Change Report

Group 21
Generic Games

Josh Thomas
Andrew Palombo
Oscar Gunn
Scarlet Desorgher
Immanuel Ghaly
Madeleine Nielsen

Change Report

We first copied all of the artefacts. We placed them under version control by creating a new shared drive and git repository. We used version control to track the changes made to each file, and it also allowed us to revert to a previous version if needed. We also set up CI, which automated the integration of code changes and made development more efficient.

We then chose to identify the changes that needed to be made to the requirements first, as it ensured that all changes to the deliverables could be traced back to the requirements. We planned our changes by creating a to-do list of every needed change. This list was held in the shared drive, and during our meetings, each member would choose a task to complete. Any changes we made to the documents were logged in a shared file on google drive, which specified the files, the date, and a brief description of the change. This allowed us to keep track of the changes made and allowed for a more organised approach.

Before we made any changes to a section of the code, we implemented automated tests. This ensured that any changes wouldn't inadvertently break a section of the code, which may have gone unnoticed until the later stages of the project. When necessary, we would create a change request, including a description of the change, files or components that could be affected, the risks, and who requested the change. The implementation subteam would review the requests together and would implement the changes. We used a spreadsheet to manage and track the change requests instead of an alternate platform, such as GitHub issues, as it was simple and familiar [1].

Once a change had been made, we marked it as complete on the to-do list and during our weekly meetings, we reviewed the to-do list and the changed deliverables.

Requirement Changes [System Req.] [User Req.]

The previous group stored their requirements in 4 tables [link]; user, functional, non-functional, and constraint requirements. We removed the constraint requirements table as it wasn't necessary, and most of the constraint requirements identified should have been non-functional instead.

Group 13 also included a risks and assumptions column in each of the tables, which allowed us to add more information about what to consider when implementing the different features of the system. We decided to remove this from the non-functional requirements table as it did not apply because non-functional requirements specify the general qualities of the system. The previous group only identified four non-functional requirements. We added more non-functional requirements specified by the customer, each with objective fit criteria. We did this to ensure that the qualities specified by the customer influenced the design and implementation. Furthermore, we could prove that we met the customer's needs by including objective fit criteria.

Due to the increased complexity of the second assessment, we added more detail to the description of the requirements. This was done after the design process to implement the system changes directly from the requirements. This prevented us from making unnecessary changes and ensured we only implemented what the customer requested.

Method Selection and Planning Changes

The last development team used SCRUM and an Agile method for their software engineering [Link]. We are also using this approach as it is suitable for the project, and our team is used to this method. Like the other team, we had weekly meetings. However, we held and planned these differently. As we did for assessment one, we split the project's main tasks into equal parts and then assigned them to the most appropriate team member. On the shared drive, we had a document with regularly updated to-do lists where each team member would claim their associated tasks.

We decided not to adopt the Trello that Team 13 used as we were unfamiliar with the tool and wanted to avoid assigning the project or product owner role to any member. We carried on with Discord as our primary form of communication and where we would update other team members on any progress. This was enough for efficient task management and communication at the start of the project. However, communication was dropping nearer the end as we were working remotely; luckily, this did not significantly affect the project's progression. Looking back, Trello, or similar tools, might have been beneficial. We also could have used Discord's remote meeting functions more to allow us to video call over the easter holidays.

For the first assessment's architecture diagrams, we used LucidChart, we found this tool helpful,I but we changed to PlantUML. This was because we needed to ensure our updated diagrams were in the same style as the initial ones. We also found that even though we had to learn the syntax for UML, it was a lot easier than expected, and the diagrams changed layout as new parts were added. This was a lot less hassle than modifying diagrams on LucidChart. Our only problem with PlantUML was the auto-cropping of the diagrams. As a few diagrams were large, we split them into two to remain clear.

We also kept a document with change requests logged, this allowed us to view what had been changed and by who. It also allowed us to backtrack if a change didn't work properly. We only kept this method for changing inherited code because backwards compatibility wasn't as important for any new code added. All the added code was logged through our Github repository which was a lot less overhead.

Change Request [1]

Change Requester	Change ID	Date	Requested Change	Files or Components Impacted	Potential Risks	Implemented Y/N	New Version
Immanuel	1A	10/02/2023	Delete some redundant files	OldKitchen.tmx, Kitchen.tsx	The files might have been used in an area I didn't check	Y	1.0.1
Oscar	2A	17/02/2023	Create texture atlas for chefs	assets/Chef	May remove textures if referenced incorrectly	N	1.0.1
Immanuel	3A	18/02/2023	Change movement system, to make testing possible	Chef.java, PlayScreen.java	Movement may act differently from before the change	Y	1.0.2

