

Software Engineering Principles & Testing ACCB5030

by Matthew Hatherly

Submission date: 02-May-2025 10:41AM (UTC+0100)

Submission ID: 257574459

File name:

49459_Matthew_Hatherly_Software_Engineering_Principles_Testing_ACCB5030_648183_635378116.docx
(4.68M)

Word count: 4476

Character count: 24101

Software Engineering Principles and Testing

Software Engineering Design & Testing

ACCB5030

Matthew Hatherly

2306287@student.uwtsd.ac.uk

Table of Contents

<i>Introduction</i>	<i>4</i>
<i>User Interface Design.....</i>	<i>5</i>
Amended Game Rules	5
User Interface and Wireframe Designs	5
<i>Connect 4 as a Web Application</i>	<i>6</i>
Frontend.....	6
Backend.....	6
Development Tools.....	6
Potential Future Concepts	6
Easy Difficulty	7
Medium Difficulty	7
Hard Difficulty	7
Integration of Artificial Intelligence.....	7
Wireframe Concepts.....	8
<i>User Stories.....</i>	<i>11</i>
<i>CRC Card Design.....</i>	<i>13</i>
Benefits of using CRC Cards	14
CRC Card Relating to Connect 4 Exercise.....	15
<i>Software Testing.....</i>	<i>16</i>
Black Box Testing	16
White Box Testing.....	17
Testable Code.....	19
<i>Unit Testing.....</i>	<i>20</i>
<i>Conclusion.....</i>	<i>22</i>
<i>Bibliography</i>	<i>23</i>

Figure 1: Home Screen Wireframe	9
Figure 2: Game Screen Wireframe	10
Figure 3: Final Screen Wireframe	10
Figure 4: User Story for a Pizza Business	11
Figure 5: User Story Map for Connect 4 Exercise.....	12
Figure 6: CRC Card Layout [3]	13
Figure 7: Example of a Hand-drawn CRC Card [3]	14
Figure 8: Another CRC Card Example [3]	14
Figure 9: Black Box Testing [6].....	17
Figure 10: White Box Testing [8]	18
Figure 11: Unit Testing [9].....	21

Introduction

This report aims to outline the design and evaluation process for an enhanced version of the Connect 4 game, developed as a web application. The game introduces additional rules that modify traditional gameplay and necessitate a more complex backend logic, enhanced user interface (UI), and rigorous software testing strategy. The report reflects on the collaborative and individual aspects of the design process, evaluates the effectiveness of design tools such as CRC cards and user story mapping, and discusses the application of software testing methods including black and white box testing.

User Interface Design

Amended Game Rules

Traditionally, Connect 4 is [1] “a classic strategy game in which two players go head-to-head in a battle to own the grid”. It is traditionally played on a 6w X 7h grid. Players alternate turns to drop different coloured counters into a column. The counter falls to the lowest point in the column that isn’t already filled by another counter. The objective of Connect 4 is to be the first player to form a sequential line of four counters. This can be done vertically, horizontally or diagonally.

For the purpose of this project, there has been a number of amendments made to the traditional game of Connect 4. These include:

1. **Counter Capturing** – A player can capture an opponent’s counter by trapping it between two of their own counters. This can be achieved horizontally, vertically or diagonally.
 2. **Counter Destruction** – If a player is able to capture three of their opponent’s counters, then these counters are subsequently destroyed. Following this, any gaps left from the destroyed counters can be filled by any counters. Similarly to the counter capturing mechanism, these counters can be destroyed horizontally, vertically and diagonally.
- However, an important rule to note about the both the capture and destruction mechanisms is that any counters that are dropped down will not capture or destroy any opponent’s counters.

User Interface and Wireframe Designs

For the purposes of this exercise, this amended game will be viewed as a web application, offering appropriate user interface and wireframe designs.

There are several objectives that need to be met in order to produce a successful user interface. To start, the user interface should be intuitive and clear for the user. A number of other steps should be met, including:

- The board should be set up as a 6w x 7h design, just like the traditional Connect 4 board.
- There should be an indicator stating who’s turn it is to drop the counter.

- There should be an option to clear the board and start the game from the beginning.

