Method Selection

Software engineering method

To effectively manage our project it is important to agree on a software engineering method that will work well with the type of project we have been given, as well as being comfortable to use for our team. It was identified that the development method we picked must be flexible to the customer, with the ability to adapt to changes in the projects end goal requirements. An agile method such as Scrum or Extreme programming would are good candidates for methods to use as they are both simple to implement, and we will be able to manage our project with greater transparency making it easier to work as an effective team.

Extreme programming (XP) is a software development approach that primarily focuses on the ability to handle changing customer requirements quickly whilst still offering high-quality products. To achieve this XP encourages the use of rapid development cycles which gives more space for the customer's requirements to be implemented between the releases. XP suggests using tactics such as pair programming, implementing features only when necessary to simplify the design. XP also stresses that there should be a heavy focus on code review, often through the prior mentioned pair programming strategy.

Scrum is a software development framework that separates work into smaller subtasks (called “sprints”) that can be completed in short periods of time, generally around two weeks. Each working day starts with a “scrum” in which all team members engage in a discussion and report the work they have completed previously, the work they aim to accomplish today, and any risks they think might occur. After sprints, there will be a scrum review that looks at the work that has or hasn’t been completed and the team then discusses ways to approach the following sprint with greater efficiency.

Ultimately we decided to use the Scrum development method as we thought that it would be flexible than XP, giving us greater control over our project as we will be able to adapt and grow as a team by reflecting on previous ideas in order to become more effective as the project progresses. By adopting the Scrum framework as an agile method we will maintain our project whilst allowing us to incorporate any new requirements or changes to previous requirements as requested by the stakeholder. Using agile methods will have a significant benefit considering the stakeholder will have continually changing requirements, therefore an appropriately fast software engineering method will increase our ability to fulfil the requests of the stakeholder; producing high-quality work on time. Furthermore, a team of six members is an ideal size to use the Scrum software engineering method as it will allow us to quickly discuss progress and split tasks effectively.

Tools

To maximise the time spent on the project as efficiently as possible it is essential to use development and collaboration tools that suit the project and team. This will reduce downtime which might occur if poor tools are chosen which might hinder the development process in some way. Being able to create the product without worrying about the tools used allows us to focus fully on each task, resulting in a higher quality product being developed in a shorter time. Using communications tools that are intuitive and easy for all of the team members to use will significantly streamline the project by allowing members to share ideas and update others on the progress of current tasks.

Document file sharing - Google Drive

We decided to use Google Drive to keep track of any documents such as the assessment specifications and scenario we need as well as any document deliverables. Having a universal storage for any admin files in our team allows us to collaborate on creating any deliverables - and makes it easier to refer to any assessment documents. By backing these files from the team drive also

decreases the chance of losing data since we can store a copy locally as well as on Google's cloud services.

Project source code & resource management - Git

Using a version control system to manage our source code and any other resources (such as image files or sounds) will make the actual creation of the game much easier. Git gives us the ability to coordinate and work on the project at the same time, increasing the speed at which we can develop. In addition to this, GitHub offers version control and branching, decreasing the risk of data loss. Having access to this also means that if mistakes are made or decisions on development approaches change we will easily be able to revert back to a previously agreed upon version.
We decided that we would use Git over the other suggested VCS, Apache Subversion - SVN, mainly because we are more familiar with Git, as well as having the perk of being decentralised to reduce the chance of data loss risks. Git is also more popular, meaning sharing the project with others will likely be easier than if we chose to use SVN instead. 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.

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 - Gantt Chart

For task management, we are using the gantt chart shown below as a method to keep everything on schedule. We felt that the gantt chart is very easy to read and interpret and it breaks down large tasks into smaller tasks which have their own individual time frame planned allowing for greater concurrency when completing the tasks. This is because we can very easily assign the smaller tasks to specific individuals giving them a more realistic amount of work in each aspect of the project. This also aids the flexibility of the project. We made sure we were on schedule from the gantt chart by communicating on Facebook Messenger which is discussed in the paragraph below.

Communication - Facebook Messenger

At the very start of the project, we decided to create a group chat on Messenger to initially get to know each other and then begin planning the first steps of the project. Having access to a common communication tool allows us to quickly update each other on project events and progress, and can be used to arrange team meetings. Using a group messaging service like Messenger seemed to be the best choice as all of the team members already had accounts.

