ScavengeRUs – Process Report

Team 3 / Coding Wizards

To start our tier one agile process, our team developed a Trello workspace to define our project backlog. The product backlog was composed of user stories as well as developer stories. Categorization took the form of either features, technical work, or knowledge acquisition. Upon categorization, these backlog items were prioritized based on the complexity to implement and by how independent they were. The stories with the most dependencies typically fell to the bottom of the backlog, while the stories that were relied upon for other stories to function were put to the top.

We split our backlog across each sprint into lists in Trello in order to easily manage our workspace. We also created a list for in-progress user stories, user stories being testing by the developer, user stories that were being validated by the Scrum Master and Product Owner, and finally the user stories that have went through the validation process and have been completed. This was done for each sprint.

Developers had the ability to add themselves to user stories and place these into the ‘in-progress’ list. Each user story contained multiple subtasks listed inside the card and allowed the developer to mark off as he/she completed these tasks. This allowed the rest of the team to view the progress of a certain user story at a high level. Upon completion of a backlog item, the developer would place these stories into the 'Dev Testing’ list in Trello. There, the scrum master along with the responsible developer would test and verify that the code worked as intended. Once verified, the branch with those changes would be compiled into a pull request. The user story in Trello was then moved to the ‘Validation’ list. Once validated by the product owner, the product owner places the backlog item into the ‘Done’ list for archival and merges their pull request into the main branch.

Each card in Trello was labelled by their iteration number, type of story, and feature. These labels also serve as the branch name for easy reference in our repository. For example, i1-USER-AccessCode refers to iteration 1, a user story, and the access code feature. This practice allowed us to associate backlog user stories with current branches very easily and worked well. Our approach, as evident by our naming scheme, was branch by user story. Once a user story was completed, verified, and validated, it was pulled into the main branch. However, midway through the semester we found ourselves running into many merge conflicts as several branches contained dependencies with classes currently being worked on by other developers. To solve this problem, we migrated our approach to make pull requests with each subtask within a user story. That way, we were keeping the main branch as up to date as possible in order to reduce the merge conflicts created.

A few agile practices that worked well for us are described in this paragraph. We found ourselves using the Planning Game approach often. During meetings, we attempted to predict what will be accomplished by the end of the sprint and what the next steps needed to be taken to accomplish those tasks. This practice led to an effective backlog prioritization on a per sprint basis. Another agile practice we established was Collective Code Ownership. Many of our developers took part in the implementation of multiple user stories. This led to a collective understanding of what our code was doing and how it was doing it. This also resulted in easy maintenance. Any one of our developers can jump into a task outside of their original domain and be comfortable with expanding or testing it.

Communication was a major factor in the amount of progress made in tier one. Discord was our primary source of communication. In addition to in-class meetings and Discord voice call meetings, our team regularly checked in with their progress over Discord. What worked well for us is that the Product Owner, Scrum Master, and Developers all communicated what they were currently working on, what stage of their user story they have progressed to, and any issues they were having. Also, many check-ins were made over the course of this sprint by the product owner and scrum master to the developers to gauge progress and/or assist the developer with any hindrances they were encountering. Since we were practiced Collective Code Ownership, issues were easily solved through collaboration and pair programming.

In order to track progress of user stories, we used the subtasks checked off on the user stories in Trello in addition to our velocity and estimation guidelines. Our backlog for each sprint consisted of user stories and developer stories. The estimation for completion were assigned the following labels and values:

- XXL – 5/6 days
- XL – 3/4 days
- L – 2 days
- M – 1.5 days
- S – 1 day
- XS - .5 days

These values represent our estimation point system for how long a story might take where each day equals 8 hours. In addition, we used Clockify to track our collective time spent on each sprint. The resulting time sheet at the end of the sprint was used to create a burndown chart to show our progression across the timeframe of the sprint compared to an ideal linear burndown. This allowed us to view how effective our progress was and gave us some immediate feedback on things we could do better.