Change report

Cohort 4 Group 6

Javengers

Braithwaite, Max
Faruque, Amber
Fu, Zhuoran
Kocaman, Melike
McDermott, John
Rissen, James
Scott, Charlotte

Summary

For Assessment 2, our team focused on understanding, extending, and safely modifying the deliverables inherited from Team 1. We began with a structured review of the existing codebase and documentation to identify design decisions, assumptions, and areas requiring refactoring or extension. This process helped us become familiar with the project and ensured that new work aligned with the original architecture and documentation without introducing inconsistencies or duplication. We also implemented automated JUnit tests to validate both inherited functionality and new additions, helping us detect issues early and maintain reliable behaviour.

To manage changes to inherited artefacts, we used a careful, step-by-step workflow. Team members first reviewed existing code and discussed potential impacts. Changes were developed separately and reviewed before integration, helping us check new behaviour against the original and reduce errors. This approach was especially important when working with unfamiliar code from the previous team.

To support this process, we used GitHub version control to manage and track all changes made to the inherited artefacts. By working in separate branches, team members were able to experiment with extensions or refactoring without affecting the stable codebase. Commit histories and pull requests provided a clear record of what was changed, why the change was made, and how it related to the existing implementation. This made it easier to review modifications, discuss design decisions, and revert changes if issues were discovered. Overall, GitHub version control allowed the team to safely extend the inherited system while maintaining code integrity.

To plan, create, and track updates to documentation we inherited, we followed a structured, collaborative workflow. Team members reviewed the existing material to understand its structure and content before making additions or edits. Any changes were first discussed within the team to ensure consistency.

We used Google Docs as our primary tool for editing documentation, which allowed multiple team members to contribute simultaneously and leave comments for review. Version history within Google Docs helped us track changes over time, see who made updates, and revert if necessary. For task planning and progress tracking, we maintained a Kanban board in GitHub Projects dedicated to documentation tasks, allowing us to visualise pending, in-progress, and completed edits alongside code-related tasks.

In terms of conventions, we preserved the structure and naming used by Team 1, including headings, sections, and CRC card format, to maintain consistency. All documents were shared openly within the team so anyone could review updates, provide feedback, and ensure that the documentation remained accurate and up to date. Regular sprint discussions resolved ambiguities, approved major changes, and reviewed completed updates. In addition, weekly reviews and checkpoints were held to monitor progress and coordinate updates across both code and documentation, ensuring alignment throughout the project.

The website was also an inherited deliverable. We cloned the previous team's site and added Assessment 2 material, including all new documents, the game's executable JAR, and a link to our code repository. Updates were made using GitHub version control, reviewed locally and collaboratively, and deployed via GitHub Pages. We maintained the original structure and style, ensuring all links worked correctly and content remained consistent with the existing site.

ORIGINAL AND EXTENDED DOCUMENTS:
https://team-6-eng.github.io/WHALES-Website/deliverables.html

i. Requirements

The Requirements document inherited from the previous team was fairly extensive, the three tables required were already complete. A few changes have been made to them in order to account for how we wish to continue the game. All changes to the table are highlighted in the Requirements document.

The first change that we made was to implement a menu system, as the previous system was clunky and cumbersome and required you to either wait a set amount of time or press a key that was not clearly indicated. Hence, we wanted to add a traditional button system so we added the requirement MENU_UR. Several menus link into this user requirement including the pause, main, leaderboard and achievement menus. Another change we added, although small, is to add a UI_UR requirement to be specific about how we want the game to appear to the user, this specifically had a sub-requirement about buttons as the game we inherited didn't have any on-screen buttons/ and we believed that needed to change to make the game more intuitive to players.

Through discussion and interaction with users, we felt that the game was simplistic and easy to complete. So a new requirement was added that meant several pieces of the goose must be collected. This requirement culminated in Goose Pieces which is under UR_EVENTS. Additionally, we developed a more specific definition of the “energy” system within the game, the previous energy system was difficult to understand and users were struggling to understand what it was linked to or what it was used for.