IDE - IntelliJ IDEA

For our development environment, we decided to use IntelliJ IDEA. This is because IntelliJ is advanced, but still simple to use - sharing similarities with Eclipse used by all team members in our first year. Also, the IDE also supports and has a simple importing process for either of the game

libraries we were considering to use (LibGDX or LWJGL). IntelliJ also includes Git integration which will is especially useful considering we will be using Git for our version control.

UML diagram production - *StarUML & LucidChart*

We decided to use StarUML to produce the UML diagrams as it is freely downloadable software that you can download onto your personal computer. Since no one has used any UML diagram creation tool prior to the project no one has any preference towards any specific software, and being as this was recommended and we had seen it was used for previous years projects, we decided this was a suitable option. We also did give some testing using other UML software such as draw.io because we saw that it had google drive integration, however actually designing the UML models we found easier in StarUML so we thought this benefit was more important than the group collaboration facilities.

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 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.

Project 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. The Gantt chart we designed is evidenced below however a more readable version is available in the extra documents section of assessment 1 on our website www.limewire.me

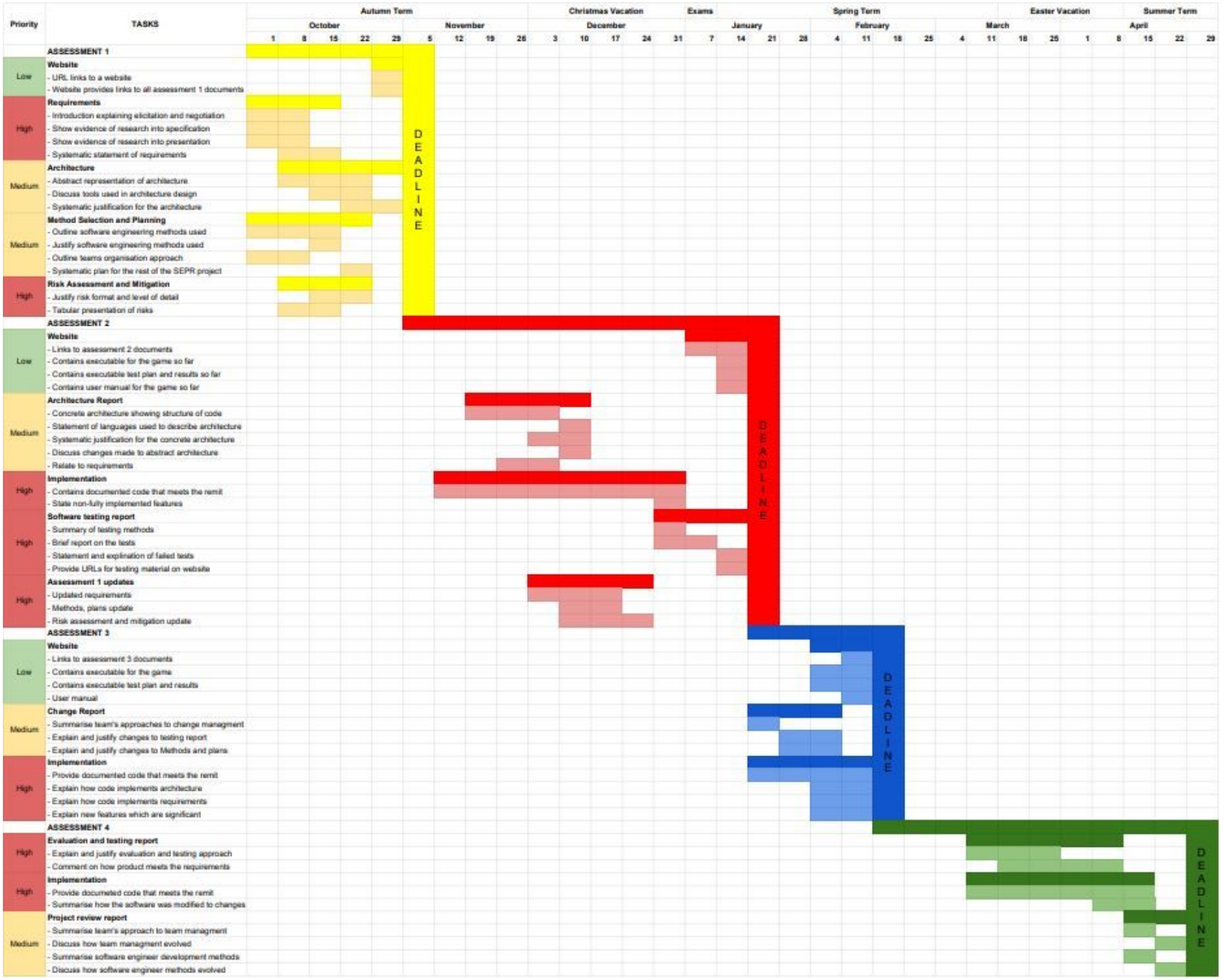
For assessment 2, we felt it was necessary to plan for development of the game throughout all weeks leading up to the deadline apart from the penultimate weeks. This is due to bug fixes and code reviews taking place throughout the assessment. The reason we planned to finish implementation 2 weeks before the deadline is because we wanted to leave ourselves a crumple zone if anything went wrong. The testing planned to overlap slightly with the implementation as, once we had the testing complete, we would need to isolate and fix bugs discovered in testing. The

