

2011907__Software Engineering Principles and Testing

by Josh Archer

Submission date: 28-Apr-2023 08:09AM (UTC+0100)

Submission ID: 203807553

File name:

24846_Josh_Archer_2011907__Software_Engineering_Principles_and_Testing_490085_959655846.docx
(956.07K)

Word count: 4794

Character count: 23940



SOFTWARE ENGINEERING PRINCIPLES AND TESTING

ACCB5030

Josh Archer
2011907

Josh Archer
2011907@student.uwtsd.ac.uk
Page 0 of 20

Contents

1.	Assignment Overview	2
1.1	Abstract	2
1.2	Introduction	3
2.	Main Body	4
2.1	Design Evaluation & User Stories	4
2.2	CRC Cards	8
2.3	Team Evaluation	8
2.4	Software Testing	10
2.5	Black Box Testing	10
2.6	White Box Testing	12
2.7	Testability	13
2.8	Unit Testing	13
3.	Conclusion	15
4.	References	16
5.	Appendix	18

1. Assignment Overview

1.1 Abstract

The Purpose of this assignment work as a team to produce suitable software design to make the changes necessary to a connect4 game. As a team we must produce suitable User stories, CRC Cards and Flow diagrams that will help come produce the most efficient design. The team must work together and document the discussions made.

Individually, they must then evaluate the design that was produced then must document and discuss the experiences that incurred when producing the design. The strengths and weaknesses must be recorded and any ways the design could be better.

The individual must then evaluate how important suitable testing methods are to software engineering. Black and white box testing will be discussed as well as which tests the individual feels will be the most suitable to ensure their code is acceptable.

1.2 Introduction

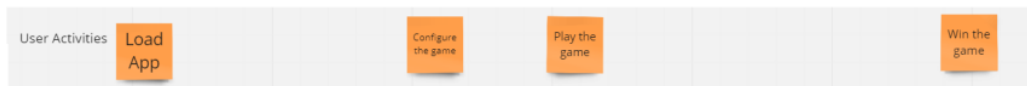
Software development is becoming a more and more complex task with ever evolving technologies. With this being said the testing of software has become vital to ensure the end product of a design is defect free and is what the client has asked for. To make this more achievable, collaboration between all members of development team is essential. Using the correct tools to plan and design the product will set teams up for success. During this assignment, these tools will be discussed, and a groups use of these tools will be evaluated. The testing methods whether black or white box will explain and what testing techniques to be used on the new feature to be evaluated.

2. Main Body

2.1 Design Evaluation & User Stories

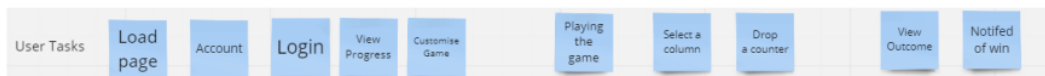
When first considering a software design a technique must be first used to outline a new product or a new feature within the design. The technique that has been used is known as user stories & mapping. Using this method, the group were then able to collaborate efficiently to produce the best possible design for a new feature. This means that the group were able to prioritize certain aspects of the work whilst sharing an understanding of what the client needs and wants from the feature. [1]. This technique comes under Agile framework. Adhering to an Agile framework will allow for continued planning, changing or prioritise, testing and collaboration. Working towards the values of Agile Frameworks, the team were able to focus more on the individuals and there interactions rather than the actual process and tools of developing the feature [2].

Firstly, the group were able to come together and produce User activities or Epics to decide what the group wanted from the new feature.

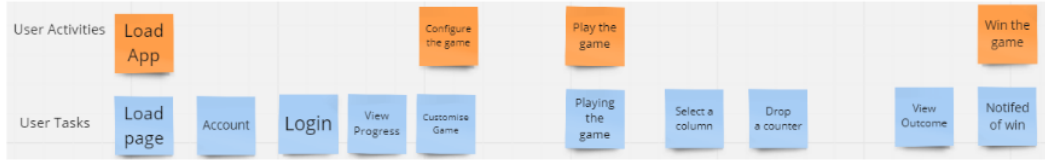


By doing this, everyone in the group were able to fully understand what is needed from the feature and would then be able to move forward from there. From this step then, the group were able to efficiently identify the steps needed to then achieve these activities and in turn helped the team keep in mind the big picture of what they were trying to achieve. Producing the User Activities enabled the team to break large activities into smaller steps. As you can see from the Sample above, the User activity LOAD APP for example, will generally have a lot of code to be able to complete this activity. At this stage the group found it challenging to not get to bogged down in making this stage to complicated. This stage is to provide a basic description of what the client wants from the new feature. One of the things that could be improved at this stage would have been the addition of Ending or Restarting a game. However, when designing a new product, the group have found that it is vitally important to adhere to an Agile framework throughout. By doing this, the group would then allow for this change to be added later.

