Change Report

Team CASEUS

(Team 7)
Eliot Sheehan
Matthew Turner
Daniel Atkinson
Hannah Pope
Mayan Lamont
Divyansh Pandey

Software Engineering Methods

In order to meet the constraints of the stakeholder's outlined requirements, a methodology that would assist this development in a beneficial way was needed. When deliberating on which methodology should be chosen, the following points were considered: time constraint of the project and its size, the requirements that must be fulfilled and the need for flexibility. This narrowed our decision to three types of methods: Plan-driven, Soft Systems and Agile methods.

Plan-driven methods provide a structured way to create software by predicting and anticipating all future features that may be implemented. Planning of the project can be promoted by providing comprehensive explanations of tasks, workflows and responsibilities [1]. There is an emphasis on continued review and risk management analysis. This detailed level of planning requires the project to be developed in a stable environment and is mainly used in projects that have a long working period [4]. Plan-driven methods that were researched include Capability Maturity Model and Team Software Process (TSP). The focus of TSP is to create an environment that supports disciplined work and maintaining a self-directed team, that would help to establish an effective management and structure to the project. However, with TSP requiring the need for experience of managerial skills and Plan Driven Methodology not catering for the project's need for flexibility, the use of Plan-Driven methods was not chosen [7].

Soft System methods look to break down complex problems in order to truly understand the problem outlined by the stakeholder, and use this to develop multiple conceptual models that can be compared to the original real world problem [2]. The time taken to understand this problem and discuss potential solutions is a lengthy procedure that takes weeks. The model has limitations in comparing the conceptual model to the real world problem and it does not provide a framework for how the design should look, instead focusing on the search process for the best solution [9]. The time-constraint and potential need to integrate other Models meant this plan was not a rewarding model to use for this project [6].

Agile methodology focuses on the "concept of ongoing waves or sprints of project planning and execution" enabling the project team to adapt the design, scope and execution of the project deliverable [5]. Code is distributed in minor releases with short periods, where the project team works together with the Customer in close cooperation. The approach is easy to understand, and its core principles were aligned to the scope of the project and the stakeholders requirements. The focus on frequent software delivery allows for the development team to produce the product in a shorter time frame, when compared to other methodologies, and its allowability to iterate and innovate on functionality of the system would help to maintain a higher motivation within each individual of the team. The scope of the project and its delivery fell in-line with this methodology and was decided on.

With the agile methodology chosen, we decided upon iterative development as the framework because we feel it best suits this project. Since many of the tasks needed to be completed in a specific order, iterative development best suits this as it allows us to make an initial prototype based on our first ideas and then iteratively improve it until it reaches the desired state. This is particularly important for the second assessment because the majority of the new features need to be implemented in a specific order so that they will work correctly.

Development and Collaboration Tools

Documentation tool and Deliverable Repository: Google Drive acts as the central location to store all the documents related to the project. It allows for simultaneous work on single or multiple documents. With support for different document types such as Docs, Sheets, Slides and third-party tools, such as UML diagrams, it stood out over alternatives. These alternatives included Microsoft Word which was not chosen as it only allowed for asynchronous collaboration and the transferring of this document type required additional services. Dropbox and OneDrive (with its integration of further Office tools) were noted but their feature sets are not as vast as those offered by Google Drive. Confluence was also researched (which integrates with Jira) but its cost was too great.

Communication tool: Discord is used to communicate with other team members via voice and text chats. We found that this works well for us as all team members are already familiar with the interface. Furthermore, Discord pushes notifications to ensure that every team member is kept informed about any issues or queries surrounding the project. In terms of voice communication, Disocrd provides voice channels and the ability to share the current user's screen to other users, which is useful when tackling tasks collaboratively.

Version Control tool: GitHub will be used for the version control tool for the project. It works as an online repository and allows us to collaborate on code development. The feature of branching will prevent unnecessary confusion and damage to the project, when team members are coding at the same time. As GitHub is a widely used VC tool and offered large amounts of integration with other software, it was chosen.

UML diagram tool: Lucidchart will be used to create the architecture diagrams for the game. It allows for early discussions of basic entities to be conceptualised and the relationships between them, before the implementation process is started. It allows for flexibility when adding or removing functionality in later iterations of the diagram. The tool is easy to learn, free and allows us to work on the UML diagram synchronously with integration with Google Docs.

