Method Selection and Planning

Cohort 1, Team 4:

Robert Watts
Adam Wiegand
Josh Hall
Travis Gaffney
Xiaoyu Zou
Bogdan Lescinschi

Selecting a Software Engineering Methodology:

Approaching this project as a small team, we knew that first and foremost a choice had to be made as to which methodology we would be adopting throughout the development process. During a discussion in our first meeting, we looked at a couple of different methodologies that are available when developing a piece of software.

Waterfall was the first to get eliminated- whilst it is the easiest to implement and forces us to plan thoroughly before we perform any designing or coding, we knew that concrete plans were unlikely to be kept for very long with this being the first time creating something of this scale in the Java language for all of us. This uncertainty in the early development phase, coupled with the nature of the Waterfall Model which only provides a working version of the system in the later stages of the development would overall prove detrimental to us. The Spiral Model was the next to go, because it required a certain level of prior knowledge beforehand to create rules and protocols which should be strictly adhered to throughout the development. We simply did not have this expertise as relative beginners to Java game development, so we thought we would struggle to make it work.

This led us to the Agile Model, and we agreed that Agile appealed to us the most. This was because the ability to deliver incremental progress on our project would not only allow for constant self-evaluation of how we are performing and where we would like to be, but also provide a close and inclusive relationship with our customer. This usually helps groups who are dealing with volatile requirements, and whilst our requirements may not have been volatile, they were quite broad in some respects. Being able to have frequent meetings with the customer to discern if our product was heading in the right direction would be a huge help in the production stages. Furthermore, the nature of the Agile Model where the project is broken down and divided into its smaller parts would also work in our favour, because this would allow each member of the group to have something to work towards between meetings and be actively involved.

With this in mind, we believed the basic Scrum development best catered to our needs. As previously stated, the project would entail completely greenfield development in an area none of us are particularly familiar in, so therefore our early designs and implementations are realistically likely to not work. This is mainly down to not immediately knowing how certain parts of our chosen game engine fully function yet, or us potentially making false assumptions based on previous projects- creases which could only be ironed out through gradual research, development, and weekly evaluation with other members in our group. Using scrum provides the flexibility we need to allow these changes to happen whilst also allowing us to keep track of our gradual progress and risks along the way, without having to completely redo prior sections of our program, which is supremely beneficial.

Tools Used in the Development:

In order to begin our development, we also needed to think about some of the tools and programs we may use along the way to aid us, both collaboratively and individually. GitHub was the first program that we all chose to use. We saw this as the foundation on which our project could be built on for a few reasons, but mostly because some members of our group were already familiar with its features and interface. It facilitates group collaboration by creating a code repository for our work which all members of the group could access, add to, or modify at any time, allowing us to keep up with the dynamic nature of the project and its different iterations. This is especially necessary in our case due to one of our group members being overseas and in a different time zone, so a cloud platform was not only suitable, but imperative.

GitHub's suitability for us as a team only grew further when one of our members suggested we use GitHub Pages as the platform to produce the website for our implementation, something which was a requirement in the deliverables. The main benefit of using this GitHub feature over other programs is the seamless updates between the website and our existing Git repository. Our website would be automatically up to date throughout our coding- a crucial factor when the Scrum method which our group selected would lead to very frequent updates to our code.

With a workspace set up, we turned our attention to the need for group planning to take place. Initially we thought about using Google Drive and having a document with everyone's current tasks, however it became quickly apparent that this would only be suitable at the start of the project- when aspects of the development process would start to get more complex or nuanced, or concurrent tasks would increase in number, we would quickly outgrow this environment and our productivity would be hindered. The alternative we found, and subsequently decided to use due to its suitability, has been Trello. It is overall a much more specialised and refined tool for project management which would help us to much more easily organise our resources. It allows us to create categories on our group board with tasks which we plan to do, are in the process of completing, are evaluating, or are completed, all the while allowing us to refer back to our original customer requirements in order to minimise 'scope creep'. With a Trello free trial, we could even create Gantt charts so that we could visually see the whole project with its phases, and clear relationships and dependencies between our tasks, or instead we could use Microsoft Excel for this purpose with similar results.

We returned to Google Drive and instead decided to use it as a platform on which we could host our team's research and ideas for the project. Because we would be using a game engine none of us had experience with (namely LibGDX), before development could begin we needed to research and familiarise ourselves with the package and its functions to get us started. Documents were made which set standards for our group with regards to how functions should be used, which is helpful when everyone is working separately. Not only that but ideas and assets that we wanted to utilise in our game also needed a place to be stored so that anyone in the group could access them, so Google Drive was suitable because these items could be published for the whole team.

