Exploration of the Role of Co-Design in Agile Development

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

Emily Lu
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

Advisor
Sean M. Ferguson, Department of Engineering and Society

Exploration of the Role of Co-Design in Agile Development

Introduction

Agile development has become a widely adopted programming cycle across most tech companies and development teams to deliver quality products to clients as quickly and efficiently as possible. As implied by the name, agile development is extremely fast paced and involves making many quick decisions, requiring a high degree of flexibility in order to accommodate last minute design changes. While co-design and collaboration can help accelerate this process, there are times where it can cause significant slowdown from delayed or unclear communications.

Co-design is "critical to service development" because different perspectives are needed to understand the users' needs, how they intend to use the service, as well as the developers' available resources to implement the desired service (Steen et al., 2011). However, the way software is created creates problems for software engineers due to the ever-changing nature of technology (Takalo et al., 2021). This can cause significant slowdowns in an agile development setting as changes need to be communicated to clients in a way that is both efficient and easy to understand. In this paper, I will use the co-design framework to explore the effectiveness of agile development in creating an optimal solution within deadlines across companies of different sizes and cultures. Since the scope of different project types in software development is too large, the exploration of agile methodologies will be focused on the development of products directed towards external clients.

Background on Agile Development

As consumer expectations have increased, more companies compete against each other to deliver the optimal product to consumers as fast as possible, leading to companies adopting agile development. However, having both speed and the perfect product is nearly impossible to achieve, and at times one of the two will have to be sacrificed throughout the development cycle. In software engineering where methods of implementation and design choices are endless, input from all involved developers and primary clients is crucial to keeping the project on track and user-friendly.

According to Katsonis (2019), co-design is a "co-creation practice that allows users to become part of the design team as 'experts of their experience'" that drives innovation and moves design "towards using the collective creativity of a team with members from different backgrounds and interests." Co-design in agile is crucial to understanding the full scope of the consumer's needs while maintaining a clear objective and having consistent communication between developers and clients sets the foundation for developing a user-intuitive product most closely aligned to the user-defined specifications (Steen et al., 2011). However, extraneous or unclear communication can easily cause delays in delivery times and development progress as user requirements change.

Out of the five stages of agile, the client is the most directly involved in the first two stages: the requirements and design stages. In these two stages, clients will define what sorts of features they want in a product as well as set deadlines for when they want the project or certain parts of the project to be completed. During these two stages, developers and clients agree on a middle ground to find a set of expectations that are both reasonable and as closely aligned to the client's needs as possible. With heavy involvement from non-technical parties, the "non-designers become equal members of the design team" and can "directly contribute their knowledge and fresh perspectives to exploring problems and possible solutions" (Katsonis, 2019). Without extensive co-design in these two stages, the foundation for the remaining stages of the cycle will be unstable and more prone to unnecessary delays from uncommunicated expectations and limitations (Fridman, 2016).

Since the requirements and design must be agreed upon by all parties involved, having a smaller development team makes it easier for communication and provides fewer opportunities for long, repetitive discussions over a single decision (Dubreuil, 2017). Larger teams bring in more dissenting opinions and leave each team member with too many people to keep track of, making it easier for important information to be lost in between communications and hence reducing the effectiveness of co-design. Especially in larger projects with dependencies from other teams, even if individual teams are small, communication is easily lost or delayed between teams, slowing down the development process.

In a study analyzing Amazon's approach in agile development, each team was split into two smaller sub-groups with no more than six to seven people each, also known as "pizza teams", with a Single-Threaded Owner (STO) to manage one or more of such teams. Having a structured and smaller subdivision of the large company allowed teams to operate "with little to no dependencies on other teams when developing new ideas" but STOs were having problems with having "too many meetings, challenges obtaining approvals, and aligning on priorities" (Golden et al., 2019). Prior to Amazon's move towards the pizza team approach, teams were much too large and resulted in "over-communication", creating unnecessary chaos and confusion as tasks became decentralized within large teams. While pizza teams have significantly improved focus distribution within Amazon, Amazon now has so many teams and products even within the same sub-organizations that continue to make communication chains largely inefficient at times. As a large company, Amazon has adjusted its internal structure to accommodate an agile approach by reducing clutter generated from inefficient communication chains, but there are still many limitations on efficiency due to the sheer size of the entire company as a whole.

