University of Southampton

COMP2211: Software Engineering Group Project

Deliverable 5: Final Team Project Deliverable

Group 30: Benjamin Lellouch (brl1u18), Daniel Best (db5n17),
Deniz Matur (dm1n17), Iman Dzulhadi (ids1g18) Rajpal Dhillon (rsd1n18)

September 28, 2020

1 Evaluation of Team Work

Overall, the team worked well together over the duration of the project, in such a way where we were able to effectively produce the required artefacts in a timely manner. We were able to deliver all of the required artefacts on time in each sprint, with the system itself meeting all of the core requirements with the production of increment 3. For the most part, we received exemplary feedback from our supervisor and the additional reviewer at each sprint review in regards to these artefacts — particularly when it came to the subject of testing. This was achievable due to our professional presentation style during each sprint review, where we employed slides as well as a PDF document to convey our progress in as clear a manner as possible. We were also able to effectively spread the workload such that, for the most part, we were able to split the work up between us evenly. This meant that each member of the team was free to invest a significant amount of time into each aspect of the project, which therefore increased the quality of what was produced. We also made a conscious effort to setup a good work environment that allowed us to work effectively as a team, which will be discussed in-depth in section 3. Prior to the COVID-19 pandemic that occurred during the course of this project, we also met with our supervisor on a weekly basis — we took advantage of these meetings by ensuring that we always had something to show him, such that he could provide instantaneous feedback prior to the sprint review that we could use to improve that feature or artefact. We also always compiled a list of questions to ask our supervisor prior to the meeting when we were otherwise unsure of how to proceed. Using our time with our supervisor in this manner allowed us to reduce the amount of problems that would have otherwise occurred during a sprint review, thereby improving the quality of our work. Last, but certainly not least, the team did a good job of overcoming the unforeseen circumstances that occurred due to the COVID-19 pandemic. We ensured that we still communicated frequently with one another over this time, which allowed us to produce good quality work over the course of sprints 2 and 3, despite not being able to physically meet with one another over this time period.

There were, however, also elements of the project that could have been improved in hindsight. Firstly, we missed three of the optional extension tasks in our last sprint: the screen reader, the 3D view, and the zoom and pan tools for each of the visualisation views; we decided that we did not have enough time to implement these to a good quality given our other module commitments. Having said this, if we had planned so in advance, we could have pushed more work into increments 1 and 2, and would have therefore had more time available to work on these tasks and implement them into the system. We also did not formally communicate the delegation of our tasks, which luckily did not lead to any issues, but could have possibly resulted in team members working on the same activity. This was because, although we used JIRA at the initial planning stage of each sprint, we did not use it during the main body of the sprint to assign tasks to members when required, and instead simply discussed what we were each doing via Slack. We also could have had more daily scrums, especially in sprints 2 and 3 during the ongoing global pandemic, as we became rather lapse with formally following this aspect of the scrum methodology during this time. Also, whilst we did assign a new scrum master at the start of each sprint, the individuals assigned as scrum masters never really took the formal leadership role that they should have according to the scrum methodology. Having an individual in a leadership role at each sprint would have probably helped with the previously described issue of formally delegating tasks, and minimised the risk of work duplication.

For this project, we applied an agile approach, using aspects from various methodologies including scrum and XP; this approach provided both advantages and disadvantages over the duration of the project. In terms of advantages, using an agile methodology over the course of this project allowed us to easily iterate on our artefacts, and create increments that enhanced the system by adding new features and fixing pre-existing problems. This is in stark contrast to a more traditional methodology, like the waterfall model, where it is exceedingly difficult to modify and update artefacts in this manner. The use of scrum also ensured that the team communicated with one another, and with their customer (supervisor), on a much more regular basis than they may have otherwise done; this was achieved by means of a variety of different types of meetings, in particular daily scrums and sprint reviews. By using the scrum methodology, this also guaranteed that we wrote user stories in a style that conformed to the INVEST acronym, which in turn could be used to check that the system satisfies the customer's requirements. Using the MoSCoW prioritisation model on the product backlog allowed the team to order the tasks in such a way where the application's features that were implemented in each sprint were selected based on the value that they would provide to the customer, and therefore ensured that the best possible product was created for increment 3 with the time given. Giving user stories scores allowed us to produce a burndown chart that effectively communicated the team's progress in a given sprint, and illustrated to the customer how the team managed to progress over the duration of the sprint.

