Module	Engineering1 (Eng1) - COM00019
Assessment Title	Assessment 2, Cohort 2
Team	Quakthulu (Team 16)
Members	Aaron Price, Alec Coates, Charlie Curedale-Rayner, Eleanor Griffin-Smith
Deliverable	Change Report

Formal approach to change management

Requirements: In order to manage the proposed requirement changes for assessment 2 we followed “requirements change management” [1] from I.Sommerville’s book “Software engineering”.

1. *Problem analysis and change specification* – the change proposal was delivered by the stakeholder and will be review by our team to ensure it is valid and possible. We will then meet with our stakeholder to confirm we have interpreted the new requirements correctly.
2. *Change analysis and costing* – we will then go through the existing code and the previous teams’ requirements (traceability information) to determine what modifications need to be made.
3. *Change implementation* – Finally the documentation is modified

Architecture: From the updated requirements and using the assessment 1 architecture we can add new functionality to the existing abstract architecture documentation. We will be using OOP therefor we can also use this to add new classes onto the existing UML diagram for the concreate architecture.

Method selection and planning: Like the previous team we will also be using an agile SCRUM development process which lends itself to changing requirements so we will approach the project like a new iterative prototype. Also due to the shorter timescale of the assessment and our small team we will be using collaboration tools and organisation methods that are different to the previous team.

Risk assessment and mitigation: Risks have been identified and analysed in by the previous team so we must plan ahead to monitor the risks, so they do not hinder our project.

Website: The website will be updated with new documentation in a new section of the website dedicated for assessment 2.

Implementation: Like the previous team, we prefer to use a GitHub repository for code so we created a new branch from their project into a new one that we could edit for ourselves.

Changes made to Assessment 1 deliverables

Each section of every deliverable is recorded, and the changes made will be listed. The original deliverables from assessment 1 as well as the updated versions are available on the website.

Requirements:

Introduction:

- No changes made as this introduces the requirement documentation with relevant research done by the previous team before they formally recorded the requirements. This research also aided our team when alerting the requirements.

How requirements were elicited and negotiated:

- The method of acquiring requirements in assessment 2 was the same as in assessment 1 so the initial paragraph containing the method was unchanged. Except for a sentence saying that it was repeated.
- A paragraph after this was added to explain that due to our small team size certain new requirements were dropped from the project so it was possible to complete in the timeframe.
- The SSON was updated to better reflect the old and new requirements.

Why requirements are presented as they are:

- No changes made to this justification as our team agrees with it and have added requirements in the same way.

User requirements:

- Assessment 1 requirements have been condensed into a single requirement called UR_MAINTAIN_PREVIOUS. This has been done as the majority these requirements have already been implemented so just need to be maintained in future releases. It also helps with the clarity of the document for our team, so we know which requirements have not already been implemented.
- 3 new user requirements have been added to the table with their associated risks and priority level to reflect the project changes. These are UR_CHOOSE_DIFFICUTY, UR_SAVE and UR_LOAD.

System requirements – Functional:

- Assessment 1 requirements have been condensed into a single requirement called FR_MAINTAIN_PREVIOUS for the same reasons as in user requirements.
- 6 new functional system requirements have been added to the table with their associated risks and design requirements to reflect the project changes. These are FR_DIFFICULTY, FR_DIFFICULTY_SCREEN, FR_SAVE, FR_SAVE_SCREEN, FR_LOAD and FR_LOAD_SCREEN.

System requirements – Non-Functional:

- Assessment 1 requirements have been condensed into a single requirement called NFR_MAINTAIN_PREVIOUS for the same reasons as in user requirements.
- 2 new non-functional system requirements have been added to the table with their associated risks, fit criteria, and user requirements. These are NFR_DIFFICULTY and NFR_QUICK_LOAD.

Constraints:

- No changes made to the constrains table as overall project constraints have not changed.

References:

- No changes made as no additional references used.

Appendix:

- Previous teams use cases remain as they are still relevant to project.
- New use case “Resume game” is added to reflect the additional requirements for assessment 2.

Architecture:

Architecture – Part A:

- This section mostly unchanged except for the additional features and classes added to the flowchart and UML diagrams. Also, we did not have to pass on the project for another assessment, but reusability is still a good trait to have in code.
- New diagrams have been drawn with Draw.io instead of Lucidchart because as a team we are more familiar with it.
- New requirements meant that all forms of architecture would need to be updated to account for new deliverables of implementation of both how the game would behave and be structured within abstract architecture as well as concrete architecture needed to portray the code accurately.