The most major requirement changes were in relation to the additional requirements for part two including the extra events which we were required to add. The additional two positive event requirements that we added consisted of a score bonus item which the player can collect and access to a mini-map, which allows the player to locate items more easily and complete the game quicker. The negative events that we added were, a security guard that will hinder your attempts to find and reconstruct the goose, a large frog that will swallow you and prevent you from moving for a few moments, and also a contradiction to one of the positive events that was previously added - that the player can drink too much from the vending machine and this will result in a sugar crash. We also altered how the coffee slip affected the player as the previous effect looked unprofessional and could be mistaken for a glitch. Finally, the additional hidden events consisted of vanity items to collect and dress the duck statue, a professor who rambles once triggered and a windy effect which may randomly appear on the main path when the player joins it, and may speed them up or slow them down depending on which way they are attempting to travel. The previous team had listed their power cut event as a hidden event however we changed this to be a negative event.

Some of the requirements we left unchanged were ones stating the need for a variety of player characters, along with a need for a top down camera to view the game. The requirements that were left unchanged were left that way, as they aligned with the direction we were intending to take the game and required no alteration. Other requirements such as FR_ENERGY were redefined to make their purpose clearer and more direct. This was done to streamline the project and reduce confusion.

ii. Architecture

Identified Gaps in Inherited Documentation:

- CRC cards: **Section D3a** mentioned original CRC cards being provided, although this was not available on the document and was difficult to find on the website with no proper online version (only paper photos) available. There was also no iteration or final updated CRC cards were provided
- CRC requirement traceability matrix: No mapping between CRC and requirement ID's
- No links to website containing diagrams in appendix section **D3C** or in justification section **D3A**

Changes to Assessment 1 Deliverables & Justification:

- Final CRC cards for their submitted code, this allows us to keep better class cohesion, lower coupling between classes and an easier to understand and edit codebase. This also allows for us to run through class responsibility during development, reducing redesign work later. An example of this would be when we extracted AudioManager, the CRC card clarified its single responsibility being to: manage all audio, reducing coupling in number of classes. It also allows non-coding team members to quickly understand the system's structure and make changes without causing issues.
- Requirement traceability matrix created for the end of assessment 1 to fulfil deliverables along with CRC cards (as mentioned above) and requirements from their submitted deliverables. This provides us with full coverage of each requirement and shows architectural support for each of them, or informs us of ones which are not fulfilled yet. It also allows us to make changes to our code management (some mentioned below such as audio manager class). Finally, it demonstrates that the architecture design is based on the requirements that we/the original team gathered.
- Website links to diagrams/deliverables relating to architecture. Despite the lack of a page limit, we will change this in order to show consistency and direct links between the presentation on the website and documents. This shows the workflow process via only diagrams and no text, completes the required deliverables asked for, and is good practice to link all documentation. It is also necessary since, despite the interim diagrams for classes being complete, they were not linked or included on the document, meaning that you are unable to easily trace the evolution of the classes and architecture.

Why we maintained the Layered, OOP structure:

By maintaining a clear layered structure from the original project, this allowed for clear task allocation by each layer, independent development without the need of excessive coordination and easier separation of tasks by team member strengths due to the reduced cross over of layers. This also helped reduce merge conflicts since most layers are relatively isolated within themselves, barring a few method calls or initializing variables.

Using layered architecture has also enabled improvements such as adding the headless testing platform to the presentation layer without touching game logic, splitting collision

methods in the domain layer while preserving all visual/presentation interfaces and integrating Scene2D UI to the presentation layer without having to modify any logic.

Enhanced Documentation of architectural structure:

We strengthened **Section D3a** by adding:

- Clearer concrete/abstract architecture distinction
- More in-depth UML/notation justification
- Framework implementation trade-off analysis
- Enhanced assessment of benefits and losses

The Planned additions for Part 2:
All iterations of sequence/behavioural diagrams have been created and presented on the website and document, with proper justification and all relative diagrams or links to requirements. This provides us with continuous documentation and will allow them to serve as the justification and thought behind our current architecture. As well as enabling team coordination during parallel development to keep diagrams up to date and frequently iterated. It also allows for immediate feedback when these architectural plans prove to fail or not work as intended, allowing for active changing and upkeep to produce the best architecture for this system. This is also vital due to the fact that these iterations provide traceability from requirements to implementation, as well as serving as validation that the architecture is supporting the intended interactions via the sequence diagrams. Each iteration will have a permanent URL leading to our team website, allowing for a visual trace of each step in our project's life cycle.

