<u>**University of York**</u>

<u>**Department of Computer Science**</u>

SEPR - Assessment 3

Change Report

Team Craig

Thomas Burroughs

Huw Christianson

Joseph Frankish

Isaac Lowe

Beatrix Vincze

Suleman Zaki

Approach to Change Management

For Assessment 3, our team had to take over another SEPR group's project. Before each group presented their product during that week's SEPR practical, the group met to define the necessary criteria that the product must meet for us to successfully complete Assessment 3. Examples of our specific criteria were as follows:

- Concise, reachable (within Assessment 3-time frame) and traceable requirements
- Well documented, succinct and tested code abiding by well-structured concrete architecture.
- Precise documentation that meets the requirements and standards set by Assessment 1 and 2.
- Similar or well-justified approaches to testing and risk management.

Members of the team made notes during each of the team's product presentations and noted which sections of the discussed criteria were met. A meeting later in the week then commenced were team members discussed each team's product before an executive decision was made to select Team Geese Lightning's Zombie game.

On selecting a product to take over, the team began to research suitable approaches to change management to ensure a successful and risk-free transition. Our research revealed that IEEE's Standard for Software Configuration management [1] outlines a suitable process for configuration control (Section 3.2). The standard revealed that change management process needs to be put in place to identify the need for change; the analysis and evaluation of a change request; the approval of a request and finally the implementation of that change [1].

To follow the IEEE Standard the team decided to create a Change Table which took the form of a more relaxed Change Request Form as stated by Sommerville [2]. A tailored down version of the Change Request form was created to better suit our small team and project size. For example, unnecessary headings, such as decisions by SGA app were omitted as they were not relevant for our project. When a change was requested by a team member, the requester and the remainder of the team assessed the change against the project requirements and goals. This took the form of the Configuration Control Board (CCB) as suggested by the standard [1]. To better suit our small project and team, only one layer of request approval was needed. If the request was accepted, based on the team's evaluation of whether the change helped to meet the requirements of the project, the request was assigned an owner and implementation began. Ownership provided more structure to our strategy for change management as it allowed team members to establish who was doing what and helped to avoid miscommunications and confusion over collaborations. This prevented team members from working on the same changes independently. The table gives a description of each change with a Change ID for traceability, the effect of the resulting change and the status of that change along with an issue and completion date. This table helped the team with monitoring changes throughout the project and allowed for greater traceability of changes in documentation.

Due to the nature of the assessment (full project takeover), it was important that our change management process involved communication with Geese Lightning. Communication was established to help quickly resolve issues that arose because of confusion with their code, documentation or trouble with editing file types. This helped to reduce the number of risks that transpired because of changes and hence minimised time wasted mitigating these risks.

The team made a significant effort to appropriately manage changes made to Geese Lightning's source code. This was done as unmanaged changes to code may affect the game's previous functionality, creating errors. These errors may have been caused by the incorrect functioning of code, due to the overriding of functions/variables. To help avoid these risks, the team's developers spent a considerable amount of time becoming familiar with the architecture, coding standards and functionality of each section of code. When each developer completed a discussed change from the Change Table, testers and other team members were notified allowing them to update the concrete architecture, requirements and other documentation, as well as create new unit and black-box tests. Testers were also responsible for running Geese Lightning's Unit and black-box tests as a form of regression testing to ensure that changes did not negatively impact the game's previous functionality.

Changes to documentation were also included within the change table. Most changes to documentation where evaluated on whether to continue with Geese Lightning's proposed methods, plans and risk management or to revolve back to our own approaches from Assessment 1 and 2. If the benefits of Geese Lightning's approaches outweighed the benefits of our own familiarity they were often incorporated despite the uneasy transition.

Link to Change Table: https://teamcraigzombie.github.io/assets/downloads/ChangeTable3.pdf

Explanation and Justification of Changes to Testing Report

During the selection phase at the start of Assessment 3, the team reviewed the formal testing methods and approaches deployed by Team Geese Lightning. This review revealed that the previous team followed similar methods to us, with both teams using black-box testing and white-box testing in the form of automated Unit tests. We will continue with team Geese Lighting's test-driven development strategy as we found from Assessment 2 that creating tests before implementation began helped to structure our implementation process and prevented programmers from implementing potentially unnecessary features. Despite this, further black-box tests and Unit tests will be created after implementation to maximise testing coverage.

