ENG1 Group 2 Marlin Studios
Part 2

Change Report

SKYE FULLER

JONATHAN HAYTER

ALEXANDER MIKHEEV

MARKS POLAKOVS

FREDERICK SAVILL

HAOWEN ZHENG

Briefly summarise the team's formal approaches to change management, including change management of all deliverables, documentation and code

Change Management of Deliverables

Firstly, our approach to change management in relation to the deliverables was to assign members of the team to different deliverables, setting deadlines as jobs were assigned. We used Zoom for calls and team meetings in order to get the team organised and to clearly set out what needed to be done. We began by analysing their deliverables as if they had been handed in under the mack scheme specified in part 1. We then began updating the Requirements deliverable, whilst updating Risk Assessment at the same time.This allowed us to add Associated Risks once we had finished updating the Requirements, and proved an effective way of updating both deliverables, in a quick manner. Throughout this process, we were sure to check our part 1 feedback frequently to ensure we did not make the same mistakes again this time round.

Change Management of Documentation

Our approach to change management when it came to the documentation of our project was to include both self-documentation and a game manual to explain aspects of the game which were not adequately explained in the tutorial. A GitHub Action was also set up in order to automatically generate Javadocs from our code. We made sure to only make changes to the previous documentation where it was necessary, mostly adding to it.

Change Management of Code

Finally, our approach to change management in relation to our code remained largely unchanged to our approach in part 1. We used GitHub to host our code, along with a selection of git tools to make commits and change branches. This allowed us to make changes to the codebase whilst ensuring other team members would have an up-to-date version of the code on their work computer. We had also set up a GitHub Actions to automatically run JUnit tests and provide feedback on said tests. We also made use of Pull Requests on GitHub, and got other members of the team to review changes before merging branches was allowed.

All updated Deliverables can be found on our website, or at the respective links:

- **Req2**
  - https://2020-eng1-team2.github.io/pdfs/Req2.pdf
- **Arch2**
  - https://2020-eng1-team2.github.io/pdfs/Arch2.pdf
- **Plan2**
  - https://2020-eng1-team2.github.io/pdfs/Plan2.pdf
- **Risk2**
  - https://2020-eng1-team2.github.io/pdfs/Risk2.pdf
- **Gantt Charts for Assessment 2**
  - https://2020-eng1-team2.github.io/#planning

Requirements Introduction:

The negotiation and elicitation of requirements in the introduction was to a large extent already explained well by the previous team, and thus only infrequent grammar/word choice changes were made; e.g. changed ‘3’ to ‘three’ because it is a written report.

Whilst the presentation format of the requirements is stated, the reason for using that particular format is omitted. We therefore added our team’s reason to their introduction.

The ENG1 lectures were a main source of research used by the team, and thus we decided to follow the tabular requirements format suggested in ENG1 Lecture 5.

Single Statement of Need:

Another change to the requirements was the addition of a single statement of need:
“The system should be an infiltrator game that is challenging but not impossible, and fun to play for first time users.”

Added/Edited User Requirements:

ID	Description
UR_PLAYER	The player can navigate the map, use power-ups, heal and arrest hostiles
UR_PAUSE	The user must be able to pause and continue the game whilst the game is running
UR_SAVE	The user must be able to save their current game and resume it after closing and relaunching the game

UR_PLAYER is a requirement that has been modified. The new cohort brief states; “Implement five special power ups that Auber can obtain”. Thus, the user requirement for the player must account for this, therefore ‘use power-ups’ has been added to its description (see underlined text).
The cohort brief also states; “allow players to save the state of the game at any point and resume a saved game later”. Therefore UR_PAUSE was added which facilitates pausing (saving the state) of the game, and also UR_SAVE which allows the saving and reloading of the current state of the game.

