Aston University
Engineering & Applied Science

CS2010 Group Project

2016/2017

Quality Evaluation Report

Team 2

Report Authors

Clare Buckley, Siu Chan, Iva Gerogieva, Benjamin Gladstone, Saima Kawal, Deepa Mangrati, William Warren Hardwick

Abstract

In this report, the planning and development employed by Team 2 in creating high quality software and processes throughout the project are evaluated in order to gain a more extensive understanding of how well the team achieved the 'high quality' end result which had been previously specified.

Contents

1. Introduction

Quality Evaluation System is a necessary measurement that is used to assess the process of quality development to show which characteristics of the product needs to be improved. According to Azuma (1996), the quality evaluation process model ISO/IEC 9126 outlines that there are four steps of the quality evaluation process:

1. Evaluation requirement analysis
2. Planning and implementation
3. Measurement and metrics
4. Total assessment

Throughout this report, the V-Shape model (fig. 1) will be used to evaluate the different methods the group used to ensure a quality system. The model represents the development process as considered by the group to manage, control and evaluate the quality of the end game. This model allowed the team to evaluate the project in two different phases; verification and validation. The verification phase included the evaluation of planning, requirements, architecture and detailed design of the project. The validation phase involved testing, system validation, maintenance and improvements that could be implemented in future iterations of development in order to improve quality.

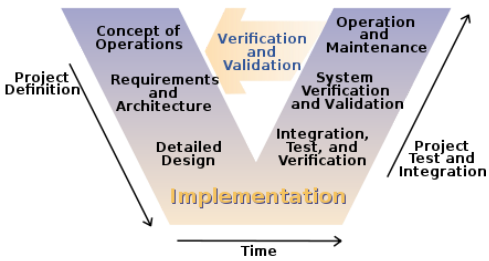


Figure1: *V-Shape(aka U-shape) Model (Azuma,1996)*

2. Planning

The Quality Evaluation System process defines this phase as important as it recognizes the needs for new product development by allowing the developers to have an awareness of areas that can be executed at a higher quality (Azuma, 1996). The group has followed this definition by ensuring quality control throughout the entire development process by defining and following a plan about how the game should function.

The initial plan and pitch (consisting of the team's intentions) served as a useful tool for evaluating the progression of the games development at each milestone. In the concept of operations it was decided that a turn based city builder game should be developed, in which the player progresses through different historical eras. Planning sessions allowed the group to prioritise tasks based on three different milestones of development: the hack, MVP and final product. Smaller milestones also encouraged a quality evaluation at regular weekly intervals by adapting an agile approach to development (see Tutoring and Meetings). The requirement specification in Section 3 (developed in the requirements and architecture stage) gives an overview of these stages.

The MVP provided a general picture of which improvements should implemented to achieve the desired final product. Figure 2 depicts what was expected from the group's members in order to achieve total quality control of the finished game - this was used as an effective means to evaluate the quality of the project throughout the process.

The knowledge gained from a meeting with the project mentor after the MVP showcase forced an essential stage in evaluating the progress and quality of the development in its early stages. From this, the team understood where in the processes undertaken at that time improvements could be made (in terms of both project management and code). This phase is necessary in evaluating system quality as it allows the developers to plan reviews and define achievable measurements to fulfill the design goals

(Azuma, 1996,) - such goals were designed and accepted by the team after reflecting on the mentor feedback.

The following list is taken from the meeting dated 31/1/2017, which founded the basis of this quality evaluation:

Figure2: *Quality evaluation based on mentor feedback*

1. ***Revisit diagrams***
   In evaluating the diagrams as submitted in the quality assurance report, the team took a new approach to documenting processes diagrammatically. To ensure a high quality, all diagrams created were smaller in scope and focus on a specific feature of the system. A higher quality process towards designing the diagrams was also taken, meaning that correct notation was held at a higher importance.
2. ***Attendance at group meetings***
   By enforcing a higher priority of all group members being present at every group meeting, this allowed higher quality processes to be undertaken throughout the project. Whereas initially attendance at meetings was lower than anticipated, after referring to the design team alliance (Team 2, 2016) an evaluation on how meetings should be approached was taken. Due to this change in the team functionality, a higher quality team morale and higher quality development was allowed throughout the remainder of the project, which ensured that the quality described in the quality assurance report was achievable.
3. ***Reflection on solving problems***
   To gain a deeper appreciation of a high quality solution to problems in the project it was required to develop on the initial understanding before meeting with the mentor by devising a method to encapsulate this data. By organising problems into subproblems and sub-systems, this applied a principle-based systems analysis (Bocij et al., 2008). Through this, an iterative method in solving future problems was allowed by applying similar previously defined solutions in order to improve the problem solving process. One example of this improved the quality of software by using similar steps to debug a feature, which saved time in trying a new approach.
4. ***Contribution from members***
   The meeting highlighted a quality of contribution which could be improved on through referring to the strengths and weaknesses of each group member as highlighted by their identified Belbin team role (Belbin, 2016). By reassigning team members to different aspects of the project, the team morale increased and therefore a higher contribution and higher quality end result was allowed.