Visual Studio Code was the IDE most of us are choosing to work on. This was mainly down to the group's personal preference, and it being the one we all felt most comfortable using without hindering our overall productivity being relative beginners. However, VSCode also has extensive Java support through downloadable extensions, which will allow us to set up our programming environment more quickly, and hopefully allow us to start creating a concrete product which we can then build upon much sooner in the development process. Visual Studio Code also supported the Javadoc commenting system, which we are planning on using to organise our code as we go. The nature of Agile development will mean that additions will be made to the code very frequently, so it will be beneficial for all of the team to know exactly what each piece of code is doing in a project which will increase in scale quite rapidly. In addition, GitHub Pages supports this commenting system too, so our code will be better documented and organised even on our group website.

Finally, we plan on using draw.io in order to complete our UML diagrams for this project. It is much more suitable for this purpose compared to drawing them ourselves in another vector program, because it facilitates the symbols and features needed for these diagrams natively. Not only that, but it is free, and has strong links to Google Drive (a platform our group is already using) which allows for easier real-time collaboration between members – something which will no doubt be necessary along the planning and production of our game.

Team Organisation:

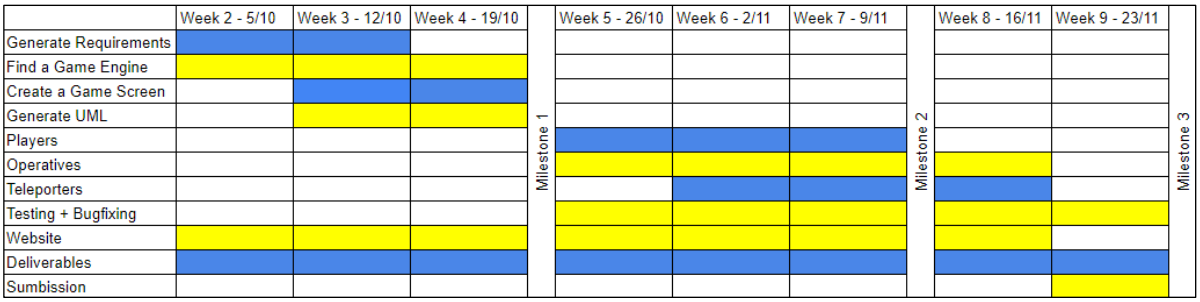
After choosing to use the Scrum software development method, it gave us the principles required to begin organising our team and thinking about how we will proceed with the project. We knew that we would have to divide our work into sprints which we decided to employ weekly, due to our pre-existing Thursday practical slots. This meant that each group member would have something to work towards through the week, which could then be presented to the group and evaluated when we all met in the next scrum duirng following Thursday practical. Due to Scrum being an Agile methodology, we knew that for these small improvements to occur weekly we would have to break down our requirements into smaller, more manageable tasks. A complex requirement like the player needing a map to walk on could be tackled by a group of 2 team members for example – one working on the visual aspect of the map, and one working on how the map is actually represented in our code for the player - and these smaller units within our group could also have scrums of their own in order to ensure the end product when the task is finished is cohesive and can easily be integrated. We realised that this communication between members of the whole team and these subunits is a key part of Scrum and any Agile methodology because without it, everyone working on their own part of the program could lead to integration issues at a later point in the development.

Scrum also gave us the opportunity to involve our customer in the development process more than other methods. We had a meeting available to us once a week, and in our scrum meetings we discussed and evaluated whether we would need any customer input to aid us in the next stage of production. If we agreed that this would be favourable, we would have the meeting with our customer and then immediately go into another scrum to assess the information given and how our plans for the current sprint would change or stay the same. This approach to organisation was suitable for us, because it allowed our group to focus on creating a product sooner in the development cycle which we could then build upon through consistent research and feedback from our customer meetings. It becomes much easier for a customer to understand what exactly it is they want from a product if they have something tangible which could show them features being implemented and progress on a weekly or bi-weekly basis, and this organisation would make the project easier for us.

Systematic Plan:

When devising a roadmap for this project, we settled on certain key tasks which needed to be worked on in order for our project to be a success and to meet the requirements given by our customer. These included “generating requirements”, “finding a game engine”, “creating a game screen”, “generating UML for the project”, “producing players”, “producing operatives”, “producing teleporters”, “testing and bugfixing”, “producing the website”, “deliverables”, and “submission”. As a group we understood that these key tasks would need to be broken down further in order to more effectively progress through the development cycle by splitting the workload.

