Automated Replication of Resource Type Permissions

The Role of Co-Design in Agile Development

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

Emily Lu

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

Sean Ferguson, Department of Engineering and Society

Rosanne Vrugtman, Department of Engineering and Computer Science

Jim Cohoon, Department of Engineering and Computer Science

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' resources available 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 will need to be communicated to clients in a way that is both efficient and easy to understand. The STS research will explore how co-design is both a necessary yet limiting component of agile development.

My technical project is loosely coupled with my STS topic in the exploration of co-design and agile development. The goal of the technical project is to create a complex, internal resource-type permission replication infrastructure in an extremely agile environment with a gargantuan codebase. In the world of software engineering, agile development is important to making full use of the allotted time and completing the assigned task especially with larger-scale projects. Learning how to effectively work in an agile environment together with other developers and clients is a crucial skill for present-day software engineers in keeping up with rapid technological developments and faster expected turnaround times.

Technical Topic

Manually replicating permissions across multiple regions is not only time-consuming but can also lead to potential mistakes such as replicating in an incorrect region or creating/deleting a permission with

the wrong configurations. To reduce such mistakes, automation of the replication workflow is crucial to improving the efficiency and consistency of future related operations within the organization.

As an intern for Amazon Web Services (AWS), I created an automated workflow for creating and deleting internal permissions over certain resource types for the Resource Ownership Service (ROS) team. I was solely responsible for the entire development process-- writing APIs for creating and deleting resource type permissions, synthesizing the APIs with another workflow using Amazon Web Service's Simple Queue Service (SQS) and Lambda to automatically replicate the corresponding create or delete across all applicable regions, and providing end-to-end testing for the entire workflow. Through the workflow, users can create or delete internal permissions by calling designated APIs, and the resulting creation or deletion is then automatically replicated through all regions in which the corresponding resource type exists. Having such an automated replication workflow saves time spent on carrying out repetitive actions, as well as time spent on fixing bugs resulting from manual errors.

Given that confidential resources need to be strictly regulated, the process for defining who is allowed to access or modify the permissions to those resources needs to be secure and consistent to ensure that unauthorized users do not unintentionally gain access to those resources. In order to achieve this, extensive testing and documentation were a necessary component to the project. Being an internal tool, the process of designing and implementing the workflow required lots of flexibility and co-design with the client as well as other developers. Therefore, while the development itself was primarily an individual effort, I consistently submitted code reviews as well as regular design documentation updates to other developers to have them approved before pushing my changes to the repository.

The development process began with creating and finalizing a design document to synthesize the architecture and overall workflow between each component of the project while defining requirements for each component. Such requirements were defined through evaluating the client's requests for certain features and analyzing how they can be implemented through code with available resources. Once the design document was finalized, each component of the workflow was individually implemented starting with the APIs and then the AWS SQS and Lambda workflow. Once the components had been

individually tested, the components would then be synthesized to complete the overall workflow. Lastly, integration tests were written to test the workflow from end-to-end and maintain that future changes to the workflow do not break the system or change the base functionality.

Throughout the development process, I considered various architectural designs with the client to align my work with the existing structure of the rest of their codebase, and integrated various tools into the workflow to promote efficiency as well as simplicity so that any problems sourced from future updates to relevant dependencies would be easier to locate and resolve. Design choices were also never completely set in stone and were always subject to change, and such changes were applied and integrated as soon as possible to reduce future confusion and refactoring. Lastly, design choices were made to be intuitive and easy to use for the client for maximized client satisfaction. This project helped me better understand how to balance typical co-design processes with the agile development cycle.

STS Topic

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, 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. 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.

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, 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). As a large company, Amazon adjusted its internal structure to accommodate an agile approach, 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.

To further explore the sustainability of agile development in different software development environments, I will analyze a case study conducted by Vishnu Yash Tripathi and Anoop Mishra on agile in software development, which will provide information on different methodologies in agile, how co-design is incorporated in each methodology, and which types of projects each methodology is best suited for. This case study along with the provided evidence will be used to assess the effectiveness of different agile methodologies across different software development environments.

Next Steps

- STS Research
    - Interviewing developers working on different kinds of projects from different work cultures to help answer the questions I have listed below (to be conducted throughout the rest of the semester)

    - Get a better understanding on how difficult it is to communicate technical knowledge to clients without any technical background and whether or not this could help improve agile development
    - Conduct research on how to efficiently conduct regular meetings with the client without straining developers for time even more than they already are
    - Find what is the threshold for too much communication vs too little communication
    - Find more evidence and statistics on how often developers are actually able to deliver the product on time without skipping over certain parts of agile development such as extensive testing
    - Determine what sorts of companies are best suited for using agile with extensive co-design, and which companies are better suited for other development methodologies
- Technical Project
    - The below goals are listed assuming that I still have access to the company's resources
    - Write more integration tests for the workflow
    - Migrate data structures the workflow is dependent on into other databases
    - Create more detailed documentation on how the APIs are called and used, as well as other requirements that were implemented to help users understand certain error messages and use cases

References

Amazon. (2018). *Regions and Availability Zones*. Amazon. Retrieved October 30, 2021, from https://aws.amazon.com/about-aws/global-infrastructure/regions_az/#:~:text=AWS%20has%20the%20concept%20of,AZs%20within%20a%20geographic%20area.

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.

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.