Despite these changes to improve quality in the project, small changes which deviated from this plan were made to some parts of the game during the development process after the group faced an unexpected problem (see Team Management).

3. Requirements & Architecture

Clarifying the quality requirement and development policy makes the input process a higher quality (Azuma, 1996). Undertaking this feasibility study and eliciting the requirements for each stage of the development process founded the basis for high quality software which is relevant to the end goal. This ensured that any initial problems with the first game concepts were highlighted and addressed before the requirements were brought together for the hack, which resulted in a required understanding of using Unity and C# to implement the game.

Further requirements for the hack, MVP, and final product were elicited through diagrammatically abstracting what was needed from the initial plan, and determining which of those points were feasible considering constraints such as ability and time limits. Ghezzi, C., Jazayeri, M. and Mandrioli, D. (2003) describe abstraction as “one of the fundamental principles of software engineering in order to master complexity”, and through mastering the complexity of what was required, a higher quality end product is

feasible.

By prioritising and rationalising these suggested features, the requirements specification resulted in a plan to produce software of the highest quality for each important milestone of its development (in the context of the time constraints). These requirements were translated into Trello cards and delegated to the most suited group member.

Issues with meeting requirements during the periods leading up to the various milestones suggested unmanaged expectations, and more importantly a low quality approach to processes within the project. After each weekly meeting, these requirements were revisited to allow a progress and quality evaluation. In some cases, the Trello card requirements were modified (split into smaller requirements or elaborated upon) based on this evaluation, which allowed for a higher quality specification to work towards.

MVP Requirements (Due 30/11/16)	
Ability to win the game	
A main menu which passes user defined options into the game scene	
Functional companies that the user can easily interact with	
A market for players to interact and exchange resources	
Randomly occurring natural disasters	
Complete Game Requirements	
All of the above MVP requirements	
A fully functional AI	
Online multiplayer	
Moveable units to traverse map and interact with other player	
Ability to save the game	
Ability to load saved games	
Different difficulty settings	

Figure 3: *Requirement list after the MVP*

This specification allowed the team to follow high quality processes which also contributed towards time and team management throughout the entire development process. These requirements allowed a means to assess the quality and completeness of the software at each stage of the process, and where specific areas should be improved to produce a higher quality end result. This, along with meetings with the mentor and critic and well defined software requirements, ensured the highest possible quality of software and processes by managing expectations which were achievable.

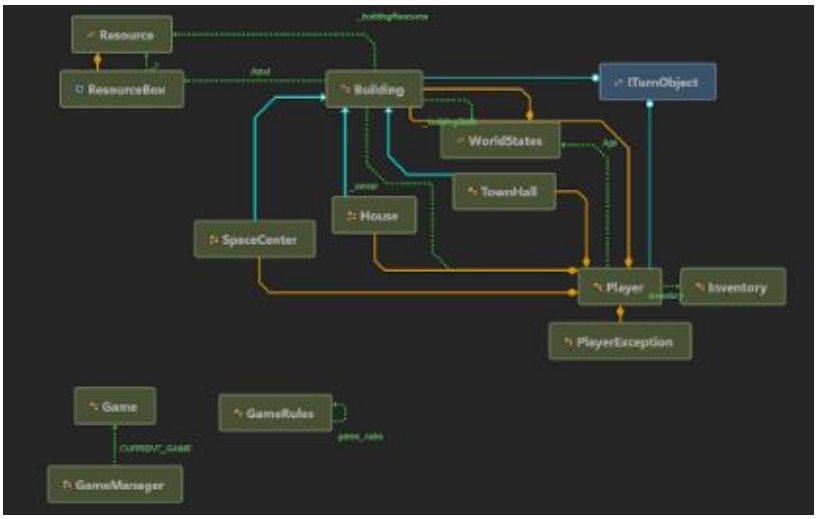
4. Detailed Design

A detailed design was followed throughout the process so that each group member had a detailed understanding of each feature of the game. By having a clear diagram of how features of the game should function, a higher quality end product was allowed as problems could be broken down and abstracted from the game in its entirety. “Modelling is the most important engineering technique; models help us to understand and analyse large and complex problems.” (Kramer 2007). This process also highlighted any unknowns within each problem and any assumptions which had been made: overall, this saved time in highlighting errors in future code implementations which in turn improved the quality of the code.

