.edu/story/agile-methodology-advantages-and-disadvantages

Broza, G. (2012). *The Human Side of Agile: How to Help Your Team Deliver*. 3P Vantage Media.

Cao, L. (2022). *Estimating Efforts for Various Activities in Agile Software Development: An Empirical Study*. IEEE Access, 10, 83311-83321. doi: 10.1109/ACCESS.2022.3196923

Chirra, S. M. R., & Reza, H. (2019). A Survey on Software Cost Estimation Techniques. *Journal of Software Engineering and Applications, 12(6)*, 226-248. doi: 10.4236/jsea.2019.126014

Cohn, M. (2005). *Agile Estimating and Planning*. Pearson.

Cohn, M. (2022, January 7). *Writing the product backlog just in time and just enough*. Mountain Goat Software. https://www.mountaingoatsoftware.com/articles/writing-the-product-backlog-just-in-time-and-just-enough

Latour, B. (2005). *Reassembling the Social: An Introduction to Actor-Network-Theory*. OUP Oxford.

Latour, B., & Woolgar, S. (1979). *Laboratory Life*. Sage Publications.

Ponnraj, A. (2022). Agile in the Workplace: Personal Technical Experience Under Agile at a Fintech Startup | the Social and Personal Implications of the Agile Coding Methodologies on Computer Science Workplaces. Charlottesville, VA: University of Virginia, School of Engineering and Applied Science, BS (Bachelor of Science), 2022. Retrieved from https://doi.org/10.18130/tv7z-1945

Sage, D., Dainty, A., & Brookes, N. (2011). How actor‐network theories can help in understanding project complexities. *International Journal of Managing Projects in Business, 4(2)*, 274-293. doi: 10.1108/17538371111120243

Schwaber, K. (2010, June 15). *About scrum.org*. Scrum.org. https://www.scrum.org/about

Sismondo, S. (2004). *An Introduction to Science and Technology Studies*. Blackwell Publishing.

State of Agile Report. Institute Project Management. (2022). https://instituteprojectmanagement.com/wp-content/uploads/2023/01/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf

Sutherland, J. (2014). *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Currency.

Technologies, T. (2023, December 20). *Top 5 benefits of REACT for interactive web development*. Medium. https://medium.com/@marketing_26756/top-5-benefits-of-react-for-interactive-web-development-69875dd8cab5

Waterfall methodology for project management. Atlassian. (2024). https://www.atlassian.com/agile/project-management/waterfall-methodology