Connect 4 as a Web Application

In order to work on creating the enhanced version of Connect 4 as a web application, there are several different tools and technologies that can be used to achieve this. Choosing the correct stack is vitally important to ensure efficiency, particularly when it comes to testing and debugging.

Frontend

For the frontend of the enhanced Connect 4 game, you have several good options, but for ease of use, HTML and CSS would be the best applications for this. With the use of HTML, you can create the overall game board, in addition to the design of the buttons. Thereafter, CSS can be used for the style elements, particularly relating to animations when the counters are captured and/or destroyed.

Backend

As the enhanced Connect 4 game is somewhat simple, a highly functioning backend is not necessarily needed, but should you want to introduce concepts such as a multiplayer function, then Node.js and Express.js can be useful for creating a simple and clean backend for your program.

Development Tools

For the enhanced Connect 4 game, the best development tool would be GitHub. This allows for good version control as well as collaboration with other potential users.

Potential Future Concepts

You could leave the enhanced Connect 4 game at a basic stage, and it would perform its intended purpose at a basic level. However, if you intended to develop the game onto a larger scale, then you could consider allowing the user to play against a computer, or AI opponent. This would eliminate the purpose of having another physical player with you, thus improving replay-ability and accessibility for a larger scale project. In order for the AI opponent to work successfully on a larger scale, there should be several difficulty options, ranging from easy to hard. It should also be able to understand the rules and perform

these functions at a level reflective of the difficulty that the player chose. Below offers an explanation of the various difficulties:

Easy Difficulty

This is the least challenging difficulty that the user can select. With the easy difficulty option, the AI opponent selects random paths as opposed to calculated game paths. It will not try to predict the player's paths, and it won't make use of the capture or destroy features. This is the ideal feature for players who are new to the game and are learning, as well as young players.

Medium Difficulty

The medium difficulty is a step up from the easy difficult option. It is intended to pose a higher challenge but still be suitable for those playing at an intermediate level. In the medium difficulty, the AI opponent knows to block potential threats, such as the player being on the verge of getting 4 in a row. The AI opponent will also perform at a higher strategic level in that they will favour the central columns in order to give themselves a higher strategic advantage. The AI opponent in the medium difficulty will set up its own sequences and patterns, and it will also know and take advantage of the capture and destroy functions, thus adding a further challenge to the player.

Hard Difficulty

Finally, there is the hard difficulty, which is the most challenging difficulty that the player can choose from. In this difficulty, the AI opponent can predict the player's moves at an extremely high level. It can predict these moves and perform subsequent functions using data and algorithms to generate the highest probability of emerging victorious. With these insights, the AI opponent on hard difficulty is capable of playing several steps ahead of the player, as well as being able to leverage the capture and destroy counter options to an extremely high level. The hard difficulty is designed for experienced players or for those looking for a challenge.

Integration of Artificial Intelligence

The integration of AI into the enhanced Connect 4 game could prove to be a welcome addition, as it allows you to move the game onto a larger scale, making it potentially more fitting for a larger player base. In order for the AI functions to work effectively, it is vital that the AI logic can understand the

enhanced features of the Connect 4 game, such as capturing and destroying counters and the changing of the board after these functions occur.

There would be some challenges in terms of implementing artificial intelligence at this level. For example, as Connect 4 is a game that relies on the laws of gravity, breaking these rules could negatively affect the player's experience. When counters are destroyed, they should not break the rules of gravity, and this is an area that should be tested in various scenarios to ensure that it works effectively. In order to make sure that AI rules work as intended, there should be a lot of performance optimisation taking place, particularly in relation to how they work with various difficulty levels. If an end user says, for instance, that the capture feature is too difficult for medium difficulty, for instance, then this could be something to review.

In future versions of the enhanced Connect 4 game, artificial intelligence could be used for several other features. For example, a 'training' feature could be implemented that allows the player to face the AI opponent in several different scenarios. This could improve the player's strategic nous when it comes to beating the AI opponent or other players in future games.

Having multiple difficulties to choose from should hopefully fit a larger player base and subsequently encourage repeat play as it is suitable for players of all levels. It also promotes accessibility which could be an important focus for this game. The AI-system doesn't have to be perfect at first, over time and with numerous testing scenarios it can be developed progressively. At an initial stage, the AI opponent can just offer random moves, and with time it can evolve into a system that can make strategic decisions at a high level, offering a considerably more difficult challenge for the player. With time and development, you can have a system that perfectly fits the range of difficulties suitable for the enhanced Connect 4 game.

