Change Report

Team 5:
Kevin, Zuhur, Erin, Jack, Aidan, Ryan, Sam

Our Process

In order to adapt Team 9's original documentation and implementation, to meet the needs of assessment 2, we had to go through a systematic process to analyse and modify their work. To achieve this, we used tools such as structured planning, team collaboration and task management to ensure the updates were meaningful and effective.

To begin the process, we had to do a thorough review of Team 9's documents for Assessment 1, and identify areas that required changes. To take a more efficient approach when it came to reviewing, we split our teams into two groups, mirroring the structure we used for Assessment 1: one group focused on implementation-related tasks, whilst the other concentrated on documentation and planning. This allowed us to make use of individual experience collected in assessment 1, ensuring that the work we produce would be more efficient and up to our best standards.

We closely examined key elements of the original documentation, such as the architecture plantUML diagrams, the planning methodology, and the structure of deliverables by comparing these to the second assessment brief. To manage the update process, we created a detailed table in Google Docs that broke down the brief into bullet points. This table outlined the sections which needed updates, assigned responsibilities for each task, and established deadlines. This table acted as a central reference point, helping us to organise the workload and maintain precision throughout the process.

The PlantUML architecture diagram was updated to reflect the latest requirements we added for assessment 2. Initially, we made behavioural diagrams using PlantUML to plan the architecture for new requirements. These diagrams were built on top of Team 9's existing behavioural diagrams, as the code and architecture for assessment 1's requirements hadn't changed. We used this as a solid foundation on which to further develop and, by utilising PlantUML, we could easily modify and develop their existing architecture.

One of the critical adjustments we made during the adaptation process was updating the Gantt chart from Assessment 1. This is because it was only designed for Assessment 1, and lacked the points needed for the additional requirements coming from the Assessment 2 brief. To address this, we created an extended Gantt chart which broke tasks up making sure each had clear deadlines. We followed this closely, ensuring that all deadlines were met.

By employing this structured approach, we were able to make sure that all changes were well-coordinated with the new objectives of Assessment 2. This method not only improved the quality of the updated documentation, but also reinforced collaboration and understanding within the team.

Requirements

Original Requirements - LINK
Updated Requirements - LINK
We updated the requirements to ensure that there is some clarity and proper categorisation. In the assessment brief for assessment 2, we were asked to implement two new features in the game, so in order to follow the brief we added these as requirements.

The first one was UR_LEADERBOARD, this is a user requirement which states that users should be able to input a username to be placed on a leaderboard. The system requirement for this (FR_LEADERBOARD) the system shall display the leaderboard in both the start game scene and the game over scene, and the system requirement FR_SCORE shall record the score.
Additionally, there was another requirement UR_ACHIEVEMENTS, where users shall receive awards based on their interactions with the game e.g amount of buildings placed. The system requirement for this is that the system will be designed to analyse user behaviour to award based on gameplay.

We've also decided to update some of the previous requirements into the right categories, here's some of the changes made:

From team 9, the UR_PERFORMANCE has been refined by splitting into NFR_MIN_SPEC (the system should run on final specifications) and UR_UX (users must have a pleasurable user experience). This change is fitting as the original description had two different requirements in one, they were also inaccurately placed.

We also deleted the user requirement, UR_MONEY, this is because we felt the usage of money overcomplicated the game, we wanted it to be easy for users of all abilities to play. There are already constraints placed in the game, e.g, space on the map, and time, so the addition of money as a constraint wouldn't be justifiable.

The user requirement, UR_INTUITION, was edited to refine its definition, eliminating any unnecessary and unclear language.

Got rid of the part in FR_BUILDING where building buildings takes time, this change was because, due to the game having a short run time already we thought it would take away from gameplay.

UR_DIFFICULTY, FR_DIFFICULTY_SELECTION, and FR_DIFFICULTY_EFFECTS were deleted because of time constraints, due to the task at hand already being demanding we felt that difficulty wasn't a necessary requirement.

We combined FR_EVENT_TYPES and FR_EVENT_VARIETY into one called FR_EVENTS, this is because we thought having three different types of events wasn't necessary as it created a time constraint in the implementation process.

FR_SATISFACTION_BUILDINGS was edited just for better understanding and clarity.

~~This restructuring ensures a clearer understanding of system capabilities and user needs.~~

Risk Assessment and Mitigation

Original Risk Assessment and Mitigation - **LINK**
Updated Risk Assessment and Mitigation - **LINK**

We updated the risk tables to better suit the structure and dynamics of our team. The revised register, shown below, shows these changes.

In our original documents, a key difference between our approach and that of Team 9 is how the ownership of risks is assigned, whilst Team 9 assigned a single owner to most risk, our team opted for a collaborative approach, which involves multiple team members working together on managing risks. We chose this approach as we thought it highlighted our commitment to teamwork. However, after reviewing Team 9's documentation, we decided to adopt their approach of assigning no more than two owners to a risk. This change made sense as it allowed for less confusion and clearer accountability, ultimately leading to more efficient risk mitigation.

Due to the nature of most risks being mitigated or avoided, we also decided to change the likelihood and severity of risk. This was important because it meant we could accurately shift our focus to more severe and probable risks. One notable example is R4: "Product is unable to be built/run on the required hardware or Operating Systems". Given Assessment 1, we know that the product has been successfully built, thus making this risk easily avoidable, allowing us to prioritise other potential risks.
(All changes can be seen in the updated document)

We did decide to include some extra risks to watch out for which we carried on from our Assessment 1 submission. This is because we felt as though there was some risk management missing. For example, we added "Implementation falls behind set team deadlines", to ensure that potential delays in the project were actively mitigated.