Added/Edited Functional System Requirements:

ID	Description	User Requirements
FR_DEMO	A demo mode feature should be implemented	UR_UX
FR_TUTORIAL	A tutorial explaining how the game is played, is presented.	UR_UX
FR_DISGUISE	Once hostiles are shot and exposed, they attempt to regain their disguise	UR_UX
FR_POWER_UPS	The player is able to use at least 5 distinct power ups	UR_PLAYER
FR_POWER_UPS_ABILITIES	Auber should have following abilities: Invincibility, Invisibility, SuperSpeed, InstaBeam, Vision	UR_PLAYER
FR_PLAYER_SPEED	Player’s movement speed should be the same as infiltrators	UR_PLAYER
FR_DIFFICULTY	Game should have ‘easy’ and ‘hard’ difficulty settings	UR_LOGIC
FR_RANDOM_SPAWN	Spawn locations for aliens and infiltrators should be random	UR_MAP
FR_HOSTILES	There are a maximum of three active hostiles on the map at any given point	UR_HOSTILES
FR_HOSTILES_ABILITIES	Hostiles should have following abilities: Blinding player, player slowdown, and confusion	UR_HOSTILES
FR_HOSTILES_SPECIAL	All Hostiles should have special abilities	UR_HOSTILES
FR_SYSTEM_DESTROY	Systems should be destroyed by attackers in 5 seconds	UR_LOGIC

FR_TUTORIAL and FR_DEMO are both new functional requirements, however were not added to meet the new specifications, but rather because these were already implemented features from the previous group's project, that were not included in their requirements. FR_DISGUISE is a new requirement that was added for the same reason.

FR_POWER_UPS is a new functional requirement that addresses specifically the "use power-ups" part of the UR_PLAYER description (see above). This functional requirement was added in order to meet the "special power ups" feature specified in the new product brief. The group then carefully negotiated which five special abilities are available to the Auber, and thus elicited the requirement FR_POWER_UPS_ABILITIES.

FR_PLAYER_SPEED is an existing requirement that has been modified, due to a design change made by the group. Previously the player's movement speed was faster than that of the infiltrators, however the group decided to reduce the player speed to be the same, because otherwise arrests could be made too easily.

The new cohort brief states; "Implement support for different levels of difficulty", and thus FR_DIFFICULTY has been added as a functional requirement. This feature allows the user to adjust the difficulty of the game to their own preference, thus meeting the brief.

FR_RANDOM_SPAWN is a modified functional requirement, renamed from "FR_ALIEN_SPAWN". Previously there were two separate requirements; FR_ALIEN_SPAWN and FR_HOSTILES_SPAWN which intended to specify random spawn locations for aliens and hostels, separately. However, FR_HOSTILES_SPAWN had an error in the description; instead of describing the random spawn for the hostiles, it repeated the random spawning for aliens. The group thus decided to delete FR_HOSTILES_SPAWN (see below) and update FR_ALIEN_SPAWN to include for both groups, as this was the simplest and most concise solution.

FR_HOSTILES was updated from 8 hostiles on the map to a maximum of 3 active hostiles on the map at any given time for the sake of clarity. This was the case when we took over the project.

FR_SYSTEM_DESTROY was changed to 5 seconds to help with the flow of the game. FR_HOSTILES_ABILITIES & FR_HOSTILES_SPECIAL were updated for the purpose of clarity.

Deleted Functional System Requirements:

ID	Description	User Requirements
FR_ARREST	Player can't arrest hostiles if they aren't attacking anything	UR_PLAYER
FR_PLAYER_TILES	Player should have free movement between tiles; should not snap to tiles	UR_PLAYER
FR_TELEPAD_DESTINATION	The player can teleport from any telepad to any other telepad	UR_PLAYER
FR_HOSTILES_SPAWN	Spawn locations for aliens should be random	UR_HOSTILES
FR_MAP	The map will be made of tiles on which the player and the AI can traverse, from tile to tile	UR_MAP

FR_ARREST was deleted for a number of reasons. This feature had not been implemented by the previous group, and on consideration, the team decided that implementing this condition would make the game too difficult.

The previous group had two requirements with a duplicated ID; FR_MAP, but with different descriptions. The first FR_MAP has been kept, however the FR_MAP seen in the above table has been removed on account of being too implementation specific. Splitting the map into tiles is an implementational decision; an alternative could be to perhaps import the map as a png file without tiles. This implementational decision should not be specified in the requirements; only the features of the map should. FR_PLAYER_TILES is an extended requirement from splitting the map into tiles (deleted FR_MAP), and therefore must also be removed on the same account.