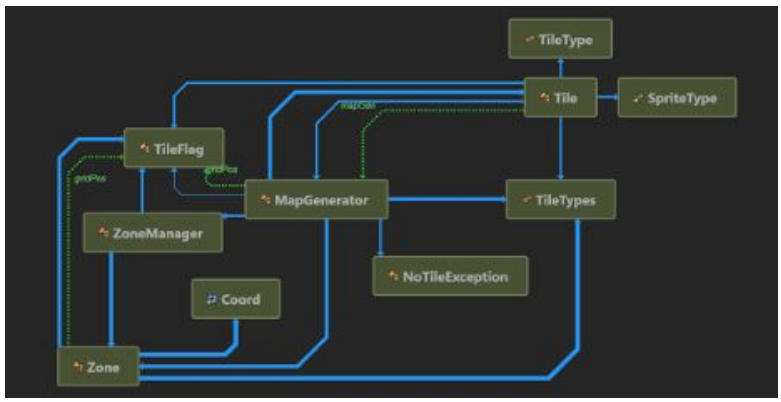
The following models were used during development in order to produce a higher quality product.

4.1 Class Diagrams

Class diagram to show game classes:

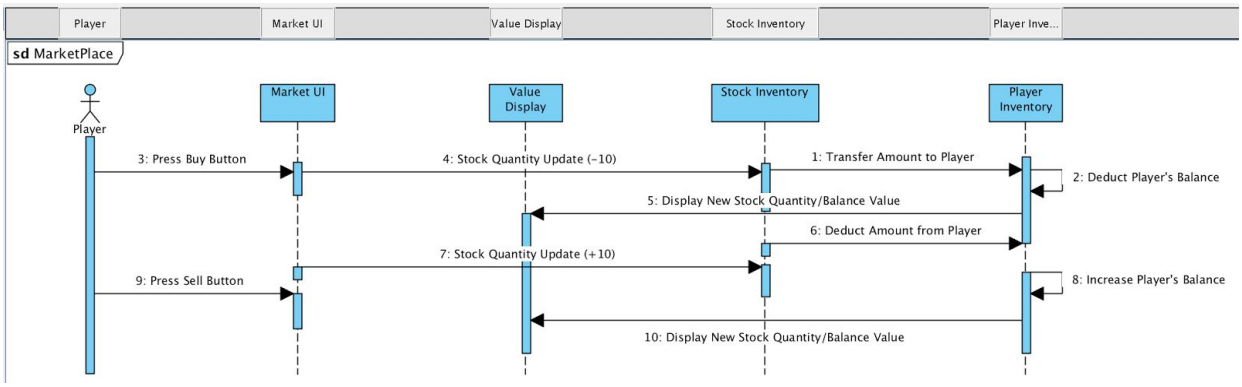


Class diagram to show map classes:



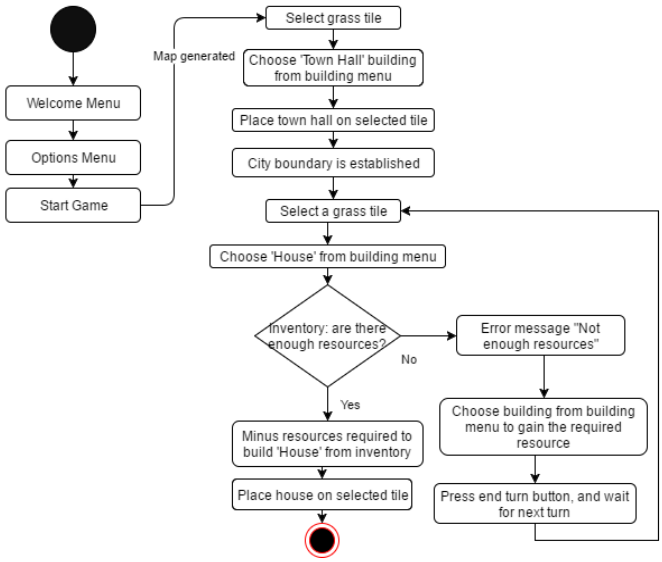
4. 2 Sequence Diagram

This sequence diagram shows the functionality of the “Marketplace” in the game. Although this part of the game hasn’t been implemented, it was an initial planning of the user interface which shows the functionality of marketplace to buy and sell resources. The diagram shows the process of when the user’s balance adds up onto their resource basket, that can enable them to build townhall.



4.3 Activity Diagram

This activity diagram shows the process of a player building a house from their first turn.



5. Implementation

At this stage of the V-Shape model, the team applied the knowledge gained from the meeting with the project mentor to implement the internal quality of the game. This section includes utilizing high quality resources such as workflow, techniques, skilled people and better working environments, which are all considered to be important for developing a high quality software (Azuma, 1996, pg 145).

5.1 Workflow

The team were split in two teams; 'UI design' and 'developers'. The members in developers ensured all the code that was implemented was functional to a high quality, and the UI team ensured all the GUI of the game, including the sound effects were designed to a high quality. In order to ensure the quality of the whole game met the team's specified standard, the team took specific measures to integrate each sub-team's work together.

5.1.1 *Version Control*

GitHub was used to collaboratively work on code and provide a version control system. Each group member had their own branch to update their work and access the whole project. As not everyone from the team was familiar with using GitHub, to ensure that it was used appropriately and to a high quality an extra team-ran tutorial session was arranged. The basic concepts behind its use were explained, and it was ensured that everyone in the team understood processes behind using GitHub through the action of drawing graphs for easier visualization. All members were required to create their own branch to work on and push their code/related files to their branch.