The group would then move on to produce user tasks. At this stage, the team started to really develop the User mapping process by producing the User flow.



By completing this step, the group were able to discuss a process of how they thought the user would interact with the feature. Through discussions they were able to decide what the first thing the user would see or do. It was decided that the first thing the feature should do is actually load the page they were trying to access. Then from there the group could then run with the ideas of what they thought would come next e.g. the users account details, then logging into the game.



As you can see by the image above, the user flow or user tasks, correlates with the user activities. By doing this the group were able to visualise what the user would need to do to be able to complete the activities that they require from the feature. As with the User Activities, one of the challenges the group faced, was to keep the tasks simple. Using short phrases and keeping it basic is again, vital at this point. The key to this stage is to enhance the User activities by giving breaking them down and adding slightly more detail. The group did this well, providing a clear map as to how the group should then progress, what is expected to come and making the writing of the User Stories much easier.

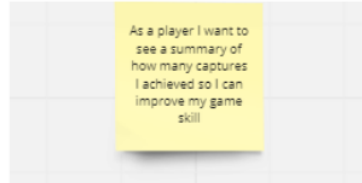
As this is a test-driven development, and a feature that will come under numerous unit testing procedures, which will be covered further in the report. A Walking Skeleton was implemented by the group which is found in the image below.



A Walking Skeleton must be designed to ensure that the end features basic functions are able to be performed. In this case, functions such as the game simply loading or being able to drop a counter when it is the players turn, will all need to be functioning, however only in a fundamental way [3].

The Group produced this to ensure that the basic features and components were connected and could provide a minimalist implementation of the feature the group have designed. The benefits of this would be, the group would have a very basic source code to be able to test against to ensure that the fundamentals of the feature were working.

Once the group were able to produce a walking Skeleton, the first drafts of the user stories were produced through many discussions and collaborations. The group came up with Minimal Viable Products (MVP). This really helped the team focus on the “must haves” of the feature. One of the other benefits of the group producing an MVP, is that when providing feedback to the client, we would have a product that would have enough of the features wanted by the client for them to give feedback on. As a group we realised that not every feature would have to work, but there would be indications of what the feature would be doing. An example of this would be the story of the player wanting to see a summary of the game they have just played (See image below). At the early stages of development, this may be to be complicated to actually have in place. However, having it within the MVP will show the client that the group intends to implement this in a later release.

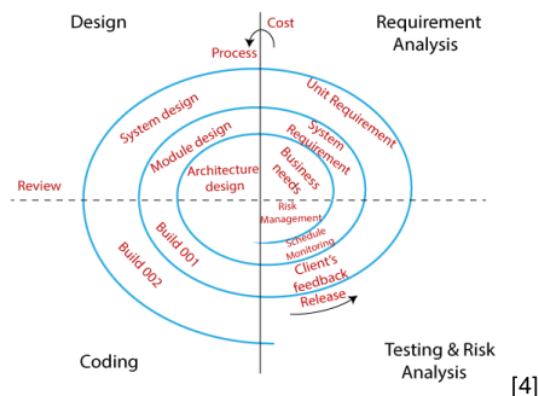


The group were then able to collaborate together to start producing the user stories. The group would need to identify what would be needed at each stage of release. What is meant by this is that when developing a new product or feature. You don't want to completely develop a product and then when the client gets it, they provide feedback that indicates they are not happy with the product. This would be known as a waterfall methodology of software development. This method allows for development in stages such as the design stage, then the testing stage. The disadvantage of this type is, the group would have found certain features or functions missing at a very late stage of development. This method usually results in cost of not only delays but money also. The reason being that the group would have found it very difficult to move back up the waterfall.

Instead, the group went with a spiral model, this model is usually broken down into 4 separate stages of development.

- Requirement analysis – This is where the business needs are collected, ranging from unit requirements or sub system needs. The benefit of this stage is that it encourages discussions and collaboration between the group and the client. This has then helped the group easily understand the system requirements and makes the next stage much easier.
- Design – This is where the group came together to develop the user stories implementing a logical design to develop the feature.
- Coding – this is the stage in which the group will be coding to develop the new feature.
- Testing and Risk Analysis – Once all the above has been completed, the group move onto the testing stage. After the first cycle the group would test the build of the feature [4].

After passing the test stage the product is then passed onto the client for them to test the feature and then give the group feedback. The cycle will then be restarted from the client's feedback. For an example of the spiral model see image below.



