2. Change Report

Team 10 | YorKorsairs

Abdullah
Tom Burniston
Omer Gelli
Adam Kirby-Stewart
Sam Mabbott
Benjamin Stevenson

A. Summary

The initial part of this change management is about problem analysis and change specification. Before going in head first as a team, the project's problems and unimplemented requirements must be identified. In this section all proposals must be evaluated and only followed through with if they are validated (one way to do this is by putting clarifications to change requestors which can elicit more specific and improved changes).

When changes have been validated the analysis of their costs can be conducted. This section uses common sense and general knowledge (as well as traceability information) of the general requirements and the system. The benefit vs drawback weighting can be assessed, in as far as is possible before implementing (reasonable conjecture). This is the point for concrete decisions and whether to go ahead with the proposed change.

We have decided to implement changes which underpin how the game will eventually be finalised. These changes will be identified in the change report document in a managed and methodical way. Formally our approach to change management will be documented chronologically with sufficient detail that provides clear and justified reasoning as to the alterations, these changes will be added when discussed by team members and can be reviewed if ever the need arises. A mechanism that helps safeguards the project from errors is by having peer member oversight and routine meetings to discuss all parts of the project, including changes made. Another process that will be employed before commits and changes documented there will be highlights and adaptation notes that will be assessed and if agreed upon, they will be enacted; the original will not be altered until there is an accord but potential beneficial modification can still be noted and therefore not overlooked. This key aspect which is talked about above (highlighting/notes etc.) but not explicitly named is that of traceability. This method keeps track of relationships, sources, and the system design. This section allows for the analysis of the reasons for proposed changes and the impact of them (including source, requirements and design all of which aim to show the consequence of the actions of change in order to plan effectively). Ultimately this property reflects the ease of finding the related content which is essential when keeping track of the changes and impact made on the project.

Finally, change implementation is concerned with the modification of deliverables. This will engender the required documentation with the advantageous alterations. For effective procedure during this section organisation will be needed so that substantial and unnecessary rewriting does not occur (minimising the time wasted). Modularity here is also favourable because individual sections can be amended without interfering with other parts.

Overall, using this formal plan will allow for effective time management and will also mean team members will be able to trace all changes right from the start, all of which will drive the successful adoption, adaption and usage of the project to lead to an effectual end result.

B. Justification

B.1. Requirements

Team 3 used methodology from Sommerville's Requirements Engineering in order to display their requirements. This included dividing all requirements into functional and non-functional, as well as displaying this in a clear tabular format with distinct IDs for each requirement. In terms of formatting, this approach was close to our original game's.

We decided not to change the formatting due to our team's familiarity with Sommerville's original concept, which is beneficial as no additional research into the model was required. In addition, we appreciate the level of clarity and the ability to see each requirement laid out in an understandable and direct way that comes with this approach.

In terms of the listed requirements, we came to the conclusion that some of the requirements listed were either redundant, too vague, or would be too much hassle to implement without satisfying any of the base requirements of the game. The first requirement we changed was ID 'FR_CROSS_PLATFORM_GNU_LINUX'. We believed as the game is targeted to an audience who will play on the university computers in Windows, that to give support for Linux was unnecessary and didn't match any of the fundamental requirements. As a result, we decided to delete this requirement. A similar discussion was had for requirement 'FR_CROSS_PLATFORM_MAC', however it was deemed that this requirement was already implemented and required no further work.

The second requirement we addressed was 'FR_FRIENDLY_AI'. The wording of this requirement reads 'The game shall control the actions of friendly ships'. We believed this was too vague, as it implies the game has control over the player's ship in addition to the AI ships. We addressed this by changing the requirement to read 'The game shall control the actions of friendly AI ships'.

Requirement 'FR_FRIENDLY_INTERACT' talks about the 'interaction' of friendly ships and the player. However, there's no reference to what's classified as an 'interaction'. Furthermore, with the game version we were provided, the only interaction possible with friendly ships was collisions - given that this is implemented, we reworded the requirement to match and marked it complete.

The next issue we encountered is in regards to requirements 'FR_XP_TRACKING' and 'FR_XP_UPDATE', which detail how XP will be earned. We decided that 2 metrics used to determine how well a player was doing would be too complicated. Therefore, we decided it was best to remove any requirements regarding points, as the brief favours a XP based approach.

'FR_BOSS_UNLOCK_TRACKING' details how the game tracks quest progression as the game progresses. What it doesn't detail, however, is whether this should be explicitly displayed in the form of text on the screen, or implicitly and not revealed to the player. We believed for the sake of clarity for the player, it would be best to include some form of tracking system in which the player can view. To support this idea, we decided it was best to reword the requirement in order to support this change.