As Geese Lightening also used an adaption of the Scrum method we will continue with their approach to iterative testing. After each sprint testers will perform suitable black-box tests and automated unit tests on either partially or fully implemented code. Team Geese Lightning did not state whether Testers and programmers were sets of mutually exclusive groups. As a result, we have decided to introduce our methods for testing groups from Assessment 2. Programmers will be responsible for Unit testing their own code after each sprint and a group of testers, with no knowledge of the code of the game, will perform black-box tests. We believe this change is justified as it helps to avoid potential bias with black-box testing which may skew the 'actual outcome' and hence the pass/fail result and allows the testers to accurately replicate a potential game user which may help to raise more issues with the game's usability and functionality.

The tests undertaken by Geese Lightning had a large scope, seen by their IntelliJ coverage tests indicating the quality and proper functioning of their software. As mentioned within our Assessment 2 deliverables, we struggled to unit test all functionality since some functionality only works in the context of events fired by LibGDX and these events cannot be triggered on demand. The coverage tests implemented by Geese Lightning, as stated in the test report, help to identify exactly which functionality has not been covered by Unit tests and must be tested by black-box testing if suitable. Our team will therefore continue with Coverage tests, as stated by the previous team, as it helps to ensure the full scope of the project is covered and minimise potential bugs or defects in our additional code.

Geese Lightning's final approach to testing was the inclusion of requirements testing. We will continue with this approach as we agree that it helps to ensure that the software meets all the requirements and eases traceability. However, we believe that the format of the group's requirement testing makes it difficult to visually identify how many test cases cover each requirement. A traceability matrix will help to optimise testing by limiting the duplication of tests and preventing the testing of already tested functionality.

Due to time constraints, limited Unit tests were added to test new functionality, however, to ensure that all new code additions function correctly and are free of errors, additional black-box tests were added. 15 new black-box tests were added as can be seen in the updated black-box test evidence document. All 15 of these tests passed first time and no further actions were needed. Furthermore, tests previously written by Geese Lightning, which had failed due to the lack of implementation, have been updated to a pass status now the additional features have been implemented. The result of this can also be seen in the traceability matrix and requirement testing document. We believe that this testing method was sufficient for this assessment, as this method was successful in previous assessments, and all functionality added for this assessment performed as expected. In terms of the presentation of Black-box tests, little was changed as the previous team followed a similar style to us. The only addition was the inclusion of an 'Associated Requirement' column to give greater traceability. This addition was justified as it should hopefully allow team members to edit documentation with greater ease should test or requirements change.

Limited Unit tests needed to be added as Geese lightning's previous Units tests considered the necessary additions to implementation, meaning these tests could be used to test our new functionality. All new Unit tests added as part of this assessment passed first time. Finally, Geese Lightning's Unit tests were ran multiple times throughout the project at the ed of each sprint as described by our agile methodology, as a form of regression testing to ensure new functionality had not negatively impacted previous functionality from Assessment 2.
Updated Test Report: https://teamcraigzombie.github.io/assets/downloads/UpdatedTestReport3.pdf
Updated Black-box test: https://teamcraigzombie.github.io/assets/downloads/UpdatedBlackBoxTests3.pdf
Updated Unit tests: https://teamcraigzombie.github.io/assets/downloads/UpdateWhiteBoxTesting3.pdf
Updated Requirement tests https://teamcraigzombie.github.io/assets/downloads/UpdatedRequirementTesting3.pdf
Updated Traceability Matrix: https://teamcraigzombie.github.io/assets/downloads/UpdatedGanttChart3.pdf

