1.4 | Software Engineering Methods and Tools

*a) Give an outline and justification of the team’s proposed software engineering methods, and identify any development or collaboration tools that the team plans to use to support the project or the team working (10 marks, ≤ 2 pages).*

**1.4.0 Introduction and Outline**

Within an engineering task of any scale, it is wise to consider the processes and engineering methods through which the project will be completed. Using methods should combat the proclivity for complex team-based tasks to tend towards disorder in the absence of well-defined plans and aims. In response to the game-development brief, and with this mind, this white-paper will document the processes, methods and tools *Un-Two* intends to use to deliver on the client’s brief.

*Un-Two* have chosen the Agile development paradigm to guide the principles of the project’s software development. Project development methodology will be guided by the RAD (rapid-application development**)** approach. In brief, we believe the core tenants of agile – namely its flexible response to change and efficiency with small teams (both stemming from its non-deterministic and non-linear nature) – are most appropriate for a small-scale videogame design project: *​The​ ​Mysterious​ ​University​ ​of​ ​York*.

Game development shall take place in *Unity* and will make heavy use of the engine’s integration with *Visual Studio* for *C#* scripting. *Visual Studio* provides native support for *Visual Studio Team Services (*VSTS), Microsoft’s version control system. It is through this that the team will code collaboratively. As such, the code repository will be hosted on a *visualstudio.com.*

Aside from group meetings, discussion pertaining to both project and group-meeting planning will be conducted over Facebook Messenger.

Documentation

White paper documents are worked on collaboratively via Google Docs.

**1.4.1 | Software Engineering Method Justification**

**/agile manifesto**

The Agile method has become a staple Software development methods that were considered but not chosen are waterfall and Scrum. Scrum had been considered, but emphasised bursts of rapid development within a longer timeframe - something that does n0t apply in our circumstances. Given the exceedingly short time-frame by software engineering standards, we doubt that scrum method will provide enough development time with the large amounts of short bursts that would have been necessary in our project.

The relatively short-time frame of the project coupled with the size and inexperience of our team quickly ruled-out many methods that large, established industry firms may consider. Examples include

The RAD method has been selected because it enables us to work best, given the client’s loosely defined requirements, as RAD works best with frequent client meetings and review of product. As our workspace and the client is the same we can organise almost weekly client meetings to review work completed at the time allowing requirement change and update in almost constant time. This gives us a product which is in full agreement with the set cliental requirements and repeating the process with an outside party of testers gives us significant user feedback on all features of the game leading to quick bug identification and streamlining of systems.

Timely manner, greenfield project, concurrent small-scale testing and implementation/development…

RAD puts less emphasis on planning and more emphasis on process. Particularly relevant given the and the lack of established practices and common engineering methods amongst the team, it also makes the process easier given that over the majority of time for this project the team will be working in different locations and will need to do independent work, in this case RAD’s rapid construction technique is extremely useful letting multiple people update and produce code in quick succession of each other.

The one possible downside of RAD is that it requires a higher level of competency from each team member than other similar methodologies[1]. In our team we have three members with a higher level of ability in game development, to compensate this our original planning session will be splitting the tasks in our driven plan into phases and assigning specific tasks to each separate member based on ability while still maintaining a fair split in the work. This plan should make the most of RAD’s fast changing code and not become a struggle to our team members.

**1.4.2 Engineering Tools**

**Game Engine: Unity**

Unity has been chosen for the following reasons: Unity’s scripting language is C#. Of the six group members, three have existing competency in C#, and all have experience with Java (a decidedly similar programming language). With existing experience, the team will be able to devote more time to producing client deliverables, rather than learning to use a new technology.

Next, Unity provides pre-existing tools and features (such as animation support, graphics rendering, object transformations) that will allow the team to focus upon implementing gameplay features rather than technical dependencies.

Unity’s asset store provides a fallback for a source of graphical resources if we are unable to produce our own or produce assets of a sufficient standard. Quality graphics are pursuant to an engaging videogame as we strive to provide client satisfaction.