In the development, testing, and release phases, the client maintains consistent communication with developers to communicate any changes in requirements, and tests different features that developers have completed. While testing, clients are expected to communicate any confusing aspects of the product and clarify the expected behavior of each feature. Having people with no technical background testing the product, developers will be able to determine whether or not their product is intuitive to a broader audience before releasing the features into production. This is a generic skeleton as to how the agile development cycle is typically structured, but different work environments and cultures will have differing nuances from each other in their approaches to agile development.

Problem Definition: How Co-Design Adapts to Different Agile Methodologies

Given that there are so many types of work cultures as well as agile methodologies, some agile methodologies are guaranteed to be more suited to certain work cultures than others to maximize efficiency while maintaining company goals. Some companies are focused more on putting out fast rather

than perfect solutions, leaving less time for code refactoring prior to product release, while other companies are more focused on the best solution in the first release. A company's available resources will also affect the types of agile methodologies that are optimal—a company with few employees cannot afford to have many specialized sub-teams with different tasks and would probably operate better under an organization where most employees are equipped to perform a range of tasks.

A company's work culture can be dependent on a number of elements, such as company size, age distribution of the employees, age of the company, as well as the personalities of the company employees. An example comparison between two common types of work cultures is startup work culture versus corporate work culture. A startup is usually a small company characterized by a culture of high uncertainty and rapid evolution, exploring solutions to a problem in a volatile market, while corporate culture is more commonly a large company characterized by stability and steady growth.

Like how different work cultures have defining characteristics, different agile methodologies also have their own respective characteristics. The most commonly used agile methodologies are Scrum, Kanban, and Extreme Programming (XP). At a glance, Scrum based development typically places heavier emphasis on planning and complex design prior to implementation while the XP and Kanban methodologies have more of a continuous delivery nature starting with simple designs (Matharu et al., 2015).

Co-Design in Different Agile Methodologies

While co-design and collaboration are evident in all three agile methodologies, the hierarchy and amount of collaboration at different phases of agile varies between the three methodologies. Such differences primarily stem from different team organizations that would be best suited for a particular work environment. For example, Scrum based development has more designated leaders such as the Scrum Master or Product Owner that are responsible for various aspects of a project, whereas XP and Kanban focus more on team ownership and responsibilities as a whole. In terms of co-design through

customer interactions, XP is known to involve high levels of interaction with customers specifically during the development process, Scrum attempts to maintain consistent levels of communication and collaboration across all phases of the agile cycle, and Kanban tries to achieve “waste minimization” by only executing tasks when they are required and does not have a set expectation for how often collaboration with customers should be (Matharu et al., 2015).

With internal collaboration and co-design, Scrum is heavily dependent on cross-functional teams, which indicates that different people with different skill sets are spread across different teams, requiring communication between these individuals necessary for consistent progress to be made. Scrum based development is also characterized by daily meetings which allow for a basic set communication schedule amongst team members and requires co-design to be prevalent throughout the entire agile cycle. However, having a small group of leaders in Scrum would imply that co-design would be more centralized within those leaders during the planning and design phases of the development cycle, and not so much with the general developers. In XP, pair programming is a crucial component where developers work in pairs of two to increase work efficiency, while Kanban operates in teams made of specialized resources such that teams will not have to cross-reference each other that often. Neither XP nor Kanban have as many designated roles or leaders and start out with simple designs. With these observations, it can be concluded that Scrum has a wider range of consistent, internal communication, while XP and Kanban have more small-scale collaboration and individual authority (Matharu et al., 2015).

Research Methodology

A survey in the form of a Google Form was distributed amongst social networks and acquaintances within the community of The University of Virginia, with most respondents being undergraduate students and a few being new graduates working in the software development field or other, more experienced, hired professionals in the field. The survey was split into two sections: one section that contained questions that could be answered by everyone who took the survey, and another

section that was catered specifically towards respondents with experience in the software development field. The first section was composed of questions regarding respondents' preferences and opinions on group organizations in a project while the second section asked about organization of agile methodologies and communication in their workplace. The questions were designed to mostly leave responses open ended and avoid influencing respondents from answering in a certain manner. Table 1 documents the questions asked in each section.