Despite mostly being of benefit to the project, using an agile approach also came with some disadvantages. One such disadvantage is that the documentation that we produced during the envisioning stage of the project has to be frequently changed to accommodate for any unforeseen changes that were made during that sprint, since the details of the system are not extensively planned out as they would be when using a more traditional methodology. The considerable amount of meetings that using an agile methodology enforces can also be a double-edged sword, since sometimes this takes away valuable time that would have otherwise been better spent working on the increment when that level of communication is not necessary.

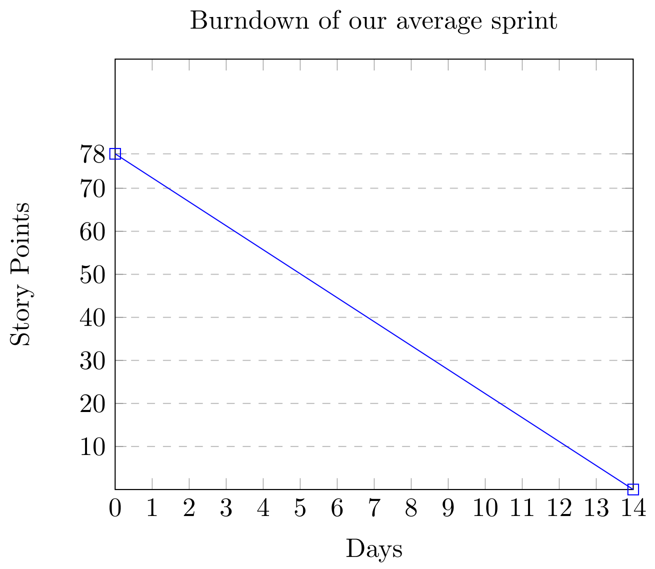
We also took into account the XP values to varying extents over the course of the project. Firstly, in relation to **simplicity**, since we defined user stories that adhered to the INVEST characteristics, we were able to ensure that we did not exceed the bounds of an assigned task. The fact that we also defined sub-tasks for each user story allowed us to take small steps to achieve the overall goal for a given user story, although some of these sub-tasks were not as specific as they maybe could have been. For the most part however, this meant we were able to be certain that we produced what the customer wanted, and guaranteed we did not overstep any boundaries that may have resulted in crossover with another task that was being implemented by a different team member, or a feature that potentially was not desirable for the system at all. When it comes to **communication**, the team pretty rigorously enforced this value by communicating with one another using Slack on an almost daily basis. This communication was definitely beneficial as it allowed us to delegate work effectively (although informally), and help each other solve any problems that we may have, whilst also making sure that the work that was being done by each team member would all fit together nicely when integrated into the wider project. This level of communication was particularly of benefit after the effects of the COVID-19 pandemic came into place, as it allowed us to continue to work effectively as a team despite being in completely different time zones and quite literally on the other side of the world from one another in some cases. Thirdly, the **feedback** value was once again pretty strongly adhered to — as discussed earlier, we always made sure that we demonstrated as much of the work done for a given increment to our supervisor as early as possible to collect feedback that can be applied before the sprint review. The formal feedback that we received at each sprint review was also taken into account during every subsequent sprint, and we received praise during most subsequent sprint reviews for our ability to adapt to this feedback. The XP value of **respect** was also followed by each team member — there were never any altercations between team members, and we largely worked well together without any significant issues, which definitely improved the outcome of the project. The final value of **courage** was perhaps not followed as closely as the other four; whilst we were definitely truthful with our progress, we perhaps could have done a better job of estimating how long tasks would take, especially in relation to our user story scores.

2 Time Expenditure

In our first sprint, we were able to complete all of the user stories we have chosen to implement this sprint. These 9 user stories, which were originally prioritised as 6 “MUST” and 3 “SHOULD” but were then all re-prioritised as “MUST” , amounted up to a total of 68 story points. Even though we implemented all of them in the allotted time, we underestimated the time required to write our report which made our first sprint pretty stressful and is one of the reasons why our first submission was late by a couple of minutes.

