Project Review Report

Team 12

April 2020

1 Approach to team management and structure

At the start of the project the team tried to adhere very strictly to the team management practices from the Lean Software literature[1]. Some of these were better suited for our group than others, so over the course of the four assessments we have adapted our approach, maintaining the best elements of a lean approach to management and planning, and replacing others.

Initially the idea was that all members of the group would assign themselves to tasks from the AzureDev taskboard. Trying to avoid overplanning, we were skeptical of the Gantt chart, and split the development into a series of 'milestones', which acted as internal deadlines. Each milestone would be paired with a customer demo and interview. The customer interviews would serve to ensure the group was meeting requirements, and also keep us accountable to meeting our own internal deadlines.

Over the course of the module the team has developed and matured. Through the course of working together we've become more aware of the strengths and weaknesses of each team member, which has made collation easier, and organising the group more natural. As a result, our approach to the following best team management practices has grown more lax, and we've worked together more on intuition than textbook practices. Another change we've faced is more of the team's communication being done over the internet than in person, due to the spring break falling in Assessment 4 and the exceptional circumstances in Term 3.

By Assessment 4 we were using a loose Scrum method, as discussed in the Software Development and Tools section. Weekly meetings had remained a key part of the team's coordination, but their role had changed. The scrummaster would now assign work to team members for the week, after progress had been reported. This made it simple for team members to know what was expected of them, in contrast to the old system. Assigning the best person for the task was now much easier, as we had a good understanding of the different skills and competencies of each member. We continued using the AzureDev taskboard for making it clear what everyone needed to work on, but the bulk of our communication took place on Facebook Messenger, which we found to be ideal for a team of our size. We embraced the Gantt chart at this point, not because we found it helpful during development, but because the process of creating a Gantt chart forced us to consider task dependency during planning, and because it was a good way to communicate with our customer. A large factor in our decision to use Gantt charts and Facebook messenger was that SEPRet Studios had used them before us. For the Gantt chart, this made it easier for us to follow the plan they had already laid out. Despite moving away from Lean we continued using milestones as a way of setting internal deadlines. Unfortunately we couldn't pair each milestone with a customer meeting, which meant we didn't get as much feedback as would have been ideal. In response to more of the communication taking place over the internet, our weekly meeting became a video call. This generally achieved the same goals as meeting in person, but some aspects were more difficult, such as explaining technical concepts and resolving issues with software. If the team had to continue working in this format, we would explore other telepresence software and ways of working over the internet.

2 Software engineering development methods and tools

Our team initially used the lean software development, which is based on agile methodology. This primarily focuses on seven main principles as outlined in our plan for assessment one (found here). The choice to use this approach was made because the principle of eliminating waste allows for maximum productivity of the team so for example, implementing unnecessary items or engaging in activities that add no value to the project are avoided. Moreover, the iterative cycle structure involves the delivery of a prototype of the product to the customer so that we can continuously make changes based on regular feedback. It has been shown that having regular cycles of refactoring is a good way of giving every team member to learn and solve problems more effectively [1]. We also decided to use the Azure DevOps Platform tool to organise and allocate tasks to members, make sprints and set deadlines for tasks. This tool was chosen over the others because of its wide range of Agile features and having the ability to integrate the GitHub repository which we selected to store our code. Github, Azure DevOps and Gradle were the tools we had kept for the entire duration of the project as they were easy to use and maintain.

Although our group had stuck to the implemented plan created in assessment one initially, as the project progressed, we ended up dropping the idea of everyone taking part in the code. This was due to the observation of everyone having different strengths which could be applied to other parts of the assessment such as testing of code and the write-up. Another issue which appeared was regarding writing the tests before the code was completed- it proved difficult to predict how the software would be implemented so there was no clear way to structure testing. Although, some progress was made with black box testing since it mostly relies on the end result rather than the internal workings of the code.

After this section of the project we found that applying the lean methodology was too free form in terms of having to make design decisions as late as possible and rigid in areas such as writing tests early as well as producing prototypes for the customer on a regular basis. As every member of the team agreed that we were facing challenges in following the lean principles, we decided to follow Scrum methodology. This method was also taken by Mozzarella Bytes, whose project we had selected to further develop. After reading their documentation on how well it has been working for their group and careful research, we made the decision to change. Scrum also has agile development as its foundation and places strong emphasis on team collaboration. Its framework involves building a product backlog in which a list of tasks to complete are ranked based on their priority. After completing a sprint, the positive and negative occurrences are reviewed to learn what strategies could be improved in the next sprint. [2]

Adoption of the Scrum framework led to productive discussions about the architecture and complex user requirements which ultimately made the team better prepared on how to implement code for these sections. Moreover, as the methodology was focused on meeting objectives rather than specific tasks, every group member had the opportunity to self-manage their deliverables. Weekly sprints allowed consistent communication and transparency of problems or anticipated risks. Overall, the success of applying Scrum principles was demonstrated by a significant increase of marks for assessment three compared to assessment two. For this reason, the same approach was taken for assessment four. It was initially followed well, however, over time maintaining weekly sprints was challenging due to group members unable to meet after University closure. Alternative virtual meetings took place when possible, but this did not assist in when some individuals who faced technical difficulties in operating IntelliJ. As this fell into the "R _RESOUCRCES" category of the risk register created in assessment one (found here), we followed the necessary mitigation of reorganising responsibilities to other members of the group and ensured good communication was kept throughout.

2.1 References

[1] M. Poppendieck, T. Poppendieck, Lean Software Development: An Agile Toolkit, Canada, Addison Wesley, 2003

[2] P.A.G. Permana, "Scrum Method Implementation in a Software Development Project Management", International Journal of Advanced Computer Science and Applications, Vol. 6, No. 9, pp. 198-200, 2015