

Software Engineering Principles and Practice - Kieran Perrin

by Kieran Anthony Perrin

FILE	24902_KIERAN_ANTHONY_PERRIN_SOFTWARE_ENGINEERING_PRINCIPLES_AND_PRACTICE_-_KIERAN_PERRIN_302868_1242996349.DOCX (37.77K)		
TIME SUBMITTED	08-JAN-2021 05:43PM (UTC+0000)	WORD COUNT	1908
SUBMISSION ID	141288907	CHARACTER COUNT	8700

Kieran Perrin

Software Engineering Principles and Practice

Connect Four Design Evaluation

Contents

Introduction	2
Design Evaluation.....	2
User Stories	2
User Story Map	3
CRC Cards	4
Software Testing	5
Black Box Testing	5
White Box Testing	6
Version Control	6
References	7

Introduction

In this document I will discuss the elements of design and testing used when creating a connect four game. The aim is to evaluate the methods used to create the design and discuss their benefits as well as discuss the different types of testing that should be done to make sure the product is ready. Lastly, I will discuss some of the benefits of implementing some form of version control during coding of the project.

Design Evaluation

The aim of the scenario provided was to generate a design specification and implement the design for a connect four game. The scenario is specific in what the product should include, which is a working connect four game with some added new rules.

To start the design process, I broke down the connect four game into its simplest parts and listed it as the basic functions required starting with the simplest to implement down to the hardest:

- Draw 6 wide 7 high grid
- Set player 1 and player 2
- Give players a rotational turn
- Allow players to choose a column
- Chosen column has counter occupy the lowest unoccupied row
- If a row is occupied the counter will go to the row above
- Player wins by having 4 counters connected in a line horizontally, vertically or diagonally
- If a counter has enemy counters flanking on either side, the counter converts to enemy counter
- If 3 counters in a line have enemy counters flanking, counters are destroyed, and counters above drop down
- Counters that drop after destruction cannot capture or destroy other counters

From this point I then started to look at the problem from the eyes of the player of the game and started to identify the needs the player has, to be able to play the game properly. I then started to relate the basic functions I mentioned prior to the player and worded them in a way from the user's perspective.

Overall, I found that creating the design did help in the coding of the product. Knowing all the small features individually and working on one at a time helped in making sure the program had everything it was supposed to. It also helped relieve stress to not think about the entire scope of the project and just focus on a single feature at a time. I feel the methodology used would be better if I were working as a part of a team and was in constant contact with the end user.

User Stories

The design methodology chosen was to create a user story map with user stories that outlined all the features the user would require when playing the game. A user story is something that "describes functionality that will be valuable to either a user or purchaser of a system or software [1]." The reason why we would use user stories over other methods is because with user stories we are able to recognise the features that the users of the product want and can prioritise those, so the user gets a product in the end that is as close to what they wanted as possible.

There are many benefits to using user stories and user story maps to other methodologies. One of the main benefits is that it fosters collaboration [2]. Because there is not a large requirements

document involved and there is minimal writing of the users needs done, it encourages more discussion with the users which allows the designers a better perspective of what the user requires. There is also more transparency when it comes to the development of a product due to the user stories being written on index cards that can be seen by everyone including the user. This allows the user to see what features are being worked on and can dictate what is a priority or not. Another benefit is the ability to split into smaller coding teams and work individually on each story, each adding features to the overall product, which can improve in productivity and the amount of time it takes to complete the project. In 2016 Utrecht University published a paper looking at the use and effectiveness of user stories in practice, using surveys to ask questions to practitioners. This survey found that there is an overall positive view on the use of user stories and allows practitioners to create the “right software” [3].

User Story Map

Below is the user story map that I came up with for the design of the connect four game.

User persona - Player							
User activities	Open game	Play with friend			Place counter	Capture counters	Win game
User tasks	Show grid	Player 1 create session	Player 2 connect to player 1	Players take turns	Player chooses column	Player places counter to capture enemy counter	Player places 4 counters in a row
User stories release 1	As a player I want to open the game and see the 6x7 grid ready to play	As a player I want to be able to play with a friend	As a player I want to join a session by entering a code	As a player I want to take a turn in rotation with the other player	As a player I want to choose a column to place my counter		As a player I want to win the game if I get 4 counters in a row horizontally, vertically or diagonally
		As a player I want to create a unique session for a friend to join			As a player I want my counter to drop to the lowest row of the column chosen		As a player I want the game to draw if all counters are placed and there is no connect 4
		As a player I want to have a code to give to player 2 to			As a player I want my counter to be placed in the row above if the		

		join the session			other players counter is in the row.		
User stories release 2	As a player I want the grid to look like a connect 4 board			As a player I want to be able to skip a turn	As a player I want the game to not allow me to place a counter in a column that is full	As a player I want to capture the other players counter by placing my counters on either side(flanking) horizontally, vertically or diagonally	
						As a player I want to destroy the other players counters if I place my counters flanking 3 connected counters	
						As a player I want counters to drop into empty spaces if counters are destroyed	
						As a player I want dropped counters to not be able to capture other counters	

