Final Report - Group 38

Team Members

- Alex - tll1g19
- Carlos - cab1g19
- Jamaal - jm3g19
- Stoyan - sv1u19
- Tom - tdh1g19
- Vlad - vgh1u19

Introduction

This report will reflect on our experience of working as a team, what went well and what did not go so well, along with brief descriptions about our project and interactions with our client and supervisor **Thomas Davies** (t.o.m.davies@soton.ac.uk).

Evaluation of Teamwork

Evaluate your work as a team

As a team, we got along very well from the beginning and worked with each other in a respectful and friendly way. Starting as strangers and having to get acquainted with your teammates was a new experience for many of us and getting the hang of it took a bit of time, however, after talking about each other, their skills and their way of operating we managed to find a way of making everyone work as a team on a schedule and deliver something of value to our customer for every increment and, most importantly, have a final product we are proud of.

What do you think went well?

We think our distribution of tasks and communication were two of the most important parts of this group project and we managed both very well. Having regular meetings helped keep everyone on track and informed about how the others were doing and how the project was coming along, as well as if there were any issues encountered along the way. Distributing tasks according to our abilities also was a major part of why we worked so well as a team.

What could have been better?

We believe the planning could have been better. Having a clear picture of the project from the beginning is hard as many have different views and that is why communication was important in this project. We forgot to take into account the research that needed to be done along the way when working with new technology and features such as JFreeCharts and Swing as a whole. Furthermore, as we were working through the increments new user stories were created and tasks were added that we did not account for in the beginning. For the last increment, after all the testing had been done, during the presentation with the client we found a small bug that disabled the date filtering which was easily fixable by changing an operator from “==” to “<=”. We were still able to display a previous version with working date filtering and the client was happy about the presentation and did not mind the small problem we encountered unexpectedly.

Advantages and disadvantages of adopting agile methodology

Advantages:

- Maintain a steady workflow instead of leaving it last minute.
- Communication is absolutely necessary.
- New tasks come up along the way and the view can change.
- Can deliver a project as close as possible to what the client wants.

Disadvantages:

- Collaboration and focus are difficult to maintain.
- It was more time consuming and harder to plan accordingly since documentation is not as thorough as with a normal project and also the view can change regularly.

Did you follow the XP values?

- **Simplicity**: We tried to deliver something of value for every increment, focusing on the important “must do” tasks first and then on adding extensions.
- **Communication and Respect**: Frequent communication is a big part of the XP values and we followed it accordingly, talking and meeting regularly. Respect is also necessary when working in a team and we tried to help each other as much as possible. When one of our teammates would miss a meeting we would get in touch with him and update him on what happened and what we are going to do from then on.
- **Courage:** It is very important when designing a product for a customer as you might have to redo things from scratch many times such as when we had to redo the histogram from the start because the team working on histograms misunderstood what they were supposed to show and how they were supposed to look. However, with courage, when given feedback from their teammates, they redid the whole thing before the meeting with the supervisor and it worked perfectly.
- **Respect:** Meetings can get intense, especially when the deadlines are closing in and everything needs to be finished, so maintaining a level of respect between teammates was necessary. It is easy to blame someone when something goes wrong, but that will not get anything fixed, only communication and respect will get things done, one such example being after the last code increment when we found the small bug just during the presentation. Tensions were at an all-time high, but we managed to keep a cool head and talk it out among us as well as with the supervisor.
- **Feedback**: We held the feedback from the client in high regard and tried to work according to the client’s needs, adapting and changing the product to their needs. Some of the things we changed according to the feedback from the client was displaying the “Total click cost” in numerical format instead of scientific, adding the “settings” button in the left side vertical menu instead of it being on top near the “load campaign” button, testing the product in detail to ensure it is working and making the application full-screen for a seamless, distraction-free, interaction with the application as well as a cleaner and more simple look.

Time Expenditure

Overview

We used the t-shirt method of effort estimation during our envisioning, categorising tasks from small to large. We also labelled certain tasks as extra-small or extra-large. This helped us highlight tasks that would be extremely difficult, and allowed us to plan how to tackle them in advance.

During the project as we got more accustomed to the tools we were using, we were able to more accurately estimate the time needed for most of the easier tasks. However, we sometimes ran into unpredicted issues, causing us to take longer than expected.