In a quality evaluation of the group's use of Git in a meeting after the MVC, it was determined that this process could be improved by using less vague commit messages. The team agreed that by pushing commits with more detailed messages, it would be easier to both understand specific changes that have been made and to revert changes to a specific point. By agreeing to make this change, for the remainder of the project version control was used to a higher quality.

Figure 4: *Example of better merge descriptions after changing approaches to Git*

5.1.2 Assigning Tasks

In the initial stage of the development process everyone was assigned a role based on everyone's preferences, strengths or ambitions to learn new things. During the first term a few changes were made in order to ensure that everyone who needs help receives it and we tried to work more as a team rather than individually. The diagram below illustrates these changes.

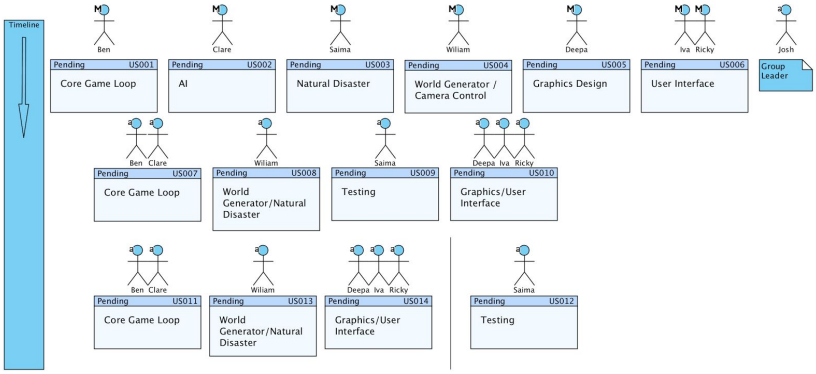


Figure 5: *Timeline of changes in member's role*

Halfway through the first term the UI and GUI sub-team was formed since these team members' tasks were very closely related so it was easier for them to work together. They had several meetings devoted entirely to UI and GUI apart from the team's general meetings in order to evaluate the quality of each other's' work. The graphics for the tiles and buildings were gradually made during the project by the development sub-team.

After the initial group leader (Joshua) left the team, nobody was assigned the role of group leader. This is debatable as to whether this was a downfall of the team, as up until the next group leader was allocated months later, the leadership role was passed around the group depending on which team members were confident to lead in that area (e.g. one group member may take the role as development leader for a specific feature, another may take the role as leader in organising the team for a certain deadline). Due to lower quality processes under this approach, towards the end of the project another group leader was allocated (Iva) in order to produce a higher quality end product at the end of term.

After the hack submission Clare switched from AI development to helping Ben with the core game loop as it was decided that this was of a higher priority. Later on in the project AI was excluded from the plan as it was wouldn't be completed to a high quality considering the time constraints and the amount of other essential features to be completed for a functioning game.

By the end of the first term everyone contributed towards the Quality Assurance report and Clare also took care of the design and final editing.

Halfway through second term Will took over Saima's task of creating natural disasters, and Saima started working on creating tests for the code as Will was more confident with completing this task to a high quality. Deepa, Iva, Rick, Clare and Saima created the final report.

5.1.3 Trello, Google Drive and WhatsApp

The team utilized online platforms to share resources and ensure that team was on track for their individual tasks. For written documentation and file exchange, the team used Google Drive, where all the documents and non-code files were uploaded (Appendix 2). This enabled everyone in the group to review and make changes to the shared documents. For example, the UI team found this very practical for sharing images and agreeing on what was of a high enough quality to be added to the end game. During the early development process, a few team members reported incidents of losing work due technical difficulties. Therefore, in order to improve the quality of this process frequent backing up of non-code files was prioritized throughout the remainder of the project (code files were backed up using Git).

In addition to this, Trello was used to record all tasks throughout the project (Appendix 1). All assignments were added to cards on Trello, each with a deadline for completion. This helped to organise time accordingly and observe the team's overall progres. If a deadline had been missed, this would be addressed in the closest weekly meeting and actions would be taken to determine the reason behind this and to ensure that this would be avoided in the future. One action to overcome missing a deadline included splitting the Trello card in question into smaller cards. Similar situations since missing deadlines have avoided by learning how to identify tasks that are too large for one group member to complete by the deadline and at a high quality.

