Change2

Cohort 1, Group 9
Chris Sewell, Fedor Kurochkin, Matt Durham, Max Peterson,
Oladapo Olaniran, Wojtek Tomaszewski,
Yuqi Fu.

Change Management Processes, Tools, and Conventions

When we took over Team 11's project, we followed a systematic process for planning, making, tracking, and reviewing changes to the original Assessment 1 deliverables. All proposed changes were first identified by examining each inherited documentation against the assessment 2 deliverable. In doing so, we identified areas that required extensions and continuations.

For documentation changes, we chose Google Docs because it supports real-time collaboration, allowing multiple team members to work on the same document simultaneously. Furthermore, Google Docs allows us to track changes to the documentation, ensuring all team members can see them. We also continued to use our WhatsApp and Discord channels to communicate and discuss any future changes we would like to make regarding either the code or documentation.

With continuous updates to the method selection and planning, we are able to set deadlines and assign team members to specific tasks. Ensuring we are able to cover the whole assessment 2 deliverables. Furthermore, it keeps the team accountable and helps us identify when a team member is struggling to meet deadlines, so we can take appropriate steps to address it. We can assign multiple people to a task to increase the speed and efficiency with which we complete our set deadlines.

For code changes, we first held a meeting to discuss the new events we wanted to add to fulfil the product brief. We also continued to use GitHub for version control, issue tracking, and pull requests. In doing so, we can track code changes and review them before merging. Also, code comments will help us identify new or extended functionality introduced in assessment 2.

Method Selection & Planning

The methods and collaboration tools used by Team 11 are almost identical to the ones we chose. However, in group 11's documentation, they do not explicitly mention the tools they used for documentation. The use of a shared Google Drive for collaboration is vague, and they also do not discuss any alternatives that they considered. This change will outline how we provide documentation as a team and justify the reason why we chose to use Google Docs as our method for documentation. [Highlighted in the collaboration software section of the method selection and planning documentation.]

Our team also took a completely different approach to team organisation, where we first identified each other's strengths and weaknesses and assigned tasks accordingly. This contrasts with team 11's approach, where they initially worked on tasks together, which wasn't sustainable, and they eventually had to redesign their whole approach. Changing the report to fit our approach makes work allocation easier, reducing bottlenecks and confusion throughout the project. Additionally, we would review the previous week's work in our Tuesday sessions and create a Gantt chart for that week, rather than waiting until the next day to do so. [Team organisation section]

The weekly plans our team created also differ slightly from those Team 11 created, as they did not include who was assigned to the tasks they outlined. Adding names to tasks improves transparency and holds our team accountable when assigning tasks. [Team organisation section].

Furthermore, we had to change the project plans, plan evolution, initial goals, and weekly plans sections because the weekly plans for assessment 2 differ from those for assessment 1.

Requirements

The team extended the inherited requirements document because we obtained additional information during handover. The content of the prior version of the document remained almost fully unchanged. [Req1] The only changes made to the inherited document were corrections of wrongfully assigned codes to certain requirements. Therefore, in order to limit consistency errors in future development, these were removed. Previously seen NFR_PAUSE, NFR_END_GAME and NFR_TIMER have now been completely cut out from the document. [Req2 - Requirements Elicitation pre Handover, p. 5] However, we added a new section, featuring all new goals that were revealed after the handover. [Req2 - Requirements Elicitation Post Handover, p. 8-10] Regarding the format, the team retained the original table structure and overall document layout. Therefore, this was not modified by the team and instead continued in the final version. Although, the team did not add new constraint requirements beyond those already documented in the older version of requirements.

As for content, post-handover brief changes presented the team with new requirements. This meant that the document needed to contain updated information on all new features expected to be present in the final version. In addition, apart from having simple user requirements later converted to functional requirements, the brief went more in depth on success criteria. This required the team to record additional non-functional requirements, which further extended the document. For example, UR_LEADERBOARD, FR_LEADERBOARD_STORAGE and NFR_LEADERBOARD_PERSISTENCE were all included in their respective tables. [Req2 - User, Functional and Non Functional Requirements tables, p. 9-10]

Further content additions were based on feedback received from user evaluation. The tests revealed requirements focused on previously implemented features. This evolved into new functional and non-functional requirements being recorded in the tables. Furthermore, the non-functional requirements were based on subjective opinions of users. This created a need for extra information to be recorded in the tables including fit criteria as one of the fields. Examples include UR_GEESE_IMPACT, FR_GEESE_IMPACT, NFR_IMPACT_VALIDATION being added to the tables accordingly. [Req2 - User, Functional and Non Functional Requirements tables, p. 9-10]

Finally, the last content additions were based on the requirements that carried over from the first version of the brief. The initial demands from the client remained relevant alongside new requirements introduced post-handover. Therefore, all requirements that were either not recorded or not implemented by the last team, needed to be acknowledged in this version. We reviewed the inherited requirements and identified such gaps. We were then able to add missing requirements and assign appropriate priority values ensuring the core gameplay mechanics' implementation. This completed the document with information that kept all unfinished requirements present, improving consistency. One example is UR_PAUSE deriving into FR_PAUSE and NFR_PAUSE_MENU_NAVIGATION. [Req2 - User, Functional and Non Functional Requirements tables, p. 9-10]