Finally, the requirement 'FR_BOSS_SPAWN' details how the game will spawn in a boss entity when the final objective condition is met. However, our team agreed that our final objective might not necessarily be based around boss combat. An example of a different final mission objective would be perhaps to purchase an item from the shop that costs more than most items. We decided to change the wording of this requirement's name and description to match our concept.
In regards to the other non-mentioned requirements, it was deemed that they still had a degree of relevance to the second leg of the project. It was decided to keep these

requirements as they were because of this, as without them the product will significantly differ from the intended final product.

In addition to altering the previous team's requirements, it was clear that we needed to add our own in order to support the new requirements.

https://aj141299.github.io/Kroojel.github.io/pdfs/Req1%20Updated.pdf

B.2. Architecture

Firstly it is important to mention that Team 3's architecture and implementation is designed in a way which makes changing and redesigning the architecture potentially difficult. This is predominantly down to the extensive and perhaps unnecessary use of class managers (which has consequently led to time consuming repairs of bugs as well as difficulty in changing the total in a structural way). Despite this, useful and effective changes have been made which have led to an improvement of the project.

The core components of the game will be kept (including the managers and current entities) after the numerous bug fixes including those errors in the AI system as well as collisions. These core functionalities have been kept because they both work in the correct manner and there would be little to nothing gained from making changes in these areas. However new UI elements will be added as well as new entities for the assessment 2 requirements.

For the abstract architecture diagram, the current state is acceptable as due to its nature of abstraction many of the categories are catch all and do take into consideration all new additions and alterations. Because of this, it is very justifiable to keep this part of the architecture as any change would not provide any benefit.

Also the visual categorisation in the Diagram of Entity and Component classes was over complicated and the layout was confused. With customer consultation, category lines have been removed and background colours with a legend have been added. This reduces the cross lines and makes the visual information more easily readable.

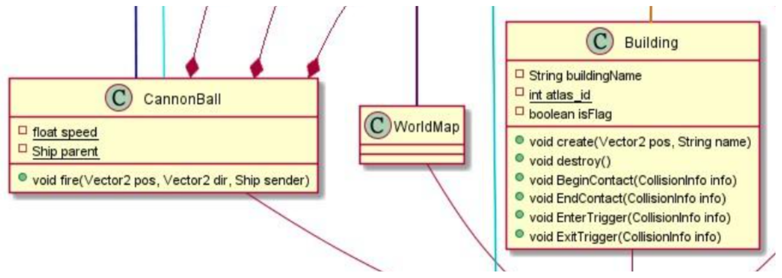
CannonBall
float speed
Ship parent
void fire(Vector2 pos, Vector2 dir, Ship sender)
WorldMap
Building
String buildingName
int atlas_id
boolean isFlag
void create(Vector2 pos, String name)
void destroy()
void BeginContact(CollisionInfo info)
void EndContact(CollisionInfo info)
void EnterTrigger(CollisionInfo info)
void ExitTrigger(CollisionInfo info)

Figure 1: Extract from architecture

One definite and main new change of architecture will be the use and destruction of the cannonballs thus changing the cannonBall class and the Building class (see architecture). Firstly the implementation of the cannonBall class is inferior to the correct standard of basic requirements. This is because the sprite isn't destroyed but is sent off screen when finished with which ultimately wastes computational power, so implementing this boolean is necessary, that is when the sprite should be destroyed a boolean value is set which removes the object. Aesthetically, the change will also be impactful as the user will feel the combat system is more true to life and has resilience i.e. cannonball will destroy on impact with a building and not continue unhindered by the amount of building it destroys. Resultantly this change to the architecture will bring benefits for the game and the player so can be quite easily identified as a justifiable change.

The addition of a shooting mechanism in the Building class (linking the new fire/shoot function with the cannonball class) is a requirement from assessment 1 that hasn't been implemented. This link between the Building and cannonBall classes will enable this prerequisite to be completed and ensure adequate functionality for the user to be able to play the game as per the conditions elicited in the customer meetings and brief. Within the Building class the different sprites of various buildings are included, with this we can allow for specific types to have the mechanism for shooting. Also, when implementing different levels of difficulty there is an option provided whereby we may also increase the number of sprites able to fire.

We intend to add a manager that will enable a prospective user to load and save the state of the game as per the requirements. We may not have introduced a new manager however, do to the current architecture of the game we have inherited, this new manager will not only fit in well with the general style of the code (making this section easily accessible and separate), it should also provide a natural flow for the use of the game with the proper mechanism put in place in this way.

As saving functionality and a load facility is required, a way to change the game state was needed. This change in architecture is in the miscellaneous classes, where the concrete structure was updated with a new class which, like the other page subclasses, inherits from the Pages superclass. But additionally the actual mechanism to store and retrieve the game data was added to the architecture's visual representation in the form of the void functions.