A result of Unity’s popularity, each feature is extensively documented and numerous tutorials are available. Such support will smooth the development process by giving the team the knowledge to overcome software barriers encountered during development.

We recognise that professional use of Unity likely correlates with the

Professional game developers think highly of Unity (evidenced by use)

Unity is most used engine

**Version Control: Visual Studio Team Services**

As previously discussed, VSTS

Noteworthy features (/beyond other VC services)of VSTS are its stakeholder support (which provides the client with instant access ,should they wish to

**IDE: Visual Studio**

**UML Tool: StarUML**

Free, ease of use,

**Communication: Facebook Messenger**

Given Facebook Messenger’s integration with team mobile phones, it offers the fastest

When new messages are received, team devices are pinged, ensuring that team members can review messages as soon as possible and ultimately facilitating a fast alleviation of issues (this is congruent to the RAD approach and timescale). Communication applications Slack and Discord were considered but Facebook Messenger + Remote working

**Documentation: Google Docs**

Files in a private virtual drive. It is hoped that ((fast delivery)), ease of use and its effortless collaborative nature ((of the platform)) will minimise any potential impact on the software engineering phase. Indeed, Google Docs is universal, having little-to-no software or OS dependencies,(int no hardware failure dependencies. Non-destructive commenting system for rapid review of documents.

Automatic version history

Universally accessible virtual drive, folder hierarchy

Paper prototyping of combat system.

which will aid client feedback for agile development)

Unity offers native integration with visual studio online through Visual Studio repository…

Agile is go-to method for software development firms

An outline of our needs identification and associated software method is shown in table form below:

|  |  |  |
| --- | --- | --- |
| **Need** | **Software** | **Notes** |
| Version Control System | Visual Studio Team Services | Hosted on visualstudio.com; Allows revision, viewing, branch, merge conflict resolution |

b) *Outline the team’s approach to team organisation, and explain why the chosen approach is appropriate for both the team and the project (5 marks, ≤ 1 page).*

Un-Two has adopted (will adopt) a flat organisational for the software engineering task, with an emphasis given on individual responsibility for certain tasks.

We believe a non-hierarchical management approach will work with our team because conflicts and inquiries can be resolved very quickly. In short, we believe any form a middle management is redundant given the team size, the scope of the project, and team personalities. Where conflicts in decision may arise, a democratic process will be undertaken to resolve the conflicts

The assignment of strict team roles has been considered.

team roles etc, devops (methods and architecture)

Team leader?

Code,graphics,sound etc

Flat organisational structure; too few of us to justify middle-managers

**Documentation**: For each given document, one member is assigned responsibility for the delivery of that document. Other members can contribute towards this document using Google Docs but the assigned person remains liable for its delivery. In spite of our not having a sole project manager responsible

It is hoped that such an approach will maximise collective input into the document (/by utilising all members) from varying members of the team meanwhile maintaining incentive for it to be completed and delivered.

c) *Give a systematic plan for the rest of the SEPR project. Your plan should lay out the key tasks for assessments 2 to 4. You should provide a detailed plan for the the software engineering tasks required for assessment 2. Tasks should indicate the earliest starting date and the latest finishing date, as well as task priorities. The plan should also identify a critical path and task dependencies (5 marks, ≤ 1 page).*

In architecture report, members responsible for implementing each feature will document it.

A redacted list containing only major assessment milestones follows:

2. Update Website

Architecture Report

Implementation

Software Testing Report / GUI Report

Update Assessment 1 documents

3. Extend another team’s game

Change existing report and justify

4. Final architecture report

Testing report

Implementation

Much of assessment 1 documentation depends upon a sufficient implementation of the game

A GANTT chart seems to most appropriate form of

//

A GANNT chart is not the most suitable form of plan because exact order of ,(ambiguity of dependencies)

Reference

1. https://airbrake.io/blog/sdlc/rapid-application-development