Evaluation and Testing Report

Stakeholders: Richard Paige, University of York Communications Office
Team: Barney Morgan, Cameron Smith, Harry Berge, Jake Phillips, Matthew Wilkie, Rob Weddell

Justification/Evaluation:

Before any evaluation took place, it was deemed important to do some research into what factors make a successful project. This research found that four key factors generally come into evaluating the success of a project: Schedule, Quality, Stakeholder Satisfaction and Team satisfaction [1].

In this case, the team believed that the most important factor driving the evaluation of the project's success was the stakeholder satisfaction. Thus, in order to effectively measure this satisfaction, the team ensured, to the best of their ability, that the final product met the requirements of the initial and updated brief [2] dictated by the stakeholder themselves. In order to achieve this, it was deemed essential for the team to consistently be referring to the requirements elicitation documentation in the previous assessments [3]. In order to efficiently track the mapping of features to requirements, the team maintained a clear and concise traceability matrix [4]. At the start of assessment four, during the distribution of tasks in the first SCRUM meeting, the team went through the traceability matrix and evaluated whether each requirement had been accurately met by the work of previous teams. In addition to this, extra requirements were added to the table as requested in the updated assessment brief. At this point, the remaining implementation tasks were distributed among the team.

This was deemed to be an almost completely comprehensive evaluation of the success of the product in the stakeholders view, and, with the traceability matrix as proof, the team could confidently say that the final product fulfilled the requests of the clients. However, it is also expected that the client may be unhappy with the team's interpretation of the initial, rather vague, requirements. In order to mitigate these differences of interpretation as much as possible, the team made sure to correspond with the client after making the initial requirements document [5]. Also, in further attempt to mitigate this, the requirements were consistently updated throughout the three previous different stages of assessment in response to feedback from the client to ensure they were as accurate as possible in reflecting their vision.

However, it is important to note that the success of a product does not entirely depend on whether the client's needs are satisfied. The team believed it was important to be satisfied with their own product in and of itself. This was evaluated rather informally through casual play testing throughout the process of development. The team believed this was very important, as it is completely possible for the game to fulfil every requirement but still be a failure. It was concluded that, in the eyes of the user, enjoyment and player satisfaction is the most important factor to judge the success of a video game [6]. Thus, to measure this, the team found willing participants to play-test the game and gave them a quick questionnaire afterwards to gauge their enjoyment and understanding of the game [7]. The team also completed similar questionnaires themselves in order to document their own satisfaction with their game [8]. In order to avoid a possible bias corrupting the results, these questionnaires were kept separate from the user questionnaires and simply used to evaluate the development teams own feelings. However, with all of the aforementioned evaluation techniques, it was possible to estimate that there was a positive overall satisfaction with the final product after some slight alterations in response to the playtest feedback [9].

In addition to this, the team also decided to judge the quality of the project based on how well the typical project management practices were followed, as well as the quality of the deliverables themselves. Due to the comprehensive documentation and research that went into following professional project management practices, the team can comfortably say that the practices were followed to the best of their ability and to the highest possible standard - and the feedback and marks from previous assessments have proven this. This was especially important for this project as the aim of the entire SEPR module is to teach and enforce

good quality software engineering project management practices. Moreover, by proving both user, client, and developer satisfaction with the finished product, it can confidently be said that the entire project was completed to the highest quality possible.

Lastly, throughout the entire assessment, the team followed a rather relaxed Agile method in order to keep on schedule throughout. The potential risk of falling behind schedule was well documented from the start [10] and, whilst there were some issues during the start of the project with completing everything before deadlines, the later portions of the project were completed on time to a high standard. This was due to the meticulous planning done during assessment one in which multiple gantt charts were created [11] to ensure that the team stayed on schedule. However, following the addition of new requirements for assessment four, new scheduling risks arose (risk 22) [10]. This put quite a lot of pressure on the team to ensure that all the new requirements were completed before the final deadline whilst still leaving enough time for comprehensive testing and evaluation, however these risks were also well documented and thus avoidable. Overall, this massively contributed to the success of the project as it ensured all tasks were completed punctually to the highest standard.

Testing:

Before being able to accurately test and evaluate the quality of the code, some research needed to be performed into the qualities and aspects that make up good quality software. It was found that the ISO 9126 software quality characteristics [12] was the most comprehensive resource as it follows the international standard drawing from previous work done by McCall (1977) [13] and Boehm (1978) [14]. This model identifies six main characteristics: Functionality, Reliability, Usability, Efficiency, Maintainability, Portability. This was deemed a very reliable method of measuring quality given its extensive history and popularity. However, regardless of its reputation, the team analysed the document together and determined that it was fit for use. It covers the six main characteristics of software comprehensively and in full, whilst explaining in simple terms how to apply the method to one's own work. From this, the team attempted to map each characteristic to aspects of the code in order to provide proof that the teams own software fits the criteria of 'good software'.