Risk Assessment and Mitigation

The inherited document already provided a suitable risk management process and risk register structure which were retained for continued documentation. The introduction section documented the process of risk analysis and development of mitigation strategies. The team decided to keep the existing risk management process description unchanged and instead extend the document with handover-relevant updates [Risk2 - Risk Management Post Handover, p. 5-6]. The risk register was also completed with all relevant information clearly displayed. Due to all listed risks continuing to stay relevant, the initial version contained critical information on potential problems with mitigation strategies that could be useful in this iteration. [Risk1 - Risk Assessment and Mitigation, p. 2-5] Therefore, all information within the description and mitigation columns remained consistent. [Risk2 - Risk Management Pre Handover, Risk Register Columns Description and Mitigation, p. 3-4]

Even though the older version of the document showcased important information, the columns for owners and some severity/likelihood ratings needed changing. We reviewed all ratings and recalibrated the values to better represent certain risks. Therefore, some entries in the likelihood/severity columns were changed. All these changes were marked with an asterisk (*) to highlight edited information. [Risk2, Risk Management Pre Handover, Likelihood and Severity columns, p. 3-4] Due to the handover procedure, all data presented in the owner column also became irrelevant for this iteration. Hence, the team altered all records within the column assigning the current members of the team with previously established risks. [Risk2 - Risk Management Pre Handover, Owner column, p. 3-4]

Following the edited version of the inherited document is a new section called Risk Management Post Handover. This new section contains information on the handover-specific risks and the additional identification activities introduced for this iteration. The risk management procedure builds on the inherited risk management approach, adapting it best for handover-specific risks. The overall risk register format remains unchanged. However, the Type column was extended to record handover-related types of risks. These types are as follows: Incorrect, Missing, Misinterpretation and Inconsistency. [Risk2 - Risk Management Post Handover, Type column, p. 6]

Architecture

The choice to extend development of the current project was based on a number of factors, with the existing Entity-Component-System structure being most relevant to architectural extensibility.
The team's previous experience developing an ECS structure meant updating existing components and systems and implementing our own much more adaptable while maintaining consistency.
After reviewing the existing architecture deliverable [Arch2] (and the diagrams webpage) we identified the type of structures used and where we needed to update them with the new implementations in order to meet the new requirements described in the previous section.

To implement the new additions such as the 'Bob hidden event' and the 'attendance system' we needed to update the Entity - Component packages with our own which already utilise existing systems such as the 'message' system and integrated our own relevant systems. The decoupled, modular nature of the ECS structure meant our extensions were easily made compatible as illustrated in the updated table entries and diagrams in the Arch2 deliverable (pages 10-12).

Following the existing deliverable format, detailed descriptions of the new implemented systems were added starting from page 12.
The Attendance system was added in order to meet the updated brief requirements of extending the number of positive and negative events of the prototype. This new mechanic will reward or penalize the player depending on the number of check-in codes collected. By re-engineering this existing collectable into our new event system successfully demonstrates architectural consistency and extensibility whilst meeting the multiple new requirements.
The details of this system are on pages 12-13.

In order to fully meet the newly identified requirements from the updated product brief, A leaderboard and Achievements system needed to be implemented.
In order for multiple players to track their score on the leaderboard, the system was designed separately to the ECS architecture, and uses LibGDX preferences to store serialised entries via JSON.
This system is described in detail on pages 12-14 of Arch2.

The Achievement system also needed to exist outside of the typical ECS hierarchy to preserve and track players' event and state data and compare them against a list of unlock conditions needed to trigger the achievements.
This system is described in detail on page 14.

Following the Arch2 deliverable format, the complete set of architecture diagrams follow the system descriptions section from pages 15-18:

A behavioural sequence diagram was chosen to represent the Bob hidden event system based on its multi step quest like mechanic.
(Note the original diagram was updated to correct a bug within the event)

A Sequence diagram was also chosen to best represent the Attendance system based on its multi-step, event state and entity tracking attributes that span the entire length of the game from start to escape.

Due to its external implementation, the leaderboard system has a basic structural diagram to accompany its sequence diagram to illustrate how it was designed independently with a closer look at how it communicates player escape data.
Similarly the Achievements system has a sequence diagram to show how player event data and counters are employed to track achievement unlocks across multiple playthroughs.

As mentioned in previous sections, the existing requirements needed to be updated alongside the newly identified ones. The Arch2 system-requirements justification table was updated and extended to include these necessary changes.
The appropriate fields relating to NFR_TIMER and NFR_END_GAME were updated to: FR_TIMER and FR_END_GAME respectively to better reflect their purpose to the corresponding systems.
Additional entries were added based on the newly identified requirements from both the updated brief and user evaluation findings data.
Updated Systems-requirements table found on pages 19-21.