The group reproduced this model exceptionally, especially in with the user stories. As you can see from the image below, they have broken the cycles down into release stages. Allowing the group to develop the product whilst being in communications with the client to ensure that when the product is finished, the client is getting what they wanted and within the timeline that was predicted.



One of the things the group also did exceptionally well is ensuring the stories are kept short and precise. As a developer, you want to be able to focus on one small thing at a time. Anything bigger than what the group has come up with above would be known as epics. These would then have to be broken down into smaller tasks. One of the benefits in the way the group has worded their stories is that they have truly thought about everyone that would be involved with the product, especially the client. By using the "As a player I want" format, they have considered what the player would want from the product. Then using the "So that" format, they have then been able to justify why having this as part of the product would be a benefit.

2.2 CRC Cards

Class Responsibility Collaborator (CRC) cards are a way for the team to collaborate on ideas for the new feature especially when it comes to an Object Orientated Program (OOP). This model of development revolves around the team collaborating and discussing to produce certain index cards. The group will produce cards that contain the most important classes within the new feature. The team will then start to populate these index cards with the collaborator's names and responsibilities of the classes. When talking about the collaborators these are the other classes that will affect the behaviour of the class in question. The responsibilities are the things that the class should know or do [5]. Next, the group should then be able to discard any design ideas that they think won't meet the requirements by playing out as a team all computation possibilities. The group are also able to see any class inheritance by organising the child class cards under the parent classes in rough rows.

Some of the biggest advantages of using this model is as the team has to scrutinise and examine each card, they inadvertently are doing the same to the whole system itself. This allows for the full team to see the bigger picture and have a full understanding of where they are in development. Using CRC cards also make it much easier for the team to be able to identify any problems that may arise by providing a visual representation. When it comes to testing, CRC cards make it much easier to do so, by clarifying the roles, responsibilities and expectations of the systems and different classes. Finally, CRC really encourage collaboration. Just like most of the other models and methods used throughout the development of the feature, CRC cards promote discussions, brainstorming and will actively involve all members of the team [6]. With this all being said, there are drawbacks to using CRC cards. CRC cards are limited, especially when it comes to large systems where the classes are extensive and will be difficult to track. CRC cards should also be used alongside other systems as they are just a small part of the user requirements. Use cases should also be considered when developing CRC cards [7]. One minor disadvantage is also, because the CRC cards are usually a physical card, it could be a problem if these became misplaced or damaged. However, as the group produced these digitally with back ups in place, this was avoided.

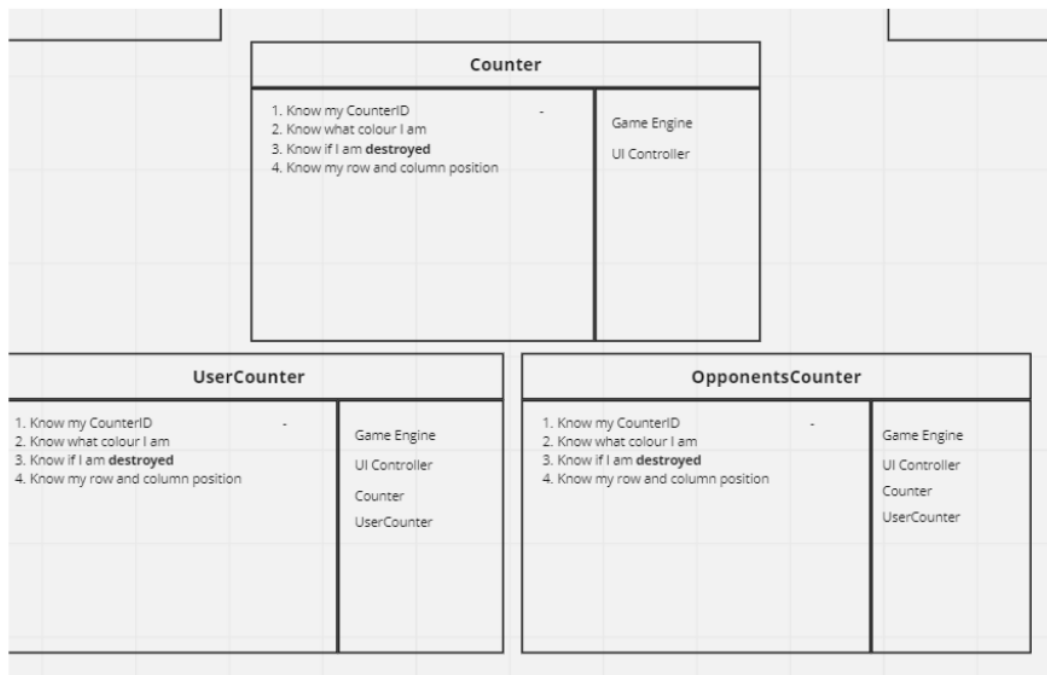
2.3 Team Evaluation

One of the things that were vital to the group producing adequate CRC cards was communication and collaboration. The group were able to ensure that the CRC cards contained enough information in them to be able to meet the criteria set out in the user stories.