Wireframe Concepts

Below lists a number of wireframe concepts, which will include:

- **A home screen** – this will display an option to start a new game, options to adjust the rules before you start, as well as an option to review the rules of the game.

- **A game screen** – this will show the game board, as well as the available counters, a key showing what counters have been captured, as well as an option to restart the game.
- **A final screen** – this is the screen that will be displayed when a game is finished. The victor will be announced, and the final grid will be displayed. There will also be the option to restart a new game.

Below are the wireframe designs that display these concepts:

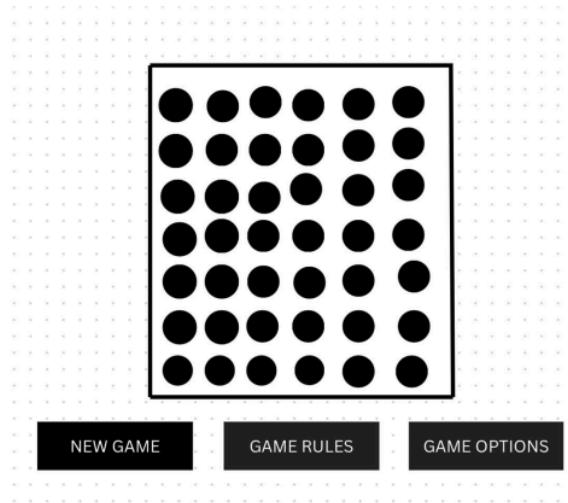


Figure 1: Home Screen Wireframe

The home screen for this Connect 4 game is designed with simplicity in mind. From the left moving in, it allows the user to start a new game, review game rules or change the options, which could allow the user to change the rules should they wish.

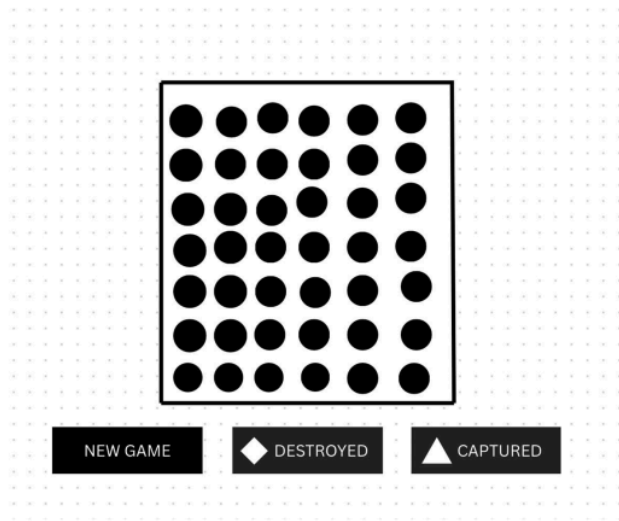


Figure 2: Game Screen Wireframe

As shown in the above image, this shows a game currently in progress. The connect 4 board is displayed with a key signifying which counters have been destroyed and/or captured.

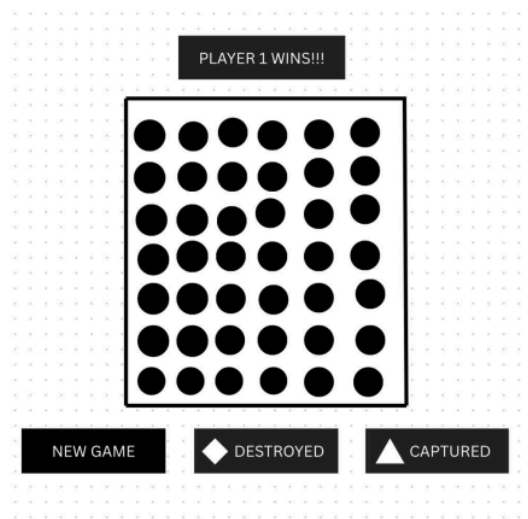


Figure 3: Final Screen Wireframe

As shown in the above image, this is the final game screen. At the top it shows the winning player alongside the final game board.

