Ming Lin

SSW555

Professor Ens

November 02 2020

Homework 07

Lean Principles in Microsoft

In Poppendieck's paper on the principles of lean software development, he points out that some of the principles of lean philosophy have been practiced by software developers in the past before agile methods were invented. In the case of Microsoft, their development process consisted of building small features in a short time by small cross-functional teams. Similarly to some agile methods, Microsoft would implement daily builds and which allowed them to assess and fix bugs with ease. Microsoft would also base their timeline in terms of features and milestones. Utilizing automated build tools, rapid testing, and continuous integration to streamline the testing and deployment process. The code and implementation strategies created had to adhere to strict standards set forth by Microsoft. This allows team members to overlap responsibilities and collaborate with ease as the structure of code was similar throughout Microsoft's immense codebase. Lastly, at the end of a development cycle teams would reflect on their work, and this feedback and focus on the product's postmortem allowed Microsoft to refine their development process.

Value Stream Mapping

A value stream is a flow of goods, services, or information that traverses through an organization for it to provide their goods or services to their customers. Value Stream Mapping is creating a

visualization that shows how value is being exchanged throughout a process. This exercise allows team members to aggregate their thoughts about a process and think about the effectiveness of each step from a higher level. The goal being of this exercise is to find points in a process where resources are not being used effectively and waste is created. When waste is identified at a certain point plans should be created to remove or minimize it. After this assessment is completed and plans are discussed to remove waste the results should include two mappings. One mapping showcasing the current process and another showing a potential future process after the plans are implemented and waste is no longer being created.

Current Process

Our GEDCOM team is currently communicating through chat messages using Groupme and verbally using Zoom. As a team, we meet weekly to discuss the context of the current sprint, distribute tasks, and resolve any blockers. At the beginning of the project, we decided as a team on a structure and standards for how code should be written in the codebase. This allows each team member to easily integrate code and reuse other team member's code throughout the codebase. The physical process of integrating code involves a team member pushing up changes from their git branch, opening a pull request into the main branch, and receiving approval from three team members before merging their code changes into the main branch. Tests are required with every user story implemented and must adhere to standards agreed upon at the beginning of the project. Each test must follow a specific naming scheme, be added to the testing suite once completed, and be able to pass all tests within the testing suite. This is because when we package and submit our results we are submitting the entire testing suite, so this process keeps the testing suite always up to date with the latest test and results.

Future Process

After evaluating my team's current process, we found that implementing some additional procedures could prevent waste in the future. One procedure is forcing every team member to pull down the latest version of the main branch before pushing up changes. This will result in fewer merge conflicts being pushed up to the repo as team members would be able to quickly realize and fix the merge conflict before pushing up to the remote repository. Another added procedure is implementing a new codebase standard where each test's seed files must be within their own folder within the seeds directory. This should avoid the event where a team member tampers with seed data to pass their own test while causing another team member's test to fail. Up to this point, our codebase has a limited amount of seed data and oftentimes we must tweak seed data in order to test certain edge cases. With this new file structure, each test's seed data will be neatly organized and tweaked without causing other team member's tests to fail.