Module	SEPR
Year	2019/20
Assessment	1
Team	Dalai Java
Members	Jack Kershaw, Max Lloyd, James Hau, Yuqing Gong, William Marr, Peter Clark.
Deliverable	Method Selection and Planning

Software Engineering Methods and Collaboration Tools

The software engineering methodology that we chose was Scrum. Scrum is an agile methodology allowing smaller teams to work efficiently to strict deadlines. This means that the focus is on iterating smaller sections of the whole project rather than trying to build the entire thing in one go. The advantage of this is that any short notice requirement changes are easier to deal with and so means the team is more flexible. We will use Discord as our communication platform for our Scrum meetings which we will hold regularly**.** In these meetings each member takes a couple of minutes to explain what they are doing on the day and how other members' tasks may coincide with their own. Discord is the obvious choice for our team as we live all over York; we can communicate verbally at any time and this allows explanations of ideas to take much less time than text-based messaging. Further to this, we will use Trello to track tasks, whether that be tasks to do, tasks in progress, or completed tasks.

Trello allows us to assign members to each task, allowing us to see at a glance what needs doing and who could potentially do it. This meshes well with the agile methodology as it allows tasks to be quickly edited and swapped out. We will also not hesitate to contact the customer if any design issue or ambiguity arises which is another core feature of an agile methodology. Embedding the customer in the team allows us to better understand their needs.

The reason for choosing Trello over other organisation software was the clean and clear interface it provides. Options such as Wunderlist and Todoist had either too much going on in the interface itself or too many menus. The fact that Trello offers a basic and neat interface was important as it allows us to focus on the tasks we need to do rather than to be distracted by the software itself.

On top of Discord and Trello we will also use GitHub, this is the obvious choice for any software development project as it allows easy collaboration and has many 3rd party applications that can push/pull etc. This means each team member can use their preferred application and it will have no adverse effect on the other members of the team. GitHub allows for easy collaboration between a group as code can be written over multiple branches and these branches can then be merged to the main deliverable branch when all code is working. GitHub also allows for rollback to older versions meaning that if a commit breaks the whole program then the software can be rolled back to the previous version while the initial problem is fixed.

Further to this we are and will continue to use Google Drive for all writing-based tasks in this assessment. Google Drive is excellent for collaborating on documents as it allows multiple people to edit the same document in real time. This means that writing tasks can be done much quicker and there is no need for merging like we would need to do if we did the same thing using GitHub. Google Drive does, however, come with some disadvantages. The formatting isn't always consistent and this can lead to frustration when preparing the final documents. We will overcome this by downloading the Google Docs version before the assessment and formatting each document in Word. Word is far more consistent in this regard and so is the obvious choice for us to make sure that each assessment document fits the required page limit whilst still looking respectable.

We chose to avoid classical and plan driven methods as these wouldn't allow us the flexibility of being able to deal with changing requirements. One such classical methodology is the Waterfall model. This is a very linear model where all requirements are drawn up first, then the software architecture is designed and then all the software is implemented. This linear model does not offer much scope for change and as such does not suit our project as we anticipate change to the requirements that exceeds 1% change per month. Further to this, plan driven methods focus heavily on documentation and so tend to weigh the project down with unnecessary bureaucracy. This may well work for scenarios such

as the client being remote or where extensive documentation would be needed upon release anyway but we thought that neither of these apply to this project.

As we are coding this project in Java, we are following an object oriented design which has many benefits for our team. Firstly, the code is much easier to compartmentalise, we can get each member to write specific classes given a specification and they should all mesh together in the final product. Secondly, object-oriented design allows us to abstract out key design features into objects. For example, a fire engine is its own object with its own methods and attributes and this makes designing the overall product more manageable and more easily translatable to code. Furthermore, we can use inheritance to reduce the file size of the game and make designing certain aspects easier. Following our fire engine example, this object could inherit from a class called “moveable” which is an item in the game capable of moving and therefore we only need to write the “moving” code once.

In order to design and prototype the architecture we are going to use UML (Unified Modelling Language). This will allow us to graphically represent all of the classes and the relationships between them, enabling us to refine the architecture well enough for us to implement. It will also enable us to bring this visual representation of our architecture to meetings with the client, if they so wish. Additionally, a visual representation of our system will be easier to understand than a written one meaning we spend less time re-reading documentation and more time designing and implementing. To design this we researched a number of different programs to achieve an effective UML diagram. Initially we looked at PlantUml which allows a textual representation of the UML diagram. This automatically creates the links between classes ensuring the correct formatting of the finished diagram. However, we decided against this method over an online GUI based program called LucidChart. This allows multiple members to collaborate in real time to design a UML diagram with the main disadvantage being that it does not force the correct syntax meaning that we had to be careful to ensure that the correct arrows were used in order to make it easier to follow in the implementation stages of the project. We decided that this trade-off was worthwhile due to multiple people needing to work together on the diagram remotely which would greatly decrease the time needed to complete it.

