Changes Report
Group 30 Triple 10
Team Members:
Kelvin Chen, Amy Cross, Amber Gange, Robin Graham, Riko Puusepp, Labib Zabaneh

A)

For this project, we used Tools such as google docs and VSCode for the main parts of the deliverables. To keep our records of changes, we also used the google docs version history. We recorded changes to the code through the GitHub commits.

As a group, we also followed specific processes of first comparing our documents for assessment 1 to the documents we have to develop in this assessment. This allowed us to combine ideas to create thorough and well-rounded deliverables. We also extended the deliverables by reviewing the Assessment 1 criteria and using the feedback. Then, we used the Assessment 2 brief to identify new criteria and write the relevant information into each deliverable. Finally, to ensure that we had covered all the criteria needed, there would be a peer assessment of each deliverable to ensure everyone was satisfied.

Conventions that we found for change management followed a similar format which we simplified into a 5-step plan: prepare, plan changes, Implement changes, familiarise with changes, then review and analyse. This follows how we ordered the processes where in the step ‘prepare’ - collected all the documents we needed, and ‘Plan Changes’ - identified what needed to be changed and how. This was then followed by the main stage of ‘implementation’, where all these changes were added, whether on the documentation or the code. ‘Familiarise with changes’ is where we ensured that each team member was aware of the changes and how they affect different areas of the project. The final stage of ‘review and analyse’ was where the part of the project was tested or reviewed by other team members for verification.

B)

Requirements

For the requirements of this project, we made some slight changes to some of the defined requirements from Assessment 1. For example, small things like adding pizzas and jacket potatoes to UR_COOK_FOOD, changing two cooks to three cooks and allowing a user to burn food referenced in UR_FAILING_STEPS and FR_NOT_OVERCOOKING renamed to FR_OVERCOOKING. These changes have been made to comply with Assessment 2 criteria when progressing from Assessment 1, as some requirements would contradict without these changes.

We also added some requirements from the Assessment 2 brief so we could carry on implementing the project. This is where most of the changes were made due to many new criteria. Changes included more general user requirements like UR_Upgrades (Allow upgrades throughout gameplay). This forms the basis of the more detailed functional requirements of FR_KITCHEN_UPGRADES, FR_POWERUPS and FR_STAFF_UPGRADES, which are directly referenced in the criteria of the game or the additional criteria the stakeholders defined. For example, ‘you will be able to invest some of your earnings to enable the other cooking stations’ for the requirement FR_KITCHEN_UPGRADES and ‘call more kitchen staff back from leave’ to create the requirement FR_STAFF_UPGRADES.

For non-functional requirements, we removed NFR_AVAILABILITY (The system shall be highly available) as this is in reference to presenting the project at the university open days. We decided that this is too general as it is not clear who ‘highly available’ relates to. So to make the requirements more precise, we removed them and replaced them with more specific requirements.

These requirements are NFR_GRAPHICS (the game is clear and easy to understand), NFR_OS (can be played on multiple operating systems) and NFR_SCENARIO_TIME (The scenario should have a fast-paced rhythm). Our group initially defined these requirements in Assessment 1 that we didn’t feel were represented in the documentation we took over. We particularly felt that NFR_GRAPHICS and NFR_SCENARIO_TIME needed to be included in our requirements as the game is designed to be an easy-to-pick-up game that doesn’t require any particular skills or prior experience but is still enjoyable to play.

NFR_OS was implemented to replace a previous user requirement that we have removed UR_COMPATIBILITY. We decided on this change because there were no NFR or FR requirements built off this user requirement. We felt like it had been defined as the wrong type, and therefore it was replaced by a non-functional requirement as an expansion of a slightly reworded UR_ACCESSIBILITY. Similarly, we also replaced UR_SOUND and UR_GRAPHICS with FR_SOUND and NFR_GRAPHICS, respectively. Any previously dependent requirements were relocated to UR_SETTINGS.

