### **School of Electronics and Computer Science**

### **University of Southampton**

### **Runway Redeclaration Tool - Deliverable 5**

### **COMP2211 Software Engineering Group Project**

Team 10: Andrew John Evans (aje3g16)

Lyubomir Tsvetanov Hristov (lth1u17)

Suyash Datt Dubey (sdd1n17)

Kemal Yesildagli (ky1g17)

Denis Kovachev (dk1g17)

[**COMP2211 Software Engineering Group Project**](#_kklxet85tyh)

[**Teamwork evaluation**](#_qazuv6rhq9qd) **2**

[**Time Expenditure**](#_ucjyyfjsod13) **4**

[**Tools and Communication**](#_3ajsggz3mv1t) **6**

[**Advice to Next Year’s Students**](#_y8p47j7xmfb3) **8**

### Teamwork evaluation

Overall, we are really satisfied with our performance because not only did we work well as a team in general, but there were also few to no breakdowns in communication over the course of the project. Even though everything ran quite smoothly throughout, there are certainly improvements to be made and we’ve collectively learned a lot about teamwork and developing software as part of a team - which will help us later on in our education and careers.

One reason why our team worked quite well, was the fact that we all had a sense of urgency and approached this project with a high degree of professionalism. We would arrive at a scrum meeting with actual progress and problems which we needed to solve. We held extra meetings once every week, where we discussed our current plan, implementation ideas, etc. which helped us get in rhythm of doing the work and working with each other. We ensured that we communicated regularly (both online and in person) to indicate progress and made use of tools that allowed use to collaborate easily. We made use of our team members’ strengths, particularly in the development phase, allowing people to implement parts they had a good understanding of, could develop well and enjoyed developing. Also, everyone contributed to every deliverable in a reasonably fair way, which made sure we all felt accomplished at the end of a sprint and caused no unnecessary tension due to a few people being overworked.

However, due to the fact thateveryone has unique work habits and models, as well as conflicting schedules, we would more often than not, work at different times. That was problematic, as sometimes meetings had to be rescheduled to fit the dynamic work life of the group. Moreover, we put a lot of the functionality of the application into the first two development sprints intentionally, however this made us do a large amount of work in those weeks. Continuing at that pace would’ve been unsustainable for a longer period of time and did cause some sense of burn out after each deliverable was handed in. Furthermore, at times, we had trouble starting an increment, meaning sometimes things were left to the last minute. However, this certainly improved over the course of the project.

We organised scrum meetings twice per week where we discussed ideas and talked through the plan for the current sprint and each person’s progress. Additionally we had meetings with our supervisor where we showed our work so far and shared our ideas with him, in order to get valuable feedback about how to proceed. We also used **eXtreme Programming** techniques alongside the development process of our application. This includes code review, [pair programming](https://stackify.com/pair-programming-advantages/), [unit testing](https://stackify.com/unit-testing-basics-best-practices/), and frequent communication with the customer. Code reviewing each other's work was very useful, as in this way we were able to make sure that a feature is implemented as planned. Additionally, this allowed for a faster error detection/correction of code. Most of the difficult to implement features we coded in pairs, which greatly improved the quality of code and the speed of writing. In this way we could figure out how to approach the problem more easily. We created unit tests to verify that the already implemented functionalities are not affected by any new ones we have decided to add.

On a few occasions we had to wait for other team members to finish their planned work, so that another member can do their job. This waiting added some stress during the development and lead to things being left to the last minute. One way we could have made this better is by putting a soft deadline on the work that we need to do for the increment a few days before the actual deadline. In this way, even if we were running behind we would be sure that we have given ourselves enough time to finish the increment since we have planned the schedule earlier. Also, we often found it difficult to communicate about implementation details online, with our points being misinterpreted and causing some amount of tension and stress. Some team members were quite reserved in some meetings and perhaps we could’ve collectively ensured that everyone was included and had a chance to participate/put their point across.

One thing that really helped us stay organized and manage our work during the development process was adopting an agile methodology. All of the team members found this very beneficial because of the various advantages it has. Itprovided us with a method for streamlining planning. In this way everyone had a clear idea of what their task was for each deliverable. By using burndown charts we were able to assess how much work we had completed and how much was left for a deliverable. It improved the overall communication among the team members, which made the development process easier and quicker, and allowed us to plan everything as a team, avoiding any mistakes and misunderstandings. Furthermore, the customer was involved in the development process, allowing to review our progress and get feedback promptly after implementing any new functionality. This also allowed us to make changes as they arose and ultimately led to, in our opinion, a superior product. Because we developed the software using Scrum, we had to pick a team member for the role of Scrum leader. Whenever we a teammate had a problem of any sorts, the Scrum leader was the first point of contact. This helped the other developer feel supported, listened to and respected, and the impediments (informed through the scrum meetings) ended up being resolved in a timely fashion.