The most difficult tasks for us were displaying metrics as histograms, filtering metrics and charts, and adding additional chart functionalities. This was mainly due to the backend structure being fairly complicated, which we underestimated in our first increment. It was very important to us to try and make the performance of our program as high a priority as we could. We managed to achieve fairly respectable speeds by the end of the third increment but it took a couple of reworks to get there, and hence too a little longer than we hoped.

Team Balance

We all tried to plan the coding so that we would all spend a fairly equal amount of time which we almost achieved. At an estimate, we would say each person spent almost a day per week (~8 hours) working on the code or the report, with more time spent on the code at the start of the sprint and more time spent on the report at the end.

Team A, who were mostly working on the backend code, took the most amount of time due to the complexity of the data computation. However, it was very difficult to gauge the progress of this team as they were creating the systems for the other teams.

Team B, who were mostly working on the GUI, spent about the expected amount of time on their code. It was very easy to keep track of their progress as we could visually see the changes to the interface. However, a few hiccups caused delays, normally caused by very minor bugs that could mess up the visuals and were quite tricky to track down and fix correctly. This team used Swing for coding a lot of the GUI and hence had to learn a few techniques along the way.

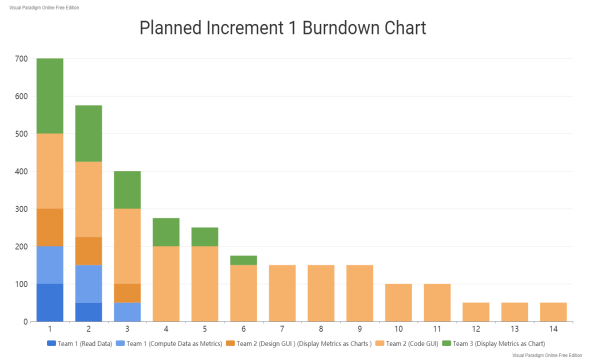
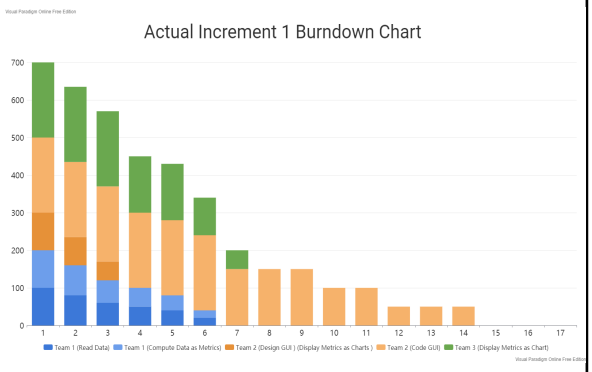
Team C, were mostly working with the additional libraries of the program like the charts and the histogram. Once the backend was done for each section they had the important job of making sure the data provided got translated correctly into the correct chart data.

Increment 1 (24/2 - 10/3)

For increment 1, we planned to finish in 14 days, with group 1 set to finish the framework for reading and computing data in the first three days. Team 2 would work on the design for the GUI and team 3 would get acquainted with JFreeChart during that time. After the framework

was finished, team 2 could implement the GUI designed, while team 3 displays the charts using given metrics.

In actuality, setting up a controller for the metrics took longer than expected, and team 1 had to take 6 days to finish, however, this was not a problem as we had taken into account things not going exactly as planned and left some days to spare. Team 2 and 3 did not have any issues. We finished with time to spare since the deadline was extended. We could have organised time more efficiently by spreading the workload more efficiently. We expected reading and computing the data to be more difficult than it was in actuality while coding the GUI was more time consuming than expected. After team 1 finished in the first 6 days, they didn't have any work for the rest of the increment, so they could have either split the task with team 2 or had a head start for increment 2.