User Stories

User stories have a number of uses, particularly relating to Agile Management. For example, per Atlassian [2] “user stories are one of the core components of an agile program. They help provide a user-focused framework for daily work – which drives collaboration, creativity, and a better product overall.”

A user story is simply a brief and simple explanation of a feature that is articulated from the point of view from the perspective of a customer or an end user. They aren’t intended to be detailed, but they are beneficial for breaking down large work items that can be broken into smaller, more distinguishable stories.

Below are some examples of user stories:

User persona								
User activities	Choose pizza		Payment		Customer Information		Update user	
User tasks	Customize pizza	Add sides and drinks	Payment type		Insert customer details	Insert delivery address	Update where delivery is	Update customer information details
User stories	As a customer I want to choose different pizzas as I have different preferences.	As a customer I want the option to choose if I want sides or drinks as well as my pizza so I can have more options.	As a customer I want the option to pay with credit card or paypal as I pay with different methods.		As a business analyst I want the customer details to gather basic information about the customer, as this helps with customer identification (CIN).	As a manager I require the address of customers to know where to send the required food.	As a customer I want to be updated on where my delivery is as I want to know when they are coming.	As a customer I want to be able to update my details, as I could move address, change phone number or email.
Release 1	As a customer I want to be able to customize my pizza as there may be preferences that are unavailable.	As a customer I want to choose from a variety of sides and drinks as I want more options.						As a manager I want up to date customer information so that I can send pizzas and drinks to the correct business.
Release 2	As a customer I want different crust options as this will improve the options.		As a customer I want the option to be able to tip based on my experience with driver time taken.					
Release 3								

Figure 4: User Story for a Pizza Business

This example user story above details how a pizza business can be best run. As shown in the image above, the user stories primarily relate to the customer’s perspective, such as being able to pay using various methods including cash and card, as well as being able to choose different types of pizzas and toppings. In future releases, different features can be included, such as being able to add a

custom tip based on customer experience and being able to customise the product more.



Figure 5: User Story Map for Connect 4 Exercise

The above image shows user stories relating to the custom Connect 4 game. The user stories are primarily related to the user's experience. The main user activities are related to playing the game, understanding the game rules, reviewing the status of the game and having the opportunity to restart the game.

The user tasks are related to making a move on the board, learning the rules before beginning the game, following the movements before the game starts, and starting a new game.

The user activities are related to the user wanting to know who's turn it is to drop a counter, understanding the rules of the game before playing, understanding what counters have been captured or destroyed, and having the ability to restart at any time before, during, or after the game.

Some more user stories which could potentially be added in a later release include being able to amend the rules before starting a new game and being able

to start a new game in custom circumstances (e.g. 3 opposition counters destroyed).

Overall, user stories are vital, particularly in relation to agile management. With the successful utilisation of user stories, developers have a clear process in terms of understanding what is important for the user. In relation to the enhanced Connect 4 game, user stories can help understand what the user wants for the game going forward. This can help define the overall scale of the project going forward, and it can also help in creating an enjoyable user experience.

CRC Card Design

Class Responsibility-Collaborator Cards (CRC Cards) are a [3] “collection of standard index cards that have been divided into three sections”. This source follows by stating “a class represents a collection of similar objects, a responsibility is something that a class knows or does, and a collaborator is another class that a class interacts with to fulfil its responsibilities.”

Below shows some example figures related to CRC Cards:

Class Name	
Responsibilities	Collaborators

Figure 6: CRC Card Layout [3]

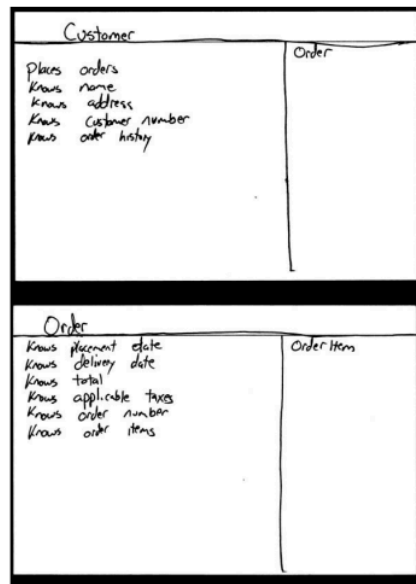


Figure 7: Example of a Hand-drawn CRC Card [3]