During our second sprint, we implemented 10 user stories (All “MUST”) summing up to 81 story points. Whilst our first sprint was more focused getting the Model/Back-end working with a minimal Controller and a simple View, the second sprint focused more on getting the visualisation part of the product working as well as adding notifications and support for import XML files. The largest complication that we encountered during this sprint was the sudden end of term as well as the beginning of confinement measures. Thankfully, we had completed most of the user stories by the time the original deadline had passed meaning that we didn’t add anything major during our break.

Finally, in our third and final sprint, we had set out to implement 14 users stories (4 “MUST”, 5 “SHOULD”, 5 “COULD”) which totals up to 124 story points. Unfortunately, we decided not to implement 3 of these user stories mainly due to a lack of time. Therefore we only accomplished 11 user stories (4 “MUST”, 4 “SHOULD”, 3 “COULD”) amounting to 84 story points.



According to the burndown of our average sprint and sprint reporting, we have averaged 78 story points per sprints which means that each member expended about 16 story points per sprint. On average, we each worked 10 hours per week which falls in line with what is expected of us. Therefore, a single story point is worth about 1.5 hours of work.

These following user stories were the most expensive to implement:

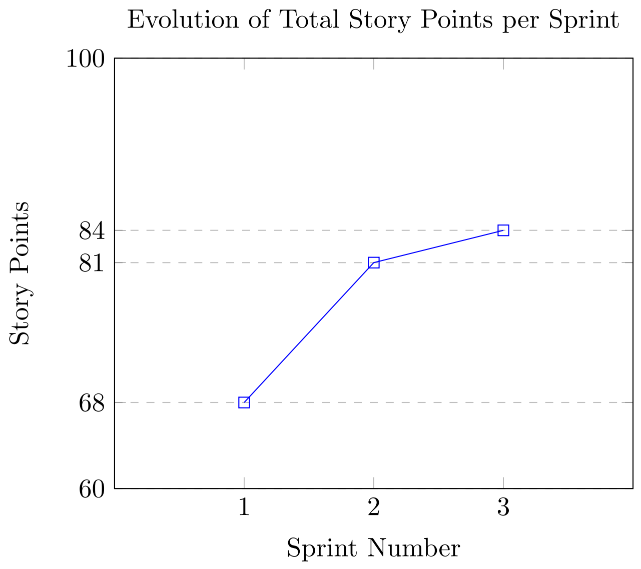
- Revised Runway Calculations (13 points, Sprint 1)
- Side-way View (13 points, Sprint 2)
- Bird’s Eye View (13 points, Sprint 2)
- Real World Overlay (13 points, Sprint 3)

They revolve around visualisation of the runway as well as calculating the revised runway both of which are calculation-intensive. These estimations turned out to be precise as the user stories required a lot of manual testing as well as a considerable amount of tweaking to get the expected results. We also found the production of design artefacts and the writing of the report to be a lot more expensive than we first expected.

For effort estimation, we decided to use the Fibonacci sequence because a linear scale (1-10) uses numbers that are too close to one another which makes it impossible to distinguish as estimates since, according to Weber's Law, the difference we can identify between objects is given by a percentage. Therefore the Fibonacci sequence works well as each number is about 60% larger than the preceding value whereas on a linear scale the percentage of the difference between two consecutive values shrink when values become larger.

We found this effort estimation to be particularly useful as the nature and size of our user stories fitted the sequence perfectly. Our smallest user stories were estimated at 5 points (e.g. Runway Revision Notification) whilst our largest stories were estimated at 13 points (e.g. Side-way View). Finally, our mid-sized user stories were estimated at 8 points (e.g. Import Airport).