Oscar	4A	20/02/2023	Reorganise chef assets	assets/Chef, Chef.java	Chef assets could be removed if implemented incorrectly	Y	1.0.3
Oscar	5A	03/03/2023	Remove old asset files	assets/Chef, assets/Chef New	None, previous change was implemented correctly	Y	1.0.4
Oscar	6A	09/03/2023	Add new assets for ingredients	assets/Chef	Incorrect assets could be displayed if code is not adjusted accordingly	Y	1.0.5
Oscar	7A	23/03/2023	Reorganise ingredient assets	assets/Food, src/Ingredients	Ingredients will not be displayed if code is not adjusted to accommodate the change	Y	1.0.6

Architecture Changes

Original	Change	Justification
Original team made a component diagram [Link] for the system.	We did not develop this any further	Not required for this system as it is small
State diagram [Link] included option for customer to accept wrong order	Customers can no longer accept the wrong order and instead just waits for their order.	Allows continuous gameplay and means that the player has to make more decisions, leading to a more fulfilling experience.
One state diagram was made for scenario mode	We made two state diagrams, one for scenario [link] and one for endless mode [link]	Maintain clarity between the two modes. As the game ends at different points.
No state diagram with: reputation points, ailing prep stages, multiple orders coming at once, user profiles	Reputation points have been added to the end of the state diagram when the customer gets their order. Ability to fail preparation stages and for multiple orders added to the endless mode state diagram. We added the user profile to the start of the state diagrams	Fulfil new requirements like saving games and difficulty levels.
In the previous class diagram 'InteractiveTileObject' [Link] was not marked as an abstract class	'InteractiveTileObject' is now marked as an abstract class	Correct syntax for UML and correct for the code
We found that the sequence [Link] diagram was thought of in the wrong way by the previous team.	We changed the language used in the diagrams and adjusted the point of view of the interaction flow.	In the use case diagram, actors should only represent real-world interactors rather than game concepts like the "customer".
No sequence diagram with user profile and game-saving as it wasn't a requirement	User can select to start a new game or old game in sequence diagram [Link]	Shows that the user can save the progress in an old game and restart it by having a profile
LibGdx sprite class [Link] was shown in class diagrams	We have chosen not to show the sprite class in our class diagrams	The class 'Sprite' is already in the Libgdx library so it didn't make sense to include it in our class diagrams

One use case diagram	We have made two use case diagrams, one for the gameplay [Link] and one for the profiles and pause functionalities [Link]	This helped us understand the systems better and how it would work.
Classes did not exist as requirements did not exist	'DifficultySelectionScreen', 'OptionScreen' [Link] and new ingredient and recipe classes added	User now needs to be able to pick between different game modes and levels
Attributes and methods did not exist	New attributes and methods added to classes [Link]	Provide needed functionalities and fulfil new requirements: shop, pause, save, scores, powerups, fail prep, new ingredients and recipes
Order and recipe did not have relationship	In our class diagram they have a relationship [Link]	Orders have a recipe related to them.
Class diagram with multiplicities and relationships [Link]	Different format without attributes and methods. Added new functionality classes and changed the relationship statuses. [Link]	Attributes and methods were redundant in this diagram, and some classes did not exist. New requirements needed new classes. Old relationships were unclear between classes.
Change ID: 3A	See table [1]	Make testing possible by removing the movement system from the graphical interface. Can now simulate pressing movement buttons
Change ID: 2A	See table [1]	Simplify code for easier testing and reduce file size as all assets are now one image
Change ID: 7A	See table [1]	Simplify code for easier testing and reduce file size as all assets are now one image
Cook collision was set to 0.3 seconds and -1 velocity vector	Cook collision is now set to 0.2 seconds and -0.25 velocity vector	Could not remove all the cook collision function as it kept crashing the game. So Instead we toned it down to make it less intrusive on the gameplay and less of a distraction to the user.

handleInput() in Playscreen is a very long method	Method is now a lot shorter by being split into multiple methods	Readability and maintainability

Risk Assessment and Mitigation Changes [Risk Register]

The previous group used a risk register to store their risks and their risk ratings [link]. We did not change the format of the risk register as it contained the necessary information, such as ID, type, description, likelihood, severity, mitigation and owner, and we only wanted to make changes that were necessary so that we could prioritise our time with the most critical aspects of the project. The risk register also contained categories such as technology, people, requirements and estimation, which helped organise the risks by grouping similar risks. We kept this the same as it allowed us to identify the most vulnerable sections of the project and apply our mitigation strategies accordingly. For example, in our risk register, the technology category had the most and most high severity risks, so we had more people monitoring those risks.

We found that most of the risks identified and their mitigation strategies were appropriate and effective. However, we removed a few risks, such as R6, R11 and R14, as they were incorrect and unnecessary. We also changed the likelihood and severity ratings. These changes were made for various reasons, such as the increased scope of Assessment 2 and the complexity of the required features, which led to higher likelihoods and severity ratings. New risks identified throughout the project were added to the risk register with new owners. The original group loosely prioritised the risks by their severity. We prioritised the risks by severity and likelihood, making identifying the risks that required careful monitoring easier.