Table 1: Survey Questions

Section 1 (Universal)	**Section 2** (Software Development Experience Only)
What do you think is most important to efficient teamwork in larger group settings?	Provide a brief description of how teams were organized for agile development in your most memorable internship/job experience.
What do you find to be the hardest part about working in group projects?	What size company did you work at?
For smaller projects that may take a few weeks to a month to complete, what size group would you prefer to work in to maximize efficiency?	Did you feel that the organization was efficient, or if there was anything that you felt was lacking?
For larger projects that may take several months to complete, what size group would you prefer to work in to maximize efficiency?	What did you feel was the most important aspect of team organization that helped maintain agile development?
	Was the amount of team communication and collaboration throughout your projects consistent through each phase of the agile cycle or did it fluctuate?
	If you answered "Not consistent" to the above question, please elaborate on which phases had more collaboration and which phases had less.

Research Analysis and Related Works

A total of 17 people responded to the survey, with ten of them having experience in a professional software development setting. In the first section of the survey, roughly half of the individuals specified that they would prefer to work in a single group while the other half preferred to work in several small groups when tasked with a large project that would take several months to a year to complete. On the other hand, nearly 83% of respondents specified they would like to work in a single small group for smaller projects while 17% of respondents indicated they would prefer to work alone. Furthermore, nearly all respondents indicated that the most difficult aspect of working in group settings was a combination of communication, work distribution, and having aligned visions on the tasks at hand.

Such responses from the first section indicated that respondents generally preferred to work with smaller groups for the sake of more effective communication amongst team members and ease of maintaining expectations and responsibilities. Respondents voiced that they felt having a larger team made it harder to consolidate all opinions into a single conclusion, and that it was more difficult to schedule common times where everyone could meet to provide updates or redistribute tasks. Survey results also pointed towards the importance of proper delegation of tasks such that responsibilities would be assigned according to peoples' skill sets to improve quality control and reduce potentially extraneous and confusing communication. However, even in cases with efficient overall company organization, the nature of software development as a career can be, as one respondent pointed out, "slow when taking a product from the planning stage … to the production stage" due to security checks, quality assurance, and discussions between numerous teams.

Practically all respondents who had professional experience in the software development field indicated that they were part of a small sub-team under a larger team and felt that daily meetings with the team along with clear communication of expectations and roadmap were crucial to working with maximum efficiency. Some indicated that a good manager or small group of leaders was one of the most

important aspects to maintaining a clear direction since having a small group of decision makers would help prevent project design and timelines from becoming too convoluted due to too many opinions.

Although several respondents had similar team organization structures at a similarly sized company with some even at the same company, there were individuals that felt team organization was efficient while others did not, indicating that even within the same company, work culture can be very different from team to team. Differences in work culture appeared to primarily stem from the type and scope of the project that was being worked on, implying that even if a variation of an agile methodology has been "catered" to a certain team's work style, it will not necessarily be appropriate for all projects assigned to that team. For example, XP development is not suited for projects where consistent communication with the customer is required, or for projects that do not have a nature that allows consistent, intensive testing (Tripathi et al., 2017).

Certain individuals also indicated that communication and co-design was not necessarily consistent throughout their projects, sometimes not due to the formal structure of team organization but simply because of circumstantial situations such as a team leader being absent during a crucial time of the development period. In the presence of miscommunications, and lack of communication or customer involvement, projects are more prone to failure as a result of misunderstanding customer needs (Tripathi et al., 2017). Respondents that pointed out inconsistent communication and co-design generally indicated that they felt work was not as efficient and was prone to project failure, aligning with the case study conducted by Tripathi (2017). Although there is a common consensus that consistent co-design is crucial to ensuring a successful project, some respondents had raised concerns of having too many dependencies on other teams for the sake of quality assurance, causing significant slowdowns in making ample progress on the project from slow response times or high volumes of discussions with too many conflicting ideas despite having "good" company organization.

Discussion

The aforementioned survey was conducted to determine recurring trends in preferences for group organization as well as team organizations within an agile environment. Since the number of responses gained from the survey was on the smaller side, it was difficult to generalize the responses to a larger population. Furthermore, most of the responses were from college students or new graduates, which implies that the survey pool mostly consisted of individuals with minimal experience in a professional work setting. A lack of experience in the survey pool could have limited the amount of insight gained through the survey, such as undermining the importance of certain aspects of working in a group environment that normally would not be experienced in a non-corporate setting.