Player	
1. Choose counter colour 2. Knows Player name 3. Choose column and drop counter 4. Whose turn is it? 5. Know if I have won 6. How many games I've played 7. How Many I won	Counter

Take this example from the groups CRC cards. The class would be Player, the Collaborators would be the Counter and the Responsibilities are laid out in the left hand column. From this, the group can then see what the Player class knows and does. The class itself adequately meets the criteria proposed in the user stories. However, the collaborators could have been thought about more. If the team had focused more on the collaborators, they may have realised that actually this class in particular, relies on numerous other classes, such as the Counter, GameEngine & Grid. The player class interacts with all of these in relation to how the player interacts with the game grid, how the player places the counters and over all just how the player interacts with the Game itself.

The group were able to show well examples of inheritance throughout the CRC designs. As you can see from the example taken below. It shows how the UserCounter and OpponentsCounter classes are Child classes of the Counter Class.



One of the things the team found difficult was minimising the responsibilities within the classes. It is important to limit these to ensure that the cards are used for the purpose which is essentially a tool for collaboration and communication [6].

2.4 Software Testing

The testing of the software the group has developed is essential to being able to produce a product that not only the client is happy with but is defect and bug free. Software testing generally comes with 7 Principles. These are as follows:

- The testing shows any defects that are present – doing this allows for you to identify, track and fix any problems before the product is deployed to the client. However just because no defects were identified, does not mean there aren't any. It simply means that the tests produced didn't cover the scenarios in which the errors would be invoked. With this being said it is important to manage the stakeholders and clients' expectations.
- It is impossible to Test everything – every possible combination of inputs, preconditions or general function of a code can be tested. Not only is it a waste of time and money to do so it will never really affect the quality of the code you are trying to test. Instead, risk assessment and carefully planning the tests will ensure that the test will cover all said risks without testing every line of code.
- Early Testing - testing early will keep cost and time producing code down. By testing early bugs or defects are caught early in the requirement and design stage when developing from a Spiral Model. After each cycle, testing takes place which ensures that there is no time wasted by not having to rewrite huge lines of code.
- Defect Clustering – This is the idea where the majority of defects are found in a small area within the code. Finding a defect in an area will usually lead to finding more within this same area. This will allow the developer to focus there testing to one area in particular.
- Pesticide Paradox – This is the principle in which running the same tests will eventually lead to the failure to find new errors. Using a variety of testing techniques and approaches simultaneously.
- Context Dependent - Remembering that no strategy or technique will apply to every scenario will help the developer ensure that numerous techniques are used to find defects. The techniques a developer deploy are always affected by the scenario and context.
- Absence of errors fallacy – Remembering that testing isn't really about finding bugs or defects, but is about ensuring that the software is what the client wants and meets the user requirements [8]

With all this being said, the testing is vital to the quality assessment of the software. Appropriate testing will measure the quality of the product and then give details into how the product can be improved [9]. Software testing is hugely important to ensure that the product adheres to the product specification. Whether that is the product not doing something it should do or doing something the product specification says it shouldn't. Finally if the product will in any way be viewed as incorrect by the client, it should be tested.

2.5 Black Box Testing

The testing of lines of codes usually comes under manual testing. Manual testing is needed to ensure that applications that have been given to the clients are stable and defect free. Manual

testing is usually tested from an end user perspective. The different types of manual testing are known as Black and White Box testing.

Black Box testing – This is a form of testing on the functionality of the software. The tester should not be concerned by the internal coding or structure of the coding. Instead focusing on whether the product does what it is meant to in accordance with the products specifications. The procedure in which this test takes place is usually done by taking a function and interacting with said function. If the function gives the correct output, then it passes the test, if not then it fails, and further development will need to take place. The tester will usually input a positive test and a negative test. This could be done by inputting a valid input to a function and then putting an invalid input. For the test to pass, it should accept the valid input and reject the invalid [10].