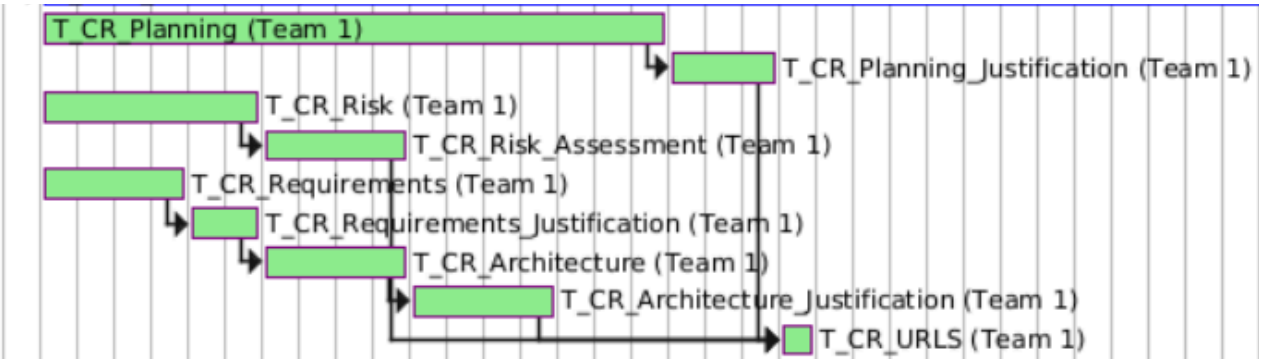
An important risk we included was "Misinterpretation of previous teams work". This was important because any misunderstanding of Team 9's work could create incorrect assumptions which could lead to flawed implementation and setback in project deadlines. By identifying this risk we were able to create a proper mitigation plan where we make sure to contact the previous team to clear any ambiguities.

Planning

Original Planning - **LINK**
Updated Planning - **LINK**
There were several key adjustments and additions to the original project plan which we had to make to align it with the group takeover and new requirements of assessment 2. Firstly, Team 9 had assigned job roles to each member of their team, and whilst we had more team members than them, we adjusted the job roles to allow for them to be split between the most fitting team members within our group. Furthermore, we had to expand the scope of these roles by adding new tasks and requirements to certain roles, to ensure full coverage of the new deliverables and content required for assessment 2 such as testing and continuous integration.

Team 9's original plan and schedule were limited in coverage as they concluded at the submission date for Assessment 1, and did not account for the requirements and deliverables of Assessment 2. To address this gap, we reviewed the updated product brief and aligned the plan with the expanded requirements, ensuring everything was included. This involved constructing a comprehensive new plan that outlined all tasks and work packages necessary for Assessment 2, including detailed timelines and priority levels. From this breakdown, we developed updated Gantt charts to provide roadmaps that now spanned over both Assessment 1 and Assessment 2, ensuring full coverage of the project.

Rather than specifically writing out each task's dependencies and priority level like team 9 did, we decided to focus on making more in depth Gantt charts, which showcase the tasks dependencies within the diagram using arrows, as shown in the example below.



Architecture

Original Architecture - **LINK**
Updated Architecture - **LINK**
For the architecture we started by looking at their code and architecture and then decided what needed to be added and what needed to be changed.

For our first iteration we didn't plan to change the code that implemented the requirements for assessment 1. We wanted to focus on using this foundation to start building the architecture for our new requirements. This meant we could make a copy of Team 9's architecture diagram using PlantUML shown in diagram 6. Then in diagram 7 we looked at the new and remaining requirements that we needed to implement. The first requirements we looked at were, FR_EVENT_TYPES, UR_ACHIEVEMENTS, FR_SCORE and UR_LEADERBOARD. We first looked into designing the architecture for events and scores. We began to add the classes we thought would be needed. These ended up being: EventManager, Event, Score and EventBar. We added these as branches of the classes that they would either need to inherit code from or interact with. We added eventManager as a branch of world and Score and EventBar as branches of GameScreen so their information could be displayed to the player.

For our second iteration (diagram 8) we had to first update some of our previous architecture. During the implementation of the last iteration we made a few changes to the logic of the code, for example we added the tick() method to the Event class as this was needed for scoring. We then added AchievementManager, achievements and achievementRequirements as a branch off world. This was so we could start planning the implementation for UR_ACHIEVEMENTS. We made some changes to the UI by adding more branches to the gameScreen as this would be needed to display new information.

In our third iteration (diagram 9) we created a new branch to our GameScreen so we could look into implementing the leaderboard. This would need to be displayed to the user and have access to attributes in the score class so we decided this would be the best place for it. Similar to our last iteration there were a few changes made during implementation so we made sure these were also updated on our architecture.

For our final iteration (diagram 10) we focused on the readability of our architecture. We decided to create separate trees for each screen so the diagram was less cluttered. It then made it easy to add a tutorial screen as we could copy the architecture of a previous screen and make the appropriate edits. During this iteration we also created a state diagram (diagram 11). We used Team 9's original diagram (diagram 3) as a template and then made it reflect how the game now behaved. This made it easy to make sure the game flowed correctly and every screen was accessible.

Finally during implementation we created some more behaviour diagrams (diagram 12 & 13). We used Team 9's original diagrams (diagram 4 & 5) as a reference for design. We created a diagram for both achievements and events. This was so we could plan out how classes would interact during gameplay. For both of our diagrams we expanded upon the building placement diagram (diagram 4). As the user would be placing buildings we didn't need to redraw these actions as this architecture hadn't changed. For example in diagram 13 we simply added in the new actions that would occur if the user completed the requirements for an achievement. We made these so we could plan how user actions would affect gameplay and how we would need to relate this to our architecture.