IFB299 IT Project Design and Development

CRC Release Documentation

Team 54

Student Number	Team Member Name
n9741283	Liam Dale
n9762116	Marco Fiumara
n9723200	Develyn Evans
n9366164	Alice Hendicott
n9936947	Christopher Stock

Semester 2 - 2018

Contents

Process Adherence and Quality

Agile

While working on the CRC website / application for the customer, our team made sure to adopt the principles of Agile: Transparency, Inspection and Adoption.

Transparency: Each team member was trusted to take on jobs that they felt comfortable doing with their skill level, also during SCRUM we made sure that each member of the group felt comfortable to advise if they were having any trouble with the task they selected.

Inspection: Our team originally worked together using the bitbucket website, midway through the first sprint we ran into some technical problems which caused us to change to GitHub during our Sprint. This was to make sure that we followed the Inspection principle, as to enable everyone to have the opportunity to inspect team members work in order to create efficiency and reduce redundancy within the project.

Adoption: Using these made adhering to the last Agile Principle Adoption easier. Utilising the first two principles allowed us to work collaboratively with one another and keep a constant stream of communication and input open as we all worked through our designated tasks.

The Agile principles assisted our team in being fluid while working on this project, making sure that each iteration of our project was a product that would work for the CRC, allowing for changes to be made and communicated effectively. This aided in keeping everyone motivated and clear on their tasks.

Scrum

Benefits of SCRUM, 3 questions – what did you do yesterday, what do you aim to do today, do you have any impediments.

The SCRUM methodology allowed our team to stay on top of tasks, with the aid of GitHub task-board, weekly revised burndown charts, multiple weekly meetings and notes taken by the SCRUM Master. All these methods of communication enabled our team to work effectively, in a lean manner that maximised development progress for the time frame we had.

The GitHub task-board allowed anyone (usually the SCRUM Master) to create ‘issues’. These issues are similar to notes on a whiteboard, in which everyone can see the issues that need to be addressed, and team members can nominate themselves for these tasks, with the ability to clear issues when resolved. These issues became integral staying on top of story completion and thus creating a burndown chart that is much closer to linear progression than our sprint 1 results.

The burndown chart was an excellent way for the team to track our progress with the website development. We gained a further sense of direction from setting a minimum number of issues to be cleared per week and working collaboratively to get them done. This reflected in the burndown chart each week as we updated it, showing our progression to be on track and closer to completion with each task done.

Having a meeting each Sunday, at the start of the week, allowed us to get ahead of our tasks and have some direction with the importance of certain tasks. The meetings were run by the SCRUM Master and started with discussing objectives, assumptions and outcomes. Each were predefined before the meeting based on that week's tutorial content that will be covered. This was done in order to stay ahead of our work and remain flexible on tasks that had a greater urgency for completion.

Product Backlog

The product backlog while being a creation of the product owner, also received the input of the entire team. Team 54 made sure to come up with a large amount of functions required by the business and other functionalities that would improve the product outside of the scope of the task.

The backlog was was initially made early on during our first sprint. We had a large list of features and used poor estimation techniques for the amount of time it would take for each feature to be completed and tested. Near the end of our first sprint we completed our retrospective and during this we heavily revised our product backlog.

The product backlog is heavily influenced by the user stories. As the user stories were revised and updated, the product backlog received similar treatment. The changes made to the backlog revolved around the testing that was required for the revised stories. During this we also modified some functions to better explain their purpose while continually checking with the product owner to ensure the product delivered was up to her expectations.

Customer Engagement

User Story revision & Release Planning

The user stories defined in our Sprint and Release Plan were ambitious. We created highly demanding stories which were underestimations of the time given to us as well as the task requirements. The tasks were made difficult to deliver by Sprint 1 due to a poor job in using the poker method effectively in calculating story points. From this, the developers had an overbearing amount of difficult tasks to complete before Sprint 1.

The subsequent result of this was that Sprint 2 required a larger amount of work than originally anticipated. We revised the story points and came up with a plan for developers to spread the workload for more difficult tasks based on their story point values. This was key to our success on delivering a working product on time with all Must Have, Should Have and Could Have features outlined; leaving the Won't Have story's for 'Sprint 3'.

Although we were able to adhere to our delivery schedule, what was delivered was varied from what was documented for Sprint's 1 and 2. Despite this however; holistically Release 1 contained all story's as outlined in the original Sprint and Release Plan.

Due to the gross underestimation of time it would take to deliver on the agreed upon story's, the estimated velocity in our Sprint and Release Plan falls short of the real velocity at which the development team worked at. This is to be revised to encompass all working time for each developer per week.

UAT Agreement

Each story created goes through a process of naming, estimating story points, determining MoSCoW importance and writing up acceptance criteria. This acceptance criteria reflects how the function defined in the story is to be executed. These acceptance criteria aided in the user testing process necessary for every story, as they are predefined steps that are able to be executed by a tester.

As user stories are created, they are checked over by the development team to ensure the story is able to be coded. The story is then brought to the client for confirmation. If changes needed to be made the development team would amend the story. Once agreed upon, the stories would go into development and then testing upon being completed.

Our testing was done using the Selenium add-in, used through the Mozilla Firefox web browser. Selenium enabled testing to be done efficiently and accurately, staying true to the acceptance criteria documented in the user stories. Testing through Selenium consisted of opening the website and inputting key/button presses as defined in the acceptance criteria, including username and password entries, to ensure each function delivered was working as defined.

The test cases were done over multiple sessions and consisted of the client watching and critiquing the Selenium test suites as they ran. Once satisfied, the client would sign off on the tests to finalise completion of the story.

Customer Prioritisation

During the creation of each story, we assigned a MoSCoW rating to each, indicating its importance to the CRC. These were features that allowed the CRC to utilise what we have done in the short working time. These functions were confirmed as necessary by both our client and tutor. This aided in clarifying the significance of certain functions, thus guiding the development team.

During our first sprint we had to make some changes to our stories and sprint plans to better explain our functions and split them up so they could be implemented faster. Moving certain lower priority stories into Sprint 2 allowed for the development team to write functions of higher importance to a greater standard than previously possible.