An example of black box testing technique that would be used on the new feature would be the boundary value analysis. This technique is used to test the boundaries values due to the fact that the values close to this point have a higher chance of defects and errors [11]. A precondition for this test would be the grid of the game. A tester could then try to drop a counter outside of the 6 wide and 7 high grid. An acceptable pass would be the game not allowing the user to do this, a failure would obviously allow it. The tester would then place a counter on the edges of the grid boundary, Then again in the middle. What this test would be trying to show is whether the game will recognise the size of the grid that has been designed, and whether the user will be able to use it adequately.

Another test would be a Decision table technique. This test allows for various input combinations and then records the behaviour in a table. This test in particular is done in a systematic manner and a benefit of this type of testing is that it saves time and offers a large coverage of testing area [12]. Again a precondition for this test in regards to the feature the team is designing could be the login function. A correct login name and password would be entered, if so then the tester should be able to login correctly. The tester will then go on to enter a correct login name but incorrect password. At a minimum the tester would expect the function to decline to log in. The same would be done with a correct password but incorrect login name. The tester would expect the same result as before. The results would then be recorded in a table as below.

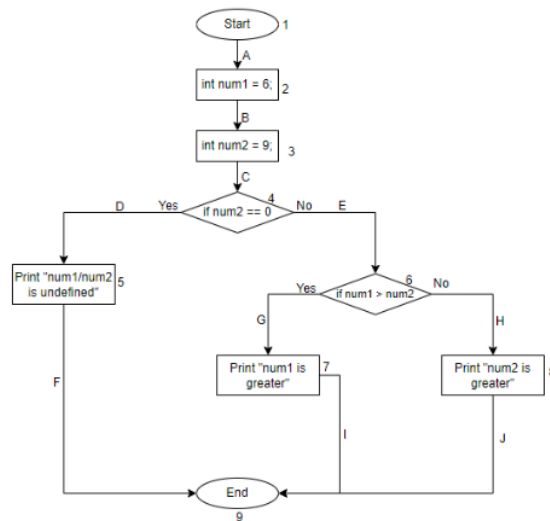
Email (condition1)	T	T	F	F
Password (condition2)	T	F	T	F
Expected Result (Action)	Account Page	Incorrect password	Incorrect email	Incorrect email

[12].

2.6 White Box Testing

White Box Testing allows the tester to be able to fully inspect and understand the internal structure of the product they are trying to develop. It provides the tester with full knowledge of the feature and the source code [13]. In comparison to other techniques white box allows for a much larger coverage of testing area, especially when given the source code.

An example of White box testing that could be carried out on the feature would be the Path Testing technique. This test will test all possible paths of a program shown through a control flow graph [14]. An example of a control flow graph is found below.



[14]

However, in the feature in question, the test could be run on the part of the code that runs the players turn. To start, the player enters a counter, has the counter made a connect 4. If yes, then the game will end. If not then has the counter trapped an opponent's counter if yes, then the opponents counter is replaced with the users. The test will continue like this until all paths have been tested. As a tester you would expect all features that have been written in the product specification to be executed without failure. Once this has been completed a formula will be written to calculate the cyclomatic complexity which is as follows: $E - N + 2P$. E = the amount of edges within the graph, N = the number of nodes and P = the number of connected components [14]. The tester is then able to identify the independent paths and then be able to design a test case.

2.7 Testability

With all this being said there needs to be some sort of Testable Code. For the code to be testable then the code must contain certain properties:

Simple – When the components or the application itself are fairly simple, testing of these becomes much easier.

Observable – Ensuring that the internal structures and data are visible to make sure that the test, whether pass or fail, is accurate.

Control – making sure that the thresholds of the application are able to be reset and set.

Knowledge – When testing, specifications or other types of documentation should be available. This is to ensure that the results of the test are correct.

When making the code more testable, it helps the developer fully understand the requirements of the software, the reason for this is being able to test different parts of the application or code to examine how they work together. It also provides the application with a greater flexibility. This means that the code can be changed without breaking or malfunctioning prior tests. Overall achieving a testable code will greatly enhance the success factor of a developed feature. Attaining the knowledge of what will make an impact on your testing will allow the developer to address these problems early in the development stages [15].

2.8 Unit Testing

Unit testing is testing a small piece of the code that is isolated within the system. This could be a function, a method, property, formulae, and equations. It is important to ensure that the code is isolated so that it's not influenced or relying on external systems [16]. Unit testing usually takes place over 4 phases:

- **Setup** – the Objects that are required and the variables are then declared.
- **Exercise** – This is executing the code performing the test.
- **Verify** – Check the results of the test are correct
- **Teardown** – Shutdown any databases, files or other resources that were opened to conduct the test.