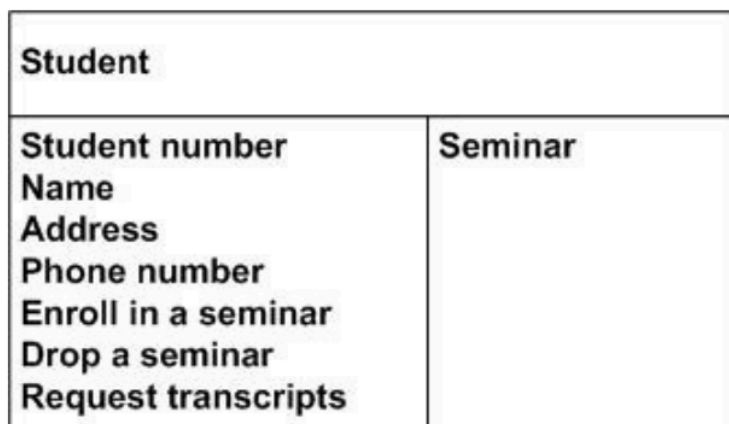


Figure 8: Another CRC Card Example [3]

Benefits of using CRC Cards

Some benefits of using Class Responsibility-Collaboration Cards include: [4]

- **Visualisation** – The use of CRC cards could provide a visual backboard of the overall system design. For developers, it could be useful in terms of being given the opportunity to conceptualise the structure and relationships between the various classes within the system.
- **Collaboration** – When participating in a group setting, CRC cards can allow for efficient and active collaboration across several parties. Every

team member note put their ideas on the cards and subsequently collaborate and trade ideas.

- **Ease of Understanding** – CRC cards can allow for ideas to be clearly articulated into bullet points. In a group setting, this can be very useful in terms of dividing tasks and responsibilities. Giving every team member a clear knowledge of what they need to do can allow for a more efficient and streamlined work environment.
- **Iterative Design** – CRC Cards allow for an iterative design process. In numerous situations, task requirements and designs may change. Therefore, CRC cards being easy to change and adjust could prove beneficial relating to this process. Using CRC cards can allow for an easy adjustment during various task stages.
- **Rapid Prototyping** – With the proper effective and proper use of CRC cards, organisations and teams would be able to use these cards to experiment with various [4] “class hierarchies, responsibilities and collaborations.”

CRC Card Relating to Connect 4 Exercise

Class – Connect4Board
- Responsibilities
Falling Counters
Capturing Counters
- Collaborators
Player, Counter

Overall, Class Responsibility-Collaborator Cards can be a valuable tool for understanding and conceptualising the structure prior to beginning the code development process. Relating to the enhanced Connect 4 game, CRC cards can allow for a more modular environment, as well as understanding the classes, responsibilities, and collaborators. This should lead to cleaner, more modular code that should improve the development process, and assist in testing and debugging.

Software Testing

Black Box Testing

Effectively, black box testing means that the tester reviews how an application works without having a clear idea of how its coded or designed. Some other terms of black box testing are called [5] “specification-based testing, closed box testing, or opaque box testing.”

Generally, [5] “the black box methodology is used to test most software applications.” The benefits of black box testing include the fact that many test cases are covered, which subsequently gives the best chance to discover bugs. Black box testing can be used in nearly all stages of software development.

Some important things that black box testers look for are [6] “how the system handles expected and unexpected user actions, response time, and usability and reliability issues.”

An example provides by Testlio is [6] “for instance, you check if the checkout process works smoothly when a user adds items to their cart and proceeds to payment. You also test how the system handles errors, like when a user enters incorrect payment information. You may need to make sure it provides clear error messages and prompts for corrections.”

Testers don’t generally need a large technical knowledge in black box testing as it is more geared towards relating to and refining the user requirements. A lot of knowledge can be gleaned from black box testing as its generally less precise than white box testing, which while it has its downsides, does also mean that the tester can stay true to the bigger picture.

Generally, black box testing occurs [5] “post-completion of development, and both processes are independent.”

