UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

ENG1 Assessment 2
Group 18 - Octodecimal

Change Report

Group Members:

Izz Abd Aziz

Phil Suwanpimolkul

Tom Loomes

Owen Kilpatrick

Zachary Vickers

Michael Ballantyne

Summary of how changes were made

The first thing we did as a team when we inherited the previous deliverables was decide on a strategy to track the changes we made to those deliverables and how these changes were progressing towards completion. Our main strategy was to make use of a Kanban board, split into sections for each of the deliverables that needed altering. Within these sections, we set out sub-tasks (specifics for what needed changing, for example a section or diagram within a deliverable), added a brief description of what needed to be done and assigned an owner to complete the task, along with a progress tag: “to do”, “in progress”, “needs reviewing” and “done”. We also assigned an additional team member to each task to review the work that had been completed, check that it met the initial brief and covered the necessary changes in enough detail.

Finally, we had weekly in-person scrums where we briefly reviewed changes as a team, removed completed tasks from the board, discussed and added any new tasks we had found in the past week, and task owners provided progress updates to the rest of the team to let them know how close their tasks were to completion. During these scrums, we also reassigned tasks to different owners if necessary and updated timescales for the completion of subtasks and the entire deliverable itself. As for the specific changes within the documents themselves, before the changes had been reviewed and signed off by the team, additions to the documents were written in **bold** and anything that was set to be removed would be ~~crossed out~~; this made it quicker and easier for the reviewer, and subsequently the team, to see and approve the proposed changes.

For each of the deliverables, we iteratively adapted them throughout the project to meet the new brief and our team’s personal methods. Details on how we determined what changes to make for each are described on the following pages of this document, as each deliverable was uniquely related to the project.

When updating the code to meet the new requirements, we made sure to document all modifications to the original code. At the top of each class that had changes, there is a section titled “/* ASSESSMENT 2 CHANGES” which bullet points every change made in that class. For methods that are new, the first line of that method’s docstring reads “* NEW METHOD FOR ASSESSMENT 2”. For all of the brand new classes, the docstring for the class definition at the top of the class reads “* NEW CLASS FOR ASSESSMENT 2”. Additionally, we performed a significant amount of refactoring to make the code more modular and reduce dependencies to streamline the testing process. This mostly came from moving a large number of methods out of the “GameScreen” class into their respective classes. These changes are also documented, with the first line of the method docstring for each reading “* NEW METHOD FOR ASSESSMENT 2 - Refactored from GameScreen” .

Assessment 1 Deliverable Changes (8 pages max, <= 3 each)

Requirements

Original document
New document

For the requirements deliverable, there were few changes that need to be made to match the updated product brief. The requirements were iteratively changed and reviewed by 1 person before we finalised the change report as a group. This approach helped our team to exchange ideas from other people's perspectives and make a better improvement to our deliverable. Firstly, every number used in identifiers was deleted and changed to more clear and meaningful ID. For example, FR-INTERACT1 and FR-INTERACT2 were changed to FR-INTERACT-PLACING and FR-INTERACT-MESSAGE . These new IDs names were changed to more specific and descriptive names of that particular ID. Then, every description was amended with more detail according to the product brief and client meeting. The idea to change some description was based on the feedback that was given to our group to give more detailed descriptions for requirements. For example, in UR_DEVICE we changed the description to state that "The user is able to play on all Windows devices with Windows 10 or above" instead of "it is playable on desktop and laptop". Also, we changed the description of a few IDs, for example FR-NAVIGATE was updated by our group. Instead of just moving around the map using the arrow keys, the user shall use the WASD keys. Additionally, we deleted FR_MENU_OPTIONS and FR_MENU_SELECTION as these 2 were duplicates.

Next, we added extra IDs for user requirements UR_LEADERBOARD, UR_SCORE, UR_ACHIEVEMENT, UR_USERNAME, UR_SIMPLICITY, UR_CREDITS, UR_NPCS, UR_CITY and UR_AVATAR. UR_SIMPLICITY was added to match non-functional requirements while UR_SCORE was added to match functional requirements linked to it. Meanwhile UR_NPCS, UR_CITY, UR_LEADERBOARD, UR_ACHIEVEMENT and UR_USERNAME were added to meet extra requirements for assessment 2 specified in the updated product brief, and conversations with our client. For non-functional system requirements we simply added details to the descriptions and filled in some empty cells in the user requirements column.

Furthermore, we added functional requirements IDs that match user requirements. New functional requirements include FR-SPRINT, FR-NO-TICKING-TIMER, FR-GAME-WALKING, FR-GAME-DUCKS, FR-GAME-FRIENDS, FR-GAME-BAR, FR-GAME-STUDY-SESSION, FR-SCORING-STUDY, FR-SCORING-EAT, FR-LEADERBOARD-RANK, FR-SCORING-RECREATIONAL, FR-SCORING-STREAKS, FR-SCORING-SLEEP, FR-STATS, FR-TIRED, FR-USERNAME-LIMIT. Some FR IDs were added because of new user requirements. For example, FR-LEADERBOARD-RANK is linked with new UR ID (UR_LEADERBOARD), FR-STATS is linked with new UR ID (UR_ACHIEVEMENT), and FR-USERNAME-LIMIT is linked with new UR ID(UR_USERNAME). In other cases, we also added FR IDs because our group decided to add more features to the game. For example, FR-SPRINT was added to the ID because of feedback we received during user evaluation about our game being too slow.