Unit testing can check the behaviour of an array of characters of which are part of the system that is under test. Usually however, it only comes down to 2 categories, State based or interaction based. State based is verifying that the system that is being tested is providing accurate results. Proving that the system is accurately recruiting different methods is interaction based [17].

The main principles of Unit testing are that the testing code is simple to write. This means that the code shouldn't take a huge amount of effort to code. This is due to Unit testing usually requires the same code to cover numerous cases. Next is readability, a unit test should be transparent. If a test fails then as a test it should be easily discoverable. Another principle is that the code should be reliable and finally the code should be fast. Running just one slow test wont cause much of an issue however, like previously stated that numerous cases are tested.

If this is the case and the code is very slow, it will delay the testing of the feature dramatically [17].

With this being said, applying this in a white box test such as the path test would prove beneficial. Isolating a small function of the code such as the Capturing of an opponent's token horizontally, will allow for a small unit test to be carried out without having to go through the entire scenario, of checking the vertical and diagonal tokens also. These can be done after and by running all 3 the developer can ensure that the code will work individually.

3. Conclusion

In conclusion, the user stories and CRC cards truly promote collaboration and discussions throughout. By doing this, a more efficient design is created, leading to a greater product. It is drastically important to stay agile throughout the development process. Through collaboration and discussing different ideas, the design constantly changes to meet with the new user requirements.

Testing is vital throughout all stages of the development cycle. Ensuring that testing is done early and often, will ensure that any deficiency, bugs or just overall errors will not affect the timeline and cost of the end product.

The coding should be kept simple through out which will make the testing much easier.

4. References

- [1] Nimble Work, "The Ultimate Guide to Agile User Story Mapping," Nimble, 2023. [Online]. Available: <https://www.nimblework.com/agile/story-mapping/>. [Accessed 24 04 2023].
- [2] Mendix, "Agile Frameworks: An Overview," Mendix, 06 2023. [Online]. Available: <https://www.mendix.com/agile-framework>. [Accessed 25 04 2023].
- [3] T. Contributor, "walking skeleton," What Is, 10 2016. [Online]. Available: <https://www.techtarget.com/whatis/definition/walking-skeleton>. [Accessed 2023 04 26].
- [4] JavaPoint, "Spiral Model," Java Point, 2022. [Online]. Available: <https://www.javatpoint.com/spiral-model>. [Accessed 26 04 2023].
- [5] Agile Alliance, "CRC Cards," Agile Alliance, 2023. [Online]. Available: <https://www.agilealliance.org/glossary/crc-cards>. [Accessed 26 04 2023].
- [6] Chisel , "What are CRC Cards?," Chisel Glossary, 6 04 2023. [Online]. Available: <https://chisellabs.com/glossary/what-are-crc-cards/>. [Accessed 26 04 2023].
- [7] S. W. Ambler, "CRC Modeling: Bridging the Communication Gap Between Developers and Users," Ambysoft Inc, 2020.
- [8] T. Cser, "7 Principles of Software Testing," Functionize, 16 06 2022. [Online]. Available: <https://www.functionize.com/blog/7-principles-of-software-testing>. [Accessed 26 04 2023].
- [9] International Software Testing Qualifications Board, "Why is testing Necessary?," ISTQB, 2023. [Online]. Available: <https://www.istqb.org/>. [Accessed 26 04 2023].
- [10] E. Aceska, "Types of Black Box Testing Techniques," TestDevLab, 17 08 2022. [Online]. Available: <https://www.testdevlab.com/blog/types-of-black-box-testing-techniques>. [Accessed 26 04 2023].
- [11] JavaPoint, "Boundary Value Analysis," JavaPoint, 2022. [Online]. Available: <https://www.javatpoint.com/boundary-value-analysis-in-black-box-testing>. [Accessed 26 04 2023].
- [12] Javapoint, "Decision table technique in Black box testing," Javapoint, 2022. [Online]. Available: <https://www.javatpoint.com/decision-table-technique-in-black-box-testing>. [Accessed 26 04 2023].
- [13] CheckPoint, "What is White Box Testing," Check Point, 2023. [Online]. Available: <https://www.checkpoint.com/cyber-hub/cyber-security/what-is-white-box-testing/>. [Accessed 26 04 2023].
- [14] K. Asif, "What is basis path testing?," Educative, 2022. [Online]. Available: <https://www.educative.io/answers/what-is-basis-path-testing>. [Accessed 26 04 2023].