That being said, there were some downsides to the agile methodology. For example, it was quite difficult to gauge how much effort would be required by the project, especially in the early stages of development. For that reason, the first deliverable was quite stressful, as the effort estimation (we go more in depth later on) was quite off. Also, the iterative nature could be intense at times. After finishing an increment, we were thrown straight into the next one. As a result, we made process fast but we did suffer from a sense of burn out at times, often needing time to rest after delivering an increment. Improved planning and better time management throughout the sprint would’ve helped with this.

Another thing, that turned out to be very beneficial for the team was following the XP values throughout the project:

* Simplicity
  + We completed what was agreed upon in the scrum meetings and took small steps towards creating a completed product. We fixed bugs and other failures as they occurred. And eventually created a product we were all happy and proud of, without unnecessary over complication of the product. This simplicity made it ease for team members to understand and add to, without misunderstanding.
* Communication
  + We had two SCRUM meetings every week and everyone actively participated in the meetings. We worked on all deliverables as a team and everybody got the opportunity to contribute to each submission. When there was dysfunctional communication or tension, we took steps to handle the conflict quickly and efficiently.
* Feedback
  + Every deliverable, apart from the first envisioning deliverable, we submitted a working product. Every two weeks we demonstrated our progress to the supervisors and carefully listened to their feedback in order to make changes that would improve the design and performance of the product. We’d always show response to previous feedback in our following sprint review sessions. Taking this feedback into consideration made a considerably better product and development experience.
* Respect
  + Everyone respected each other and everyone’s ideas were valued and taken into account while developing the product. This respect helped us act professionally throughout and as result, never had any major disagreements.
* Courage
  + We were always honest and clear with each other about the progress we had made in our respective tasks and estimates on how long it would take to complete them. This courage allowed us to adapt our schedules and overcome challenges with ease.

### Time Expenditure

For the duration of the this group project, the group decided on a strict schedule (although allowed for flexibility with other urgent commitments) of when our group meetings would be conducted. We had Scrum meetings every Monday and Thursday after lectures for about 15 minutes, to update each other on the progress of the task. We also met once or twice during the week, at a time convenient for everyone, to discuss implementation issues and solve larger scale problems. Individually, we decided to be flexible and do project work around our other commitments, as long as there is enough time for the features to be tested, reviewed and soundly integrated.

During the envisioning stage, we spent a lot of time understanding the specification we had been given, the context in which is set and researching and resolving ambiguities that we found. From there, we built up an idea of who would be using our product and how they would use it through stakeholder analysis and personas. We spent some time researching and familiarising ourselves with a set of tools that we’d use throughout the course of the project e.g Trello, IntelliJ IDEA. This was also the stage where we planned our future increments. We spent some time designing the intended class structure for our application, that was later added to and revised as we implemented the functionality.

The majority of the time spent on the project was spent implementing the application. This was split between coding meetings, pair programming and individual development time. The time spent coding ramped up during the increment, starting off pretty slowly and reaching a peak around half way through the increment - where we would typically have completed some of the user stories. Some time was spent designing storyboards for GUI elements and discussing design decisions for implementation and user experience throughout, although this was overshadowed by the time actually implementing the functionality. As the deadline for a deliverable approached, we often met more frequently, to work on the report for the deliverable and to work on the last minute problems that arise. The final user stories of an increment were completed in the days prior to delivering the product. Completing more user stories earlier in the cycle could’ve been beneficial. We also met prior to each sprint review briefly, to go through the intended structure of the meeting and who will present each part of that meeting - to make it run as smoothly as possible.

The amount of time each person spent working on the project largely depended on the difficulty and volume of the work for the increment. However, on average, it was between 8-13 hours per week. Naturally, this varied somewhat between increments and between team members and some team members worked fewer hours in one week, but far more in the previous or following week - depending on the nature of their tasks and other commitments. We found that we spent a lot more time on the first two increments, as we had to implement a lot of core functionalities in these increments. Moreover, some functionality took more longer to implement than expected, for example, building the GUI took a significant amount of time in Increment 1, more than we had expected.

One of the most challenging and time-expensive parts of this entire project, was learning how to program in a group, as most of our programing related projects in university were individual. Learning how to use Version Control (Git) was quite the challenge as we all had varying levels of experience and understanding with Git and GitHub. We had some difficulties making sure everyone understand how we were managing our directory. We also had a few problems with branches throughout the project and merging incorrectly - which was time consuming to fix. Another expensive activity was creating the GUI in a way that is intuitive for the user, has all functionalities present, looks good and most importantly works correctly. We spent quite a long time in increment 1 designing storyboards for the main GUI elements, which delayed the creation of the GUI. Furthermore, we underestimated the effort required to make an intuitive, appealing design for the GUI - although, we did expect it to take a significant chunk of time. After setting up the GUI in increment 1, the GUI work was largely done - allowing us to focus on other parts of the application such as the views. In order to ensure this we performed a lot of debugging and testing, which was a large part of the process as well, but we recognised that this a vitally important part of the project to spend time on.