Finally, we deleted NFRs that we felt were no longer necessary. We deleted NFR-MAINTAINABILITY2 because the game is no longer being taken over by another group after assessment 2, so we did not feel it was relevant any longer. Also, NFR-MAINTAINABILITY-TEAM-CODE was removed. This requirement stated that "Team members not involved in implementation shall understand what is happening in the code". We felt it was not necessary, as during the method selection stage of our project we assigned 4 people to the game's implementation. These 4 people were all familiar with how the code functioned,

and this was enough to significantly reduce our bus factor. We did not feel it was necessary, nor an effective use of our limited time, to have our other 2 team members study and understand the game's code.

Architecture

Original document

New document

In Architecture deliverable, group 16 used class and state diagrams to explain their architecture, as well as a table to link the requirements to their architecture. We added updates to all of these to accurately represent the new architecture which we had to create for the assessment two brief.

For the class diagram, we first had to fix inaccuracies in group 16's original architecture. For example, they claimed to have packaged the classes, but this did not match the implementation. They also claimed to have classes such as Time and Day, when all this logic was actually in the GameScreen class in their implementation. Once the architecture matched more accurately with the original implementation, we began adding new classes such as Leaderboard and Achievement to the class diagram, which were necessary to implement the new functionalities. We followed the approach of group 16 by having one large class diagram with reduced levels of information about classes' methods and attributes, this was used to show relationships between classes and give an overall view of the packages. Then, we also created a smaller class diagram for each package, these went into more detail, giving all the relevant attributes and methods about classes in that package, but not including every relationship to classes outside of that package. Appropriate explanation was then added to explain what each class diagram shows in the document.

For the state diagram, we firstly added comments for each state so that we know the description of the state and what the state is doing. For example, "overall gameplay in the Heslington Hustle" was added in the GameScreen's state. Then, we added more state because of the additional requirements. For example, Leaderboard, EnterName and AchievementScreen to match new requirements like "UR-LEADERBOARD", "UR-USERNAME", "UR-ACHIEVEMENT" . Also we added a few transitions to add the details from the previous diagram. For example, in GameScreen state, we added BackGroundScreen, PopUp and Message so that each state can iterate through GameScreen state while the game is still running. In addition, we also added a state that was not added by group 16. For instance DayShown was added so that the screen can show the current day and how many times left before the examination. Finally, we change the comments of transitions because we think it poorly describes how each transition happens from state to state.

The paragraphs and diagrams about Component-Entity-System were removed from the document, as the architecture of this project was a simple object oriented architecture, not an ECS architecture as was claimed. These diagrams inaccurately represented the project, and were also incorrectly listed as behavioural diagrams, rather than structural diagrams.

The table with traceability from the requirements to the corresponding architecture elements was updated to reflect the changes to our requirements. This was done because some requirements were changed in name, or split into multiple requirements that better reflect the sub-tasks of the project. There were also some new requirements due to the new brief requiring more functionalities. Other changes were required as we refactored some of the classes (such as GameScreen), meaning that the links from the requirements went to the wrong classes of the architecture.

Method Selection + Planning

Original document
New document
In the method selection and planning deliverable, there were a few changes that needed to take place to fit our team's personal organisation approach.

Firstly, in the "Outline and Justification of our Software Engineering Methods" section, we removed the information about the client interview. In Assessment 2 we did not require a client interview as we had all of the requirements from Assessment 1, and we felt the additional Assessment 2 requirements were straight forward enough that we did not need further clarification.

The previous team adopted an agile team approach with elements of the scrum method. We reviewed this and decided that we would continue this approach, so no changes were made to that part of the deliverable. This is because in Assessment 1 we closely followed these same team dynamics, and very little had to be adapted from the previous team's to meet ours. The previous team talked about using Google slides to facilitate planning of sprints, as well as categorising sprints and tasks in particular ways. We changed this, as we preferred to use a more abstract method to plan sprints through our meeting log book and gantt chart. Every week during our meeting, we would plan and allocate tasks for the next week and adjust our gantt chart accordingly. This still follows scrum methodology although was less formal than the previous team's method.

The original deliverable discussed the use of the spiral lifecycle for software development. No one in our team had encountered this before, but after conducting research into it, we decided to implement it into our methodology. We already worked in a way similar to it, but having a definition for this work-style helped us understand and plan out our project better.

As for "Identification and Justification of Development and Collaboration Tools Used", nothing needed adjusting from the original deliverable. The version control system, IDE, language, game engine, art tool and map editor were all the same as we used in Assessment 1 and what we planned to continue using.