Implementation Tool: Java using the **LibGDX Module** will be used as Java is a customer requirement stated in the Product Brief. LibGDX will be used as the game library, as the documentation and online resources about it are vast (including tutorials). Additionally, there was a consensus within Cohort 1 and 2 over its use, allowing for easy transfer of projects in Assessment 2. Alternatives considered were Slick2d and Joge (Java OpenGL Game Engine).

Team Organisation

Organisation – In terms of managing tasks, we have created a Gantt chart which lists all of the tasks that need to be completed in order to finish the project. Each task is assigned a specific length of time and will be assigned to a team member in the last meeting before the task is scheduled to be started.

In terms of meetings, we are planning to have two meetings a week which all team members will attend. If any additional meetings are required due to issues needing to be resolved, the team will organise one using Discord. Meeting notes will be recorded on the home page of the website, where we will summarise the discussions that take place during each meeting.

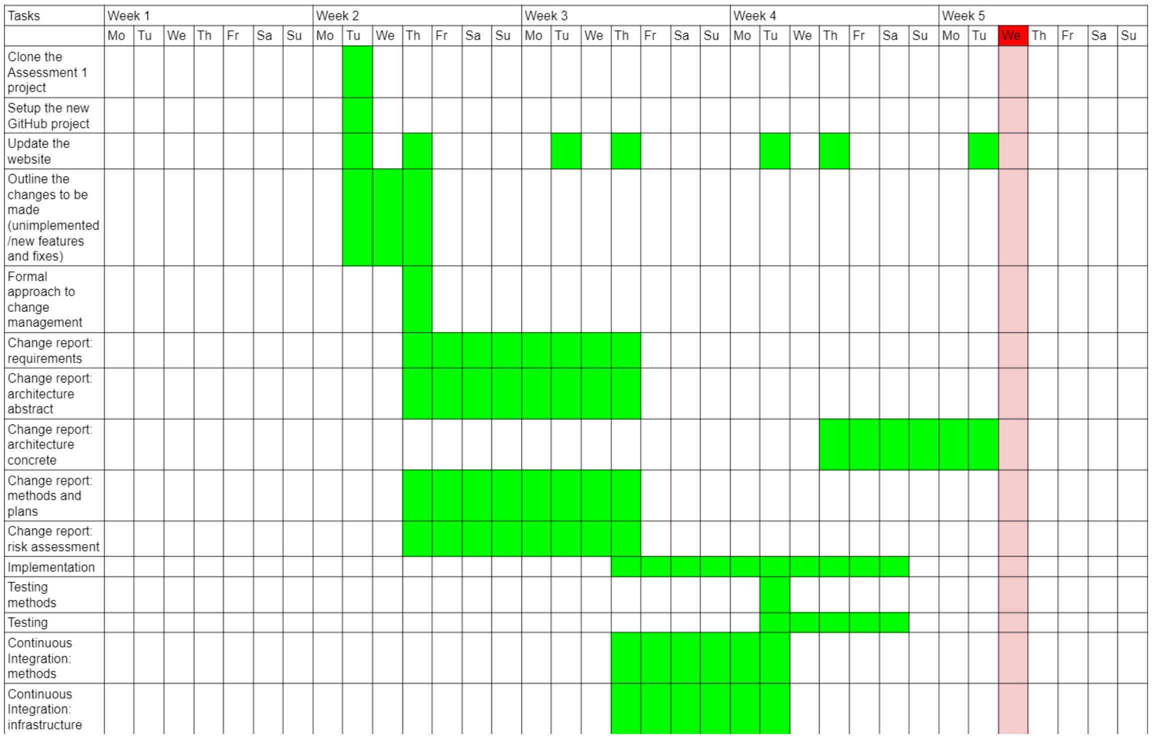
Roles and Tasks Allocation – Tasks are allocated based on the preference of each team member, meaning that the members that feel most comfortable with programming will be assigned to do so and the same goes for documentation.

