COMP2211

Final Deliverable Report - Group 23

Patrik-Tibor Csanyi (ptc1g20@soton.ac.uk)
Adam Clarke (ac4g20@soton.ac.uk)
Guillaume Comet-Vernet (gcv1u20@soton.ac.uk)
Madhav Muralikrishnan (mkm1g20@soton.ac.uk)
Andrew Sansum (ajs2g10@soton.ac.uk)
David Stefanov(ds1u20@soton.ac.uk)

Evaluation of Teamwork

In terms of teamwork, we chose to follow a decentralised approach which worked rather well. We gave a lot of liberty to our individuals to work at their own pace, and at a time of their own choosing. At the beginning of each sprint, we split up the work roughly equally, then each member finished their tasks when they had time so as to not put unnecessary burdens or deadlines on our members. We did try to split the tasks in such a way that if one had dependencies, it was allocated to someone who liked getting their work out of the way early. We also decided together how to split the tasks, and due to the fact that each member felt they needed to contribute, we did not have any problems with people not participating or avoiding doing their fair share.

In the first sprint however, the project came together quite late (the branches were merged a few days before submission) and so there were quite a few complaints and arguments and we ended up redoing some parts. To avoid such a situation from happening again, the sprint leader closely monitored each branch in the following sprints, checking in every other day to make sure everything was going well, and luckily with these precautions, no such problems occurred afterwards. Other than at the very beginning we did not have any major problems with our teamwork mostly due to the efforts we put in at the beginning to set up this structure.

At the beginning of the project, we used the SCRUM method to organise ourselves, however as time passed, we unintentionally transitioned to what we would later learn is the Kanban style. This approach felt much more natural to us, and it was better suited for the decentralised nature of our workstyle.

We attempted to follow the extreme programming values, but only insofar as it felt natural. In terms of simplicity, the structure of our workstyle, our communication as well as our final product prioritised simplicity and efficiency over superficial features (more on communication in the following chapters).

Feedback, respect, and courage were important values that allowed us to work through difficult situations. We each believed that feedback was important, not only from our supervisor, but also from one another so that we could deliver a good product as well as maintain a good atmosphere and avoid building up resentment. Some discussions were definitely more difficult than others, and many in our team demonstrated their courage to point out flaws in the project even if it had the potential to sour relations. Nevertheless, by doing our best to be respectful, and keep an open mind, we managed to avoid major conflict.

Time Expenditure

During the envisioning process, our team has set up the previously mentioned infrastructure for the project, allowing us to work rather efficiently. When we split our work amongst the three increments, we decided that the first two should have the majority of the tasks to ensure we did not run out of time, whereas the last sprint would be spent finishing any remaining tasks, adding extensions, and polishing the final product. This approach went rather well for the majority of the project, and while we did not finish all the tasks in either the first or second sprint, that was intentional, and by most measures we outperformed our expectations setting ourselves up well for the final sprint. That being said, this approach did hurt us in the end, as we did not feel pressured to do much work during the easter break, and afterwards it was quite difficult to get back to the project. This unfortunately resulted in us not being able to finish all the tasks, we also did not succeed in polishing most of our features, and worst of all we failed in conducting enough regression testing. This led to some of our features that have been fully completed not working properly in the final deliverable.

If one looks at the burndown charts, they might get the impression that for a large part of each sprint we procrastinated and delayed our work, as only in the end does our total workload start decreasing. That, however, could not be further from the truth. As mentioned, because of our decentralised approach, we had many tasks being done simultaneously, and made good progress throughout the sprint. Our workload did decrease linearly, however tasks were often finished at the later parts of the sprint.

In retrospect, there were a few things that we could have improved on. One such improvement is setting a preliminary deadline before the actual deadline. During our sprints, we would code until a few hours before the deadline, adding new functionalities. This inevitably caused certain bugs to arise and, as a consequence, we had no time to fix them. This made our product not reach its maximum potential in terms of quality. If we were to do these sprints again, we would put a preliminary deadline at least 2 days before the actual deadline which would give us sufficient time to fix any major bugs.

Throughout the whole process, our main aim was to incorporate as many features into the application as possible, which left some of them unpolished. After the sprints, we all acknowledged that we could have spent more time perfecting these, instead of adding more which would have made our application more streamlined and intuitive.

In terms of time distribution, each person spent a reasonable amount of time on the project. After some rough calculation, one can comfortably conclude that each person spent more than 20 hours on the project. Some people however did end up spending more time on it than others (upwards of 45 hours!). That, however, was less because of some people's unwillingness to work, but more so because we did not succeed in estimating the difficulty of our tasks well, and because of the generosity of some members, who spent a great deal of time helping others and taking on tasks that no one wanted. Such honourable mentions include Adam, who put in much effort completing, and helping others complete difficult tasks, Madhav who spent a great deal of time doing paperwork, tasks that many wished to

avoid if possible and Patrik who did most of the managerial work including setting up the meetings, making presentation, checking in with the members for progress and pushing us to finish our tasks in time.