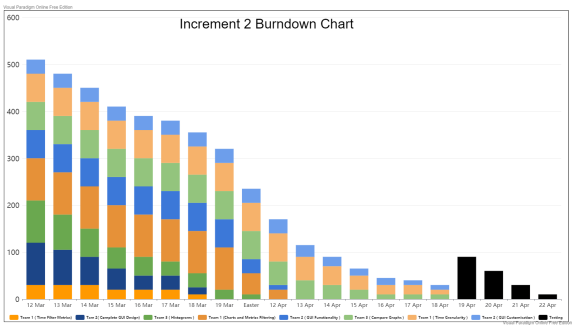
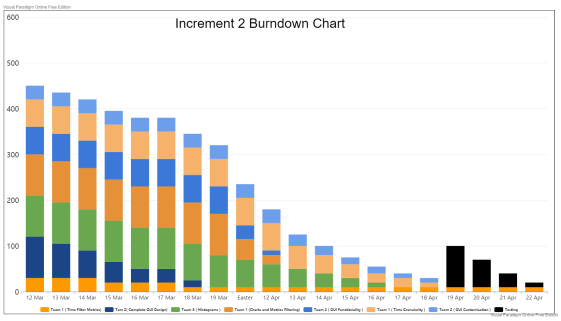
<table>
<tr><th colspan="2">Increment 1</th></tr>
<tr><td>Planned burndown chart</td><td>Actual burndown chart</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Left-to-Right (Team 1 - Blues, Team 2 - Oranges, Team 3 - Green)
Team 1 - Read Data
Team 1 - Compute Data as Metrics
Team 2 - Display Metric as Charts
Team 2 - Code GUI
Team 3 - Deploy Metrics as Chart</td></tr>
</table>

Increment 2 (10/3 - 21/4)

Since we had more than a month to finish increment 2, we planned to finish our most difficult tasks during this increment, including displaying histograms and filtering metrics and charts. We also included some time for planning at the end of the increment.

We planned to split our workload before and after the break. Before the break, team 1 would work on giving users control to filter metrics and charts. Team 2 would work on GUI customisation, and team 3 would work on displaying metrics over time. After the break, team 1 would work on implementing time granularity. Team 3 would use the new filters to display metrics as histograms.

We ended up not communicating much during the Easter break due to clashing schedules. This didn't affect our productivity much, but if we had stayed in better contact, we could have given more time for increment 3.

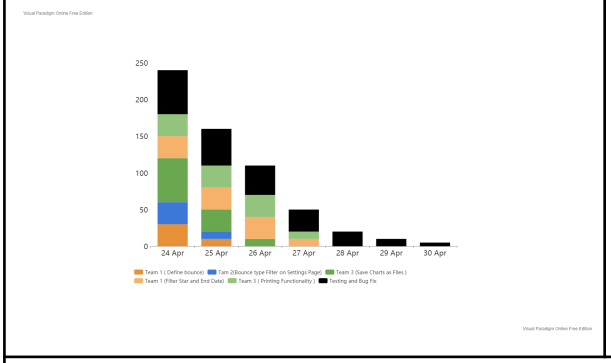
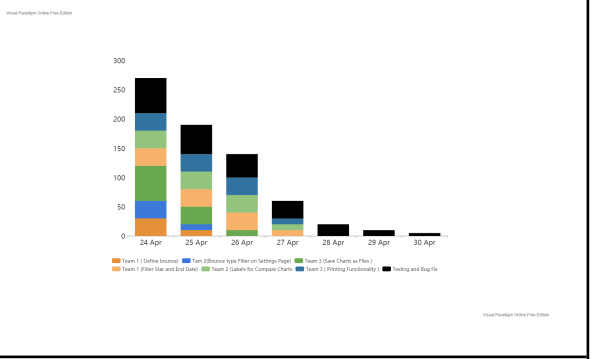
<table>
<tr><th colspan="2">Increment 2</th></tr>
<tr><td>Planned burndown chart</td><td>Actual burndown chart</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Left-to-Right (Team 1 - Oranges, Team 2 - Blues, Team 3 - Greens, All - Black)
Team 1 - Time Filter Metrics
Team 2 - Complete GUI Design
Team 3 - Histograms
Team 1 - Charts and Metric Filtering
Team 2 - GUI Functionality
Team 3 - Compare Graphs (removed in actual)
Team 1 - Time Granularity
Team 2 - GUI Customisation
All - Testing</td></tr>
</table>

Increment 3 (21/4 - 5/5)

For increment 3, we planned to finish some final minor tasks such as defining a bounce, as well as tasks that weren't required. At the end of the increment, we would work on testing. Team 1 and 2 would work on letting users define a bounce, while team 3 worked on saving and printing functionality.

