Software Engineering Project: Assessment 1
Gandhi-Inc. - Project Blind World

Methodology

Outline of Methods

The team has decided on adopting Agile for a software development methodology. This methodology has a backlog of requirements and generally concentrates on 2-4 week periods called sprints before reaching a set target. An outlay is shown in figure 1. The product is frequently reviewed, tested and adjusted where necessary throughout the sprint process. Scrum is an agile process that allows the team to concentrate on delivering a quality product in a set deadline. Our team is going to base our methodology on Scrum as with fixed submission dates and a list of prerequisites this would seem most fitting.

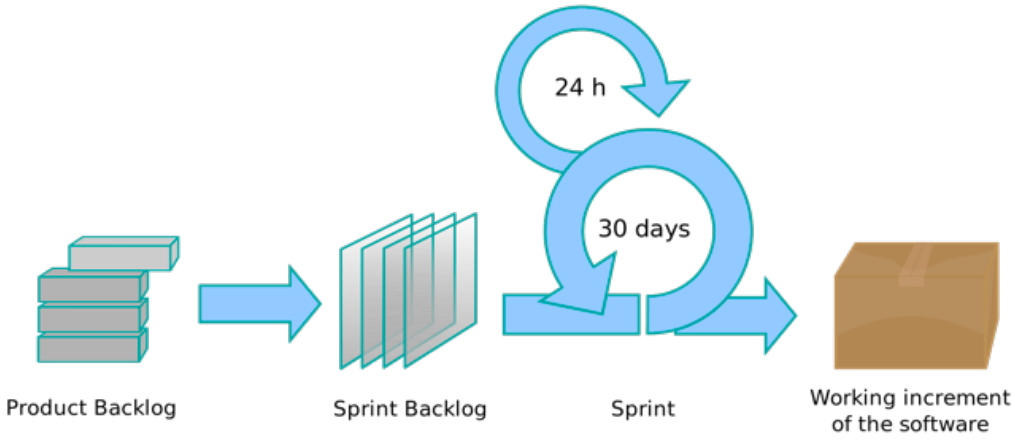

Figure 1[1]: an illustration describing the Scrum process.

The team can be subdivided into smaller groups working on different tasks and then coming together every week to realign on the project goals and make sure we are on track to reach our targets. This allows the team to self manage and to prioritize key features. Our goal is to integrate new features based on our systematic plan in each sprint, this way the focus can be on enhancing the project for further iteration.

The scrum process is based on a product backlog, this is divided into different stages which can then be put into the sprint process. Our list of requirements and our planned schedule make up this product backlog. By using the scrum process, we aim to facilitate the group workflow.

The team looked at alternate approaches including the waterfall methodology however it was concluded that not having an idea of the deliverable until the final deadline can often lead to many confusions with the requirements and make the product harder to sell at the end of the second deliverable.

[1] Lakeworks, *The Scrum project management method*. 2009 [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/5/58/Scrum_process.svg/2000px-Scrum_process.svg.png [Accessed: 25-Oct- 2016]

Collaboration Tools

The team has chosen to use Java[2] as the default programming language. The team members have all had previous experience in Java in the first year at university. Java is a high level language allowing the team to create a range of classes and objects. When implementing graphics, the aim is to use a Java API to help us use this.

The team choose Github[3] as a repository solution to store all of our code. Github allows users to collaborate together on a project and to commit and push new work to the master repository. Github also keeps a local copy stored on any devices that sync with the master. This allows us to work on the project offline and not have to rely on the Github servers if something were to go wrong.

Github education pack includes a free domain through Namecheap[4]. The team therefore registered the domain *www.gandhi-inc.me* and pointed it to Github's free hosting. The static website was then drafted on a Jekyll[5] template. Jekyll is written in Ruby[6] which is very light, and great for static websites.

Google Drive[7] and Google Docs[8] were chosen to store our documents and files, this was a unanimous decision as the University of York supplies its students with a student account linked to the university email, with unlimited storage. Google Docs also allows for multiple users to edit a document at the same time, which means all of us could be working on one document together. A nice feature is also the comments and suggestions, that allows a member to go through someone else's work and suggest and edit to the group.