We created four new sequence diagrams, a mixture coming from updated and new events/sequences (Movement, Energy, Timer and CoffeeAudio). These sequence diagrams demonstrate our architectural improvements from inherited to completed, complementing the existing diagram that are found in **Section D3c**.

A headless testing system for unit testing has also been implemented to fit our deliverables for assessment 2. Adding this functionality has allowed us to validate our architectural decisions, demonstrate professional practice for development and enabled test-driven development for any new features (e.g. new events were developed with 90% + test coverage before being integrated).

At the end of our architecture we have created a short modularity and component evolution assessment. This provides us with a metric for objective evidence of whether our architectural changes have had a positive impact. It also allows us to analyze and learn from the original team's architecture and limitations, meaning we can have relative and relevant benchmarking for our success or failures.

We also implemented various cohesion & coupling analysis through each different iteration, mentioning when and why this will improve our architectural design. Finally, we have added a distinction for the concrete and abstract architecture justification, mainly expanding on the original documents opening paragraphs in **Section D3a** and splitting them to be more clear

and concise. While also adding trade-off analysis and addressing the original lack of justification and separation of design features of the architecture.

Tools & Documentation:

PlantUML provided consistency with the inherited workflow. While we maintained diagrams in Google Docs and the website for team accessibility. The text-based nature of PlantUML syntax enabled clear documentation of changes between iterations in our architecture log.

GitHub was kept for architectural/codebase management due to the easy issue tracking for architectural changes (e.g. adding a class like AudioManager), as well as allowing for architectural review via pull requests and commit history providing verification of evolution.

iii. Method selection and planning

Software Engineering Methods:

For Assessment 2, our team retained the agile, Kanban-based iterative approach from Team 1's plan, including the use of GitHub Projects to track tasks and the integration of Kanban boards with Gantt charts. We kept these elements because they provided clear visual tracking, supported time management, and allowed us to monitor workflow effectively.

The main change we introduced was the implementation of SCRUM methodology. As our team did in assessment 1, we will structure work into weekly sprints, assigning tasks at the start of each sprint and reviewing progress at the end. We introduced key roles such as a Scrum Master and Product Owner. This change was made to improve flexibility, support incremental delivery, and allow rapid adaptation to changing priorities and emerging risks. Weekly meetings combined with daily WhatsApp communication ensure consistent progress without overloading our academic schedules. Overall, the combination of retained methods and new SCRUM practices allows us to maintain efficiency while increasing adaptability and iterative improvement.

Development and Collaboration tools:

For Assessment 2, we retained several effective tools from team 1's plan, including Git for version control, PlantUML for diagrams, libGDX for game development, IntelliJ IDEA and VS Code as IDEs, and Pixabay for copyright-free music. These tools were kept because our team believes they are the most effective for the project and the team is already familiar with them.

The changes to the development and collaboration tools consisted of the following. Documentation moved from GSuite to Google Docs to improve collaboration and ease of use for the team. Communication shifted from Discord to WhatsApp to suit the needs of team members. Photoshop replaced Paint.net/MyPaint for room and bitmap design due to its advanced features and team familiarity. Finally, GitHub Pages replaced Netlify for website hosting, simplifying deployment and linking directly to our GitHub repo. Our team also utilised this in assessment 1 so again it's familiar.

Team Organisation

For Assessment 2, we maintained a structure similar to team 1's plan, assigning primary leaders to each deliverable while the rest of the team acted as secondary members to assist, review work, and provide feedback. The key change was that secondary members are now more flexible and can contribute to other tasks beyond their primary focus, allowing us to adapt roles based on priorities. Team members also chose roles aligned with their strengths, such as game testing or user evaluation, while still supporting other deliverables when needed. This structure links closely with SCRUM, enabling flexible task allocation and collaborative engagement to ensure high-quality outcomes.