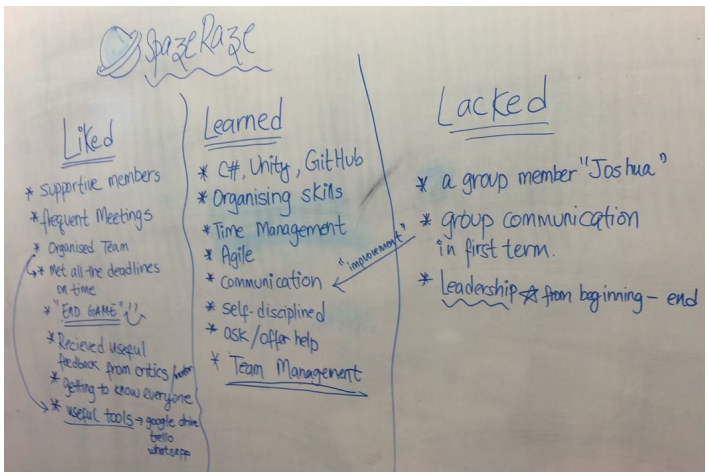
Outside of face-to-face meetings, the team communicated via WhatsApp to ask questions and raise any problems that addressed immediately. This platform encouraged resources to be shared to provide group members with extra help on completing tasks. Meeting minutes were also shared through this online chat, so members who were unable to attend meetings could still be kept up to date on developments (see Appendix 3).

5.2 Team Management

5.2.1 Tutoring and Meetings

The group ensured that everyone meet weekly at a scheduled time slot (Tuesdays 11am - 1pm at the earliest), which allowed the opportunity ask any questions and ensured team members had access to any help required. The purpose of holding a meeting was to allow any issues with tasks (or other) to be discussed in person so that all group members could be aware of any changes happening in the software. During these weekly sessions, agile retrospective activities took place in order for each group member to understand the feelings of individuals in the group. This ensured that everyone was up to date with how the group felt towards particular topics - important topics included:

- Progress in the project
- Issues that have been raised (in code and group management)
- Progress in the team culture
- How quality could be ensured (in code and group management)



The team's final retrospective activity was based on 'The 3 Ls: Liked, Learned, Lacked' (Fun Retrospectives, 2017). This allowed a final reflection on how the team has worked together throughout the project, and highlighted any areas of high and low quality processes.

Figure 6: *Fun Retrospectives(The 3 Ls: The 3 Ls: Liked, Learned, Lacked)*

Throughout the development, the group held tutoring sessions for GitHub and Unity when required in order to ensure that each group member had a high quality understanding of how to use this software. This encouraged a high quality of the end game, as the files were pushed to GitHub which allowed everyone continual access. In addition, bringing together each group member's code allowed continual testing on whether each individual's working code worked with the code overall, which avoided any last-minute incompatible code.

5.2.2 Issues

Despite having regular meetings, the team fell behind in terms of what had been expected. This occurred mainly due to changes in team and also lack of support from members. At the beginning of the project it

was easy to control the workflow, as there was a leader (Joshua), who ensured that the group was on track with task, and because there wasn't a lot of pressure on the group (in terms of official deadlines across course modules). However, losing Joshua's work contributions during the early stages of the project caused the group to fall behind. The increased use of the Trello app after this event gave a clear understanding and organisation of tasks and delegations and helped to overcome this issue.

This solution soon became obsolete due to some team members losing commitment in terms of work contribution and attending meetings, which lead to lack of direct communication and a lack of progress in some aspects of the project.

The whole group met with the module tutor to discuss this issue and solve the problem, so that the remainder of the group management would be at a high quality. After the meeting and the decision to delegate Iva as a new leader, the team has been able to improve the attendance of the group meetings by communicating directly with each member, and have also planned a successful group outing (a meal) to engage socially and boost team morale. Retrospective meetings have also encouraged the team to talk about feelings about morale more often to help avoid the situation from repeating. Due to the increased involvement by group members now the quality of the end game is ensured to be higher than previously.

6. Testing

Testing was planned and had agreed to be implemented after the MVP showcase, as after a team evaluation of the quality of development and processes it was decided that having tests in place would ensure a higher quality end result. However, a group member not uploading their assigned role, a testing structure for the game wasn't completed. This ultimately reduced the quality of the end result and made making changes to the code throughout the project significantly more difficult.

The group has carried out white box and black box testing to ensure that the program was running effectively and efficiently. By carrying out these tests it allowed the ability to test the code along the way, whilst in the implementation stage; this prevented the team from having to go back through the code all at once. The team realised that it would take up more time and also become difficult to get through.

White box testing: By carrying out white box testing whilst in the implementing stage, it made it easier to point out the errors and the things that went wrong. This allowed the team to check that the changes made didn't affect the program overall with no runtime errors. The testing also allowed the program to be split into smaller sections which simplified processes. Whilst testing, separating the code into sections and subsections allowed each individual part to be looked at separately; this allowed the code to be readable and made it more understandable. Two types of tests were into consideration when carrying out white box testing, *branch testing*(Appendix 4.1) and *path statement* testing (Appendix 4.2). Branch statement testing each individual *if* statement to be tested. Branch testing helped to check whether the boundaries and restrictions that we set were working accordingly an to a high quality. White box testing allowed the quality of the system to be ensured and confirmed that that any changes made did not affect its overall performance.