FR_TELEPAD_DESTINATION has been removed because it was deemed redundant, as there already exists a requirement FR_TELEPORTER, which already specifies that the player will use ‘teleport pads (...) to teleport around the map’.

Finally, FR_HOSTILES_SPAWN has also been deleted because the updated FR_RANDOM_SPAWN (see above) specifies random spawning locations for both aliens and infiltrators.

Deleted Constraint:

ID	Description
CON_TOP_VIEW	Game must be from a top down view

CON_TOP_VIEW was deleted because a top down view was not a compulsory constraint, but rather a group decision. For instance, a 45 degree angle would also have been viable. The decision to have the game from a top down view, is specified in FR_TOP_VIEW.

Associated Risks:

The previous group did not include any associated risks with their requirements, therefore we added these in the format of a tabe. The addition did not cause the requirements document to exceed the page limit. The table is shown below. (* = All Requirements)

Requirement ID	Associated Risk	Description
UR_UX	R25	Players will not know how to play the game without the tutorial
UR_SAVE	R23	No warning can lead to overwriting a saved file and losing saved progress
NFR_SCALABLE	R24	Scalability issues with the window size, therefore the requirement is not met
FR_*	R1	Bugs in libraries can cause issues in implementing any functional requirement
FR_*	R3, R13	Slow group productivity, poor management, and missing team members hinders the progress of all functional requirements
NFR_ENJOYABLE	R6	Game cannot be enjoyable if it cannot be played
FR_*	R27	Team members may be unable to add new features
FR_MINIMAP	R26	Players will not know their way around the map unless a minimap helps guide them
FR_MENU	R24	Player may not be able to play the game as they can’t navigate the menus
FR_LOSE_CONDITION	R12	Misunderstanding of the brief meant that the team was initially under the impression that full loss of health meant losing the game. This was not the case.
FR_DEMO	R7	Addition of demo mode to the menu, caused an error that stopped the menu from functioning properly.
FR_DEMO, FR_TUTORIAL, FR_DISGUISE	R28	These requirements were added on realisation that they were not present in the previous group’s requirements, causing increased time spent on this section

Abstract and Concrete Architecture:

No changes were made to the abstract architecture of the project, despite some consideration as to whether we should have restructured it in an ECS architecture. It was eventually decided to keep the architecture as-is due to time constraints.

As for the concrete architecture, some changes were made, mostly additions in order to meet the new requirements set by Assessment 2.

Firstly, we felt that in order to properly implement difficulties, we should implement a new UI for the menu. PauseUI, MenuUI, and GameOverUI were classes added in order to render the user interface on different screens. MenuUI contains a button that allows the user to switch between difficulties.

In order to implement difficulties into the game, an enum class Difficulties was introduced with a method nextDifficulty() to cycle through the different difficulties added. This method was used by MenuUI to decide which button to show on the menu, and a value of its type was passed into a World constructor so that the necessary changes were applied when World got initialised.

Power ups were implemented into the game by creating a new class PowerUp which extended GameEntity. The new class had a constructor which took a value of type Abilities to determine which ability to grant Auber. Abilities was an enum class implemented for this purpose, and is also used in World.spawnBuff() to randomly select the power up to spawn.

Saving/Loading was implemented into the game by creating new methods for the World class saveGame(), read(), write(). In order to load a game, a new constructor for GameScreen was created to populate the world with GameEntity data from the save file. Also, a number of entity classes had zero-argument constructors added, to enable the serialisation system to create them. To do this we had to refactor several of them so that they could be instantiated without a reference to World (as it wouldn't exist during deserialisation). Also, many classes were modified to mark certain instance variables as *transient*, meaning they wouldn't be serialised when saving (for example there's no good reason to serialise sprites or textures).

Finally, we had to make a number of refactorings to enable unit testing. Much of the inherited code had con-mingled graphics and game logic code, which made it difficult to run the game in a headless environment. We made a number of changes to ensure various classes (mostly subclasses of GameEntity) don't attempt to load sprites when running headlessly[1].