All respondents that filled out the second section of the survey worked at a large company, which also prevents the implications made from this survey applicable to a larger scope including smaller companies as well. Since most large companies use the Scrum development method, there was extremely limited insight on the other agile methodologies, but in turn there was insight provided on how the same agile methodology could be different from company to company.

Ultimately, all the conclusions made from the survey could only be safely generalized as far as college students, but despite the gap in experience levels from the survey pool and the general population, responses from the survey aligned with Livermore's (2008) survey and Tripathi's (2017) case study in that efficient communication, leadership, and team organization were the most important to any approach to a project. While work culture is a major factor in how agile methodologies are structured at a company, many other factors can cause there to be multiple methodologies, many variations of a certain methodology within the same company or even the same team depending on external factors such as project scope and team availability.

At the University of Virginia, agile is taught to students in a methodical way such that a very small team of students will work on a single project, working through the typical stages of the agile cycle.

However, independent, short-term projects fail to address the various difficulties of communicating with other teams or deciphering other projects that may be a dependency of the current working project. Such struggles are typically glossed over or only briefly mentioned in a lecture setting, and at times, take students by surprise when experiencing those difficulties first-hand in a professional setting for the first time. Educational programs could potentially better prepare students entering the software development career by teaching students more about common procedural struggles that are encountered in an agile team depending on different types of organizations as well as how to approach and mitigate such problems.

In conclusion, although agile methodologies all share the same basic emphasis on co-design, there is no rubric for which methodology or form of co-design is best suited for any given environment as external factors and unpredictable events can heavily impact the efficiency of all team structures—collaboration and co-design will only become effective in agile if structured in a way that suits the nature of the project in question, the compositions of the teams involved, and is compatible with the methodology implementation.

References

Dubreuil, J. (2017, March 13). *8 effective ways to improve the productivity of an agile development team*. Julien Dubreuil. Retrieved October 13, 2021, from https://juliendubreuil.fr/blog/agile/eight-effective-ways-to-improve-the-productivity-of-an-agile-development-team/.

Fridman, A. (2016, May 6). *The massive downside of Agile Software Development*. Inc.com. Retrieved October 13, 2021, from https://www.inc.com/adam-fridman/the-massive-downside-of-agile-software-development.html.

Golden , J., & Galetti, B. (2019, October 3). *Inside day 1: How Amazon uses agile team structures and adaptive practices to innovate on behalf of customers*. SHRM. Retrieved October 31, 2021, from https://www.shrm.org/executive/resources/people-strategy-journal/spring2019/pages/galetti-golden.aspx.

Katsonis, M. (2019, December 12). *Co-design: From expert to user-driven ideas in public service design*. ANZSOG. Retrieved October 30, 2021, from https://www.anzsog.edu.au/resource-library/research/co-design-from-expert-to-user-driven-ideas-in-public-service-design.

Livermore, J. A. (2008). Factors that Significantly Impact the Implementation of an Agile Software Development Methodology. *Journal of Software*, *3*(4), 31–35.

Matharu, G. S., Mishra, A., Singh, H., & Upadhyay, P. (2015, January 1). *Empirical Study of Agile Software Development Methodologies: A comparative analysis*. ACM Digital Library. Retrieved March 1, 2022, from https://dl.acm.org/doi/abs/10.1145/2693208.2693233?casa_token=XDWeVtek4w4AAAAA%3AdwGCgS9r2e3yPvsQ-eaYZo3ZjovXunC8snnio6IAP8tVqKpQbCHMuQLVLjFTI0WMjQHDtPmwNRxuww

Steen, M., Manschot, M., & Koning, N. D. (2011, August 15). *Benefits of co-design in Service Design projects*. International Journal of Dsign. Retrieved October 5, 2021, from http://www.ijdesign.org/index.php/IJDesign/article/view/890/346.

Takalo, J., Kaariainen, J., Parviainen, P., & Ihme, T. (n.d.). *Challenges of software-hardware co-design - vttresearch.com*. Retrieved October 5, 2021, from https://www.vttresearch.com/sites/default/files/pdf/workingpapers/2008/W91.pdf.

