Group Project – Individual Report

Matthew Henderson

Team Roles

Matthew Henderson – I was the scrum master of the project. It was my job to make sure that the group stuck to task. I chaired the meetings and liaised with other group members to ensure that user stories were getting completed and that the overall velocity of the project was maintained. With regard to the programme itself, along with christos, I wrote all the game logic contained within the game package. Furthermore, I was responsible for the TopTrumpsCLIApplication class that interacts with users on the command line, and for the testlog class that prints information about the game to the testlog. I also co-wrote the group report with Parvir.

Christos Dritsoulas – With myself, Christos co-wrote all the game logic inside the game package. He was also responsible for the StatsPrinter class that prints all the game statistics to the command line.

Parvir Chomber – Parvir was responsible for all database management. He designed and constructed the Postgres database. Furthermore, he wrote the java classes that connected to, and queried the database. He also co-authored the group report and made the UML diagram of the system.

Christopher Barnes – Christopher, along with Kaikan, was responsible for the development of the online version of the game. They were responsible for all the HTML and javascript, and programmed the rest API that communicated with the game logic. Christopher focused on the selection screen and the layout.

Kaikan Wu – Kaikan, along with Christopher, was responsible for development of the online version of the game. They were responsible for all the HTML and javascript, and programmed the rest API that communicated with the game logic. Kaikan focused on the game screen and functionality.

Experience with Scrum

Lessons Learned from Estimating User Stories

The main lesson I learned from estimating user stories was the importance of making each story as low-level as possible. In particular, our user story “play an online game” was too broad. We should have broken this user story down into more manageable stories or organised a spike. The result of this large story was a decrease in the velocity of the second iteration. This was because we started the story later than expected and it was not completed by the end of the sprint. It was also harder to predict the length of such a large user story.

Overall, our story point predictions proved to be fairly accurate. However, a few were longer than expected. This resulted in a project velocity that was lower than predicted. I think that this highlights the importance of experience when predicting user stories. For the members of the group this was the first attempt at a project this large. I believe that with more experience, members would have been better equipped to accurately predict the story points, and hence better match predicted and real velocity.

The slower than expected velocity of the project also highlighted the importance of building contingency time into the release plan. Had the team not left a week contingency, we would not have had a functioning version of the online version to hand in. Even with more experience, there is too much uncertainty in software design to be certain of the story point predictions made.

Missed User Stories

As stated, some of the user stories could have been expressed in more low-level terms. For example, "play an online game" could have been split into "Choose category in online mode", "View cards in online mode" and "Play a round when it is not your turn in online mode" amongst others. In a similar way, "View statistics" could also have been split into "View statistics in online mode" and "View statistics in command line mode". Having more manageable stories like these would have helped predict the overall velocity of the project.

Apart from those mentioned above, the team agreed at each of our retrospectives that we had not missed any user story that would have been relevant to the requirements of the project. We were happy with the iteration breakdown and workflow of the project.

Requirements that were hard to express with scrum

The requirements that we found hard to express in scrum were those that did not belong to a real user. For example, when an AI player auto-selects which category to play. We solved this by creating a user role that was "AI player" and assigning the user story to that role.

Furthermore, requirements that were not directly performed by users were hard to express. For example, shuffling the cards is not performed directly by a user, but is instead performed automatically in the setup of the game. The team considered a separate "game" role that would act as a user to deal with such requirements. However, in the end we assigned this story to the player role, although this was not an ideal solution.

Furthermore, time taken to accustom ourselves with new concepts such as rest API's, HTML and javascript was hard to quantify with scrum. There was a lot of learning involved in this project that was required to achieve full functionality. This was detrimental to the overall velocity of the project as predicted by the scrum process.

What I have learned

The project was a learning curve in many respects. Firstly, it was a great opportunity to see the scrum process in a real life setting and to follow it through to the end. It provided valuable experience with requirements capture and story point estimation. It also gave me my first experience of sprint reviews and retrospectives which I found a particularly useful part of the process, as it allowed for learning and improvement after every step. The project also showed how important communication between team members is whilst working on a project such as this.

My role as scrum master showed me how important proper delegation is on a group project. Indeed, I feel that I did not do this enough and ended up doing a lot of work myself. This was one of the most valuable lessons that I learned.

From a technical perspective I also learned a lot. I got a lot of valuable experience using git and github. It was especially useful in a collaborative environment where merge conflicts are a reality. This helped to reinforce good practices such as proper branching of the project. We also had to revert several commits which showed the usefulness of proper version control.

The project provided a good opportunity to use the model-view-controller architecture. I had used this architecture before but never on a project this big. Using it in this project gave some perspective as to why a proper architecture is important in increasing the cohesion of a larger project.