Methods and Plans:

Edited Development and Collaboration Tools:

- **GitHub**
  Centralised place for all code – allows everyone in our team to view and contribute to the development of the project (source code). Branches and versioning were utilised heavily to ensure collaborative efforts were constructive and not destructive. GitHub Desktop was also used to minimise teaching necessary to those unfamiliar with the command line utility. *In the second part of the project, we added GitHub Actions to automatically run tests, calculate code coverage, and generate executable JARs and javadocs.*
- **Google Suite (Drive, Docs, Slides, Gmail, *Sheets*)**
  Google applications such as Google Docs, Drive and Gmail were used for file sharing and development of documentation. Allows for real time collaboration on documents meaning files could be stored in the cloud and accessed by any member of the team. *In the second part of the project, we continued to use Google applications, with the addition of Google sheets to create the Gantt chart for our systematic project plan, and Traceability Matrix for testing.*

The above are existing development tools included by the previous team, which we have modified to account for extra tools/features incorporated in the second part of the project by our group. This concerns the addition of GitHub actions and Google Sheets. Modifications are in italic.

Added Development and Collaboration Tools:

- **Paint.net**
  In part two of the project, the team used Paint.net to create new graphics, which were added to finish the game. Paint.net was the graphic design tool that the new team members were most experienced with, and thus was the clear choice.
- **Microsoft Powerpoint**
  Microsoft Powerpoint was the tool used to create the Gantt chart for the systematic project plan, in the first part of the project. Powerpoint provides functionality "add-ons" and Gantt chart templates.
- **Zoom**
  In part two of the project, the team used Zoom as the primary method of communication; the platform over which all meetings took place. Free zoom accounts were provided by the

[1] There are almost certainly cleaner ways of doing it than the way we did, but all of them would require substantial refactoring, which we couldn't do due to time constraints.

University and ENG1 practicals already took place over Zoom, therefore it was the obvious choice.

- **WhatsApp**
  In part two of the project, a group chat on WhatsApp was a secondary tool for communication. This was used mainly to schedule Zoom meetings, and stay connected on progress made between meetings.
- **IntelliJ IDE**
  In part two of the project, the team used IntelliJ IDE to develop the project code. IntelliJ was the IDE the team used in the first part of the project, and the recommended IDE in the LibGDX documentation. IntelliJ is easy to install, and provides efficiency and collaboration benefits the team made use of.
- **gdx-testing**
  In part two of the project, we added the GdxTestRunner from the open source gdx-testing project to emulate a headless environment as we had trouble setting it up ourselves (discussed in Testing2)
- **Coveralls and JaCoCo**
  In part two of the project, we added Coveralls and JaCoCo to track code coverage (discussed in Testing2).
- **GitHubPages + Jekyll**
  In part two of the project, we used GitHub Pages to host our website, in addition with Jekyll as it would build automatically and offered a wide range of themes and functionality

The above are new development tools added to the method selection. GdxTesting and the Coveralls were the tools used to complete the testing section in the second part of the project, and thus added.

Our group used different communication and graphic design tools as well as a different IDE to the previous group; having two different tools for the same purpose causes a conflict (e.g. including both Zoom and Discord for meetings). The solution is not to delete the tool used by the previous team, but rather to make it explicitly clear which respective tool was used by which respective team. Thus, to avoid tool conflicts, we include “In part two of the project”, before the description of any conflicting tool we have added.

PowerPoint on the other hand, is the only additional tool that was not used by our group, but rather by the previous team. Our team contacted the previous group, enquiring as to which tool they had used to create their Gantt chart. Our team incorporated a different tool for this purpose (Google Sheets as described earlier) but regardless PowerPoint is still a necessary addition.

Finally, we have extended the logo format to account for the new logos used by our team. The additional logos are shown below:

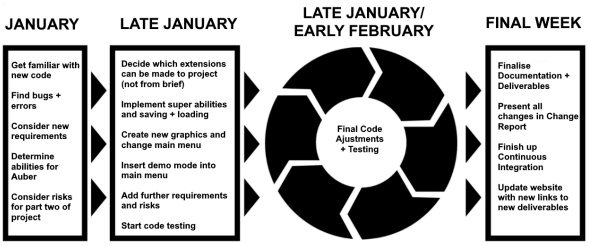
Added Considered Tools:

- **Microsoft PowerPoint**
  The team initially considered using PowerPoint to design the Gantt Chart for the project plan. However, our team found that Google Slides had similar Gantt features, but was easier to add to our shared drive folder.
- **Bootstrap CSS**
  Jekyll was the simpler framework compared to Bootstrap CSS, and therefore we opted for Jekyll due to time constraints.