The estimation of each user story was accurate but we underestimated the number of user stories we could complete in a sprint in sprint 1. This underestimation can be explained by the fact that we didn't know how competent we each were at coding in Java, but also because we underestimated the amount of free time we would have to implement the user stories we had set out to complete. We rectified this mistake in the following sprints as you can see in the graph below:



We balanced the workload for each team member by first splitting up the story points evenly between each other. Members with more experience in Java and JavaFX were usually assigned to more valuable tasks but did less of them whilst less experienced members did more tasks that were worth fewer points. We also took design artefacts into the equation. If someone felt more comfortable producing artefacts such as Class diagrams, Sequence diagrams or Storyboards, he could give up a programming task in exchange for producing such artefacts. In terms of testing, we made sure that everybody contributed in one way or another whether it be scenario testing, boundary testing, unit testing etc ...

We noticed that, at the end of each sprint, there was a couple of days with very little team activity (e.g. little communication on Slack, little to no commits on the repository etc...). This "recovery period" was probably because the team had to produce a report at the end of the sprint and then had to prepare a presentation and present it the next day. We could have organised our time more effectively by using this period as an opportunity to get a head start on the writing of the report so that we end up being in less of a rush when the time comes to submit our work the next increment. We could have also managed our time more effectively by choosing more stories to implement during our first sprint. By doing so, we could have been able to deliver a product which contains more optional features such as a screen reader.

3 Tools and Communication

In this section, the tools and communication applications employed will be elaborated on to discuss their value and effectiveness, and the team's strategy for physical meetings will be discussed. Firstly, the tools used for software engineering will be explained — the team made use of Lucidchart to create UML and Sequence diagrams; IntelliJ IDEA, Maven, GitLab and other libraries such as JavaFX and TestFX were utilised for the actual creation and implementation of the project, and Balsamiq was used for the creation of our storyboards too. The team used Lucidchart for both our UML and sequence diagrams, as we were familiar with how it worked and knew how to get the job done with it. Both IntelliJ IDEA and Maven were utilised as the team had prior experience and were comfortable using both of these tools; IntelliJ IDEA also has integration with Git, which made it more appealing. GitLab was employed mainly for its repository, as the team knew we could potentially run into coding issues; GitLab's version control, issue tracking and integration services made it an indispensable tool in our arsenal. GitLab's CI/CD feature also came in handy: continuous integration, delivery and deployment (CI/CD) allowed us to push code to our repository, with the code being validated before it is merged into the main branch. This ensured we were notified of any errors or bugs so we could troubleshoot them. JavaFX was preferred over Swing for our GUI, since we were more familiar with it and in our opinion, it produces a sleeker and cleaner looking interface. TestFX was incorporated in for automated testing of the user interface — as we had multiple scenarios to be tested, we believed having the ability to test the interface automatically would save time, avoid any human error related with manual testing, and above all else, was a more professional approach. Balsamiq was adopted to create our storyboards, the ease of creating sketches of our ideas with it made it the perfect tool for this task. All of these applications were used to achieve different goals, and each one of them was unbelievably effective in achieving it.

As for collaboration: JIRA, Git, Google Docs, Google Slides, and Overleaf were utilised. These were all used for vastly different reasons, and they were all effective in helping the team meet its objectives. Given that JIRA is an industry-standard tool commonly employed for agile-based projects, it was used by the team to track progress for the sprint, as well as allocate tasks to its members. Members could then check which user stories they have got to implement, which gives them a sense of responsibility and ownership. Git was used so that members could work independently, yet in a collaborative manner. Its version control proved to be a useful feature, as there were times when a member's code got overwritten and having an older version to revert to allowed the team to proceed hassle-free. Google Docs was almost used as a canvas for ideas, members could edit and share anything they believe to be beneficial for the team's progress; it was also heavily relied upon for the first deliverable, since it also gave members the chance to critique each other's work and provide useful advice on how to improve. Google Slides similarly worked for the team, but it was solely employed to produce the team's presentation that was needed for each sprint review. Overleaf was made use of for the creation of the user guide since the team was familiar with its format, and its collaborative nature made it an excellent choice for the task — members could edit and add more valuable information.

