Method Selection and Planning

Method

For every software engineering project, choosing the right development methodology is essential for success. With this in mind, our team carefully approach the selection process by first considering the primary constraints imposed onto our project:

- Project Size: Short project of about 7 months.
- Cost of Delay: High as we have concrete deadlines.
- Team Size: Small team of 6 members.
- Requirement Enhancement: Requirements subject to changes.
- Client Feedback: Client can be contacted as often as needed.
- Experience on Project: Team members have limited experience with a project of similar type.

Given the high possibility of changes in the requirements as development proceeds, we immediately ruled out the Waterfall model; this is because the linear structure with discrete phases means that once the application is in the Testing stage, it would be very difficult to go back and make changes for the enhanced requirement. This constraint also drew our attention to the change-friendly Agile approach and after careful consideration of the Manifesto for Agile Software Development[1] we all agreed that the adaptive and cooperative characteristics of the Agile paradigm, along with the efficiency it brought to small development teams, make it most appropriate for the development process of "York Pirates!".

Having chosen Agile as the main development principle, our team then looked at methodologies that follow its philosophy and decided Scrum[2] would best fit our constraints and preferences. This approach greatly emphasizes teamwork by having the team "try to go the distance as a unit, passing the ball back and forth" to address complex problems [3]. Scrum starts with a wish list of features in order of priority called a product backlog. Tasks will be picked from here and moved to a sprint backlog for development during the 1-4 week long sprints. Each sprint ends with a review and the team then chooses another backlog to develop in a next sprint. Brief team meetings called scrums are organised regularly to discuss progress and make plans.

Having the project split into Sprints enables our team to work with whatever requirements we have and build upon them with frequent client's feedbacks. In addition, the Sprint review, where the client checks and provides feedback on the completed Sprint, accommodates for diverse changes that can be triaged in the following Sprint. This approach makes Scrum very flexible with requirement enhancements and allows us to capitalise on our ability to contact the client as often as possible. Also, by using Sprint, we can make sure the project is completed on time based on estimation from data like the velocity of previous Sprints. This is particularly useful given the high cost of delay of our project. Finally, a small team like ours will have an easier time self-organising (a core nature of Scrum) and requires less overhead for meetings and discussions compared to bigger ones.

Despite the usefulness of Scrum, there are aspects that are not suitable for us thus our project cannot solely follow its traditional implementation and adjustments must be made. The original Scrum requires daily Scrum with every members presented, which is difficult considering our members' schedules. The client, though can be contacted as often as possible, is also very busy and requires a decent amount of time beforehand to set up meeting with. To counter this issue we are ensuring as many weekly meetings as possible but at minimum 2 meetings per week (start and end of the week).

Tools

After careful analysis we concluded that our team requires tools for: Development, Communication, Document sharing, Version Control, Task Management and UML Diagram/Gantt Chart production. The tools we use in the development process of our project should be accessible by all members of our team, the client as well as others in case we pass on our project to another team.

Development

One of the main constraint requirements of the project is that it has to be in Java, thus we need a Java-based game engine or framework to help us develop our game. After careful consideration our team chose to go with LibGDX as it is one of the most powerful framework in Java, especially for 2D games. Libgdx is also extremely popular and used widely in the industry so there are a lot of existing tutorials and guides to ease us into the development process.

Communication

We will use primarily Facebook Messenger for communication outside of in-person meetings. We chose Messenger as it allows swift responses between members through notifications and can be accessed from a wide variety of platforms (smartphones, laptops, PCs, etc). We also used the in-built chat on google drive when working simultaneously on the same document so that other group members were not flooded with document specific questions that weren't as relevant to them.

The team already uses messenger on a day-to-day basis and it has some handy built-in features such as image/video-sharing and polls. It is also fairly easy to create smaller group chats, allowing us to still communicate effectively if we split into smaller sub-teams to work on different aspects of the project.

Document Sharing and Production

We chose Google Drive as the hub for sharing our documents, logs and research with each other. Due to the fact that Google Drive is based on the cloud and has automatic backup, the risk of losing documents (Risk 6) is extremely small. All team members are experienced with it and through the university get unlimited storage and the ability to create team drives, a feature we will be utilising.