Explanation and Justification of Changes to Method Selection and Planning

During the selection segment of the Assessment 3 period, the team reviewed Geese Lightning's Method Selection and Planning document. This revealed this team also used an adaption of the Scrum method. They followed the Scrum methodology more rigidly compared to us in both Assessment 1 and Assessment 2 and we have decided to follow their lead on this. We believe following the fundamentals of the Scrum methodology more strictly will help our team to be more efficient with producing better quality documentation at the end of each sprint. In Assessment 2 it became clear that team members did not abide by our methodology as closely as was necessary with team members choosing to extend deadlines to finish deliverables leading to rushing closer to the assessment deadline. The team have agreed that meeting three times a week, as suggested by Geese Lightning, was unrealistic due to the team's University and social schedules. We also believe that the timing between these meetings does not leave an ample amount of time for the results of changes to be observed or documents edited sufficiently. As a result, we have edited Team Geese's method of meeting twice a week, face-to-face, to discuss the current state of the project. We have also decided that one of these weekly meetings will be used to discuss transpiring risks or new risks. At this meeting, the Risk manager will update the risk assessment table and the team will discuss the implementation of the mitigation plan for each of the transpired risks. We believe that this change is justified for our team as we find that setbacks to our work, due to unanticipated risks, have a detrimental impact on our plan and evidence from previous assessment suggests that this leads to the lowering of the quality of work and increases group conflicts.

One final change to Geese Lightning's method selection was the reduction of sprints to one week. This change was made in accordance with the short assessment period for Assessment 3. Having four, one-week sprints instead of two, two-week sprints will lead to the greater turnaround of work by ensuring that work must be completed each week of the assessment. This should also help to minimise the impact of potential confusion, conflicts and difficulties as they can be addressed more regularly, and mitigation plans put into place more frequently.

Geese Lightning used similar communication and development tools to our team and due to this minimal changes were needed. We decided to not adopt their use of StarUML, despite its strong readability and its professional appearance, as we believe that the simplicity and familiarity of Lucid Chart outweighs StarUML's benefits. Its simplicity will allow future teams to better understand the architecture of our game should they take over the project. We also decided to swap their communication tools to our previous tools, WhatsApp and Google Hangouts, as our team was already familiar with these tools and have proven to work well for our previous assessments. We did not want to waste valuable time, which is already limited for this assessment, having to learn the new protocols for new communications tools which provided limited advantages, if any, over our previous tools.

Conveniently, Geese Lightning's team roles suited our team's previous roles and therefore little change was made to this section of the deliverable. Specific details were added to some of the team roles to accommodate changes made to the software engineering methodology. For example, the Risk Manager role was updated to account for the introduction of weekly risk meetings. These small details should help to avoid ambiguity and minimise confusion with each role's responsibility. Another change made to the team roles was the introduction of sub-roles and experts. We decided that Geese lightning roles, such as Test Leader, did not cover the full scope of the responsibilities or left the potential for confusion between members with the same role. We therefore decided to include sub-roles, such as Unit Tester and Black-box tester to further specify the responsibilities of each role and avoid unclear collaborations. The introduction of 'experts' also falls in line with the previous changes. Geese Lightning's plan had no clear explanation on how executive decisions are made for each deliverable, should disagreements arise. As a result, 'experts' were assigned for each deliverable who have the final say on decisions made for that deliverable should a group decision not be made. We believe that this change is justified as it helps diffuse conflicts and prevents delays caused by disputed decisions.

Finally, Geese Lightning also used a Gantt Chart to plan the remaining assessments and show task dependencies. Our team has decided to follow Geese Lightning's approach to constructing the Gantt chart by including a start date, end date and priority for each task as well as inclusions of suitable subtasks. Following these inclusions should hopefully allow the team to better abide by the initial plan and have a greater understanding of which deliverables need to be completed each week and how not completing by the given date may affect other dependent tasks.

Link to Updated Method/Plan: https://teamcraigzombie.github.io/assets/downloads/UpdatedMethod3.pdf
Link to Updated Gantt Chart: https://teamcraigzombie.github.io/assets/downloads/UpdatedGanttChart3.pdf

References

[1]- IEEE. IE*EE Standard for Configuration Management in Systems and Software Engineering*. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6170935

[2]- I. Sommerville. Software Engineering. Pearson, tenth ed., 2016, pp.