Method Selection and Planning

Team Mallard

Methodology

Software Engineering Method

We will be employing the Scrum Agile method for our project. Scrum is an iterative method, in which development is split into iterations or "sprints". The process begins with the drawing up of a product backlog - a list of implementation tasks needed to create an end product which satisfies all requirements. On the beginning of each iteration, the team decides which tasks from the product backlog to implement - this list forms the sprint backlog. During a sprint, the team meets regularly to review progress towards completing the tasks on the sprint backlog. At the end of each sprint cycle, the product is presented to the client to further ensure that it meets the requirements already set out, and to allow the client to change the requirements if needed. We believe the following characteristics of Scrum make it a good fit for this project:

- Scrum allows us to continually review and improve how the team works. It is very important that each team member is being utilised in the best possible way at all times, especially due to the large amount of work required on a relatively short timescale. By regularly reviewing and discussing our progress we can more efficiently determine which tasks are suited to each team member and change them around if necessary.
- Due to the iterative nature of Scrum, we are able to easily adapt to most changes in system requirements. After each iteration, any outstanding changes of requirements are analysed, and a list of tasks necessary to implement these changes is drawn up. These tasks are then prioritised and inserted into (and existing tasks possibly removed from) the product backlog. This offers a huge advantage over other software development lifecycles i.e. waterfall, where there is a possibility that any consideration of requirements changes would have to be put on hold until after implementation is complete.
- Using Scrum allows us to more easily mitigate risks associated with team member illness/inability to work. A team member can pick up tasks from another team member's backlog when said member is ill or otherwise unable to work. This allows us to minimise the impact that an ill team member has on project schedule.
- Testing is an integral part of Scrum. It takes place after the implementation of each backlog item, which ensures that features are of high quality before being integrated into the product. It also allows us to actively check that requirements associated with backlog tasks are being met. This is in contrast to other software development lifecycles like waterfall and spiral where testing only takes place after implementation is complete. By testing features as they are implemented (*do it right first time*), we can avoid potentially many cycles of time-consuming testing and re-implementation of large parts of the product.

We also considered the following approaches before we settled on Scrum:

- Waterfall lifecycle: simple to understand and follow, works well for projects with concrete initial requirements. However, it is very difficult to manage with changing requirements, and also no testing occurs until after implementation, which may lead to us running out of time with an end product that doesn’t work/meet the requirements. Due the difficulty of accommodating the inevitable change of requirements, as well as the relative ambiguity of our initial requirements, we decided Waterfall was not a good fit for our project.
- Spiral lifecycle: overcomes some of the issues with waterfall by supporting requirements changes and testing between implementations, however Scrum is more flexible with these. Also has a relatively large focus on risk assessment, which we didn’t feel was necessary for a project of this scale.
- Extreme programming (XP): pair programming would ensure high quality code, but it is impractical due to scheduling conflicts between team members and time constraints. However, the general idea of peer reviewing each other's code is good, and we may be able to incorporate something like this into our plan.

Collaboration Tools

In order to facilitate collaboration on the project, the following methods/tools will be used:

- Regularly scheduled face-to-face meetings to discuss progress, assign tasks and raise any concerns.
- Skype group calls to be used as an alternative when meeting face-to-face would be inconvenient, and text chat or one-to-one calls for anything that doesn’t warrant a group meeting.
- Google Drive for collaboration on reports and documentation.
- Git for collaboration on code. The project repository will be hosted on GitHub. Any merge conflicts will be resolved by contacting the appropriate team member(s).

Team Organisation

After the team was formed, the following basic roles were identified as being present throughout the majority of the project. Following this, each role was assigned to a team member.

Member	Role	Role Description
Daniel Marshall	General Manager Website Manager	Responsible for the overall direction of the project and administration of the project website. Daniel was assigned this role after indicating that he had past experience in developing games and also had experience in web development.
Oliver McClellan	Git Manager	Responsible for administration of the project's Git repository. Oliver was assigned this role as he had previous experience using Git and GitHub repositories.
Jack Copland	Report Editor	Responsible for overseeing the editing of project reports and other documentation and ensuring that they are consistent.
Denet Benny	Secretary	Responsible for taking minutes during team meetings and recording them.