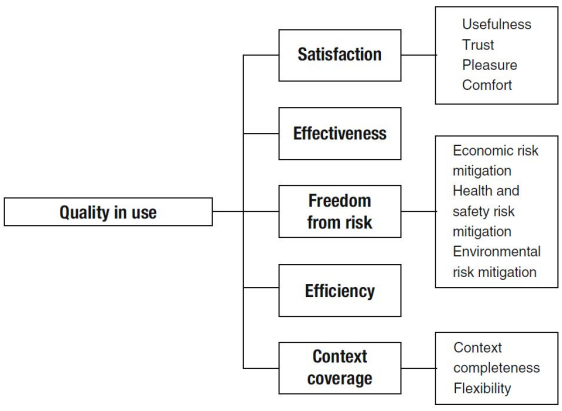
Black box testing: Black box testing was implemented by allowing the user to interact with the interface. For the creation of the test a list of expected outcomes were used, referring the aims of the game. When performing black box testing, an observation was made on how the user interacted with each individual screen of the game. These results were compared with the expected outcomes and documented in a table (Appendix 6).

7. System and Acceptance Testing

This is the final phase of the model in which external quality characteristics are validated with the quality requirements. The objective of evaluating the 'in-use quality', is to ensure all the components work

together as expected. This has been achieved by receiving feedback on the quality of the software to gain awareness on lack of compatibility between user's needs and the characteristics of the game from the user's perspective (Azuma, 1996, pg 152).

7.1 Usability Testing



To test the usability of the software, the group conducted a data collection from two users. The questionnaire used in the data collection has been designed using the Quality-in-Use Model, which shows five different criteria to be tested against (Appendix 6). This model has been used as a form of checklist to ensure all the criterias are met in terms of quality.

Figure 6: *Quality in use model (ISO/IEC 9126-1)*

Results from Usability Testing:

The usability testing was executed by allowing two users to play the game, while one of the members is there to monitor their performance and provide guidance (Appendix 6). According to the user, they rated the overall quality of the GUIs to be 4/5. One of thing that could have been improved was the graphics of the building to be more animated, rather than just blocks as it was difficult to differentiate between different types of buildings. The user gave 3/5 for the graphics. Another question raised was regarding the background image of the menu bar. It was lacking in terms of overall theme and is different from the game theme. The user was also satisfied with the context of the game such as starting from the roman era to modern and reaching to the space and was given a rating of 4/5. The user also liked how the resources were generated/unlocked, as they go up to next levels(era). However the overall functionality and the rules was a bit confusing which was given 3/5. The user pointed out that the game was less competitive and intriguing as it was lacking in quest requirement and not enough interaction between different players.

The user found the game to be very user-friendly, despite complicated game rules and less complex once you start the game, which was given 4/5. The overall satisfaction was rated 4/5 for the end game.

8. Maintenance

This phase requires the modification of the software product after delivering faults to improve future performance (ISO/IEC 14764:2006). From the given feedback on usability testing, we will be pointing out some of the the improvements for future development.

The final aim for the player is to grow from the Roman age through to the space age. However currently, only three out of five eras which had been planned for are available in-game. Therefore, in the future the team will aim to add two more ages (Medieval/Tudor and Victorian/World War) to fit between the currently available ages have to make the game feel more complete and realistic. This would involve the creation of more advanced buildings, together with expanding on features such as natural disasters and attacks across the ages. For example, in the Modern/Space era the team has planned to add a 'rocket launch' to the moon with an 'X' chance of success which increases with each attempt. For example, the team has planned to add moveable units (following a meeting with the mentor) as this wasn't possible during the time constraints. Player interaction is to be improved mainly by adding an online multiplayer option which would be possible by connecting to an online server. More interesting quest requirements and scenarios would improve the player's experience in a way that the game is more competitive and intriguing.

The team also planned to include the marketplace that allows the user to buy/sell different resources when they needed a particular resource to progress. This essentially creates an in game economy where

the marketplace act as a trader, and trade with the user for either money or resource.

The graphical design of the game can also be enhanced by adding more effects and also changing the images with better crafted ones that look smoother. AI is to be improved and added to the game too. Below are some of the list for future improvements that has been suggested by the members and after the usability testing.

- Online multiplayer
- More ages
- Advanced buildings through the ages until the space age is reached
- More player interaction through implementing a marketplace - making the game more competitive (as the marketplace wasn't completed to a high enough standard to be included in the end game).
- AI - depending on circumstances
- Expanding on current features
- Quest requirements made more interesting
- Graphical improvements - animated effects, etc.

Having a testing structure in place would significantly improve the maintainability of the code, and therefore the quality of the end game overall.

9. Conclusion

To conclude, the evaluation has enabled an understanding of sufficient quality that needs to be implemented in the software and whether the quality is acceptable to be forwarded to the next process. Evaluating the software has also enabled the members to clarify the requirements or analyse the cause of bad parts in the system. This gave the team a chance to reflect back and make improvements to develop a complete end product. Writing this report enabled another opportunity for the team to reflect on how how a high quality end product was achieved feasible end product.