For document production we will use Google Docs as it is built into Google Drive and has the classic Word-style interface that everyone is familiar with. Furthermore, it allows for simultaneous editing and has a live commenting system making it much easier to collaborate on files as a team.

Version Control

We plan to use Github for the version management of our software as creating private repositories is free for students like us. In addition, the ability to setup different branches of a repository enables changes to be applied and checked gradually without breaking the code from the master branch should problems arise, which fits perfectly with our Scrum approach. GitHub is also decentralised which allows for restoration of code from local copies should the server failed (Risk 6).

An added benefit of using GitHub is that it also provides a service called Github Pages, which are websites that describe our repositories. We can use this for making our project webpage as it is easy to pick up and we do not have to deal with server-side problems.

The only drawback to using git for our source code is, unlike drive, you cannot collaboratively work on the same document from a multi user point of view. Instead you need to wait for updated code to be pushed onto the git which can slow things down and stops multiple people working on the same document concurrently, remotely. If you were ti work concurrently on the same document it would cause clashes which would need to be resolved upon merging again, which is a lengthy process we might not be able to afford to do due to the strict deadline we must adhere to.

Atom

To counter issue discussed above, we used a piece of software called Atom. This is a collaboration tool designed specifically for software development and it allowed our development team to work on bugs and issues together making production much more efficient. The main motive for choosing Atom is the embedded git control that is offers. This made it really easy to work around the issue of git not being collaborative enough for our group hierarchy, however still keeping git as the destination for our version control. It did not require us to migrate to a completely different tool other than git which would have taken a lot of time and caused potential problems further down the line in the project. The majority of the time Atom was used for working on isolated functions when debugging. IntelliJ still remained the primary IDE during development.

Task Management

For the task management we mostly used the google drive creating checklists and tables to assign jobs to one another. This made sense as we used this method in previous assessment iterations and we are all familiar with the collaborative use of google drive. It also stores all changes made allowing us to see which tasks had been ticked off by which group member.

UML Diagram

We used an IntelliJ IDEA plugin called "Sketch It!" to generate UML diagrams directly from our Java code using the PlantUML syntax. This allows our concrete UML diagram to be 100% accurate to the actual code, which would likely not be the case if the diagrams were created manually due to the scope of the project. We then took the generated UML diagram and made alterations removing trivial methods and variables such as getters and setters.

Tile Map Editor

We have decided to use Tiled Map Editor to create our in-game map. This application is recommended for use with LibGDX, which means there is access to many useful features that can simplify the process of creating a sailing mode. Further, we found that it had a nice aesthetic but was still a free software which was very useful to our group. For our group, when taking over the project, there was some learning to do however the time spent learning was much shorter than the time the whole of the group would have spent trying to redesign the game to not use a tilemap.

Team organisation approach

To ensure all team members produce the best work we are capable of it is important that we have roles that we are confident and competent in. While people will primarily specialise on tasks based on the specific roles they are assigned, these are not strictly fixed roles. We decided to make this decision to give greater flexibility within our team - as not to restrict creativity or ideas amongst people for any given task. This is especially important considering no member has full specialised experience in one area meaning the cost of having more dynamic roles is minimal. This did cause slight issues when it came to assigning risks however so moving forward in the project, people have naturally taken more ownership of their individual roles. This does not mean that the flexible hierarchy does not still remain however.

Team Leader - Charlie Dyson

The role of the Team Leader is core to the success of the project. Overseeing all progress on the project, Charlie will track project development over time to make sure that the everything runs smoothly, and identifying any risks or problems that occur in order to reduce downtime or unnecessary backtracking in the project development. We felt that Charlie was the best person for the job as from the first meeting he naturally took up a leadership role and motivated the team. As we want everyone to be satisfied with the role they are assigned, we thought that it would be counter-intuitive not to give him the role he displayed qualities in.

Graphics Designer & Risk Manager - Shubei Qian

Shubei is in charge of designing any graphics for the project or game assets as well as risk management for the project. The role of Graphics Designer covers designing any art that is used on the website in addition to any assets in the game. The role of the risk manager is to identify track any risks that might occur and then to provide suggestions for potential solutions to these issues. We felt that Shubei was definitely the most talented artist in the team and she has a style of drawing which suits our game aesthetic. We also feel like Shubei was a good choice as the primary risk manager as she had a good balance of knowledge of the code, as well as functionality and requirements the code was trying to implement. In other words she showed that she has a good 'bird's eye view' of the project as a whole which we felt would be very beneficial when factoring in risks.