The team needed an easy way to communicate, and seeming as though all the members had smartphones, it was clear an app with push notifications was the way to go. As all team members had a google account through the university we decided to use Google Hangouts[9]. It is accessible from all smartphones and through the web platform.

Tiled[10] is a tile based map editor which is all open source. The team have decided this would be a great tool to use to make a 2D map of the university campus. Any PNG textures can be easily implemented in this map editor; we will be sourcing ours primarily from Open Game Art[11].

---

[2] Oracle Corporation. *"Java,"* www.java.com. [Online]. Available: https://www.java.com/en/. [Accessed: Oct. 28, 2016].

[3] GitHub. *"Github,"* github.com. [Online]. Available: https://github.com/. [Accessed: Oct. 28, 2016]. https://github.com/

[4] Namecheap. *"Namecheap,"* www.namecheap.com. [Online]. Available: https://www.namecheap.com/. [Accessed: Oct. 28, 2016]. https://www.namecheap.com/

[5] Jekyll. *"Jekyll,"* jekyllrb.com. [Online]. Available: https://jekyllrb.com/. [Accessed: Oct. 28, 2016].

[6] Ruby. *"Ruby,"* www.ruby-lang.org. [Online]. Available: https://www.ruby-lang.org/en/. [Accessed: Oct. 28, 2016]. https://www.ruby-lang.org/en/

[7] Google. *"Google Drive,"* www.google.com. [Online]. Available: https://www.google.com/drive/. [Accessed: Oct. 28, 2016].

[8] Google. *"Google Docs,"* docs.google.com. [Online]. Available: https://docs.google.com/. [Accessed: Oct. 28, 2016].

[9] Google. *"Google Hangouts,"* hangouts.google.com. [Online]. Available: https://hangouts.google.com/. [Accessed: Oct. 28, 2016].

[10] T. Lindeijer, et al. (2008). *"Tiled".* Tiled Map Editor [Online]. Available: http://www.mapeditor.org/. [Accessed: Oct. 28, 2016].

[11] Open Game Art. *"OpenGameArt.org,"* opengameart.org. [Online]. Available: http://opengameart.org/. [Accessed: Oct. 28, 2016].

Organisation

The team was organised in such a way that the deliverables were distributed across members, where a pair or a trio would work on a single deliverable when possible. At other times, deliverables would be developed by a single team member. The team member chosen for each section was chosen because of their particular skill set, eg software development or risk assessment.

Firstly, due to the rushed nature of the project, it was deemed necessary for the workforce to be utilised as much as possible[12], in order to make the best use of time. The obvious consequence of such method, however, is that merging developed content from separate sources can result in clashes related to contrasting expectations and opinions. This issue was addressed by having frequent team meetings throughout the week, where members can discuss each other's work while updating their own content at the same time, ultimately synchronising their progress [13]. It was also possible to remediate the lack of synchronicity through usage of the collaboration tools aforementioned[14].

The team found that the most reliable way to produce reasonable amounts of content per week is to organize regular meetings. Though there was no formalised objective regarding how often such meetings should happen, the team often met three times throughout the week, including the compulsory practicals. It was found that for Assessment 1, this sort of approach, including work that was done outside of such meetings, generated enough workload to have the deliverables equally developed across team members, and at a pace steady enough such that no crunch[15] would be made necessary close to the deadline. Obviously, crunching is naturally undesirable.

With regards to the team itself, the chosen approach also works well due to the fact that the bulk of the synchronisation depends on three members at the most: as a trio works on a single deliverable, it is usually easier to reach a consensus that only depends on three individuals rather than five, which is the total amount of team members. As a matter of fact, this makes it so that it is often (if not always) possible to collaborate remotely [3], and later on making use of the meetings to practise peer-reviewing on top of the other members' work, ultimately leading to a globalized team-wide agreement. Furthermore, in the event that a single member works on a section of the project, we believe that lacking the need to spend time with organising grouping arrangements (as pairs and trios did) contributes towards increased individual productivity. In addition, a carefully thought-out choice of which team member would work independently can result in increased productivity, especially if such member has a particularly high level of experience related to the assigned task, if compared to others[16].