Tripathi, V. Y., & Mishra, A. (2017, November 13). *Case study of agile methodologies in field of software development*. International Journal of Engineering Research & Technology. Retrieved October 31, 2021, from https://www.ijert.org/case-study-of-agile-methodologies-in-field-of-software-development.

Appendix A

Survey Responses

*Responses with similar or identical wording have been combined

Section 1 Questions (Universal)	**Section 1 Responses**
For larger projects that may take several months to complete, what size group would you prefer to work in to maximize efficiency?	- Alone (0%) - Several small groups (52.9%) - One small group (41.2%) - Several large groups (0%) - One large group (5.9%)
For smaller projects that may take a few weeks to two months to complete, what size group would you prefer to work in to maximize efficiency?	- Alone (17.6%) - Several small groups (0%) - One small group (82.4%) - Several large groups (0%) - One large group (0%)
What do you find to be the hardest part about working in group projects?	- Delegating an even amount of work and maintaining progress across the board - Scheduling a time to meet up with everyone - People not pulling their weight, figuring out organization and delegation of responsibilities appropriate to skill sets - Getting everyone on the same page and making sure everyone has a part in the final product - Accountability and timing - Communication and organization with members and other groups (different ideas conflict with each other) - Aligning everyone's vision and accountability - Quality control - Work distribution - Handling slackers - Keeping up to date with everyone's status on their assignments
What do you think is most important to efficient teamwork in larger group settings?	- Clear communication and efficiency - Knowledge sharing and regular progress meetings - Designating work and sections to groups or members - Distributing work and keeping everyone accountable for their portion - Finding a good number of meetings to ensure everyone is on track but not wasting time on meaningless meetings - Having a strong leader and responsible members - Predefined expectations and healthy communication - Compromising with other opinions - Fast response times and clear communication - Taking initiative and responsibility on your own part
Section 2 Questions (Software Development Experience Only)	**Section 2 Responses**
Provide a brief description of how teams were organized for agile development in your most	- Two pizza teams - Numerous small teams, one manager for each team, each with several engineers

memorable internship/job experience.	- Each team had a product managers well as a scrum master who helped guide interns through the agile development process. Had daily stand-up meetings as well as weekly sprint goals where each member selected tickets to work on. - 1-2 software developers on a project - Tasks are split into sections and each person is assigned a section that can be completed without much dependency on other people's actions - We had a group of 5 interns that collaboratively worked on a project, and we would divide up tasks ourselves - One big team divided into two smaller teams that reported to different managers - Don't really have agile at my company - My team was extremely small with 4 people initially, so agile development in my team was mainly in the form of daily stand-up meetings and grooming/cleaning tickets and tasks for the next two weeks (each sprint last two weeks)
What size company did you work at?	- Large (100%) - Medium (0%) - Small (0%)
Did you fell that the organization was efficient, or was anything that you felt was lacking?	- Organization was efficient and uniform - Not efficient. Depending on other teams who don't give timely responses slows down efficiency significantly—this can be related internally as well - Not really efficient. Lack of group and design discussions on new initiatives. Most of the time an idea was picked and implementation immediately followed - The organization (Amazon) as a whole was efficient, but I think since it's a company based on high user satisfaction as well as innovation, it can be slow when taking a product from the planning stage all the way to the production stage (there are many security checks, quality assurance, and discussions between many teams in order to push out a single product)
What did you fell was the most important aspect of team organization that helped agile development?	- Hierarchy of technical experience - Clear direction and understanding of what the product needed to be as well as what steps were needed to get to that point - Fast responsiveness - Regular meetings to share progress - Good manager - Having deadlines that everyone follows, making sure to help each other, and communicating each others' progress to each other - Pizza box team structure - Tech leadership. Need a very small group of people to lead design, and carefully plan the implementation phases

	- Daily stand up since everyone can keep up with each others' tasks and also assist each other if there are blockers
Was the amount of team communication and collaboration throughout your projects consistent through each phase of the agile cycle or did it fluctuate from phase to phase?	- Consistent (70%) - Not consistent (30%)
If you answered "Not consistent" to the above question, please elaborate on which phases had more collaboration and which phases had less.	- I think the phases that were more coding intensive needed more communication as our group helped each other in parts we were not familiar with - The middle part of the project had less communication and collaboration as my direct mentor went on vacation - More collaboration at the beginning just to get things started, and less communication as the project progressed, leading many projects to fail