Characteristic	**Definition**	**Evaluation**
Functionality	The essential purpose of the product. This characteristic determines whether the software does what is expected to an acceptable degree and works in union with other components of the software.	- Tested via the traceability matrix linking requirements to features which provide the required functionality - Black box testing was completed to directly evaluate the functionality of the software with a list of predefined tasks and expected results. - Play testing and questionnaires
Reliability	The capability of the software to maintain its service under defined conditions for defined periods of time. This characteristic essentially measures the fault tolerance and occurrence of errors / bugs.	- Unit tests for the code were created and used to perform white box tests. - Game compiles into a .exe without any errors. - Game runs upon clicking the .exe without any errors. - All major errors and bugs in the code have been dealt with during development.
Usability	Relates to the functionality, but focuses on the ease of use of the given functionality. This	- User questionnaires evaluated how easy the participants found the game to learn, and attempts to evaluate how much they enjoyed

	characteristic determines how intuitive the software is to use and measures the learnability of the system.	their gameplay. - Casual playtesting throughout by the developers ensured that the game was always in a playable state which is intuitive to the user. - Small tutorial screen at the start of the game teaches the player the basic controls, however more is learnt through gameplay.
Efficiency	How effectively the software utilizes the system resources to provide the required functionality. This characteristic is typically measured by the amount of memory / disk spaced used up by the software. It also ties in with the usability of the system as the usability is influenced by the system's performance.	- The game loads up and runs smoothly on a variety of different systems such as: PCs in the software labs, PCs in the hardware labs, individual laptops and PCs of the developers at home - Throughout development, the game was optimized as much as possible to reduce lag within the game by ensuring algorithms were coded efficiently.
Maintainability	The ability to identify and fix software faults with ease. This characteristic is heavily impacted by coding practices such as modularization and good documentation.	- Modular object oriented code means that changes to one area of code can be done without impacting other aspects of code. - Code is well documented and commented throughout to ensure that an outsider (with programming experience) could understand it.
Portability	How easy it is to adapt the software to a changing environment or its requirements. This characteristic essentially measures the adaptability of the program. Modularization also impacts this.	- The project has gone through multiple assessments and handovers already. All of these required good documentation and traceability which have been provided. - As previously stated, the modularization means that the program is easily adaptable.

Requirements:

As stated earlier in this document, the team believed that ensuring all requirements were adequately met was a very important aspect of producing a successful project. The evaluation of requirements was completed within a traceability matrix [4] which linked the requirements to features within the game and provided evidence that they had been tested for.

However, despite the teams best efforts, it was neither possible nor viable to complete the implementation of all requirements due to various factors such as: time constraints, prioritization of more important features, and slight alterations to the game. The following requirements could not be implemented:

Req 2.16 - The addition of collectable items was decided to not be included in the game. Instead, the function that items were going to fulfil have been passed onto the crew members (ship/ weapon upgrades). This was mostly due to the lack of development time, resulting in only the essential features being implemented. The team felt like there was no need to implement an alternative method of upgrading the ship at this point in time.

Req 2.17 - The integrated tutorial is not as fleshed out as the team would have hoped. Rather than having an interactive tutorial which teaches the player through gameplay, there is a temporary screen introducing the player to the world and showing them the controls. It is assumed that because the controls are relatively simple, the player will quickly be able to pick them up on the go.

Requirements: https://sepr4.github.io/web/submission/assessment4/updated/Req4.pdf

References

[1] M. Freeman, P. Beale, “Measuring project success”. Project Management Journal, 23(1), 8–17, 1992.
[2] SEPR Module 2019, Initial and Updated Product Briefs [Online]. Available: https://sepr4.github.io/web/submission/assessment4/Brief.pdf [Accessed 03 Apr. 2019]
[3] Element of SEPRise!, Assessment 4 Requirements [Online]. Available: https://sepr4.github.io/web/submission/assessment4/updated/Req4.pdf [Accessed 03 Apr. 2019]
[4] Element of SEPRise!, Updated Traceability Matrix [Online]. Available: https://sepr4.github.io/web/submission/assessment4/updated/testing/Matrix.pdf [Accessed 03 Apr. 2019]
[5] Element of SEPRise!, Assessment 1 Client Interview Records [Online]. Available: https://sepr4.github.io/web/submission/assessment1/requirements/ClientInterviewRecord.pdf [Accessed 01 Nov. 2018]
[6] C. Klimmt, C. Blake, D. Hefner, P. Vorderer and C. Roth, "Player Performance, Satisfaction, and Video Game Enjoyment", 2009. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-642-04052-8_1.pdf [Accessed 13 Apr. 2019]
[7] Element of SEPRise!, Playtest Questionnaire [Online]. Available: https://docs.google.com/forms/d/e/1FAIpQLSfci8fvca76FzDiSyUNYqx3SRY2a8R1Ol0wtLKF8PrzCkyCKw/viewform?usp=sf_link [Accessed 10 Apr. 2019]
[8] S. Patton, "The Definitive Guide to Playtest Questions | Schell Games", Schell Games, 2017. [Online]. Available: https://www.schellgames.com/blog/the-definitive-guide-to-playtest-questions [Accessed 15 Apr. 2019]
[9] Element of SEPRise!, Both Playtest Feedback [Online]. Available: https://sepr4.github.io/web/submission/assessment4/testing/PlaytestFeedback.pdf [Accessed 20 Apr. 2019]
[10] Element of SEPRise!, Updated Risk Assessment [Online]. Available: https://sepr4.github.io/web/submission/assessment4/updated/Risk4.pdf [Accessed 04 Apr. 2019]
[11] Element of SEPRise!, Gantt Charts [Online]. Available:

[12] ISO9126 - Software Quality Characteristics Standard Overview [Online]. Available: http://www.sqa.net/iso9126.html [Accessed 20 Apr. 2019]
[13] McCall Software Quality Model Overview [Online]. Available: http://www.professionalqa.com/mc-call-software-quality-model [Accessed 20. Apr. 2019]
[14] Boehm Software Quality Model Overview [Online]. Available: http://www.professionalqa.com/boehm-software-quality-model [Accessed 20 Apr. 2019]