A few functional requirements were also removed as they were not specific aims we would work towards when implementing our project. These are FR_GUIDE_USER (The system shall subtly guide the user to complete tasks), FR_COLOR_BLINDNESS (allows the user to choose a suitable colour palette) and FR_LOADING_SCREEN (displays the logo when the system is loading). Firstly we removed FR_GUIDE_USER as they will be provided with instructions on how to create a dish (UR_INSTRUCTIONS) but are not planning to implement any other type of guide for this. Access for users with colour blindness is also not an aim that we have in the confines of this project therefore, we decided to remove it. However, we do have the requirement of UR_ACCESSIBILITY that we will consider when creating our project to make our project as accessible to all users as we can. The final requirement is to have the logo displayed on the loading screen. Our group determined that this might be something that we would want to do in the future but is not a requirement in our project.

A small paragraph was also added to the explanation to explain how some of our new requirements were elicited and negotiated. This referenced the creation of the requirements UR_REP_POINTS and FR_POWERUPS with the parts of the criteria that these were formed from and how this was done to fulfil all the stakeholders' requirements.

Risks

Our group decided that the best course of action for the Risk deliverable would be to continue with the risks we had defined throughout our assessment 1 Risk deliverable rather than using the ones we've inherited. This is because responsibilities from the identified risks had already been assigned in Assessment 1, and we were comfortable with them, so we felt that a complete switch up of these responsibilities would be a bad idea for managing risks in our project.

However, taking advice from our feedback and reviewing the risks we have inherited, we have included a status to record any occurrences of each risk. This helps us to recognise any problems during the project and ensure that they are resolved. It also helps us to track problems that occurred and helps avoid repeating similar mistakes further in the project.

Despite choosing to carry most of our risks on through the project, we did decide to expand on our risks based on the inherited risks. This means we have added the risks of ID R18 to R21. We felt these risks hadn't been included in the last hand-in for our deliverables.

For example, we had initially overlooked two risks concerning our team members, R18 and R20. R18 being 'A team member leaves the project' and R20 'A member becomes unavailable for an extended period of time'. It did not occur to us that a critical risk to plan for was the breakdown of our team. Instead, we were focused on mistakes and issues with technology and actions that we had taken and not looking at the broader picture of external influences on our project.

This could also apply to the added risk R21 'A library being used becomes unavailable or deprecated in the middle of the project'. This is not in reference to the team but is still related to our previous error in seeing the bigger picture outside of our project.

Also, we didn't carry on the scoring system for the inherited risks defined by the other group. Therefore we removed the methodology of this where the previous team had assigned the severity of the risk through a system of averaging secret votes between team members. Our team agreed that doing this was unnecessary as we had already reached a consensus together, so we did not need this process to assign the risk's values.

Apart from this difference, we followed a similar methodology to define our risks across the project. We constructed a list of all the possible risks that we could think of and then reviewed this to create a comprehensive but useful list removing those we felt were unnecessary, like the previous group. By discussing these as a group, we could identify who in the team was most suited to manage this risk. Whether that is due to the part of the project, they were working on or in cases where multiple people were working with a related element, who knew the most about a possible solution to this problem.

The previous team had also referenced many sources in how they constructed these risks. These were relevant as we found similar information when completing our table of risks.

Unfortunately, we could not tell where the previous team had gathered their information from. So in these parts, we have included references to where we found this information.

This also meant that we had to remove a section of their writing to do with researching the areas where you should focus your risk assessment. For the inherited risks, there was no reference to the actual website they retrieved this information from, and we were unable to find a website that detailed this thoroughly.

Method

In the method deliverable part A, we made only some minor changes to the writing. Firstly, we added a small section on why we chose IntelliJ over Eclipse, as mentioned in the previous teams' feedback. Therefore it was essential to include this in the update.

The previous team did not explicitly state a methodology they followed but defined it as an iterative model. To continue this project, we followed an Iterative and Incremental Model that is made of smaller waterfall models. We chose this model as in the last assessment, we followed a waterfall model. It provided the benefits of completing requirements early, allowing us to define the entire project scope, which we found to be very helpful due to the strict timeline of this project. A common complaint of a waterfall methodology is fixed stages, but this was not a problem as the requirements were not to change throughout the project.

We chose an iterative version of the waterfall model as this part of the project does not strictly follow this methodology due to us carrying on an existing project and developing it. However, we felt like the project still followed a waterfall model of defining requirements and risks at the start of the project and carrying them through the development and moving on to the testing stage of the project after implementation. We did this because, as a group, we were uncertain about testing, what we needed to test and how, so felt more comfortable implementing the project first so that we could work together to tackle testing.