Team Organisation

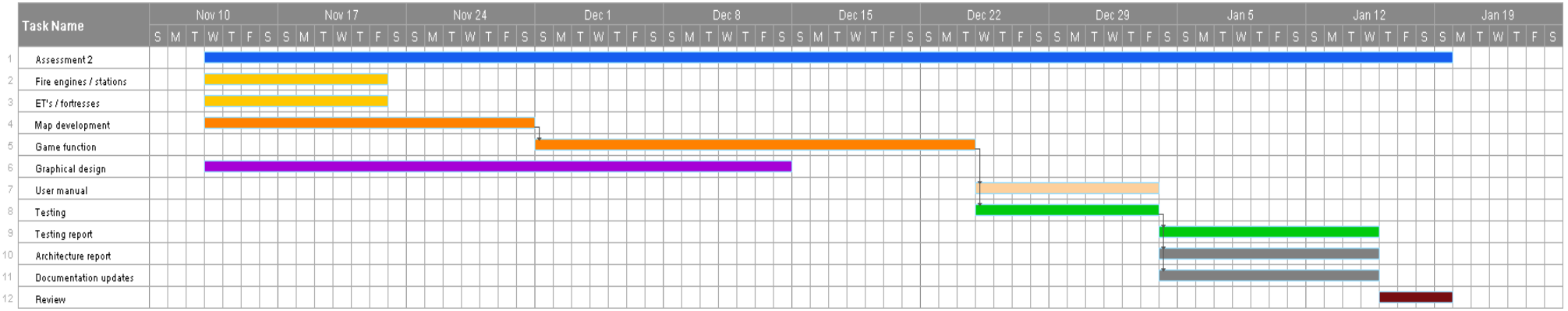
We set out an abstract team structure with the following roles:- Jack:PM, Peter:QA, Will:Art, James:Art/Dev, Max:Dev/WebDev, Yuqing:Dev. Roles were picked based on aptitude in given areas, for example, Jack took charge early on and showed good leadership skills, Peter was the first to implement JUnit tests etc. These are non-specific for a reason, for example, if people are needed on a specific task that does not generally fall under their role they will still be able to do it. Therefore, we delegated individual tasks based upon who prefers doing what and who thinks they will be best at particular things. This will result in each person putting their whole effort into each task as they don't feel they have been forced into doing an area of the project they aren't comfortable with. Obviously, there is some degree of flexibility when it comes to a plan such as this as we will not always be able to delegate tasks in this way. There may be a time where one person doesn't feel fully comfortable doing any task and this will be dealt with by them doing their most preferred option. This should lead to a higher quality final product as all work done on it would have been done to its best ability. We must also ensure to delegate tasks proportionally to the complexity of it resulting in more people being involved in the more difficult tasks for obvious reasons. It is better to delegate more people to a difficult task at the beginning rather than starting with a small group and adding to it as the task falls behind schedule. This would waste time because in order to add more people to a task they must first be brought up to speed.

Once each subproject is completed, team members from it will be reassigned to a new task in order to aid with it. It is up to the individual in this case to bring themselves up to speed as quickly as possible so as not to run into the same issue as before. The individual in question should aim to disrupt the current developers working on that task as little as possible by catching up through non-verbal forms first (reading docstrings etc) and asking questions only when necessary. For each task there will be a rough plan of what has been done and what is to be done (classes and methods that need to be built for example). This should help us overcome the problem stated above in the previous paragraph.

When we initially plan out the subtasks for each given section, we will deliberately make our internal deadlines early. This will give us some breathing room if we come across any unanticipated blockers. Some people may see this as a pointless endeavour; why plan if you aren't going to follow it? We see it as a guideline for our own internal schedule as we are aware that in the real world, plans change and so we need to be ready to anticipate that. This meshes well with the project as a whole as we anticipate change and therefore making rigid plans without any flexibility wouldn't be a smart idea.

We will aim to have a transparent working environment. This means that each member is clear about what sections they are working on, how their progress on that section is going and if any issues have arisen that are difficult to overcome. Trello has helped to manage this by enabling us to see what tasks are in progress at a glance and see who is doing what. It is therefore important that each team member keeps this updated as otherwise the tool is useless. A large portion of transparency is frequent communication. Meetings are and will be held regularly over Discord and quick questions can be asked at any time of day via our group chat. Discord meetings are a good form of communication as we can all communicate verbally which enables us to get ideas across much faster than we would via a text-based communication service. It is important however, that we don't forget about our group chat. This is a much easier way of communicating, without requiring a scheduled meeting where everyone must be free to talk; questions can be asked and answered while on public transport for example.

Assessment 3: Selection, Extension and Integration, due: 17/02/20



The first phase of assessment 3 will be the evaluation and selection of project to develop further with known requirements. The ideal priority of evaluation should be around the ease of implementing the further features, with other focuses being completeness of the project, documentation quality and the overall standard. This process will occur between 20/01/20 - 27/01/20, during which we will also be doing a presentation. We must ensure to choose a project who's codebase is clearly laid out and has substantial progress towards the final release whilst also selling our project well.

During the second phase of extending the project to include all the requirements, the features that will need to be added are: two or more fire engines, three or more fortresses, the minigame, patrols, scaling fortress strength and possible destruction of fire stations. Furthermore, any incomplete requirements from assessment 2 will need to be completed first, which should take priority as there will be dependencies. The minigame will be delegated to a small group and other features will be individual. A change report will document any necessary alterations to the base project. If we are left with any time after completion of the core game requirements we can add other features to enhance the product, we must however ensure that these additions do not stray from the brief and are checked off with the client.

Assessment 4: Selection, Requirements Change, due: 29/04/20

In a first phase of assessment 4 we will be evaluating and deciding on a project to alter. The changes made to this project will be unknown in this phase, so the project chosen should be easily adaptable. Alongside adaptability, overall quality, documentation and adherence to the current brief. This process will occur between 17/02/20 - 24/02/20, and we will present our project to other groups at this point.

For the second phase where we will be implementing features to satisfy new requirements, the requirements will need to be reviewed at the time. In doing this we will be able to form a strong development plan. A presentation to a client will be devised following the deadline to be presented on 17/05/20.