Project Plan

Task priority was decided based on:

- The time taken to compete.
- The number of marks it was worth and the number of people it requires, if related to **documentation.**
- The impact of the task not being completed and the priority value assigned to it in the Requirements deliverable, if related to **coding.**

Using this, an initial Gantt chart was constructed to display the various parts of the project and highlight any anticipated dependencies between tasks. An online tool [10] was used as this allowed for synchronous collaboration between team members and allowed for adjustments to be made in response to volatile requirements. This allows for a clear understanding of the implication of any delays in the project.



Tasks	Week 1							Week 2							Week 3							Week 4							Week 5						
	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
Clone the Assessment 1 project									🟩																										
Setup the new GitHub project									🟩																										
Update the website									🟩		🟩					🟩		🟩					🟩		🟩					🟩					
Outline the changes to be made (unimplemented /new features and fixes)									🟩	🟩	🟩																								
Formal approach to change management											🟩																								
Change report: requirements											🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Change report: architecture abstract											🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Change report: architecture concrete																									🟩	🟩	🟩	🟩	🟩	🟩					
Change report: methods and plans											🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Change report: risk assessment											🟩	🟩	🟩	🟩	🟩	🟩	🟩																		
Implementation																		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩								
Testing methods																							🟩												
Testing																							🟩	🟩	🟩	🟩	🟩								
Continuous Integration: methods																		🟩	🟩	🟩	🟩	🟩	🟩												
Continuous Integration: infrastructure																		🟩	🟩	🟩	🟩	🟩	🟩												

Above is the Gantt chart that we created to help us plan and manage the second assessment. The team is planning to meet every Tuesday and Thursday in order to discuss the progress that everyone has made and clear up any bigger issues. As can be seen on the left-hand side, we split up the various deliverables that we have to produce into several tasks, therefore making it easier for us to manage how long each task should take to complete. The amount of time assigned to each task was based on the priority, as well as the estimated length it would take to complete. For example, some tasks are allocated a week to complete (such as Change report: requirements) and other tasks are allocated a single day, as we believe they can be completed by the whole team during one of our meetings.

References

[1] H. Svensson, *Developing support for agile and plan-driven methods*, PhD dissertation, KTH, Stockholm, 2005, p.26. [Accessed: 23/10/2020]

[2] S. Burge, “An Overview of the Soft Systems Methodology”, *System Thinking: Approaches and Methodologies*, 2015, pp.1-14. [Accessed: 23/10/2020]

[3] *What is Scrum?*, Scrum.org. [Online]. Available at: https://www.scrum.org/resources/what-is-scrum [Accessed: 15/10/2020].

[4] *[No title].* [Online]. Available at: http://www.se.rit.edu/~se456/slides/PlanDrivenMethodologies.pdf. [Accessed: 18/10/2020]

[5] *[No title]*. [Online]. Available at: https://www.wrike.com/project-management-guide/agile-methodology-basics/. [Accessed: 05 Nov. 2020].

[6] *Soft Systems Methodology*. [Online]. Available at: https://www.umsl.edu/~sauterv/analysis/F2015/Soft%20Systems%20Methodology.html.htm. [Accessed: 02 Nov. 2019].

[7] *Scrum- what it is, how it works, and why it’s awesome*, Atlassian.com [Online]. Available at: https://www.atlassian.com/agile/scrum [Accessed: 26/10/2020]

[8] A. Özerten, (2016, December.26), *Benefits and Challenges of Self-directed Teams in Software Projects*, Commencis. [Online]. Available: https://medium.com/commencis/benefits-and-challenges-of-self-directed-teams-in-software-projects-fe8df2d842e8. [Accessed: 28 Oct. 2020].

[9] (2018, September.11), *Soft Systems Methodology (SSM)*. [Online]. Available: https://www.toolshero.com/problem-solving/soft-systems-methodology-ssm/. [Accessed: 24 Oct. 2020].

[10] *Online Gantt Chart Software & Project Planning Tool*. [Online]. Available: https://www.teamgantt.com/ . [Accessed: 15 Oct. 2020].

[11] U. Eriksson, (2015, July.15), *Agile Software Development and Requirements*. [Online]. Available: https://reqtest.com/agile-blog/requirements-in-agile-software-development/ . [Accessed: 1 Nov. 2020].