To stay organised, we continued using Kanban-style task tracking and Gantt charts, outlined in team 1's plan. We also considered Trello and GitHub Projects for Kanban boards, and PlantUML or GitHub Projects for Gantt charts.

The changes we made for Assessment 2 included using GitHub Projects for Kanban boards because it integrates directly with our codebase and collaboration environment, making task tracking more efficient. We also swapped to using PlantUML for Gantt charts, as it is simple to use, compatible with our documentation workflow in Google Docs, and allows standardised version control. This combination supports parallel work across sub-teams while maintaining clear visibility of progress.

Systematic Plan:

The changes we made to the plan for the game consist of the following

Negative Events Added: Frog that can eat the player, campus security confiscating found goose pieces, random power cuts, coffee slips, and speed/time penalties for overconsuming energy drinks.

Positive Events Added: Collectible coins and energy drinks from vending machines that give a speed boost.

Hidden Events Added: A rambling professor distracts the player, strong gusts of wind slow movement, and a hidden top hat for the goose.

These changes increase gameplay variety and challenge, adding both risks and rewards beyond Assessment 1.

We also restructured the project into Work Packages, each with sub-tasks, dependencies, and priorities. Examples include WP7 (Game Testing), WP8 (Game Implementation), and WP10 (User Evaluation). This is a refinement from team 1's plan, where tasks were less formally grouped, allowing for a more systematic and trackable plan.

We also introduced Milestones to mark key achievements. This provides clearer checkpoints and progress tracking for assessment 2.

Each deliverable from team 1's plan was updated to reflect changes in requirements for Assessment 2, for Example the change document, testing, and CI. Deliverables also now include multiple versions (interim and final), reflecting the iterative improvements made during Assessment 2.

Finally, a complete new set of weekly gantt charts have been created to reflect changes to the plan for Assessment 2. The reasoning behind these changes each week has been outlined on the Plan2 document. Weekly snapshots of these Gantt charts can be seen on our website.

iv. Risk assessment

Discussion:
The purpose of the risk assessment has been clearly outlined alongside the management process in their section D5a. They clearly outline the steps that are taken in the risk management process which emphasises identification, analysis, planning, action and monitoring. This hasn't been changed much as their process is very fitting to the approach we were already taking.

Register format:
Overall the format of the risk register is conventional and has been discussed in D5a . Small changes were made to the table format: decrease the font size by 1, resizing the width and shading the header for clarity. Any additional changes made or were
They have clear descriptions of the risk, its impact and ways to mitigate it. This has been slightly tweaked and expanded on in order to ensure that the mitigation strategies address ways to handle the risk once it occurs alongside the prevention strategies that they found as that is the ultimate goal of the risk assessment.

One thing that will be changed is their unclear display of who was assigned to manage each risk. They chose to assign vague groups as opposed to assigning members by name. This doesn't affect our extension of the project as our group members' names will be added on top of theirs where appropriate. This will allow it to fit the way in which our team intends to function in the second half of the project as everybody is aware of any risks that they are responsible for monitoring. However, having multiple people assigned to every risk will be continued as it is a good way to ensure somebody is always handling the risk.

Clear colour coding was assigned in their "impact level" and "likelihood" columns with a description and severity rating out of five. This won't be changed as it is a reasonable scale and the addition of colour alongside written descriptions makes it readable. The previous team also used appropriate risk categories: "product", "project", "product and project" and "business". These will not be altered as they are appropriately simplistic and encompass every potential risk and have been clearly defined in their discussion.

Identified risks:
Overall, the limited number of risks that they identified fit appropriately to the nature of the project. The risks identified were general and quite unspecific to any one deliverable.

Additional risks added that we had identified in our first part of the project and risks associated with part two. This included things such as the risks associated with inheriting a new team's code as well as those that come along with testing. Again, these risks will follow the same format as the previous team as their register accurately covers the explanation and mitigation process.