As mentioned before, the previous team had used different tools for the same purpose than our group, and therefore have in some instances included a tool under their “Tools Considered” section, which was later used by the new team. We have discussed earlier why such conflicts are not deleted; therefore to this section only additional tools considered by our group have been added.

Software Engineering Methodologies:

Our team incorporated the exact same scrum methodology of software engineering as the previous group. This method is described well, clear, justified with advantages and thus no changes are made to this section. We added the same diagram included by the previous team; updated to part two of the project:



Team Organisation Approach:

The team organisation approach has been heavily reworded; language is now more formal and concise. Our group had a similar approach to the previous team, in the sense that we assigned members to their strongest section. However, we also included our organisational approach of allowing members to try new skills [1]. We also added a section on how sub-team leaders were determined, and what leadership entailed [2]. Finally, we included our method of enquiring after uncompleted assignments [3].

[1] Nonetheless we realise this is an educational project and therefore we still wanted to give members the opportunity to work on skills they are less familiar with (...) often paired less experienced members with a more experienced member in the same sub-team
[2] the most experienced member would be the leader of that section. The section leader would take the largest portion of the work and would be responsible for organising the sub-team, ensuring that deadlines were met.
[3] In part two of the project, through the WhatsApp group, team leaders would enquire and chase up on uncompleted assignments.

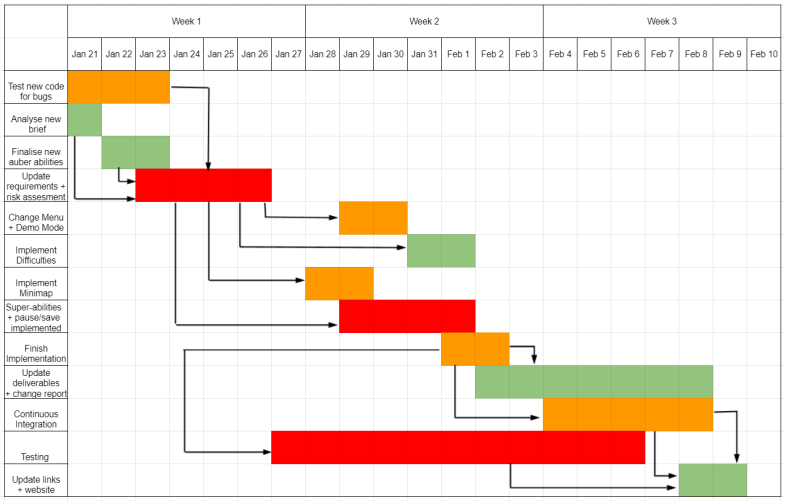
Finally, we included the breakdown of all sub-teams for the second part of the project:

- **Url2** (Website):
  - ***Marks (leader), JJ***
- **Change2** (Change Report):
  - Introduction: ***JJ (leader), Alex***
  - Requirements: ***Alex (leader), JJ***
  - Planning: ***Alex (leader), Freddie***
  - Risk Assessment: ***Alex (leader), JJ***
  - Architecture: ***JJ (leader), Marks***
- **Impl2** (Implementation):
  - New features: ***JJ (leader), Marks***
  - Changed Menu Screen: ***JJ (leader), Alex***
  - Minimap: ***JJ (leader), Marks***
- **Test2** (Testing):
  - ***Marks (leader), JJ***
- **CI2** (Continuous Integration):
  - ***Marks (leader), JJ***

Systematic Project Plan

The first paragraph written by the previous team has been entirely deleted as it is repeating the team organisation approach described earlier. Moreover, the rest of their original description has been reworded; it is now condensed and more formal. The first addition was the critical path for part two of the project with a justification provided; *Risks → Requirements → Testing → Implementation → Documentation.* Finally, a description of plan evolvement, as well as the final gantt chart were provided. Weekly snapshots showing these changes are on the website.