Head Developer - Ed Gould

The job of the head developer is to handle the creation of the game and be the primary team member to implement features into the game. Ed will work with Charlie and Luke to coordinate certain stages of the game's development to fit into the Scrum work cycle by working out incremental key stages in which additional content can be added each Scrum cycle. We felt that Ed would be the best choice for a head developer as he had the most experience in game development previously when starting this project. He is also very passionate about the code he and his team produces making him a good candidate for the role.

Design Management & Creative Director & Lead Tester - Luke Richardson

Luke will mainly be working with Ed and team members that help produce the software for the game by planning the overall design timeline of the game. Working with the team, Luke will manage any design decisions that might change during the development cycle, either due to changing requirements of the stakeholder or agreed upon changes to the design approach during the project. We felt that Luke would be the best individual when it came to creativity and ideas in the project. From the first meeting we had as a group he was very passionate about ideas and was not short of them, despite the brief being very bare. Due to the fact Luke is designing the majority of the requirements, it naturally made sense for him to also run the black box tests based off of these requirements.

Web Developer & Interface Manager - Umar Farooqi

Umar was happy to handle the development of the website where our work is presented. He is confident in learning the languages and tools required to produce the website, as well as sorting out web hosting. Umar will also relay information from the stakeholders or other customers to the team. We felt that Umar was the best fit for this role as he had previous experience in web development. He also was very active in the group communication sessions on Messenger so we felt that he would make a good interfacer and relay information to and from the client, and the team, most efficiently.

Secondary Developer & Architecture Designer - Andrew Todd

Working mainly with Ed, Andrew will assist in the software development of the game, collaborating through GitHub. With a similar role to Ed, they will both be working on different areas of the game to increase the speed at which it is developed. We felt that andrew would be best fit for the role as secondary developer because he has a good amount of experience in software development prior to starting the project. He does however have a very busy out of curriculum schedule so we didn’t want to put pressure on him as the head developer and Ed was happy to take up that role. Andrew initially started as the lead tester however, due to the flexibility of our team, Luke naturally took up this role due to him designing the initial requirements which the tests are based off of. Andrew found his place however, when designing the architecture of the software. His top changes from lead tester, to designing the structure of the classes in the software and relaying it to the development team in an easy to interpret form as a UML diagram.

Systematic Plan

In order to plan the rest of our project, we decided to use a Gantt chart. This allowed us to create a plan for our software engineering project for the entire year and include dates, priority and status to have a visual indication of what needs to be done and by when.

For assessment 3, we plan to work on the implementation and the change report in parallel, throughout most weeks of the assessment. This is so that we can update the change report as we actually make the changes in real time keeping all the information fresh in our mind and not revisiting the code weeks in advance and having to figure out what changes we make. Again the website is left until the last weeks because everything needs to be finished before it is uploaded to the website.

For assessment 4, we are continuing with our plan for jobs running in parallel as we feel that it is best when the information is current. Because of this, we plan to complete the evaluation and testing report alongside when we are doing the implementation. This is because we expect things to change throughout the implementation and we will be testing continuously as the assessment develops. This is also the motivation behind assigning these tasks to multiple weeks as we realise how lengthy the implementation process can be which will need evaluating and testing along the way. The project review report is obviously the last task we plan on completing as it is only something that can be discussed and written once all other tasks are complete to ensure a true evaluation and review of the methods used throughout the project as a whole.

Gantt chart available on next page and in additional document section of the assessment 1 tab of our webpage (http://limewire.me/docs/assess1/Gantt%20Chart%201.pdf).

References:

[1]: The Agile Manifesto. Available at: http://agilemanifesto.org/

[2]: The Scrum Guide. Available at: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[3]: The New New Product Development Game. Available at: https://ullizee.files.wordpress.com/2013/01/takeuchi-and-onaka-the-new-new-product-development-game.pdf