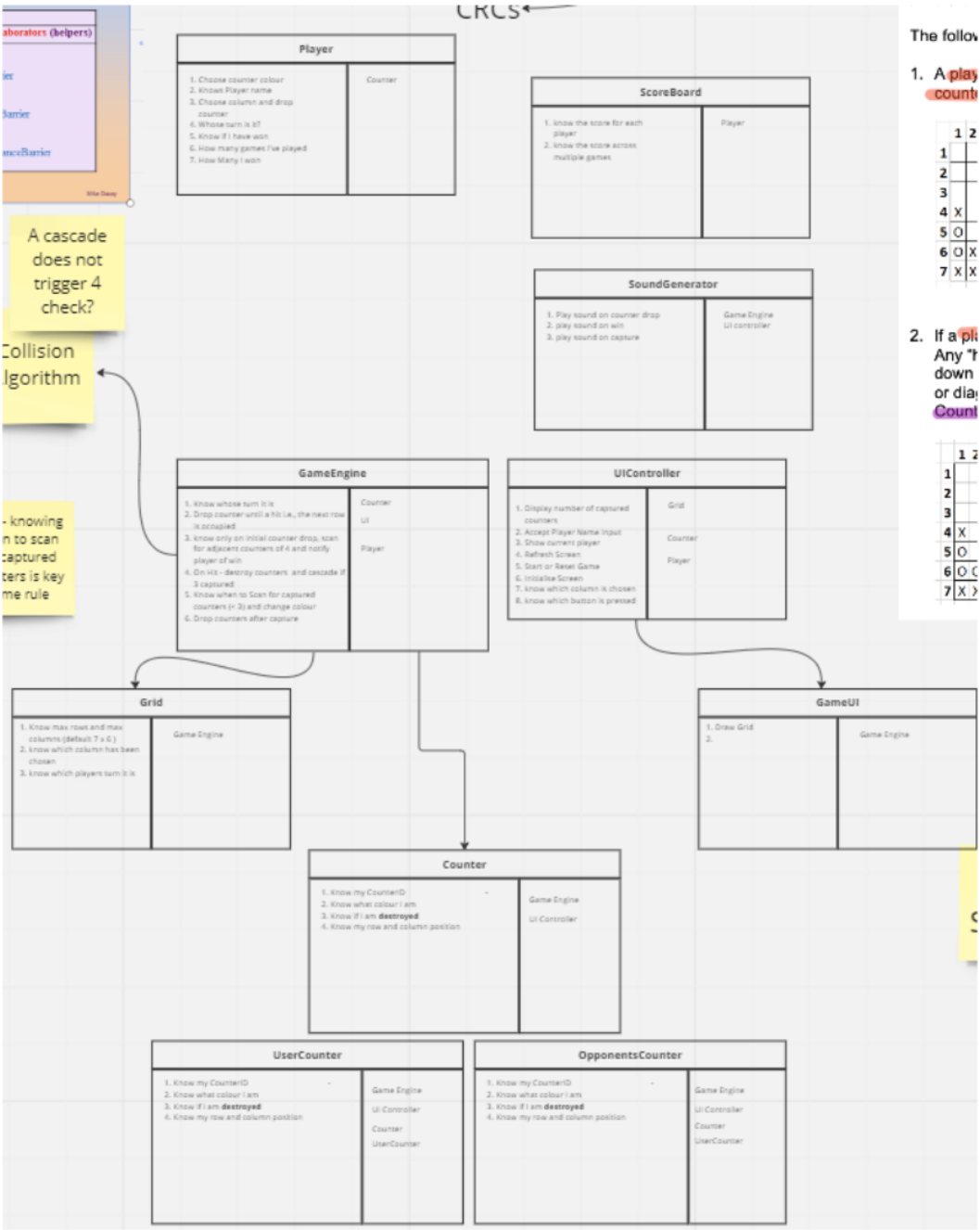
- [15] S. Naidu, "Software Testing," QA Touch, 9 01 2023. [Online]. Available: <https://www.qatouch.com/blog/testability-in-software-testing/>. [Accessed 26 04 2023].
- [16] Smart Bear, "What Is Unit Testing?," Smart Bear, 2023. [Online]. Available: <https://smartbear.com/learn/automated-testing/what-is-unit-testing/>. [Accessed 26 04 2023].
- [17] N. M. i. U. T. W. a. t. p. o. u. testing?, "What is Unit Testing? What are the principles of unit testing?," IROIDTechnologies, 19 08 2021. [Online]. Available: <https://www.iroidtechnologies.com/blog/unit-testing-principles>. [Accessed 26 04 2023].

