Change Report

Team 7 - Broken Designers

Adam Brown
Morgan Francis
Shabari Jagadeeswaran
Oliver Johnstone
Laura Mata Le Bot
Rebecca Stone

Introduction

After inheriting the project from the other team, we held a group meeting to decide on a plan and split the work between team members. We approached this task by listening in turn to each team member's preferred tasks for assessment 2. We split the change report between 5 of us, each choosing a different section to work on. We ensured that the section each member was working on was different to that of assessment 1, with the member responsible for that section in the previous assessment acting as their shadow. This was agreed upon during our Sailboat retrospective, as mentioned in the method selection and planning report for this assessment. The role of a shadow was to overlook a section of the report, helping out where needed and making sure work was getting done in a timely manner.

As we had already looked through our chosen group's deliverables when choosing a project to take over, we had a good idea of what needed to be edited and updated when taking over their project. We updated the requirements at the beginning of the project, amending it according to the new assessment brief. We also considered the elements of the other team's risk register, adding and deleting where necessary for our position at the start of the project. We constructed a new architecture, based upon the previous team's diagram, but including the new requirements and allowing for testing to take place.

One of the most helpful tools on the documentation side of the project was the feedback given for the previous assessment. We studied our feedback and considered that of the other team, organising the key points into a feedback analysis table. This considered the positive aspects of our work and any improvements that could be made. Summarising and comparing both our feedback and that of the other team in a conclusions column allowed us to improve upon their deliverables effectively. It also gave us a good idea of what needed to be done for each deliverable before we started work on them.

Generally, when working on the deliverables, we began by editing the other team's submitted deliverables and keeping a note of any changes we made in a change report. This focussed mainly on making it more readable and conveying the points we wished to make effectively. We continually added to the change report throughout the project if any problems arose that required us to change it. Next, we focussed on any additions that we wanted to make to the deliverable as a result of a difference in the two teams' approaches to the project, especially affecting method selection and planning and the risk register. New requirements provided in assessment two had to be added to the requirements and architecture deliverables.

In terms of implementation, our main focus for the first couple of weeks was refactoring the inherited code and setting up the testing environment to allow us to begin both testing and implementation synchronously using test-first development. We found it useful to have an in person meeting with limited members of the team who were tasked with the testing or implementation aspects of the project. This ensured testers all knew how the environment worked and were on the same page when they later began writing tests. Those working on refactoring the code could split the task between themselves while not using any of the others' time as a full team meeting would have.

Requirements
Original: <link>

The requirements report we took on was very similar to the one we had made for our previous project. However, we decided as a group that there were changes we had to make in order to improve its quality as well as to make it fit the new requirements we have been given.

The first thing we changed was the formatting of the previous teams requirements table. The other team never mentioned using a formatting style, and the table didn't seem to follow one either. From previous research, we decided that refactoring the previous teams table to follow the IEEE Standard Requirements Engineering document [1] would give us the best result. This document suggests creating generalised and brief user requirements, which are then referred to by more detailed functional and non-functional requirements. Our decision to change the requirements table in this table in this manner was driven by research into requirements documentation. IEEE's [1] methodology allows for easy to read and understand formatting. Additionally, it suggests grouping requirements together which helps when delegating tasks amongst team members.

After deciding to follow the IEEE Standard Requirements Engineering document [1], we refactored the requirements table around it. This meant removing, replacing and adding to the existing table. Examples of this include the refactoring of the UR_DISH_SERVE, UR_WIN and many more user requirements into functional requirements. Under the new formatting style, many of the previous teams requirements were too specific to be user requirements. This was fixed by making them functional requirements and introducing newer and broader user requirements that they could link to.

In addition to formatting changes, we also added to the requirements table in order to include additions that covered the new design requirements. New user requirements such as UR_SAVE and UR_MODE were added to broadly cover the new save feature and difficulty feature that was requested by the customer. These requirements were further elaborated by new functional requirements, such as FR_POWER_UPS, FR_REPUTATION_POINTS, FR_EARN_MONEY and FR_SPEND_MONEY. We added to the table with requirements like these in order to cover all the parts of the customer's brief, including the new additions added for the second half of the project.

Once we had changed and added to the table, we added to the explanation report made by the other team in order to explain the new formatting. The other team's report was largely unchanged in its idea, but was rewritten to include full sentences as well as proper paragraph formatting. Additionally, it was modified to include new paragraphs to do with explaining the re-formatted requirements table, as well as talking about our second client interview. These changes were done in order to make the requirements deliverable appear more professional and to explain our contributions to the previous team's project.

Architecture
Original: <link>

Method Selection and Planning

Original: <link>

The method selection and planning report we inherited was largely unchanged from the previous team. The original content of the report was kept, but was rephrased in order to make the concepts more legible.

In addition to this, we added parts to the report about our deliberation techniques used when deciding on which team to pick for assessment two. Since this was a major factor in our project, we believed it was necessary to include justifications about our decision to move on with “UnderCooked” as a project.

After rephrasing their report, we updated it to include all of the methods we had used that the previous team had not. These included things like the use of Trello, the sailboat technique and Git. These additions were made to keep the report as up to date as possible, since the methods we utilised were different to the team before us.

Finally, the report was missing information on the tools used to aid in the creation of architecture diagrams. We added a section in the report discussing our chosen tool, plantUML, and why this was more suitable for our project than the alternatives.

Risk assessment and mitigation

Original: <link>

The risk register has been updated to include risks associated with our team's experience during development. As well as this we have included further product related risks revolving around the user/customer's experience. Similar or duplicated risks (For example R4 and R5 of the original report), have been condensed into a singular risk or removed. After adding these risks, I have then tracked all our new risks, as well as all the prior risks. All prior risks had a secondary owner from our team assigned to the risk. In addition, any irrelevant or incorrect mitigations have been removed.

Website

After taking over their project, we quickly found that the source code for the website was very inaccessible and difficult to build off. Additionally, personal preference toward a website with multiple distinct pages drove us to rewrite the website in our own image. The website we took over had source code that had been formatted in an extremely unfriendly way. Entire CSS files had been condensed to one line of thousands of characters long and some html and JS files had very clear signs of being generated from a third-party website builder.

Bibliography

[1] ISO/IEC/IEEE 29148-2018 International Standard - Systems and software engineering – Life cycle processes – Requirements Engineering, IEEE Standards Association, 2018.