Black Box Testing

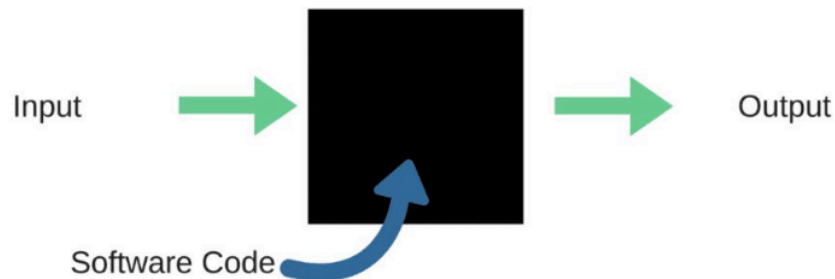


Figure 9: Black Box Testing [6]

Black box testing is beneficial as it tests from the perspective of the end user. It is extremely valuable as it doesn't require an understanding of the code, but rather just the functionality of the application. Relating to the enhanced Connect 4 game, black box testing can be useful for ensuring animations work as intended, as well as game outcomes and user interface messages. With successful black box testing, this can improve the user experience, which a reliable and efficient program that players can enjoy.

White Box Testing

White box testing is style of application testing that will give the tester full knowledge of the application that is being tested. This includes things such as the design documents and the source codes.

There are a number of things that white box testing focuses on, and these include: [7]

- **Path Checking** – “White box testing can be used to explore the various execution paths within an application to ensure that all conditional statements are correct, necessary and efficient.
- **Output Validation** - “This enumerates the various potential inputs to a function and ensures that each produces the expected result.”
- **Security Testing** – “Static code analysis and other white box testing techniques are used to identify potential vulnerabilities within an

application and validate that it follows secure development best practices.”

- **Loop Testing** – “Tests the loops within an application to ensure that they are correct, efficient, and properly manage the variables within their scope.”
- **Data Flow Testing** – “Tracks variables throughout the execution paths of a program to ensure that variables are declared, initialized, used, and properly manipulated.”

There are three known methods of white box testing. One of them is unit testing, which is designed to make sure that all functions and components within an application work to the correct standard.

There is also integration testing, which refers to the interfaces within an application, and ensuring that these interfaces work both by themselves as well as alongside other components.

Finally, there is regression testing. This is effectively where various functionality or security changes are made to an application in order to test that the application will still function as normal in spite of any changes.

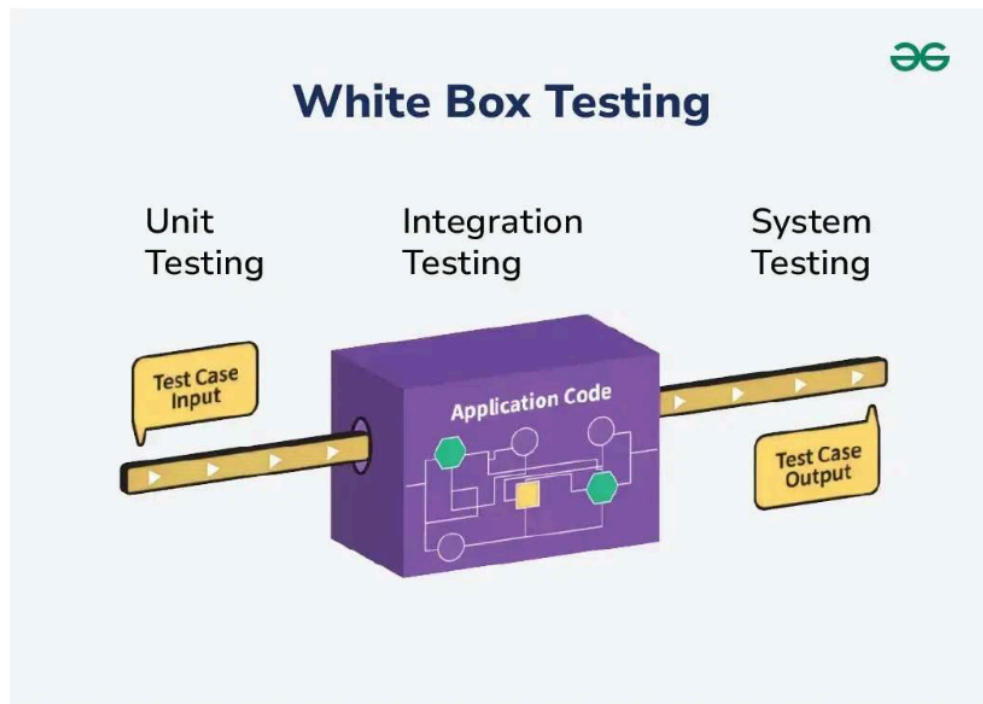


Figure 10: White Box Testing [8]