later 2 weeks were then designated for filling out documentation such as the traceability matrix seen in the additional documents section of our webpage. We planned to complete architecture alongside implementation as we knew the implementation may cause the architecture to change which we would need to justify in the documentation. The updates to assessment 1 documentation was planned in the middle of the timeline because, we wanted to get this done quickly however the priority was implementation. We ended up completing the majority of the updates documents very early on in the assessment and felt it counter intuitive not to get ahead of schedule. The website is left until the last weeks because everything needs to be finished before it is uploaded to the website.

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).


Priority
TASKS
Autumn Term
Christmas Vacation
Exams
Spring Term
Easter Vacation
Summer Term
October
November
December
January
February
March
April
1 8 15 22 29 5 12 19 26 3 10 17 24 31 7 14 21 28 4 11 18 25 4 11 18 25 1 8 15 22 29
ASSESSMENT 1
Low
Website
- URL links to a website
- Website provides links to all assessment 1 documents
High
Requirements
- Introduction explaining elicitation and negotiation
- Show evidence of research into specification
- Show evidence of research into presentation
- Systematic statement of requirements
Medium
Architecture
- Abstract representation of architecture
- Discuss tools used in architecture design
- Systematic justification for the architecture
Medium
Method Selection and Planning
- Outline software engineering methods used
- Justify software engineering methods used
- Outline teams organisation approach
- Systematic plan for the rest of the SEPR project
High
Risk Assessment and Mitigation
- Justify risk format and level of detail
- Tabular presentation of risks
DEADLINE
ASSESSMENT 2
Low
Website
- Links to assessment 2 documents
- Contains executable for the game so far
- Contains executable test plan and results so far
- Contains user manual for the game so far
Medium
Architecture Report
- Concrete architecture showing structure of code
- Statement of languages used to describe architecture
- Systematic justification for the concrete architecture
- Discuss changes made to abstract architecture
- Relate to requirements
High
Implementation
- Contains documented code that meets the remit
- State non-fully implemented features
High
Software testing report
- Summary of testing methods
- Brief report on the tests
- Statement and explination of failed tests
- Provide URLs for testing material on website
High
Assessment 1 updates
- Updated requirements
- Methods, plans update
- Risk assessment and mitigation update
DEADLINE
ASSESSMENT 3
Low
Website
- Links to assessment 3 documents
- Contains executable for the game
- Contains executable test plan and results
- User manual
Medium
Change Report
- Summarise team's approaches to change managment
- Explain and justify changes to testing report
- Explain and justify changes to Methods and plans
High
Implementation
- Provide documented code that meets the remit
- Explain how code implements architecture
- Explain how code implements requirements
- Explain new features which are significant
DEADLINE
ASSESSMENT 4
High
Evaluation and testing report
- Explain and justify evaluation and testing approach
- Comment on how product meets the requirements
High
Implementation
- Provide documeted code that meets the remit
- Summarise how the software was modified to changes
Medium
Project review report
- Summarise team's approach to team managment
- Discuss how team managment evolved
- Summarise software engineer development methods
- Discuss how software engineer methods evolved
DEADLINE