We ran into some trouble during increment 3. During the envisioning, we expected to have around two weeks, but in actuality due to changes in deadlines, we had only one week. This caused us to run out of time for testing.

During our original envisioning, we planned tasks assuming we would have less time for increment 2 and more for increment 3. After we updated our schedule, we still worked according to the original plan without change. This meant we had much more leftover time for increment 2, and ran out of time for increment 3.

Increment 3	
Planned burndown chart	Actual burndown chart
	
Left-to-Right (Team 1 - Oranges, Team 2 - Blues, Team 3 - Greens, All - Black) Team 1 - Define Bounce Team 2 - Bounce Type Filter Team 3 - Save Charts as Files Team 1 - Filter Start Date and End Date Team 3 - Printing Functionality All - Testing and Bug Fixes	

Tools and Communication

Communication

Our main method of communication throughout our entire project was a WhatsApp group chat. This came in very handy to have informal discussions and easily notify members of activity. Whilst it was a useful method of communication, it was very difficult to share our work so we needed an alternative platform. We set up a group discord server which was our main platform for when we were working, allowing us to do group calls where we could share our screen and gave us a place to share files and links for the work. This channel became quite messy with work, so we also set up a Microsoft teams space where we would host the calls with our supervisor for marking meetings and informal chats. Overall, each platform provided a different benefit but the discord channel had to be the most beneficial as it provided way more utility than we perhaps could have taken advantage of.

We met fairly regularly using our Discord channel to work on the code and the report. At the beginning of the coursework, we met most frequently, doing scrums every two days. The frequency of scrums decreased after Easter but we spent longer during them to discuss our work. Our working meetings were most beneficial, although less frequent, and allowed us to get through a large portion of each sprint. These meetings were always online as we were unable to meet in person due to travel restrictions.

During the last few days of a sprint, when we were working on the report, we would spend a large amount of time setting out our initial plans for the next increment. The days following the weekend after a sprint we always met with our supervisor for a marking meeting, at

which point we convened as a team afterwards to discuss and start work on the next sprint plan.

Agile Collaboration

Our main methods of collaborative working were to use GitHub as our code repository. We assigned a branch to each member of the team so that they could work on sections of code separate from others, and then merge those changes into the main branch once done. This was by far the most beneficial tool we used throughout our project as we took advantage of the commit history and branches quite a lot.

For the documents, we used Google docs and Google slides. These Google services came in very handy as we were able to all work on the same report at the same time and see what each person was working on in real-time. We designed the envisioning, increment 1 and the final report in Docs and used slides for increment 2 and 3 as we believed we could improve on our presentation using it for the last stages.

In addition to this, we used Jira, an agile development website that allowed us to very easily plan our sprints, their backlogs and how we shared the work. This came in very useful when we started using it for the end of increment 1, increment 2 and increment 3. Jira also provides a way to integrate code from GitHub which we could have taken advantage of.

Software Development

Our main method of software development was to use the IntelliJ IDE. We were all familiar with Java and most of us were also familiar with this IDE so it was beneficial to all use a similar platform to work on. It also allowed us to add any libraries we used much easier, which included JFreeChart and JDatePicker. IntelliJ also provides tools like IntelliSense and easy GitHub integration which worked well with our collaborative tools.We also made use of software like Visual Paradigm and Figma to design our UML class diagrams, burndown charts and storyboards respectively.

Advice

- Make sure as soon as you're allocated your groups to establish a method of communication. The faster you can get together, the faster work can be done.
- Have regular meetings so you can keep track of how much work is being done.
- We recommend having a Discord group for meetings as well as a Whatsapp group for more informal discussions.
- Allow some time for research as it may be necessary to get accustomed to working with new technology and never leave allocated work till the last minute.
- Use a version control system such as GitHub to keep track and update code (learn how to use it before writing your code).
- Using an agile project management tool such as Jira can also be helpful in planning and keeping track of progress and tasks for the whole team.
- Testing is one the most important parts of developing a working and useful program so regular testing is necessary so no issues arrive during the presentation to the client.