Liam Brew
SSW 555

Professor Ens
2 November 2020

Homework 07: Lean

1. One example of how Microsoft applied lean software development principles is its quality assurance approach under the "Build Quality In" and "Optimize the Whole" principles. Using automated continuous integration testing and fixing problems as they are discovered, Microsoft can ensure higher quality products are delivered at faster speeds compared to more conventional development methodologies. The use of predefined standards for coding, design and testing also applies to this principle. This synchs well with Microsoft's daily build cycles, itself indicative of the "Deliver Fast" principle. Microsoft's use of small and multifunctional teams ensures sufficient team engagement and overlapping responsibilities, a reference to the "Engage Everyone" principle. Finally, by focusing on small-scale features and eliminating as much bureaucracy as possible, Microsoft abided by the "Eliminate Waste" principle
2. Value stream mapping is the process by which the flow of goods and information is coordinated and visually displayed in a diagram. It contains various project metrics such as efficiency and is designed to easily show information in an organized and sensical manner. There are two types of maps: the current map, which contains information and data about the current state of the project, and the future map, which shows the ideal state of the project as envisioned by the team. An important function of value stream mapping is to identify weak points and wastage in the current cycle. By exposing these deficiencies, a plan of action can be established to transition the current map into the future map.
3. My GEDCOM team's current process features group planning and retrospective sessions followed by independent work on sprint stories. While this is independent in the sense that each member of the team is responsible for their own individual stories, this arrangement is not rigid enough in structure to exclude collaboration and teamwork. When a team member encounters difficulty in completing a story, they do (and are encouraged to) reach out for assistance from the rest of the team. These pair-programming sessions are not planned but spring up whenever difficult issues are encountered. To facilitate communication, all the members of the team are in an instant messaging group chat. Due to the realities of the remote environment and the fact that most team members have other classes/jobs/responsibilities to worry about, responses are not truly "instant" but typically occur within a few hours' maximum. Communication also takes place via GitHub merge requests and issues, wherein submitted code and ideas are commented on and discussed among members of the team. In terms of testing, alongside each completed story developer are expected to also generate unit tests for that specific story. These tests then contribute to the team's Azure build pipeline connected to the repo, which determines the test results of each development branch before it is merged into the master branch. This pipeline goes a long way in ensuring constant quality monitoring through automated testing. Before a branch is merged, it requires review from at least two team members, who may comment with feedback or even reject the merge request should it not be satisfactory. Finally, during every sprint retrospective meeting (led by two team members on a rotational basis), the team meets to discuss their progress during that sprint and any issues/breakthroughs they encountered. Sprint data is entered into the tracking spreadsheet (such as time and LOC for each story), the burndown chart is updated, and the results of that sprint prepared and submitted under the overview of the entire assembled teams.

I pledge my honor that I have abided by the Stevens Honor System

4. A major point of wastage in the team's process is the reliance on members who do not perform satisfactory work. Currently, the amount of issues assigned to each sprint is divided equally among the team members. Unfortunately, this includes members of the team who do not maintain sufficient communication, especially one who has yet to substantially contact the very team they are supposedly on. As a result of this, as each sprint draws to a close the team operates under the optimistic yet misguided belief that these members will submit their required work. When this inevitably does not happen, then it is up to the remaining members of the team to complete these extra stories. This is often done on a sporadic, "who's available when" type of approach in the last days/hours of the sprint. To rectify this, sprint responsibility allocation should exclude those who fail to contact the team and/or maintain a level of professionalism and competence unbecoming of both this course and Stevens as a whole. These members are invited to rectify these issues at any time with the team, but until that happens the team should not rely on them for anything. This will result in more realistic expectations for each contributing team member and paint a more accurate picture of the project's current status.