Module	ENG1
Year	20/21
Team	33 (Short Circuits)
Members	Jack Lord, Neo Metcalfe, Sam Rodgers, Mohammad Abdullah, Qi Tang
Deliverable	Change report

A.

Our group decided to use ADKAR change management as our team's formal approach. The ADKAR model has five steps: Awareness, Desire, Knowledge, Ability and Reinforcement. *(prosci, no date)* Actually, this method is used for employers to organize their employees but I think it also works for our team.

The first step is Awareness means identifying change requests. We created a change requests table, indicating: ID, description, improving and owner. All members gathered and brainstormed to consider the change requests in different aspects like code bugs, deliverables and so on. All possible requests should be listed to make a good project.

The second step is desire means encouraging members to change. For each change request, the proposer should provide the explanation for why it should be changed to persuade all members. This action can let the team members understand the need for change and be active to support the change.

The next step is knowledge means how to change. First, we created the process flowcharts to make sure each member is clear with this process and also set reasonable targets. Each team member needs to learn new knowledge and receive new training for the part he is responsible for.

The fourth step is ability means converting knowledge into practice. Once the project started, we monitored each other and provided constructive feedback to keep the project on schedule.

The final step is reinforcement means sustain the change. We learned from the early mistakes and gave up the old methods to make sure not repeat the same mistakes. Also keep talking about the processing and celebrate the phased success to be clear with how the change works and unfinished work. *(Matern. L, 2020)*

B.

I. Requirements

The changes for assessment 2:

1. **The requirements are not required or less important that can be deleted.**
   - **FR_LOST**(not required)
   - **FR_EXIST**(less important)
   - **UR_BASIC**(less important)
   - **FR_PLAYER**(less important)
   - **FR_START**(less important)
   - **FR_WIN**(not required)

2. **The requirements are not fit the requirement**
   - **FR_HEAL**(*The player can teleport to the infirmary to heal anytime*) replace **FR_JUDGE_HEALTH**(*The system shall need to set a minimum health before teleport to the infirmary*) because Auber can teleport to the infirmary anytime regardless the amount of the health.

3. **Confusing functional and non-functional requirements.**
   - **FR_INSTRUCTIONS** replaces **NFR_INSTRUCTIONS**
   - **FR_SETTINGS** replaces **NFR_SETTINGS**
   - **FR_DEMO** replaces **NFR_DEMO**

4. **The description of the requirement is not entirely consistent with the ID of the requirement.**
   - **FR_AUBER_INFILITRATOR_COMMISION** replaces **FR_ARREST_COLLISION**
   - **FR_INFILITRATORS_STATION_COLLISION** replaces **FR_DESTORY_COLLISION**

5. **Adding some necessary user requirements, non-functional requirements.**

- **User requirements**

ID	Description	Priority
UR_ROOM_NUMBERS	There should be at least 4 types of rooms	Shall
UR_CHARACTERS	There are 8 hostiles and one player	Shall

- **Non-Functional requirements**

ID	Description	User requirements
NFR_OPERABILITY	The game only has one difficulty	UR_UX
NFR_LOAD	The player can load in a short time	UR_UX

Updated Link

II. Abstract and concrete architecture

The first team used an entity- component system as their abstract architecture to map different parts of their classes, which allows users to see which entities share the same components. This is useful for the game at hand as many classes use their own versions of the components whilst they may also affect each other. This architecture is especially useful when you have multiple entities that contain the same components.

The concrete architecture also followed a similar pattern with a class- entity diagram. This allows each entity to be displayed as a class that will be implemented into the game. It also shows how certain entities can be used as child classes when they are similar to other entities but have certain differences which may be through their attributes or through the way they need their information to be accessed or edited.

We chose to extend on this idea with the new deliverables as their current architecture blended well with our visions to continue with their work. This meant extending the current diagrams to include the difficulty, Auber powers and save/load function.

The powers are a child class of Auber as they require access to the Auber's attributes. This allows any powers to be tracked and linked to the sole instance created for Auber, and the creation of the powers instance creates a 1 to 1 link, meaning they won't get mixed up. For example, Infiltrators' powers being referenced with the wrong infiltrator.

Updated Link

III. Methods and plans

Software development method

The software development method we decided to use is Agile. Our team is familiar with this method because we took this in assignment1. Through the first trying teamwork, our team members have a basic understanding of each member has different strengths and weaknesses that can let team members be assigned tasks more efficiently.

In assignment 2, our team still would be working closely with the customer and communicating with them in customer meetings. Working with an Agile methodology allowed us to reflect, as a team, on the current software we had, and the improvements and changes we could make to fit the customer's expectations.

We implemented an Agile approach by utilising a basic scrum methodology. Performance, progress, accomplishments, problems and risks were then discussed during our weekly meetings, which allowed for us to reflect on the difficulties each team member encountered, and allowed for other team members to provide solutions to these difficulties.