In order to cope with the expensive tasks and unpredicted issued we had to estimate the effort for each task.During the first increment we only had 2 effort-based categories of tasks: light and heavy. However, that proved to be misleading and we felt it didn’t provide us with an accurate representation of the amount of work we’d have to do in that increment (leading to us perhaps overloading the first two increments). Based on this and feedback during the sprint review for increment 1, we switched to the T-Shirt system for effort estimation we used for the rest of the project was the T-Shirt estimation system with 4 sizes: Small, Medium, Large and Extra Large tasks. Usually, the tasks that were of size Extra Large, we approached in pairs (Pair Programming). After the sprint was over, we checked if our estimates were close to the actual time spent in the tasks, and we established that the new system was far superior to the previous one. Using the T-Shirt system was not perfect however, and with more experience as individual programmers and team members, we could further improve our estimation of task sizes.

This helped us make better guesses of the task sizes for the next increments so we could spread the tasks more evenly across team members. The task estimation, agile techniques and the careful planning for each increment helped us organize our time reasonably efficiently. However it was the case that the workload was not always balanced between team members. This improved when we adopted the T-Shirt estimation technique but was still imperfect. Breaking down our user stories into more manage chunks of functionality may have improved this balance, as well as better effort estimation in the envisioning process.

### Tools and Communication

We used a variety of tools throughout the envisioning, design and implementation processes to make the project run smoother. These tools had varying usefulness, some had problems and we didn’t continue to use others. Below, we’ll briefly detail the tools we used, what they are, how we used them, and an evaluation of them and how we used them.

We used **IntelliJ IDEA** as our Java IDE throughout the project as we all had experience with it in the past and it being freely available to students, it was the obvious choice. The integrated support for GitHub was invaluable for us mitigating and removing conflicts in our project. We did encounter some problems with this tool, particularly with building the JAR file for delivering each increment (dependencies not being included correctly in the built file) but we managed to work around these and deliver on time. In general, we were very happy with this tool and likely to use it again.

For the coding increments of the coursework, we used **Github** as our main method of version control. Git is a popular choice for VCS and it is tool we have had experience with (although the experience of the team members varied somewhat). The use of a version control systems was invaluable in this coursework. We found branching to be exceptionally useful, allowing us to work on new features without interfering with others’ progress and maintaining a correct, working, tested application on the master branch. We did encounter some issues with branching and merging, but we resolved them with relative ease. We are certain to use Git and GitHub again in other projects.

When presenting, we often found ourselves needing to visually present a concept and idea. In example, in the first increments we had to show that our understanding and execution of the recalculation methods was correct. Also, in the first increment, when creating the recalculation methods, we had to understand all possible cases. To be on the same page with everyone else in the group, we had to visualise key concepts. In those cases we used **InVision** Freehand feature, which allowed us to create diagrams and schemes to visualise the technical points we were making. The Freehand feature proved incredibly useful and easy to use. Multiple members could collaborate in real time, which made working from a distance easier, as we could talk about design choices and draw concepts out and the export them in whatever way we want. I would definitely use it again as part of a project like this as it is an intuitive and powerful tool.

During the design process, we used **Moqups** to create the storyboards for the graphical user interface of our application. It is a cloud-based tool that allows for the creation of wireframes. It has a variety of common GUI elements that can be added, grouped, modified etc to create your desired design. The drag-and-drop nature of the tool, it is very easy to get started. The components are highly customisable and offer finer controls for getting the precise layout perfect. As it is cloud-based, sharing the design with others was simple and allowed for collaboration. Moqups is limited with the free account - which we were using - and some features were disabled. This didn’t limit our project too much but with a slightly larger project, could have limited progress.

We used **Google Docs** for creating the deliverable reports and **Google Slides** for creating presentations to guide discussion during sprint reviews. The cloud-based, collaborative nature of these tools allowed us all to work on the document simultaneously and allowed us to work on it without being in the same room. This was invaluable in us making these reports and presentations in a timely fashion, as we were all working separately on different components of these documents. This simultaneous editing did create some problems, when people had differing ideas for what should be put in one section. The ability for collaborators to add comments to a document was incredibly useful for communicating suggestions and edits without deleting others work.

**Trello** was used as a task management tool and we used to show who was working on which user stories in a given increment, and then moving user stories to testing and done appropriately. The cloud-based nature of the tool made it accessible to all members and the visual overview made it easy to see the progress made/to be made in the sprint. Although a useful tool, we found that it was not entirely necessary in our team. Between our regular scrum meetings, other coding meetings and our other communication channels, we were all kept up-to-date with what had been done, what is being worked on and what is yet to be started. As such, it was left and not updated as frequently as one may hope. As a tool, it one that although not that useful in this project, we see the benefits and will consider it for future projects.