Chart 1 – Abstract Behavioural View:

- The start had to be changed as a difficulty selection process would need to occur when the game would be run to pre-set the difficulty.
- The game would need a pause function and so a check to see if paused was implemented.
- Game needed a save and load feature and so a new stem off from the check if the game was paused was required to truly reflect behavioral changes the code would have to undergo to implement these features.

Chart 2 – Abstract Structural View:

- Structure needed to be updated to reflect the code more accurately from when we inherited it. For example Dragon Boat Game has been added as a controller and parent class for the game.
- A pause menu, save/load screen was implemented as part of the game screen, as with the new requirements it would have to ensure a pause menu and save/load screen was implemented into the games structure.

- In addition a difficulty screen would have to be implemented to meet the new requirements of the game having a selected difficulty level.

Chart 3 – Concrete Architecture:

- Concrete updated to include all classes that have been implemented into the code that have allowed us to achieve and fulfil the new requirements.

- Therefor as base code was not altered considerably have built off old concrete architecture and implemented new classes that achieve the new requirements they include: All Descriptor classes which act as subclasses for the given class have been added to implement save/load game feature as well as a PauseMenu, SaveLoadGame, SaveGameScreen. To fulfil different difficulty levels, difficulty screen has been added as well to concrete architecture to represent the class that has been implemented to fulfil this requirement.

Architecture – Part B:

Systematic justification for the abstract and concrete architecture:

- Unchanged as we used the same approach when adding additional requirements

Design of the abstract Architecture:

- Unchanged as we used the same approach when adding additional requirements

Progression from abstract to concrete architecture:

- Unchanged as we used the same approach when adding additional requirements

Relate the concrete architecture to the requirements:

- The original contents of the table justification and requirement ID section have been updated to reflect the new "maintain previous" requirements.
- 5 new rows have been added to justify the new classes for assessment 2.

Method selection and planning:

Software engineering methods:

- Much of this section is unchanged as the previous team's research into methods was good and helped us reaffirm our previous choice of an agile SCRUM method.
- However due to being a smaller team we have approached SCRUM less formally than the previous team e.g. No official SCRUM master. Therefore, the final paragraph of this section has been replaced by a paragraph explaining our different approach.

Development and collaboration tools:

- The first paragraph "Scrum management tool: Jira" has been removed as it is not used by our team during assessment 2. (Information on SCRUM management is explained further on in the document).
- The section "Documentation tool and Deliverable Repository: Google Drive" is mostly unchanged as we agree with previous team that this is the best option. Also, the sentence on the tool Confluence has been removed as it related to Jira which is no longer being used in the project.
- The section "Communication tool: Slack and Discord" has been replaced by" Communication tool: Facebook messenger and Zoom" as our team preferred these tools and were more familiar with them.
- The section "Version Control tool: GitHub" is unchanged as our team also used this and agrees with the previous team's justification.

- The section “UML diagram tool: Lucidchart” has been replaced by “UML diagram tool: Draw.io”. These tools have a similar function, but our team has more experience and feels more comfortable using draw.io.
- The section “Implementation Tool: Java” is unchanged as our team also used this and agrees with the previous team’s justification.

Team organisation:

- The section “organisation” has been changed to better reflect how our team works together and maintains project progress. We did not use Jira like the previous team, so these sections have been removed.
- The section “Roles and Task allocation” has been changed to better reflect how our teams allocates tasks.
- The section “Development Architecture” is mainly unchanged except for the days we have meetings and the collaboration tools used. This is because we agree with the previous team’s justification.

Project plan:

- The first section of this documentation is about task priority. This justification is unchanged as our team agrees with it. The only difference between the original is that we have used a different Gantt tool.
- A new Gantt chart and explanation that is relevant to assessment 2 has replaced the old one. We decided to do this, so our tasks were clearer and most requirements from assessment 2 have already been implemented. The old project plan is still available on the assessment 1 section of the website.
- Task dependency have been updated for the assessment 2 requirements and deliverables.

References:

- Gantt chart software reference updated with the new software used in assessment 2.

Risk assessment and mitigation:

Introduction:

- Unchanged as it is a good introduction to the risk documentation.

Justification for Risk Format:

- Unchanged as our team will use the same risk format and agrees with the previous team’s justification.

Risk mitigation:

- Mostly unchanged as we will use the same method. The only change is that the updated version does not talk about Jira as this is not used in assessment 2.

Risk monitoring & Management Process:

- Unchanged as our team will use the same approach to manage the risks.

Tabular Presentation of Risks:

- Risks are unchanged apart from their respective owners. Theses have been reassigned to member of our current team.

Refences

[1] I. Sommerville,"4.6.2 Requirement change management" in *Software engineering*, Harlow Essex, England: Pearson 2016