CRC Cards

CRC(Class, Responsibility, Collaborator) cards help us conceptualise the classes that are involved in an object-oriented program. CRC allows us to write down what an object knows and what other classes it will interact with. I think the use of CRC cards is not detrimental to the design process in any way as it such a quick process. It allows for collaborators to discuss the classes that would be implemented into a system and talk through how the classes would work and interact. I think if you

were to only use CRC cards to design a system you would come up short on key design points but as a wheel in a cog of other design methodologies it can be very helpful. As for using CRC cards after creating a user story map I think it was useful. Because I already knew the features that needed to be implemented writing up the CRC cards for the classes was easily done and allowed me to think about the relationship between the classes.

Below are the CRC cards I came up with for the connect four game.

Class: Player	
Responsibilities	Collaborators
1. Create Game session 2. Join Game session 3. Leave Game session 4. Place counters on the game board 5. Alternate turns	Counters, Gameboard

Class: Counters	
Responsibilities	Collaborators
1. Placed in gameboard by player 2. Drop to lowest column 3. Change colour if trapped 4. Destroyed if three counters trapped by opposing colour 5. Drop down into empty space if counter below is destroyed	Player, Gameboard

Class: Game Board	
Responsibilities	Collaborators
1. Display game board 2. Show counters placed 3. Update board every piece placed 4. Only allow player to place counter in empty space 5. Track counters for connect four 6. Check for counters being trapped by the other colour 7. Check for three counters being trapped by counters of the other colours	Counters Counters Counters Counters Counters

Software Testing

Software testing is a very important part of the creation of a product. It is a review of the quality of the product to make sure it is up to specification and that the functionality of the program is correct, error free and reliable. "The key to software testing is trying to find the modes of failure - something that requires exhaustively testing the code on all possible inputs. [4]"

Black Box Testing

Black box testing is the type of testing used to check if the software performs according to the design specification. To set up a black box test you should examine the design specifications,

generate some valid inputs, determine the expected outcome, create a test case for the inputs, execute the test case then compare the expected output with the actual output.

White Box Testing

White box testing is the type of testing used to test the logic of software. It is the method of testing you would use to test the construction of your code to find hidden flaws. To set up white box testing you must have access to the code. The testing usually requires you to modify the code to change values to force different paths of the program. This modification can be done by using interactive debuggers or physically changing the code.

Version Control

Using version control while developing a program can be very useful. Using version control software can help when it comes to working in a team and storing versions of the program. Every time you make changes to the development of a program, you want to be able to store it as a new version of that program in case you want to roll back the changes that you just made. If the changes you made break the program in some way, then backing up to an earlier version can be a big help. Also, when collaborating with people, you want to be able to work on the same file at the same time, using versions can allow for you both to work on the file and save as different versions then merge them later.

References

- [1] M. Cohn, User Stories Applied for Agile Software Development, Addison-Wesley, 2004.
- [2] K. Ravlani, "6 Benefits of Good User Stories," Agile For Growth, September 2017. [Online]. Available: <https://agileforgrowth.com/blog/userstory-benefits/>. [Accessed January 2021].
- [3] F. D. J. M. E. v. d. W. Garm Lucassen, The Use and Effectiveness of User Stories in Practice, The Netherlands: Utrecht University, 2016.
- [4] S. Ahamed, "STUDYING THE FEASIBILITY AND IMPORTANCE OF SOFTWARE TESTING: AN ANALYSIS," *IJEST*, vol. 1, no. 3, p. 10, 2003.

Software Engineering Principles and Practice - Kieran Perrin

GRADEMARK REPORT

FINAL GRADE

50/100

GENERAL COMMENTS

Instructor

Evaluation of design(10): 5

User Stories + Map (Benefits) (20): 13

CRC cards (20): 11

Software testing evaluation (20): 5

Version Control (10): 4

Coding (20): 12

Comments:

The design evaluation did not contain very much reflection on the design process, and needed more discussion of the design stages.

The assignment asked you to give more analysis of a couple of stories with emphasis on the benefit or ROI. This needed to be developed more in your discussion.

More evaluation of the CRC cards was required....did they reflect what was actually coded?

Test evaluation and version control discussion was very brief

No git logs provided.

More references reqd.

Some nice code with a number of functions. but no test code.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6



Comment 1

Class names should be singular i.e. Counter