**Slack** was a communication tool we identified as using in our envisioning deliverable. It provides instant messaging and is widely used for communication in projects. Our use of it was limited throughout the project and found ourselves favouring Facebook Messenger and face-to-face meetings for communications. This was largely due to the convenience of the Messenger mobile app and us finding it far easier to discuss complex implementation details in person.

We also used **Facebook Messenger** for communicating. It was a natural addition to the set of tools as all the team members had used it before and it’s crossplatform nature allowed everyone to see messages promptly (e.g through phone notifications). It was used a lot for scheduling scrum meetings, as well as to discuss progress on aspects of a deliverable. Whilst it couldn’t replace face-to-face meetings, it’s use for scheduling meetings was invaluable and did help us to resolve conflicts in our approach and understanding.

We feel that we made a good use of a variety of tools throughout the development process. If we were to do this project again, we would certainly use some these tools again, for example Github, IntelliJ IDEA and Facebook Messenger, but may choose alternatives to some of the tools we used. Tools such as Trello, which we identified as useful but made a poor use of, could certainly be taken into other projects and make organising the project easier.

We had 2 15-minute Scrum meetings every week (Monday and Thursday) that were used to disseminate information to all team members. These meetings proved to be invaluable to our progress as being up up-tp-date with the other team members allows you to plan in advance of how and when to do work. For the Scrum leader, these meetings were an opportunity to solve standing issues the team members are having, as solving impediments is one of the most important responsibilities of the Scrum leader.

### Advice to Next Year’s Students

This project is a large undertaking and has challenges associated with it. For many students, us as a team included, it is the first time programming in a group environment and as such, we have some useful advice for the students who’ll be undertaking this module next year.

Communication is key to working well within the team and should remain in regular contact with all team members. Having communication over the internet through Facebook Messenger, Slack, Discord, Whatsapp or some other instant messaging application is a good way to stay in contact, organise meetings and solve problems, without having to be in the same room as each other.

Whilst communication is over the internet is useful and important, meeting face-to-face with your team regularly is vital and cannot be replaced. Meeting around twice per week for Scrum meetings is a great way to keep track of progress of on that sprint (what has been done, is in progress and yet to be started). In addition to these, collaborative programming sessions together as a team can be beneficial, to help you work through issues. More pairs of eyes on your code or the problem may help you spot simple errors, and talking through the specification will help to resolve problems, ambiguities and get all members on the same page, with regards to the project.

Your supervisor is your main point of contact with problems with the during the coursework. Meeting them early in the project and regularly throughout the semester is important and maintaining a good relationship will help hugely. They are there to help you through problems and misunderstandings with the specification, help work out issues within the team and will give you feedback through the sprint reviews. We met with our supervisor at least once per week for either sprint reviews or a general meeting to show progress and ask questions about the specification and we found this incredibly useful, particularly at the start of the project, when trying to fully understand the specification.

To ensure a smooth experience whilst programming, it is important for you and all of your team members to know how to use your version control system of choice well. Team members will have varying levels of experience and knowledge of the tool, and problems will arise throughout the project. A good and uniform understanding of the VCS of choice should allow problems that arise to be solved swiftly. We recommend Github and the use of branches and always having a fully working and tested version maintained on the master branch (with experimental, incomplete work on other branches).

One of the key parts of this project is time management. One needs to manage your time as a team and individually, balancing the workload of the project with other modules and meeting the multiple deadlines. To do this, during the envisioning stage, your group will need to plan as to how to split the implementation of the functionality over the 3 coding increments. It is very important to estimate the size of each user story/piece of functionality and spread the load across the increments and team members in a reasonable, fair way. Good estimation techniques such as the T-shirt method will help with this.

Starting work on the increment early in each cycle is important to sticking to the increment plan and ensuring you have enough time to correct errors before your submission. This’ll reduce stress in the team as you approach the deadlines, spread the workload and will result in a better product for your customer. Often starting is the most difficult part but if can do anything early in the cycle, it’ll help you and your team later down the road. If you find yourself finishing all your functionality prior to delivery, you can start on the next deliverable, keeping any additions that are out of scope of your current increment plan on a separate branch in your VCS - which’ll make the next increment run smoother.

Also, you and your team members may have differing plans for when you will do your share of the work. This can cause problems if one section of work depends on others work. It is important to communicate and stick to these schedules - IE do the work by the time you say you’ll do it - to reduce stress on the teams relationship. If there are any problems with completing a section of work, it is important to also communicate that problem promptly with the rest of your team members, so you can devise a plan of action to solve it.