There are three main techniques to white box testing. These include: [8]

- **Statement Coverage** – This verifies that each line of code within an application is executed by one test case at a minimum. This method aids in detecting unused or unreachable code segments, which may result from programming mistakes, updates, or other issues. By identifying these flaws, developers can correct faulty conditions or eliminate unnecessary code, enhancing both performance and security.
- **Branch Coverage** – Conditional statements introduce branching in an application's execution flow, as different inputs may take different paths. Branch coverage testing ensures that every possible branch is tested, validating even less frequently used code paths.
- **Path Coverage** – An execution path represents the sequence of instructions from the start to the end of an application. Path coverage testing guarantees that all possible paths are tested, helping confirm that they function correctly, operate efficiently, and serve a necessary purpose.

White box testing plays a vital role in ensuring that the logic and structure of a program works to the desired level. Successful white box testing means developers can review their work and ensure that everything functions as intended. In context of the enhanced Connect 4 game, white box testing can review a number of in-game mechanics, such as AI-logic and how the board changes. With successful and efficient white box testing, developers can catch potential issues early on, thus saving on time and money.

Testable Code

Testable code refers to software designed in a way that facilitates efficient and effective testing. It is structured to allow individual components (units) to be isolated, mocked, and validated with automated tests. Key characteristics of testable code includes:

- **Modularity** – Code is divided into small, independent units (functions, classes, or modules) with single responsibilities.
- **Low Coupling** – Components have minimal dependencies on each other, making them easier to test in isolation.
- **High Cohesion** – Each module focuses on a well-defined task, reducing complexity.

- **Dependency Injection (DI)** – External Dependencies (e.g. databases, APIs) are injected rather than hardcoded, allowing for mock replacements during testing.
- **Avoidance of Global State** – Global variables and static singletons making testing unpredictable, so testing code minimises their use.

Testable code is vital for a number of reasons. For example, it improves software reliability by catching issues within a software early on. It is also easier to maintain code when code is testable and well-structured. If there is a clear plan to write testable code at the start of a project, then it is a lot cheaper should debugging or rewriting occur.

Overall, writing testable code is extremely important when it comes to development and maintaining software. When code is written in a modular and easy to read format, it renders it considerably less cumbersome to read and review the code, especially when it comes to revisiting at later stages. In relation to the enhanced Connect 4 game, writing testable code means that testing the game's various features will become far quicker and easier. This subsequently leads to easier testing and debugging. Ultimately, testable code should be a focus within all development areas for both small-scale and large-scale projects.

Unit Testing

Unit testing involves verifying the smallest testable components of a software application, known as units. These units are typically individual functions, methods or classes. By systematically testing each unit, developers can maintain high code quality, making unit testing a fundamental practice in software development.

A best practice in modern development is to structure code into small, independent units and create corresponding units for each. Developers often write these tests before or alongside the actual implementation – a technique known as Test-Driven Development (TDD). Automated test suites can then execute these tests whenever code changes are made, immediately flagging any failures. This allows developers to quickly pinpoint and fix bugs, reducing debugging time.

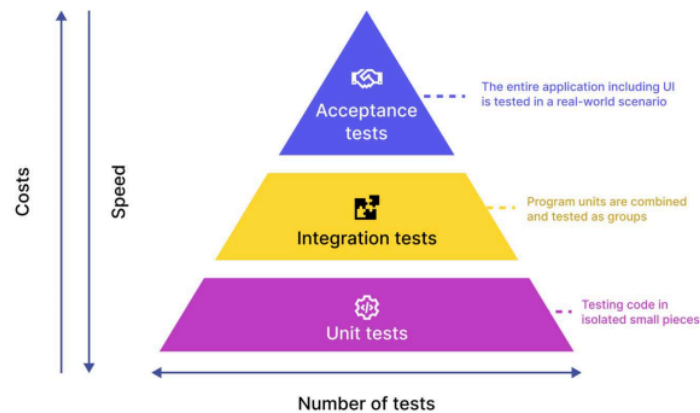


Figure 11: Unit Testing [9]

Beyond error detection, unit testing promotes modular design, encouraging cleaner, more initiable code. It also enhances test coverage, ensuring that critical logic is thoroughly validated. By automating these tests, teams save time and effort, allowing them to focus more on feature development rather than manual verification.

