University of York

Department of Computer Science

SEPR - Assessment 4

Project Review

Team Craig

Thomas Burroughs

Huw Christianson

Joseph Frankish

Isaac Lowe

Beatrix Vincze

Suleman Zaki

Team Management and Team Structure

The team's structure and management for the end of the project has remained similar to Assessment 3. Limited changes were made to the team's structure from Assessment 3 as we believe that this system maximized collaboration and minimised the risks encountered as a result of poor team management in Assessment 2. For the final assessment, the team has continued to implement a core leader who is responsible for managing sprints, leading team meetings and assessing the performance of team members. The team have continued to implement 'experts' also known as deliverable leaders. Each expert is responsible for ensuring their deliverable is completed on time, as well as having the authority to make executive decisions. Each team member has a concrete role for which they have developed significant skills over the project. This means that the team's structure is essentially completely concrete with limited flexibility to change roles. Despite this, no one individual is responsible for the entirety of a deliverable. This has been implemented so there is a greater crossover of work and team members have some understanding of each other's roles [1], reducing the risks of delays should a team member fall ill.

Although limited changes were made between Assessment 3 and 4, the team's structure and management has evolved significantly throughout the project. This has reduced the potential for risks, such as team conflicts, and also allowed team members to engage in self-development. The switching of roles early on has helped productivity and ensured that team members take on suitable roles. For example, in the Assessment 1 plan, some team members took on programmer roles based on their desire to improve their programming. Relatively new programmers were responsible for the bulk of the programming in order to drive their self-development. However, in Assessment 2 it became clear that the learning curve was too steep and the teams roles were adjusted. From this point, the team decided to structure the team with greater emphasis placed on matching roles to team members' strengths, with less importance placed on self-development. The team made use of the suggestions of Sommerville, who believes that team members strengths should be recognised and exploited to maximise productivity and quality [1].

The limited flexibility of the team's structure is a result of the project's nature. At the beginning of the assessment each member's knowledge on each deliverable was similar, however, as the project progressed team members became more knowledgeable on their own tasks, and a clear knowledge gap was created. The short time frame for each assessment meant that a dramatic change in roles would result in time being wasted learning new tools and reading previous documentation on that work package. For example, switching between a documentation and testing role between Assessment 3 and 4 would prove difficult due to the fact that the knowledge and experience gained from testing in the previous assessment is unlikely to be learned in the time frame for Assessment 4.

The clearest evolution in our team's structure was the emergence of a natural leader. Initially, as can be seen in our Assessment 1 Method [2], the plan was to share the leadership role in a self-organising team. A different team member was supposed to take on the leadership role if their skills most suited the deliverables at that given time [3]. For example, if the assessment is programming heavy, the plan was for the main programmer to take on the lead role. However, the lack of a concrete leader led to confusion on who was responsible for resolving conflicts and manging the team. This resulted in team members passing off responsibility and blame to other members who were unaware of their leadership responsibilities. To combat this issue, the team decided to elect a leader at the start of Assessment 2 and follow a strict management style. Furthermore, to avoid confusion over responsibilities, the team decided to define and document the roles of our team more heavily after Assessment 1 so we could identify the contributions by each team member and learn what missing aspects could hinder our results [4].

The greatest evolution to the team's management was made between Assessment 1 and 3 during the holidays. We planned to manage the group as a whole, as we had done for the first half of each assessment. This included continuing weekly meetings via Google hangouts and keeping weekly sprints. This plan was to be reinforced by a Gantt chart to help avoid delays and guide the team when working remotely. It was difficult to organise meetings and work collaboratively on the same tasks due to our differing schedules, resulting in less work being produced. Team members would simultaneously work on the same deliverable, producing the same deliverable twice, this wasted time, effort and resources. To combat this, we evolved our team management over Easter to avoid the same problems. Team members only completed work which didn't require much collaboration and were independent from other tasks. Workloads over Easter were reduced to account for revision and the fact that most deliverables require collaboration. In hindsight, during these breaks the team should have decomposed work packages into separate autonomous segments which could be worked on individually, to form one collaborative piece.

Software development methods and tools

To complete this project, we have continued to use an agile development method, more specifically an adaption of the Scrum methodology, as described by Sommerville [5]. The continuation of this method was approved by the team because of its appropriateness for small teams with short deadlines [6], its flexibility to embrace changing requirements and our success with it throughout the project. We have continued to have two meetings between each one-week sprint with an additional third meeting every fortnight, dedicated to managing transpiring risks. The longevity of our chosen method is due to its easy implementation and flexibility, which means that even with changing requirements and risks, the methodology can be adapted to provide the necessary structure we desire.