For the second part of the project, after our original plan was drafted, we made updates at the end of every week. In the original draft we had slightly underestimated the implementation task duration, and significantly underestimated the testing duration. In the final week we likewise, had slightly underestimated the duration for updating the documentation . The final draft is shown below, and the weekly snapshots are provided on the website.



Risk Assessment:

The first small change was the addition of “R” before the ID number for every risk in the table. This was added for clearer referencing to the risks in our Associated Risks table, and in other parts of the project.

The risk format introduction has been heavily reworded; the language is more formal, the text is concise and clearer to read. The only addition is an explicit description of the risk identification process (see italic text below) which had been previously omitted. The method described was the mechanism adopted by our team, as we cannot account for the approach taken by the former group.

The team drafted an early risk assessment before working on any parts of the project. With consideration, members devised potential risks before working on their assigned section, and would update the table with newly identified risks throughout the course of the project.

Added Risks:

ID	Type	Description	Likelihood	Impact	Mitigation	Owner
R22	Project	Lack of testing coverage for code	Low	Moderate	Researching good testing methods and implementing them, ensuring a high percentage of code being covered.	Marks
R23	Technical	Losing/ overwriting save files	Moderate	Moderate	User is able to make backups of their save file in case they accidentally overwrite it	Marks
R24	Technical	Window resizing issues	High	Very High	A hotkey (R-Ctrl) was implemented into the main menu to quickly start a game without the need of navigating menus	JJ
R25	Business	Users not understanding the objective or controls of the game	Low	High	A quick tutorial explaining all the controls, features and the objective is presented before every game is played.	Alex
R26	Business	Map navigation for first time users	Moderate	Low	A minimap will be implemented to guide the player around the map	JJ

R27	Project	Team members may have a hard time updating the codebase as they are unfamiliar with the techniques in use	Low	High	Looking through the code, and understanding it before any major changes were made	All
R28	Project	Missing or duplicated information in the previous group's deliverables	High	Low	Careful planning, organisation, and task delegation will ensure all adjustments are completed in a reasonable time. Thorough analysis of documents is required	Alex

The added risks seen in the table above, were initially drafted before changes were made to the project. For risks R23, R24 the mitigation column was later updated because the initial mitigation process had not been the mitigation method applied.

The majority of the added risks are related to the second part of the project; the risks that arise from working on unfamiliar deliverables/code, and the risks that arise from implementing the new functionality and tasks required. This is with the exception of risks R25, R26, which are customer/business related issues from the earlier stages of the project, that we thought necessary to include.

Deleted Risks:

ID	Type	Description	Likelihood	Impact	Mitigation	Owner
R5	Product	Final product quality affected by poor quality of libraries used	Low	High	Vet each library and tool used to ensure good quality and reliability	Connor
R9	People	Project progress suffering due to poor management	Moderate	High	Learning and practicing agile methodology, weekly meetings with all team members	Yi
R11	People	Different speed of working	Moderate	Low	Update the finished working and make plan each week	Yi
R20	Project/ Business	Problems delivering project to customer	Low	Very high	Ensure reliable internet connectivity to upload finished project, all team members should have access to project in case of one member's internet being down	James

Firstly, risk R5 relating to "poor libraries" is almost identical to risk R1 which specifies the issue of "bugs in libraries". Thus, R5 is also a redundant risk.

Risks R9 and R11 were removed on account of being unnecessary with the existence of R13; "slow progress due to poor productivity/management". All assignments and task deadlines were planned in advance therefore a "different speed of working" is the result of deadlines not being met; a consequence of either poor project management or poor productivity. R11 is therefore a redundant risk. Likewise, the issue of poor management is already mentioned in R13, so risk R9 is not needed.

Finally, risk R20 was deleted because of ambiguity in the description; "Problems delivering project to customer". This description can be interpreted in one of two ways; either the risk of not delivering the project because it is unfinished, or the risk of the game not functioning correctly on a customer's device. Risks R8 and R6 already address both of these cases, therefore risk R20 is unneeded.