Software development tools

- **Jira**

The project tracking tool we decided to use is jira. Jira is a project managing tool that focuses on agile software engineering methodologies. This project managing tool was very useful for our team members since it allowed us to easily to organise and lay out the tasks that had to be completed for the project, whilst also allowing us to display which issues were in progress, and ones that had been completed.

- **GitHub**

Our team decided to use Github for version control and collaboration. Github is more like a Google drive for software projects to let us share code with team members and edit the most up to date version of Auber.

- **Zoom/ Messenger**

The communication tools that we agreed on were Zoom, Messenger and Discord. We used Messenger to schedule meetings and held the meeting on Zoom twice a week to make sure each member kept on the schedule and discussed the solutions for the difficulties.

- **Google Drive**

We used Google Drive to organize our documentations. The deliverables, resources, graphics and project management documents were all stored within a google drive dedicated to our team, and allowed for us all to easily access and modify them.

- **Intellij idea/ Visual Studio Code**

The development tools we used are intellij idea and vs code. The main programming language we used is Java as the intellij idea is one of the best development tools for Java.

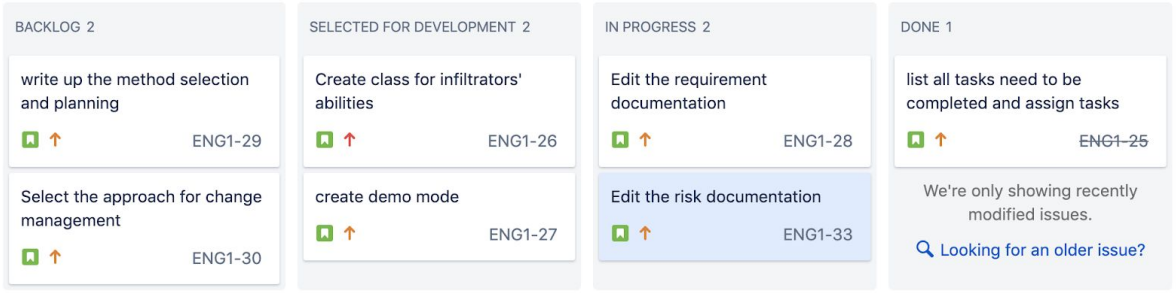
Project plan

For assignment 2, we have nearly 11 weeks to complete. We created a plan in week 2 of our project. This plan consists of 4 columns, these are: Backlog, Selected for Development, In Progress and Done.

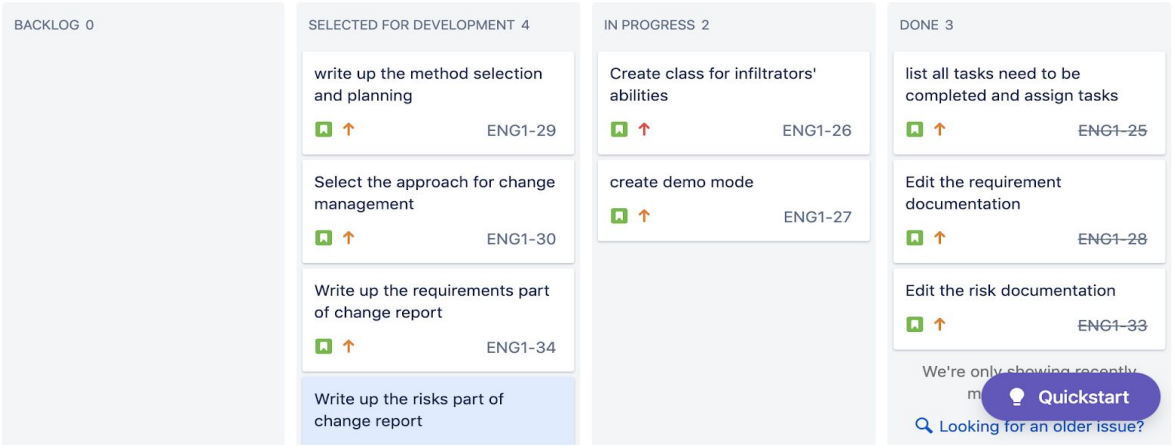
Furthermore, the plan we have created contains priorities for each task, these are signified by the coloured arrow in the bottom left corner of the specific issue. A green arrow indicates low priority tasks, while the orange arrow indicates a medium priority and a red arrow indicates a high one.

Week 2
We assigned each team member tasks and listed all tasks we needed to complete.