Our methodology evolved little over the project due to our extensive research of methods, mainly through the reading of Sommerville's Software Engineering [5], and the project itself. At the start of Assessment 1, we made our initial changes to the chosen Scrum methodology. Instead of short daily meetings, we had longer weekly meetings as this was more suitable for the less intense nature of this project and the team's university schedule. Originally, for Assessment 1 the sprints were two weeks long with one meeting each week. But, from the second assessment, instead of one weekly meeting, two weekly meetings were scheduled, and sprints were reduced to one week long.

The changes between Assessment 1 and 2 were made so our method and tools better suited the working and leadership style of our team. It became clear during Assessment 1 that some team members did not communicate well using our chosen communications tools, mainly due to a lack of confidence. Furthermore, some team members did not work well when having to self-manage their work due to a lack of motivation and worked better when assigned tasks directly at weekly meetings. Therefore, by having two weekly meetings and introducing audio-based communication tools, such as Google hangouts, a greater turnover of work was achieved from those team members. Changing requirements and risks also led to the need to increase the number of weekly meetings. An additional risk meeting was added during Assessment 3, which took place every fortnight to allow the team to make changes to our risk table and put in place any mitigation plans. For example, Risk 18 [7] transpired during Assessment 3 due to the change-over of projects, and the associated mitigation plan was discussed in our fortnightly risk meeting. This prevented future testing from being delayed and incomplete requirements due to faulty code.

The changeover of projects in Assessment 3 and 4 provided an opportunity to switch to another Software engineering methodology by following our chosen group's documentation. However, due to our success with our method we felt that any benefits that switching may provide was overshadowed by the time wasted learning the new method. The iterative nature of Scrum meant that any major changes that had to be implemented could be discussed and potentially re-planned at the weekly meetings. Although Scrum promotes the limited use of documentation, we did introduce a change table to use in our meetings to assist in the transition of projects. We introduced this on the recommendation of IEEE's standard for configuration management [8]. This allowed the team to assign further responsibilities at each Scrum meeting and thus the team felt more up to date with changes.

Over the project, few of our development tools changed due to our effectiveness using each tool and our wish to reduce any time wasted leaning new tools. For example, in Assessment 3 we chose not to swap our diagram design tool, Lucid Chart, for our chosen team's StarUML tool, despite its improved readability, due to the learning curve required. On the other hand, further tools were added when the specification differed from the previous assessment. For example, in Assessment 1 the team was not aware of all the necessary development tools needed for the implementation of the game in Assessment 2. This led to the addition of our tiled map editor, Tiled. Another reason for the evolution of our development tools was the changeover of projects for which the new game required a continuation of specific tools to satisfy unfulfilled requirements. For example, in Assessment 3 our chosen team required us to use 8-bit graphics for characters. As our team had not yet needed such a tool, it seemed appropriate to use our chosen team's choice of graphics tools, Gimp, as this would provide the most seamless transition.

An example of where our tools changed in accordance with the team's understanding was with our version control tool, GitHub. A few members who had not worked with version control before, found it difficult to use the command line. However, since Git is widely used and easily accessible the team did not want to switch to a new tool, foreign to all team members. To solve this issue, the team introduced GitKracken which provided an interface which is more intuitive. Despite this some tools remained unchanged, such as Whatsapp, LucidChart and Microsoft Excel due their comprehensiveness and appropriateness for the project.

References:

[1]- I. Sommerville. Software Engineering. Pearson, tenth ed, 2016 pp. 75-76

[2]- Team Craig. *Method Selection and Planning- Assessment 1*. Available: https://teamcraigzombie.github.io/assets/downloads/Plan1.pdf

[3]- NB. Moe. *Understanding Shared Leadership in Agile Development: A Case Study.* IEEE, Jan 2009.

[4]- J. Manjovskoya. *How to Set Up Software Development Team Organization That Will Kickstart Your Business.* Sep 2018. Available: https://www.daxx.com/blog/development-team/set-up-development-team-kickstart-your-business

[5] – I. Sommerville. Software Engineering. Pearson, tenth ed, 2016, pp. 85-86.

[6] – L. Rising et al. *The Scrum software development process for small teams.* Journal: IEEE Software, Volume 17, Issue 4, IEEE, pp. 26-32.

[7] – Team Craig. *Updated Risk Assessment 3*. Available: https://teamcraigzombie.github.io/assets/downloads/UpdatedRiskAssessment3.pdf

[8]- IEEE. *IEEE Standard for Configuration Management in Systems and Software Engineering.* IEEE, March 2012. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6170935