Lastly, the team took advantage of applications such as Slack, UoS Outlook, Facebook Messenger, and Microsoft Teams for its communication. Initially, the team had decided to use Facebook Messenger as it was a social media platform that all members were frequently active on. However, given the module's spirit, the group decided to completely immerse ourselves and adopt a more professional approach. Hence, the team agreed on Slack since it has excellent integration with JIRA and gave us a taste of using a more professional platform. Its integration with JIRA allows members to keep up to date with the progress of other team members and their respective parts. The team relied on UoS Outlook to communicate with our supervisor as well as to set up meetings and presentations. Given the recent circumstances, the team made use of Microsoft Teams to still engage with one another while we worked on completing the final deliverable as well as the final presentation. The team's most preferred mode of communication would be Slack, as we felt this was the one that worked well for everyone.

Concerning our strategy for physical meetings, we had devised a plan in our first meeting. That plan entailed meeting twice or three times a week excluding the weekly meetings we had with our supervisor; sprint planning was usually done in the first and second meeting post deliverable, and an informal sprint retrospective was held following the more formal sprint review we had with our supervisor and the additional reviewer. We would update each other on our last meeting every week, where we were currently at with the implementation of the user stories assigned to us, this gave those who needed help the opportunity to voice out. Overall, the team is pleased with our plan as we were able to stick to it, and it facilitated our ability to meet every deliverable's deadline.

4 Advice to Next Year's Students

After successfully completing our SEG project, we've found many techniques and work practices that worked well for us as a team that future SEG students may wish to implement within their own groups.

Regular meetings are key. Having frequent bi-weekly deadlines for each increment may be overwhelming if you don't have regular group meetings both with your supervisor and also just as a group in order to formulate a realistic plan with manageable tasks split up between team members. Hence we found that meeting with our supervisor at least once a week helped us to realise any shortcomings we had in our increment and anything that needed improving or changing prior to the sprint review, as well as things we needed to add in our next sprint. As well as meetings with our supervisor, we found it very useful to have as many group meetings as possible while not taking up too much time needed for other commitments. Having many group meetings allowed for us to all contribute ideas and holistically plan out how to move forward with our sprint increments, as well as giving us more group time meaning that if any individual was struggling or had issues with their tasks then other group members could help them in-person.

Communicate within your team as much as possible. As well as meetings in person, we found it vital that we regularly communicated as a group virtually whilst we were away from each other. The online tools we used to talk as a group and split up tasks helped us to communicate effectively within our team. We would advise future groups to make use of online agile tools such as JIRA to assign and track tasks within each given sprint to group members, and communication tools such as Slack to have a dedicated team communication channel. It is also useful to use tools that provide integration with other tools and online services you are using — for instance, Slack integrates well with JIRA via the appropriate plug-in, allowing for instant notification to members when it came to any updates in tasks within the sprint.

Allocate a realistic time to spend daily/weekly for your given tasks. It is important to split up the workload and start your tasks on time according to the plan you've made as a group in order to not be overwhelmed by the frequent hand-in deadlines. By adhering to this agile group working strategy, and by following your daily work plan, you should be able to produce effective increments to hand-in on time, and it should help alleviate the possibility of stressful all-nighters before the sprint deadline.

Don't neglect your design artefacts. We found that spending time on our design artefacts, such as our UML diagrams, Sequence diagrams and Storyboards, helped us to have a better understanding of the framework for our project, and were useful guidelines to help us visualise our project as a whole, allowing for our group to have a clearer objective. As well as design artefacts being beneficial to us as a group in terms of a useful guide for what to do, it also helped our supervisors to understand our group's project and vision better, allowing them to help give us feedback to issues they may not have spotted without seeing effective design artefacts.

Don't neglect your testing and try alternative testing methods. Look to branch out and use different types of testing. For example, our group implemented CI/CD testing within our GitLab repository where during each Git commit our test environment would compile the newly committed project and run our tests, notifying us of the results of the testing. It is also important to make sure to map each of your user stories to corresponding tests in order to have full test coverage within your project, something our supervisor stressed to us within our feedback.

Make sure you read the hand-in specifications before the deadline. Although this is obvious, we had an issue where we had finished all the work for one of our sprints but we didn't check the correct hand-in format required until moments before the deadline, resulting in us just missing the deadline due to us having to change our hand-in to adhere to the proper format. By thoroughly reading the specification prior to a deliverable's deadline, this issue can be completely circumvented.