We only made these small changes to Part A due to a large correlation between the tools used by the other team and us. This is likely due to there not being many popular free-to-use tools that are also appropriate for the development of a project such as this. This is how we have come to use the same tools for very similar reasons. Therefore, the explanations and reasonings for choosing the tools did not need to be changed when we took over this project.

In section B we decided to remove the *individual strengths* of the previous team as their strengths were no longer relevant to the continuation of the project. However, we thought the roles were essential to show in the planning of the project, so we are keeping a record of the original team roles, and then we have defined our new team roles for the continuation of the project.

Taking on this project for Assessment 2, we created a new breakdown of all the defined game criteria and deliverables (shown in plan2.pdf). This allowed us an overview of the whole project so that it was simpler to see the scale of the project and split it into individual tasks. To break down the game criteria, we first defined the main objectives like game modes, power-ups and difficulties. We then described in more detail the individual parts of these. For example, in ‘game modes’, there are two modes defined as ‘endless mode’ and ‘scenario mode’, with further details if required.

In part C, we decided not to remove any of the details as they were necessary for showing the history of the project and told us what had been produced and how. For this reason, we did not condense the information (for example, removing timings and dates).

However, we decided to add our own strategies for keeping track of the timings in the project and what these timings were. Some of the tools that we used were to create a logbook of each meeting and Gantt charts of the whole project.

We created the logbook to keep track of what each team member was working on each week. It also allowed us to check that we were all making progress and spending only the correct amount of time on each part of the project. As a team, this was very useful as we could check it against our Gantt chart to ensure we were keeping up with our plan.

The Gantt charts themselves were also obviously helpful for this reason during the project's development. It also allowed us to fully form our plan at the start of the assessment so that we knew the amount of work needed to complete the project to our standards. For this reason, we decided that we would need to complete some work over the Easter holidays to finish the project.

Architecture

For Figure 1, we chose to redesign the class diagram as when we added the new classes, it made the diagram very complicated and difficult to read. Redesigning the diagram allowed us to sort the classes into packages helping our code to be more readable as we followed them through into the code.

An example of this is the package of stations which closely resembles the original Figure 3. The resemblance is particularly apparent in the relationship between a parent class of stations which each type of station inherits methods from, the same as shown in Figure 3. The package also reflects Figure 4 as it shows the relationships between stations and a station controller and the game screen. It also does this without the disadvantage of separating the figures and giving the impression that they are unrelated.

We decided not to continue with the more detailed class diagrams shown in Figures 2,3 and 4. Organising our complete class diagram into packages allowed us to include all the

methods in this main class diagram. These packages reflect the groups that Figure 2,3 and 4 were designed to show without separating them from the rest of the classes.

Only small changes were made to some of the state diagrams shown in Figures 5,6 and 7. As our team did not make any major changes to the base functionality of the game. For example, in Figure 5, ‘A state diagram for controlling the chefs’ how the chefs are controlled and selected has stayed the same. However, this diagram had to be updated to include the requirement for a 3rd chef. We found that the easiest way to do this was to generalise the diagram just to choose a chef rather than a specific chef number to make this diagram easy to understand.

Both Figures 6 and 7 did not require any changes as they detail the ‘overall control of the game’ and ‘navigating through the screens’. Our team did not make any changes to these diagrams, despite the addition of game mode and levels, as the user still fulfils customer orders by interacting with ingredients and cooking stations, meaning there are no changes to be made for Figure 6. Figure 7 is similar as there are individual changes to each screen in the game to add in modes, difficulty and recipes etc. However, there is no impact on how these screens interact with each other, so no impact on the state diagram of Figure 7.

The two sequence diagrams shown in Figures 8 and 9 did not require changing. Figure 8 details how a user would create a burger as an example order. This diagram does show 2 chefs, not 3, but we felt that this was not relevant as it still represents the steps that a user would take to create a burger. Figure 9 was similar because the interaction between the chef and the station collider has not changed throughout the project, so the diagram also doesn’t need any changes.

We decided to keep all the inherited diagrams on the document as many of them contain details that are still important to the project, along with preserving a record of previous iterations of the design. This especially relates to the zoomed-in versions of the class diagrams. So any diagrams that have been altered have been clearly displayed under a new title of Assessment 2. As we kept these diagrams, their explanations have also been carried through into our documentation.