We did not assign concrete roles to each team member as we are using Scrum (with the exception of the General Manager, who will play the role of Scrum Master in meetings). This allows us to better assign work to team members once their strengths become apparent.

We also chose to identify specific roles for completion of the first deliverable which were then assigned to one or more team members. This was so that the whole group didn't have to research every part of the report. Also, it was easier to split workload roughly fairly between all group members (refer to resources chart below the Gantt chart). Members who had completed their sections then informed the General Manager who assigned the member to another section. After all sections were complete, each team member proof-read the entire report and discussed any issues with the team. The initial assignments were as follows:

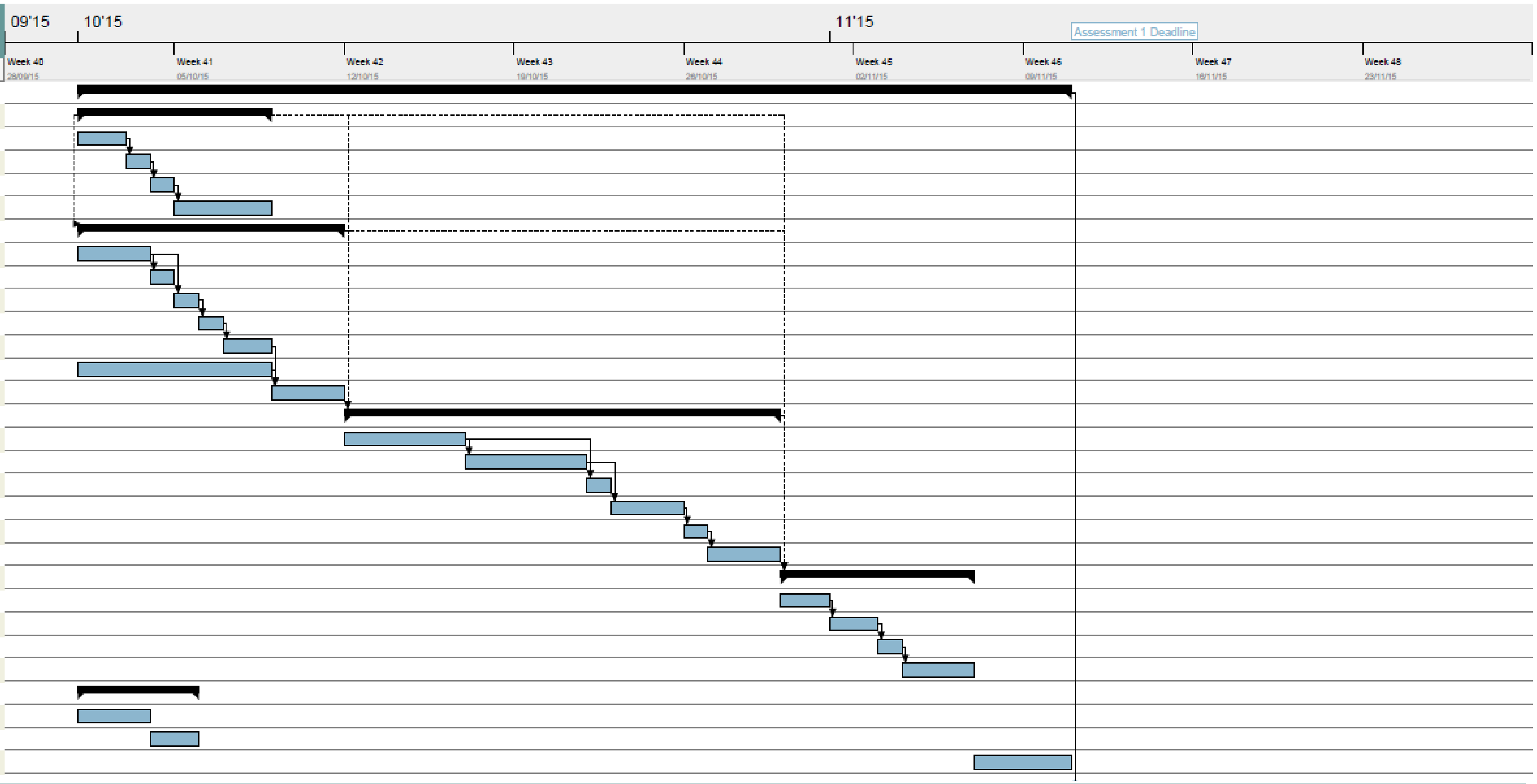
- **Requirements:** Jack Copland was responsible for refining and justifying the requirements after the team had collectively identified a list of initial requirements.
- **Risk Management:** Denet Benny and Anthony Lau were responsible for identifying risks and documenting them, along with how to mitigate them.
- **Architecture:** Daniel Marshall was responsible for devising the initial architecture for the game and creating the necessary UML diagrams.
- **Website:** Daniel Marshall was responsible for creating the team's initial website.
- **Planning:** Adrian Gralak and Oliver McClellan were responsible for documenting the team's chosen software engineering method and creation of the Gantt chart for the team's plan.