5. Appendix

Groups User Stories:

User Activities	Load App				Configure the game	Play the game				Win the game	End of the game		
User Tasks	Load page	Account	Login	View Progress	Customise Game	Playing the game	Select a column	Drop a counter		View Outcome	Notified of win	Choose to end & exit	Choose to restart a new game
Working Features	As a player I want to load a game so I can begin playing				As a player I want to customise my game so I can play my favourite game	As a player I want to play the game so I can enjoy myself	As a player I want to select a column so I can place my counter	As a player I want to drop a counter so I can play the game		As a player I want to view the outcome of the game so I can see if I have won	As a player I want to be notified of my win so I can celebrate	As a player I want to choose to end the game so I can play another game	As a player I want to choose to restart a new game so I can play again
User Stories	As a player I want to load a game so I can begin playing	As a player I want to view my account so I can see my progress	As a player I want to login so I can play the game	As a player I want to view my progress so I can see how far I have come	As a player I want to customise my game so I can play my favourite game	As a player I want to play the game so I can enjoy myself	As a player I want to select a column so I can place my counter	As a player I want to drop a counter so I can play the game		As a player I want to view the outcome of the game so I can see if I have won	As a player I want to be notified of my win so I can celebrate	As a player I want to choose to end the game so I can play another game	As a player I want to choose to restart a new game so I can play again
Release 1	As a player I want to load a game so I can begin playing	As a player I want to view my account so I can see my progress	As a player I want to login so I can play the game	As a player I want to view my progress so I can see how far I have come	As a player I want to customise my game so I can play my favourite game	As a player I want to play the game so I can enjoy myself	As a player I want to select a column so I can place my counter	As a player I want to drop a counter so I can play the game		As a player I want to view the outcome of the game so I can see if I have won	As a player I want to be notified of my win so I can celebrate	As a player I want to choose to end the game so I can play another game	As a player I want to choose to restart a new game so I can play again
Release 2	As a player I want to load a game so I can begin playing	As a player I want to view my account so I can see my progress	As a player I want to login so I can play the game	As a player I want to view my progress so I can see how far I have come	As a player I want to customise my game so I can play my favourite game	As a player I want to play the game so I can enjoy myself	As a player I want to select a column so I can place my counter	As a player I want to drop a counter so I can play the game		As a player I want to view the outcome of the game so I can see if I have won	As a player I want to be notified of my win so I can celebrate	As a player I want to choose to end the game so I can play another game	As a player I want to choose to restart a new game so I can play again
Release x	As a player I want to load a game so I can begin playing	As a player I want to view my account so I can see my progress	As a player I want to login so I can play the game	As a player I want to view my progress so I can see how far I have come	As a player I want to customise my game so I can play my favourite game	As a player I want to play the game so I can enjoy myself	As a player I want to select a column so I can place my counter	As a player I want to drop a counter so I can play the game		As a player I want to view the outcome of the game so I can see if I have won	As a player I want to be notified of my win so I can celebrate	As a player I want to choose to end the game so I can play another game	As a player I want to choose to restart a new game so I can play again

CRC Cards:



Group Discussions:

As a discussion point, I think User Stories are only half of a solution when it comes to Agile Software Development

I feel like User Stories are great for explaining little features or events that should occur while a user/actor interacts with but I don't know that they are fit for purpose for gathering larger requirements.

To me, they should be quite detailed and focus on a single point whereas, some requirements may be 'This form should include the following fields' which could be much larger and wider in scope but shouldn't be considered an entire epic.

Gareth Williams (1910285) 13/04 12:58

So I think they can work, as long as it's an iterative process e.g., the first stab and drawing up a story board will have a lot of assumptions, even if end users are actually at the board, they may not have the foresight or imagination to add the stories at the right pitch, but this is the purpose of agile. The story board will evolve. The facilitator will gain experience in knowing how to extract the right level of information. Having a mix of skills, to build the board is key. I think the focus is a lot on the front end user personas - I think at the moment it needs to be extend to personas such as "data analysts" as they will have input in to the data standards and data model that underpins an app.

Morgan Combstock (2014028) 13/04 13:00

Ah okay, so extending a persona beyond a user or type of user, taking it as far as personas who consume the output of a system or application

Gareth Williams (1910285) 13/04 13:09

I think so. In the case of health, you have many personas using the same app - med sec, nurse, consultant, ward clerk,

Miro Link: https://miro.com/app/board/uXjVMUq8YWM=

FINAL GRADE

70/100

GENERAL COMMENTS

Instructor

Eval of Design (20): 15

User Stories (20): 15

CRC (20): 16

SW Testing (20): 19

Testing (20): 5

Feedback:

=====

The evaluation of the design was quite good and there was a good discussion of the team interaction. I would have liked a little more detail on the design process.

It would have been good to see a little more evaluation of testable code.

You needed to provide some examples of tests applied to the game application.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21