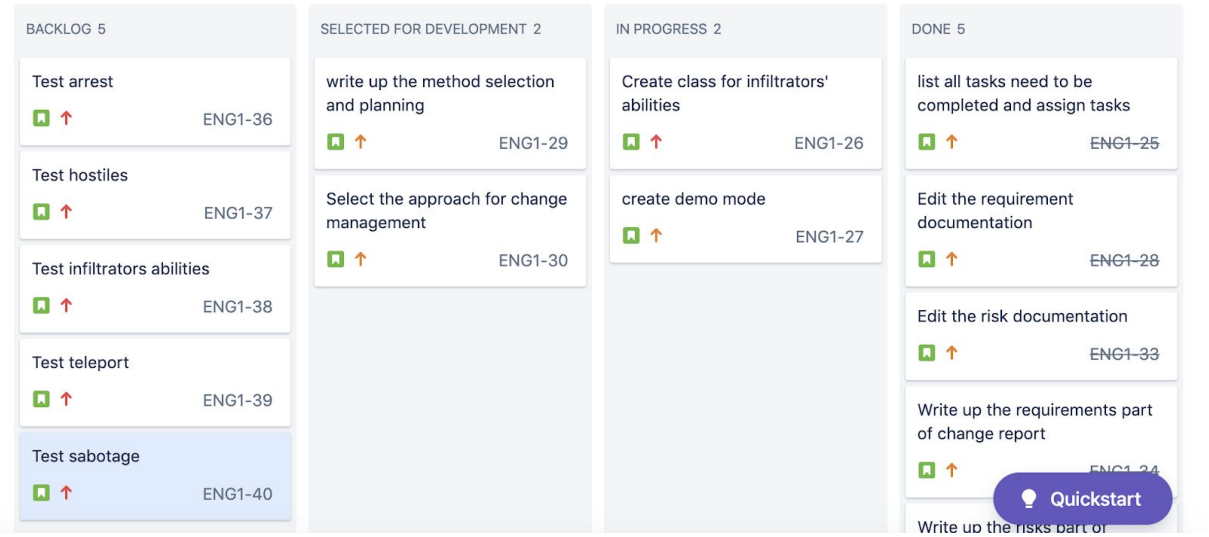
Week 3
In week3, we added missing requirements and risks and also deleted the unnecessaries. According to Impl1.pdf, we listed the incomplete tasks(e.g. the ability for infiltrators and demo code).



Week 4
In week 4, we finished the editing of the requirement and risk documentations and according to the new documentations we started to write the change report. In this week, we still focused on completing the incomplete requirements.



Week 5-8
We finished the requirement and risk parts of the change report and prepared to test the code.



Week 9-10

Most of our project was completed in week10. We focused on writing the reports and summarised the approaches.

Week 11

Added links to all of our deliverables on the website. Finished the project.

IV. Risk assessment and mitigation

The changes for assessment 2:

1. **Removing the table in risk identification and planning part because it has the duplicate contents in the risk register.**

2. **Adding the introduction of how to identify the risks.**

    - *Identifying risks is the first step of risk management. The objective of risk identification is to identify all possible risks. We followed Rebecca Webb's way [1] to identify the risks as all of us have limited experience. Engineer projects have common risk categories like technical, financial, staff, timing, and so on. We gathered all members and brainstormed to consider the detailed risks based on these categories. All possible risks should be listed to reduce the possibility of overlooking all possible risks. And then consolidated some overlapping risks which can be mitigated similarly and removed the risks with very low possible risks or have minor consequences.*

3. **Providing the table of the degree of the risk analysing.**
    - The scale we used is ranked from low to high. The table below is the degree of the risk analysing.

Likelihood	Severity
High	High
Medium	Medium
Low	Low

4. **In the register table, the previous IDs are too simple and should be detailed.**
    - **R_MISCONFIGURATION** replaces **R1**
      (note: just one example not the whole)

5. **Adding some risks with high likelihood or severity.**

Id	Type	Description	Likelihood	Severity	Mitigation	Owner
R_UNSATISFY	Product	The customer is not satisfied with the final product that needs to redesign. Result: spend longer on test, design and implementation.	**M**	**H**	Feedback and adjust timely during the course of the project	Everyone
R_UNFAMILIAR	Product and Project	Unfamiliar with the development tools leads to taking longer than expected.	**H**	**H**	Use development tools that familiar as far as possible	Everyone
R_TEAM_STRUCTURE	PRODUCT	An inefficient project team structure reduces productivity.	**M**	**H**	Assign tasks according to everyone's advantages	Everyone

Updated Link

Reference

1. Prosci (no date). *The prosci ADKAR model.* [ONLINE]. Available at: https://www.prosci.com/adkar [accessed on 3rd February].

2. Matern. L(2020). *How to take the ADKAR change model from theory to practice.* Available at: https://www.howspace.com/resources/how-to-take-the-adkar-model-from-theory-to-practice [accessed on 4th february].