Evaluating 'in use quality' raised follow up problems and enabled the team to find mechanisms for supporting and making the feedback for the next development.

However, the quality of the end product could have been improved by doing extensive testing. The testing couldn't be comprehensive since the member who was assigned for the task failed to deliver it on time. This affected the quality of the evaluation report as the member had lack of communication with the whole team leading to the poor result from their task and evaluation itself.

10. References

14764:2006, ISO/IEC. "ISO/IEC 14764:2006 - Software Engineering -- Software Life Cycle Processes -- Maintenance". Iso.org. N.p., 2017. Web. 5 Apr. 2017.

Azuma, Motoei. *Software Products Evaluation System: Quality Models, Metrics and Processes-International Standards and Japanese Practice*. Information and Software Technology 38.3 (1996)

Belbin (2016) *Belbin® Team Role Summary Descriptions*. Belbin. Accessed Web. 4 Apr. 2017.

Bocij, P., Chaffey, D., Greasley, A. and Hickie, S. (2008) *Business information systems: Technology, development and management for the e-business*. 4th edn. Harlow, England: Financial Times Prentice Hall. (ch2).

Fun Retrospectives (2017) The 3 Ls: Liked – Learned – Lacked | Fun Retrospectives, Funretrospectives.com Accessed Web. 6 Apr. 2017.

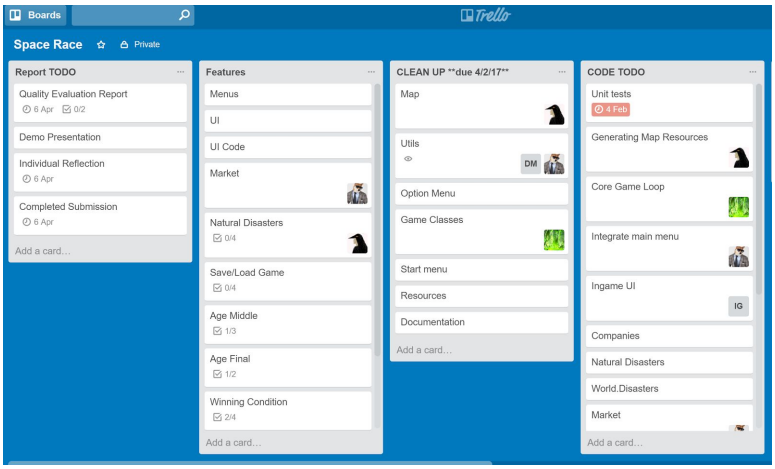
Ghezzi, C., Jazayeri, M. and Mandrioli, D. (2003) *Fundamentals of Software Engineering*, 2nd Edition. Pearson International

Kramer, J. (2007) *Is Abstraction The Key To Computing?*

Team 2 (2016) *Submission 1: Designed Alliance*

11. Appendix

11.1 Appendix 1: Trello

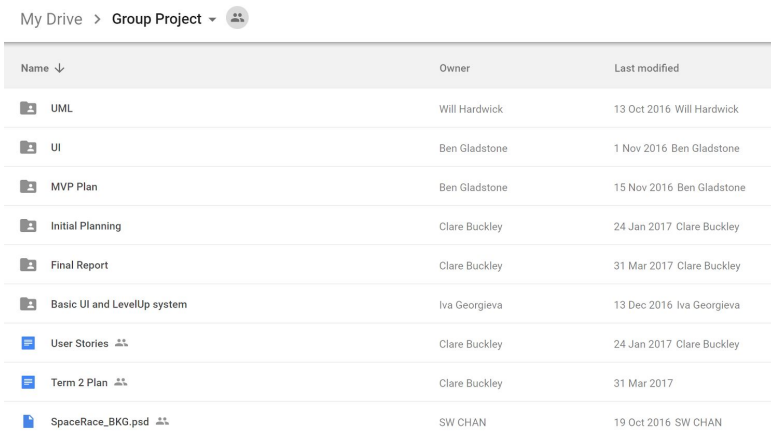


Trello app:

Each of the Cards has been listed out with tasks that needs to be completed by the given date. Upon completion, the members can mark it as completed.