For further stages of the project, it is likely that this same approach will remain appropriate, as frequent meetings are compatible with the Scrum methodology. It is expected, however, that the implementation phases might spawn several micro-tasks, which wouldn't be undertaken by more than one team member, but spread out across the team. The necessity of merging work together would then be handled by both our source control methods, and team meetings.

---

[12] G. Wigglesworth and N. Storch, "Pair versus individual writing: Effects on fluency, complexity and accuracy" *Language Testing*, vol. 26, no. 3, pp. 445-466, July 2009.

[13] G. Parker, *Team players and teamwork*. San Francisco, CA: Jossey-Bass Publishers, 1990.

[14] J. Nunamaker, et al., "Principles for effective virtual teamwork" *Communications of the ACM*, vol. 52, no. 4, pp. 113-117, April 2009.

[15] J. Brown, et al. (2004/2005). *"Crunch Mode: programming to the extreme"*. Stanford Computer Science [Online]. Available: https://cs.stanford.edu/people/eroberts/cs181/projects/2004-05/crunchmode/index.html. [Accessed: Nov. 02, 2016].

[16] G. Hill, "Group versus individual performance: Are N?+?1 heads better than one?" *Psychological Bulletin*, vol. 91, no. 3, pp. 517-539, May 1982.

Planning for the Future

Following the Scrum methodology, the team shall first identify which tasks should be prioritised in each stage, that is, the ones which have the highest amount of dependent tasks. The amount of marks awarded for each deliverable shall also be considered while defining priorities. Concerning Assessment 2, the implementation deliverable is the most important one: the GUI report, the testing report, the implementation report, the website, and some of the architecture depends on it. As such, the team will prioritise the completion of the implementation, while still working on other tasks which do not depend on it.

The schedule for the execution of Assessment 2 shall be the following:

Task (in decreasing order of priority)	Earliest starting date	Latest finishing date
Implementation	09/11/2016 (by noon)	10/01/2017
Architecture report	09/11/2016 (by noon)	17/01/2017
Updates on Assessment 1	09/11/2016 (by noon)	24/01/2017 (by noon)
Implementation report	10/12/2016	24/01/2017 (by noon)
Testing report	01/01/2017	24/01/2017 (by noon)
GUI report	01/01/2017	24/01/2017 (by noon)
Website	10/01/2017	24/01/2017 (by noon)

Figure 2: a table illustrating the proposed plan for Assessment 2.

As we have created a very abstract architecture in the design stage, we feel that creation of a Gantt chart or task dependency diagram would be unmerited. Instead, we have used a very general overview of when we need the main tasks to be completed to finish on schedule.

As stated previously, the implementation deliverable shall be prioritised. Its earliest starting date will be the day Assessment 1 is submitted, but its latest finishing date was chosen to be two weeks before the deadline for Assessment 2: the team needs some dedicated time to work on the deliverables which depend on the implementation being as complete as possible. In any case, the earliest starting date for such dependent tasks, like the implementation report or the testing report, was set to later phases of the implementation, when it is expected that there will be enough implementation content developed to start a reasonably adequate report.

Regarding tasks which mostly do not depend on the implementation, their earliest starting date was set to as soon as possible, like the architecture, for instance. However, after the implementation is finished, we might need to go back and alter some of our architecture due to development issues, or possible changes of plans. Consequently, the latest finishing date for the architecture was set to be a week after the implementation is done, so that we can make the last definitive changes before submission.

We can see that the critical path of Assessment 2 would take exactly 76 days to complete.

The team aims to follow a similar model for future assessments, by deciding which deliverable(s) should be prioritised and then building a schedule on top of that. It is expected that, like Assessment 2, future assessments will also require the implementation to be prioritised, due to it having many dependent tasks.