As for the "Team Organisation" section, our first big change was how we decided to assign team members to tasks. The previous team used an approach where they assigned two team members to each deliverable, one being a leader, who oversaw the completion of the deliverable. Our team was more interested in hopping around different deliverables and completing tasks as they came up, so we decided to instead break down all the deliverables into a comprehensive total list of tasks needed for the whole assessment, and then assigned team members to each as they needed to be completed. We did these assignments based on each member's skills and preferences, so we did end up having an average of 2 primary members on each deliverable which we formalised in our deliverables table.

The previous team used a team roles strategy, where they had 3 key roles within the team. We felt this led to some redundancies, so we reduced this to just one team leader role which we assigned to Zachary. We felt the other roles were unnecessary, as these were covered by our task assignments and risk mitigation roles.

Beyond this, no major changes were made to the previous team's deliverable. We created a new work breakdown diagram with the tasks required for Assessment 2, as well as updating the deliverable table and task table to reflect the new deliverables and tasks in Assessment 2. We also created a new Gantt

chart which was updated during every weekly meeting, snapshots for which can be found on our website, based on our new tasks and projected dates. The final iteration of this was added into the “Method Selection and Planning” deliverable, as the previous team did not include it there and had it only accessible through the website. We did make a minor change to the Gantt chart through removal of the progress bars, as we felt these were unnecessary to understanding project completion progress and just made it more complicated to create the chart.

Risk Assessment + Mitigation

Original document
New document

For the risk assessment and mitigation deliverable, we decided to introduce a few changes. One of the changes we did make was the removal of strict roles which was specified in a small table above their main risk register. This table assigned two members of their team into each of the three roles that they created, “Project Manager”, “Project Owner”, “Team Leader”, these roles were then assigned to monitor and assess risks. We decided to remove these structured roles and assign only a single person to each risk. This was done in order to create more flexibility in terms of being able to move people between risks depending on their workload and where their current work lies. Additionally, we believed that assigning only one person to each risk allowed that person to better concentrate on the reassessment and monitoring of that risk, as if people were in pairs there may be breakdowns in communication, for example, a person not checking the risk as they are assuming that the other person is monitoring, meaning that no-one is taking responsibility for the risk; it could easily get out of control. Another reason why we removed these restrictive roles was to ensure consistency for our team and their responsibilities between assessments 1 and 2. In the first assessment we had very loose roles, with more emphasis on areas of the project being the description of our responsibilities. We wanted to maintain this, so decided to remove the previous teams roles so that our responsibilities for assessment 2 were more similar to the responsibilities we had for assessment 1, something we hoped would promote more efficient and a higher standard of work as we already had some experience and expertise from the first assessment. Next, we added definitions for the different types of risk that are used to categorise risks in our risk register so the person reading our report is able to understand the different types of risks. Furthermore, we decided to rename the parameters for the risk matrix: Impact Level to severity, probability level to likelihood and Priority level was shortened to priority. We decided to do this, to simplify the language used in the risk register, in order to make it as easy to understand as possible for our team by making the language consistent with our previous risk assessment.

The first thing we did when we got to the risk register was to read through the already documented risks and make decisions about whether or not we thought they were suitable for our team. The first change we made was to remove what was the risk with ID1, as we thought the brief of that risk we better covered in what was originally risks with ID6 and ID10, and with the aim of keeping things to minimum we decided to remove this risk.

We then added new risks to the risks register that we thought the previous team missed out on.
For example risk 18 which talks about poor code documentation and the impacts that that could have on our team during the development process. We also added new risks that we thought were relevant to our team taking over their project. For example, risks 19 and 20 talk about potential undetected errors in the game and poor quality deliverables that don’t meet assessment 1 criteria, respectively. Both of these risks could require a large amount of time to fix, so it is important they are identified and monitored to minimise impact and they were not relevant to the previous team in assessment 1, but are to us in assessment 2. Risk 17 also refers to ensuring that we as a team do not infringe on copyright laws, and was something that the previous team had missed out on and we thought it was important to include. In the end, risks 17, 18, 19 and 20 were the risks that we added.

Next, we then decided to re-rate the risks, using their existing rating system (the priority risk matrix), as we thought it would provide more depth and allow risks to be compared with each other through the use of a numerical and colour coded system. We did this, as it was important to have our own opinions

on the likelihood and severity of the risks so we could form our own list of priorities on which was the most important risk to monitor, based on our own judgement. We also thought it was important to add an explanation of the colour coded system for rating risks, in order to help explain to the reader what can be seen in our risk register.

Finally, we decided to alter the way in which we carried out risk monitoring. Previously, neither us or the previous team had a centralised, structured way to keep track of risk reassessments, something which could have easily had a significant impact on our development and project as a whole. To correct this, we decided to introduce a new document to keep track of risk reassessments and the actions taken to prevent/solve risks (explained in the Risk Monitoring section of the Risk Assessment document). We hoped this addition would allow us to better track and understand the status of the risks throughout the development process and allow us to hopefully take action to prevent risks from growing and becoming a bigger problem in some instances. We also made the decision to simplify the timeframe options for reassessing a risk. The previous team had used a range of timeframes to describe when a risk should be assessed and with the aim of simplicity, we decided to reduce the amount of options to three: Weekly, Biweekly and As needed. The aim of this was, again, to make it easier for our team to understand and implement and clearly as to when the next reassessment is due.