In a similar vein, an EventManager class has been added that will enable the assessment 2 requirements to be met. Specifically, the storm (weather), monster obstacles, powerups and the features and abilities they have in game were architecturally put in place to interact with existing managers so the whole system works coherently. A superclass was used to benefit the architecture allowing the holding of any subclass object derived from it, also the use of private, protected and public attributes/functions like in most other classes have been used appropriately and effectively.

As for the other managers, they will not change from an architectural point of view, although the implementation via the code will include refactors and new additions to improve the game, due to the nature of viewing architecture none of the additional manager changes will alter the diagrams or the reasons behind them.

Shop page and related functionality has been added (which is also required by the new assessment criteria). Much the same as the other pages (hence inheritance) this class will allow the user, once implemented, to upgrade in-game. The addition of labels in this class represents the shop attributes the player will choose from.

To finalise, the architecture that has been taken over has had its core retained, despite its often over complicated nature. With the diagrams we have eliminated duplicated methods by the use of superclasses, linked disconnected classes, removed tautologies between attributes and associations, added association to make connections clearer and by breaking the diagram into multiple readable sub-diagrams the layout has been made clear. Overall, refactoring existing structures, improving efficient transfer of visulased information and including new functions and classes have all taken place so that the project can be completed in an effective way.

(Updated material:
https://aj141299.github.io/Kroojel.github.io/pdfs/Arch1%20Updated.pdf).

B.3. Method Selection and Planning

The previous software engineering method was not designed to be flexible as they believed no requirements would be changed at all. This perhaps was not a benefit to the team as requirements were tweaked both by teams as the project progressed so the final outcome worked well and new requirements were introduced by the customer part way through. This rigid structure would not have easily accommodated this and so as a team we believed it is justified to change the plan in order to anticipate the changeable nature of the project as well as allow for a more iterative and review led approach.

The previous team's method and life cycle didn't have a tremendous amount of detail from which to expand upon thus it is justifiable given assessment 2's requirements to adapt the deliverable. We believe that working via a spiral hybrid lifecycle with elements of the waterfall model will minimise risks especially not going beyond time limits. When looking at individual aspects of this assessment the team will often be working in groups of two and whilst the whole project viewed globally will be agile and spiral centric, each group of two will seem to follow linear based plans this is because the main implementation has been completed and so changes can be made all at once without a large chance of new alterations being changed again. However, the spiral would allow new changes to be introduced without disrupting the plan (this being an advantage of this life cycle i.e. coping with new changes).

Furthermore, XP programming in an agile method will be used in programming. This is due to the previous team not implementing all of their requirements and not setting their priorities correctly. XP will allow the team to implement the previous requirements efficiently but also the new requirements which might need more than one iteration to be finished at a good standard. Main reasons for using XP are exactly those needed for our project especially during assessment 2. The dynamically changing software requirements; risks, caused by fixed timescales; small collaborative development teams; and appropriate tests. All of the aspects to XP described must be considered and expected and thus XP is how our code will be developed further.

However, the frequent team meetings will be continued due to the fact it will increase productivity and organisation. Within these meetings discussions can be conducted and development decisions can be made and agreed upon. These meetings will also provide each team member to communicate their progress and as part of the required organisation new assignments can be given out when necessary. It will likely be the case that the groups of two working together will periodically have meetings without the rest of the team, this means cooperation is more effective and all sections can move at pace without wasting time on other sections being completed separately.

While not preemptively assigning tasks may allow for adequate time to evaluate those most suitable for the task, this team can and will preallocate roles due to already knowing individual and collective strengths although there is scope for reassignment. We believe this is a justifiable advantageous way of organisation as it would streamline workflow without the need for ad hoc frameworks. Moreover, this should allow for fairness with equitable amounts of work for each member. Working hand-in-hand with a plan, team members can be held accountable for their productivity.

Tools used by the previous team will be in the main carried over, Zoom and Discord for meetings both separate group meetings, whole team meetings and those with the customer. Alongside in person the online communication will continue to allow for collaboration as well as streamlining the workflow as asynchronous and distributed actions can take place meaning less wasted time.

Unlike the previous team we will not be employing the Trello board system to organise tasks. This is because our team has the ability and organisation competency to follow and stick to prearranged timelines using the accessible Gantt chart. This is a justifiable change as it adds extra unnecessary complications so the extraction of this way of working would be beneficial. We agree that the visual aspect with verbal reinforcement of timelines is a good idea so this is an added reason for the Gantt chart implementation.

(See weekly new snapshots: https://aj141299.github.io/Kroojel.github.io/).