The tasks we were the furthest from estimating correctly were the setting up of the structures backbone, such as navigation from scene to scene which we severely underestimated as well as the making of the visualisation of the runway (without the details, just setting up the basic visualisations), which we also underestimated. In contrast, the addition of the indicators to the runway were severely overestimated. We also underestimated the time it would take to thoroughly test our code, to which we did adjust in later sprints.

To estimate our tasks, we used the planningpoker.com website, where the units we used were the powers of 2. We found this tool to be rather useful, although we may have relied a bit too much on it. While the website helped each member voice their opinion, our estimations, perhaps due to inexperience were quite flawed. The website averaged our guesses, which we used to decide the final difficulty. In our backlogs, we estimated each task's difficulty in units, where one unit was half an hour. We decided to use this unit, as half an hour was the smallest reasonable unit of time that held any meaning. We wanted to acknowledge the work each member put in; therefore, we chose a rather small timeframe as a unit. That being said, the length of the unit was still large enough not to cause confusion, and make members bogged down calculating how much work they put in. This estimation was very helpful, as it quickly showed how much work each member put in throughout each sprint and allowed each member to take on tasks that (as far as we estimated) were of similar difficulty. The estimation of tasks that repeated throughout the sprints, such as testing, writing reports, scenarios and storyboards, did improve, as we often used the actual results of the previous sprint to estimate the time investment these tasks would take in the next sprint.

Overall, we are quite satisfied with our time management. The only things we would change if we had the chance, would be to avoid getting comfortable while we were in a good position, maintain the pace we set at the beginning and add further deadlines of our own making before the official deadlines.

Tools and Communication

In terms of communication, we mostly used Discord to discuss and deliberate. We did have SCRUM meetings; however, these were not daily. We tried to be as efficient as possible with our meetings and so, we preferred discussing things via text, as this saved a lot of time and more often than not, there was no need to have a voice call or a meeting. We used Discord to voice concerns and asked for help when stuck. We did conduct roughly two meetings each week, one at the beginning, and one at the end. However, looking back, it may have been better to conduct three, putting one more in the middle of the week as well, as that would have allowed us to move forward as a much more cohesive unit. However, we do not believe any more than that would have been necessary, as even with just two meetings a week, we managed to maintain good communications.

We used GitLab as version control. Each person developed their code in their preferred method. Some used IDEs, some did not. Some used IntelliJ, some used VSCode. Even in terms of version control, some used the features provided by their IDE, and some used GitHub Desktop. All this is to say that we tried to accommodate the preferences of each member so that we all felt comfortable, and could focus on developing the software, and not on learning new tools.

We used a wide variety of applications to create our documentation, as each person used what they were most comfortable with. For example, even writing the reports was done on three different platforms. Some used Microsoft Word, some used Google Docs, while some preferred LaTeX. Similarly, for charts and backlogs, we used both Microsoft Excel as well as Google spreadsheets, and even for presentations we used Microsoft PowerPoint, Google Slides and Prezi.com. For storyboards we used moqups.com, Photoshop, as well as Microsoft Paint. While in terms of Diagrams we used Lucid Charts. The one consistent element was the use of Google Drive as our primary platform for sharing all documentation not directly code related.

We cannot clearly claim any of these tools or platforms were better than others as it is mostly a matter of preference. The one thing we did agree on is the use of Discord to communicate, as it seemed to us as clearly the best platform both for text communication and voice calls. We were also in agreement with regards to the fact that there was no real need for daily meetings. Furthermore, there was no real discussion in terms of using Google Drive to be our primary platform for document sharing, as no other options even occurred to us.

Advice to Next Year's Students

For any future groups, we strongly recommend the use of the same methods we used. While we did not perfect the use of the previously mentioned decentralised method, we are certain that in the long run this is by far the most efficient and considerate approach. Therefore, learning how to manage its downsides will be of great value to any team who uses it for this project.

Planning Poker is also a great tool to help people voice their opinion, however relying too much on it is not going to be beneficial. It is important to keep in mind that this is merely a tool in the estimation of difficulty, not a replacement for all other strategies. As for our thirty-minute units, we cannot recommend them more. Some of our members use this unit for time management in their daily life as well, and it is very effective.

We also recommend the use of the Kanban style over any other agile method, as it is less complicated and easier to manage leading to great increases in efficiency. We also strongly urge any future team to not brush off the extreme programming values and practices, as they are invaluable to maintaining good working relations within any team.

Furthermore, we advise the sprint leader to take an active approach in managing the project during their term, as they are the glue that keeps the individual parts together. We also recommend, as our lecturers did, switching sprint leaders after every deliverable, no matter how good a job they did, or how willing they are to continue their work, as it is an incredibly tiring task that brings with it a lot of responsibilities, and it can easily lead to burnout.

We can also wholeheartedly recommend overloading the first two sprints with more tasks than can be reasonably finished, as even in the worst case a normal pace will be maintained, however in the best case the team will progress faster than expected. That being said, we would caution any future team not to become complacent if they manage to get ahead of schedule.

We also think there is no need for more than three SCRUM meetings per week, however that might be largely dependent on the members of the team.