Overall, unit testing plays a vital role in ensuring a high level of efficiency and reliability within a range of projects, but, particularly, it poses benefits within the software development industry. By isolating the smallest testable components of a software application, developers can understand and verify that each component works as intended. Throughout the development process, this can aid with catching bugs at an earlier stage. Relating to the enhanced Connect 4 game, unit testing could be useful for both the AI functions, but also to produce a more enjoyable gaming experience for the user.

Conclusion

This report has outlined the process of designing, testing and evaluating an enhanced version of the Connect 4 game as a web application. By introducing new gameplay mechanics – such as capturing and destroying counters – the project required a deeper level of planning, particularly when it came to the user interface design, class structure, and software testing. Personally, the development of wireframes assisted me in establishing a clear and intuitive user journey, while the use of CRC cards provided a collaborative and modular approach to class design. Additionally, both black box and white box testing strategies were explored and evaluated to ensure reliable and efficient testing.

There was also an emphasis put on writing testable, modular code that supports unit testing and facilitates debugging and future updates. Overall, the project not only enhanced but also highlighted the importance of structured design, teamwork, and thorough testing in delivering a functional and user-friendly application.

Bibliography

- [1] "PRODUCT INFO," Hasbro, [Online]. Available: <https://instructions.hasbro.com/en-my/instruction/connect-4-game>. [Accessed 1 May 2025].
- [2] M. Rehkopf, "User stories with examples and a template," Atlassian, [Online]. Available: <https://www.atlassian.com/agile/project-management/user-stories#:~:text=User%20stories%20are%20one%20of,and%20a%20better%20product%20Overall..> [Accessed 1 May 2025].
- [3] "Class Responsibility Collaborator (CRC) Cards: An Agile Introduction," Agile Modelling, [Online]. Available: <https://agilemodeling.com/artifacts/crcmodel.htm>. [Accessed 1 May 2025].
- [4] "Class-Responsibility-Collaboration Card," geeksforgeeks, 15 April 2024. [Online]. Available: <https://www.geeksforgeeks.org/class-responsibility-collaboration-card/>. [Accessed 1 May 2025].
- [5] H. Ashtari, "Black Box vs. White Box Testing: Understanding 3 Key Differences," SpiceWorks, 2 September 2022. [Online]. Available: <https://www.spiceworks.com/tech/devops/articles/black-box-vs-white-box-testing/>. [Accessed 10 April 2025].
- [6] "The Top Black Box Testing Techniques," testlio, 8 February 2025. [Online]. Available: <https://testlio.com/blog/top-black-box-testing-techniques/>. [Accessed 10 April 2025].
- [7] "What is White Box Testing?," CheckPoint, [Online]. Available: <https://www.checkpoint.com/cyber-hub/cyber-security/what-is-white-box-testing/>. [Accessed 1 May 2025].
- [8] "White box Testing – Software Engineering," geeksforgeeks, 25 February 2025. [Online]. Available: <https://www.geeksforgeeks.org/software-engineering-white-box-testing/>. [Accessed 1 May 2025].
- [9] K. Team, "What is Unit Testing? A Complete Guide," Katalon, 24 March 2025. [Online]. Available: <https://katalon.com/resources-center/blog/unit-testing>. [Accessed 1 May 2025].

Software Engineering Principles & Testing ACCB5030

GRADEMARK REPORT

FINAL GRADE

54/100

GENERAL COMMENTS

Eval Des (20): 7

User Stories (20): 12

CRC (20):11

SW Testing (20): 14

Unit Testing (20):10

FEEDBACK:

=====

The discussion of user stories, mapping and CRC are a limited and with more examples and explanation required.

The evaluation was very brief, I was hoping for more analysis.

Discussion of testing is quite gooc but the explanation of testable code needed to be expanded upon.

Unit test needed to be expanded upon such as what is it, how does it look what are its benefits? Also you needed to provide more test examples for the application.

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

PAGE 22

PAGE 23