We have already used GitHub previously and will be carried over. This is easy to explain the reasoning behind the decision, as it is a universal and widely used system and changing away from this method of working would introduce complications with no obvious benefit.

Utilising IntelliJ as the IDE due to its ease of use along with the fact of prior experience with this tool. Again changing the whole process of implementing the code would be time-consuming. Similarly using libGDX game development framework will undoubtedly be used as changing this would alter the foundations of the game and involve avoidable and unnecessary need for learning thus wasting valuable time.

GitHub pages will continue to be used to develop the website as there is no realistically superior system for the scale and requirements for our project. Also, known procedures and workings of GitHub remain a small advantage over other hosting and development systems.

The team has elected to remove the previous team's reliance on the somewhat artificial rigid team roles format. It is better if a team can democratically agree on updates and decisions without the need for a meeting chair. A secretary may have been useful in assessment 1, for assessment 2 however it isn't required as much. This is because the groups of two team members will mainly work and communicate together and then periodically inform the team as a whole meaning each member only needs to know the specifics of their own designated section but not all parts. This ensures equitable work without any member overstretching their abilities but as there are two members in each section peer oversight is still built. The librarian role is justifiably almost redundant for our team, this is due to the fact that most documents are edited in the google drive thus are automatically saved and constantly accessible. The report editor seems rather redundant especially as it seems to be made from already defined roles of the chair and librarian (which is validated by the amalgamation of the previous team roles). Due to all the reasons stated once we evaluated the use of roles it was decided that on balance they were surplus and dispensable.

Instead the hierarchy of the group will now be flexible and is based on a democratic and fair structure. This way of working may lead some projects to have less strong and decisive decision making however, for our team it is and will continue to be effective and due to this way of working a mix of ideas and perspectives are introduced to the project but member specialisation means key tasks can be completed by those most competent in that area.

As for the systematic plan, the Gantt chart that must be produced, with an advanced level of detail, is more than enough to provide required material in which to plan effectively. The 'task breakdown table' has been removed from the project from now on due to readability issues and overcomplication. The Gantt chart provides a stable view of the development processes to accomplish this simple but effective planning method which also includes the property of visibility (see link above).

B.4. Risk Assessment and Mitigation

The overall presentation style was clear and concise. The tabular format that the risks were recorded in allowed for risks and their appropriate mitigation to be found easily. The only change that was made was the colour scheme of the table, as the pattern wasn't matching throughout the table. It was updated so that the pattern was consistent throughout. It was also required to renumber the IDs. It's clear that at some point there were a number of requirements that were taken out by the previous team and they forgot to update the scheme. We've changed a few of the numbers so that it's in numerical order.

Team 3 used a similar approach to our team's original risks. This design, however, incorporates features that weren't originally included into our design, but it was deemed to be beneficial. An example of this would be the ID's. This makes it far easier to reference individual risks as opposed to a written name. It was decided that the approach would require no changes, due to its similarity to our original design, whilst every heading labelled in the table having appropriate justification.

When assessing the descriptions of the risks, most of them were deemed accurate and genuine risks that might occur. One risk that was changed was R1, as AI interaction had already been implemented by the previous team. The only element of the AI to implement would be the shooting and health bar, but this wouldn't affect the AI's movement directly. As a result of this, we removed the requirement.

A second risk that was changed was R8. It was deemed that the description of this risk was too open ended and not specific enough to identify a problem. For example, if the game was experiencing issues regarding flickering, it would be dealt in the same way as lag. This is too broad, as often these problems will have 2 different solutions. The resulting change made was to divide this requirement into 3 smaller requirements to which each can be more detailed in each problem.

The main cause for change was the mitigation for some risks. An example of a changed mitigation would be R8. It was agreed that the trivial nature of this risk mitigation was neither appropriate nor helpful in the event that that risk became a reality. As it was previously mentioned R8 would be broken into 3 individual risks, the changes made to the mitigation were unique for each new risk. It was important to consider the most likely event to cause each error, and to research into an appropriate fix to resolve the issue.

The mitigations for R3 and R12 were also changed. It was thought that trying to work around the relevant risks by implementing a solution that doesn't match the given criteria would not meet the requirements. It was therefore determined that these mitigations need to be focussed more on solving the original problem rather than finding a work around.

The previous team seemed to have quite a strong reliance on one owner for the majority of tasks. It was decided that this approach has some benefits if one person is more familiar with the code than others. However, if something were to happen to that person then it would be hard to allocate whose responsibility it would be. Therefore, it was decided that various different owners would be more appropriate.

In regards to the non-mentioned risks, it was decided that it was best to keep them as they were, because they were still relevant to this leg of the project. They still cover many important aspects of the design process, which in the event of their removal would present risks with no planned for mitigation, which would inevitably lead to harm towards the project.