At this point, it became clear that a Gantt chart could be produced using those tasks so that we could have a clear visualisation of the work ahead, and what time-frames we would expect each part of the project to take. This worked in conjunction with our Trello board, which meant that our team could be more organised. Our planning chart was as follows:



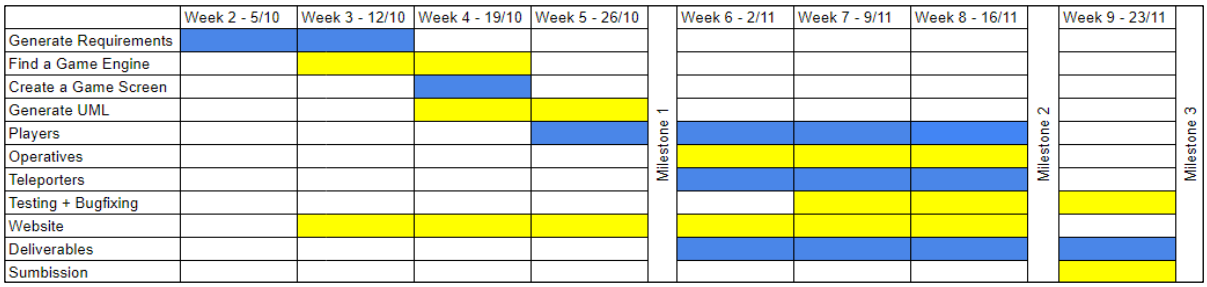
Some tasks took priority over others, and so a systematic approach needed to be taken, where certain tasks needed to be finished in order for work on the next stages to ensue. This in turn led us to the milestones which we set ourselves for this project, and which we inserted into our Gantt chart. Milestone 1 we wanted to reach before the start of Week 5, and this would be achieved when the game would have a screen and potentially a menu which the user could access and begin a game with. This was our first milestone because we saw it as the foundation on which the rest of the game would then be built upon. All future progress, such as adding the player into the game, and enemies which would destroy the ship, all depended on having a slate set up which they could be drawn on, so this to us was a key task which we needed to focus on completing in the first few weeks.

Milestone 2 we wanted to reach before the start of Week 8, and this would be the point in the game development where we would have a fleshed-out player, which could attack the operatives around the ship that were destroying the ship’s systems. It would essentially be a point in the game where all but a few final requirements would be met, and we could start thinking about the unique operative abilities which up the complexity of the game. Not only that, but operative abilities depended almost entirely on there being fully integrated operatives in the master build of the game, and a fully implemented player which the abilities could interact with. Furthermore, Milestone 2 would be the time to start some more intensive testing and bugfixing to ensure that our project build is as sturdy as possible for the final deadline, testing which could not have completely been done up until this point due to all the dynamic parts of the project not yet being implemented. Milestone 3 would just be the submission of all deliverables, and a completion of the overall project.

Plan Evolution:

When we set our initial plan and team organisation, due to our relative inexperience we expected there to be changes along the way. For a start, the organisation we had initially planned for our sprints which involved the customer almost weekly actually only ended up occurring twice. This was mainly because the group went into the first meeting with an extensive amount of questions which told us exactly what the customer required from the game in detail, which we wrote down in a requirements document on our Google Drive that we kept referring back to during our project. During the second meeting with the customer, we showcased our product before we started working on combat with enemy operatives, which was a large part of the game we were about to begin work on – this was more to ensure that all the group's prior work up to this milestone was satisfactory with their original requirements.

Other alterations came when our actual development changed completely compared to the Gantt chart we envisioned at the beginning of the project. Here is the evolution:



For example, it took a little longer to begin the game engine research because we wanted to ensure that a certain game engine could accommodate all of our different requirements, thus needing them to be exhaustive from the start. Our timescales were also a little off throughout, because it took longer than expected to initially get to grips with LibGDX. This then meant that we had less time at the end to completely finish certain aspects of the game, such as the operative abilities which we instead traded for more thorough testing and delivering a more polished product.

Due to us becoming more competent with the LibGDX game engine as the development cycle progressed though, we had a deeper understanding of what aspects of our plans were feasible, and which aspects were were overcomplicating the final product unnecessarily. One example of this can be seen in the team weekly Gantt charts located on our website – our scrum in Week 7 evaluated the fact that our game would need a mini-map in order to satisfy the customer requirement of "notifications when systems on the ship are being attacked". However after beginning the production of a mini-map in that week's sprint, our assigned group member found that it was actually needlessly complex. Thus at the next scrum we assessed our options whilst closely thinking about the requirements our customer had provided. This led to simplifying the mini-map idea to a simple text-based notification box which would tell the user exactly which parts of the ship are being affected by the operatives.

Finally, evolutions occurred as we found out each other's strengths throughout the project. Although we tried to assign people to tasks that they were willing to attempt in the scrums, some tasks proved trickier than others to handle alone. Therefore, as the team weekly Gantt charts will show in more detail, some tasks were started by someone independently but then swapped out with another group member or added a group member to the task before its completion. This was only possible because of the methodology that we chose to use, and it allowed us to more consistently make progress towards our final product, despite the fact we may not have accounted for changes like these in our original planning phase or the slight increase in time taken.