Team Mallard

08-Nov-2015

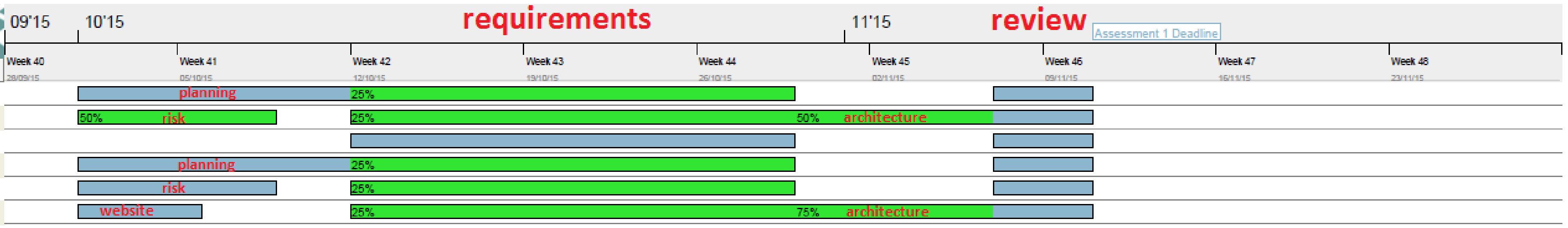
Gantt Chart

Name	Begin date	End date
ASSESSMENT 1	01/10/15	10/11/15
Risk aversion	01/10/15	08/10/15
Risk Identification	01/10/15	02/10/15
Risk analysis	03/10/15	03/10/15
Risk aversion strategies	04/10/15	04/10/15
Risk Report Section	05/10/15	08/10/15
Planning	01/10/15	11/10/15
Task Division	01/10/15	03/10/15
Workload identification	04/10/15	04/10/15
Condition identification	05/10/15	05/10/15
Role Identification and assignment	06/10/15	06/10/15
Gantt Chart	07/10/15	08/10/15
Method analysis and selection	01/10/15	08/10/15
Planning report section	09/10/15	11/10/15
Requirements	12/10/15	29/10/15
Requirements identification	12/10/15	16/10/15
Requirements analysis	17/10/15	21/10/15
Requirements conflicts discovery	22/10/15	22/10/15
Requirements implementatin and fomalisatio..	23/10/15	25/10/15
Requirements Review	26/10/15	26/10/15
Requirements Report Section	27/10/15	29/10/15
Architecture	30/10/15	06/11/15
Architecture research	30/10/15	31/10/15
Architecture analysis	01/11/15	02/11/15
Architecture review	03/11/15	03/11/15
Architecture report section	04/11/15	06/11/15
Website	01/10/15	05/10/15
Git preparation	01/10/15	03/10/15
Readme creation	04/10/15	05/10/15
Report review	07/11/15	10/11/15



Resources Chart

Name	Default role
Adrian	undefined
Denett	undefined
Jack	undefined
Oliver	undefined
Anthony	undefined
Daniel	undefined


09'15
10'15
requirements
11'15
review
Assessment 1 Deadline
Week 40
Week 41
Week 42
Week 43
Week 44
Week 45
Week 46
Week 47
Week 48
planning
risk
architecture
planning
risk
website
architecture