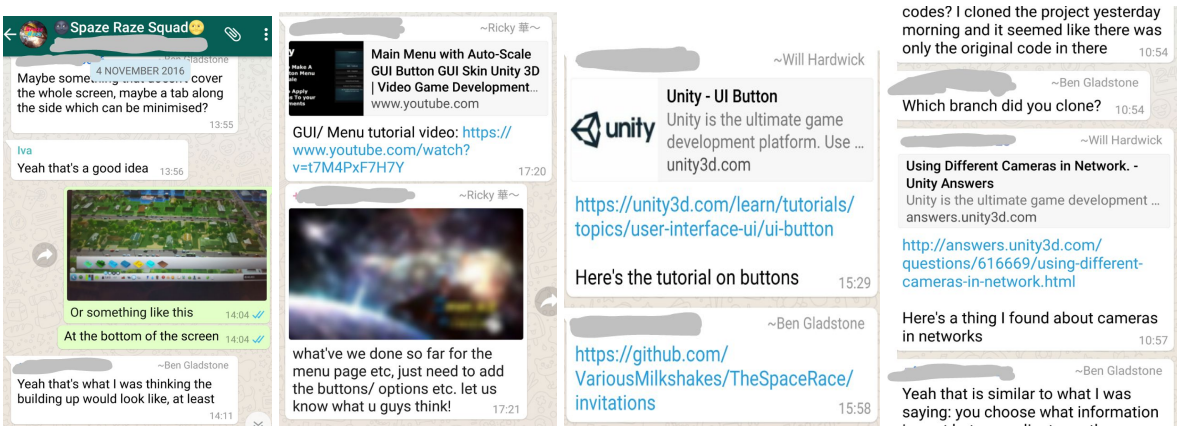
11.2 Appendix 2: Google Drive

-Managing all the documents and folders, that allowed members to share, view them.



11.3 Appendix 3: Whatsapp

Members bring up issues and finding relevant resources that is useful in the game design



11.4.1 Appendix 4.1: Branch Testing

If statement 1

When the player tries to place a building, the system will check whether the building is valid and that the player is allowed to build. Methods which define the rules of the game are called when the player is trying to place a building. First it will check the build order method, which makes sure that the first

building that the player places on the map is a town hall. It then checks the building tile method, this checks to see if the tile the player is placing on the map is valid, it does this by checking that it is the correct terrain and also whether the player owns that tile. If the building is on a later era then they cannot build on it, and filters the list to show only the buildings that the user can build. If it is invalid it will show an error message to the user.

If statement 3

After a building has been built, on each turn a certain amount of resources are added to a building inventory which checks whether its input is full. If true, it converts its input into resources which are fed back to the player's inventory. For example a bakery may require a 100 bags of flour to be able to convert these into resources which can then be sent back to the main inventory. On each turn the bakery checks to see if it has the required amount of flour, once it does, it creates food which is then sent back to the player's inventory, which they can later use for upgrades.

11.4.2 Appendix 4.2: Path Statement Testing

Loop 1 - Player phase

Each player has a phase (their turn), when one player has finished their turn, it switches to the next player and then it goes round in a recursive loop, switching between players' turns.

This nested loop keeps running until the player clicks a button to move on to the next turn. Once both players have completed their phase, a new turn is generated if the game is still running. The variable 'active players' stores a list of all the players in the game. Inside of it there is a for-each loop to go through each player, nested in that is a while loop that will run until the player has completed their phase.

11.5 Appendix 5: Black Box Testing

Test	Description	Data/Action	Expected/Actual Results
1	Does the start button open the main program	Click Start Button	The options menu form should open
			The program started and the player was presented with the options menu
3	Number of player options	Enter number of players	The number of players is stored and the main game starts
			The player selected 4 players and was sent to the main game page showing all players in 4 corners of the board
4	Does quit button exit the program	Click Quit Button	The program should exit
			Program closed
5	Store user information	Enter name and select colour	The name and colour for each player should be stored
			When the user entered their name and chose their colour this information was stored.
6	Display the player's name	Click start button	When the game starts a message should appear with the players name
			A message box appeared with the player's name and the colour
7	Does the inventory add resources	Buy a house	The inventory should increase by 10
			When the player bought a house the inventory added 10 points to the resources

8	Does the inventory minus resources	Buy a property	The inventory should decrease by 10
			When the player bought a property the inventory took 10 points away from the resources
9	Player phase	On click	After the player has had their turn the next player should be prompted to take theirs
			Once the player clicked to perform a new action the next player was asked to take their turn
10	Tile changes to the correct colour chosen by the player	Player selects a tile	When the player purchases a tile, it should change to the colour they have chosen
			When the player selected a tile, the colour matched the colour the player chose in the options menu

11.6 Appendix 6: Usability Evaluation Questionnaire

Questions that was used to get feedback from the users for the 'Usability Testing'

Rank your experience. Please circle your prefer answer

1. Design quality of Graphical User Interface(low 1 – 5 high)

   1 2 3 4 5

2. Complexity of the game (Low1 – 5 high)

   1 2 3 4 5

3. Graphic Design of the game such as buildings, resources (Low1 – 5 high)

   1 2 3 4 5

4. Rate the understanding level of the game content and its functionality(rules)(1 –5 high)

   1 2 3 4 5

5. Rate the overall context of the game

   1 2 3 4 5

6. Overall satisfaction of the game (Low 1 – 5 high)

   1 2 3 4 5

7. Suggestions for future improvements.

.....
.....
.....

8. What features do you like the most about this game?

.....
.....