SOFTWARE DEVELOPMENT PROJECT 7,5 hp

Lone Nilsson

Linnæus University

2019-03-28

Contents

Appendix

Attachment 1, Class Diagram for Hangman Game

Attachment 2, Use Cases for Hangman Game

Attachment 3, Use Case Models for Hangman Game

Attachment 4, State Chart Diagram for Hangman Game

Attachment 5, Testreport for Hangman Game

1. Reversion History

Date	Version	Description	Author
2019-02-08	v.0.1.0	First iteration: Planning and structuring work	Lone Nilsson
2019-02-21	v.0.2.0	Second iteration: Implemented models and easy program code implemented	Lone Nilsson
2019-03-08	v.0.3.0	Third iteration: Worked with testing, manual and automate unit tests.	Lone Nilsson
2019-03-28	v.1.0.0	Fourth iteration: Improved vision, project plan. Updated models and use cases. Finished code implementation and did decisions about tests.	Lone Nilsson

2. General Information

Project Summary	
Project Name	**Project ID**
Hangman	ln222tu_1dv600, hangman-game (https://github.com/onlylonely1986/ln222tu_1dv600/tree/master/hangman-game)
Project Manager	**Main Client**
Lone Nilsson	**Young students learning english words. It is usable in primary school.**

Key Stakeholders

- **Developer**
- **Tester**
- **Main clients**

Executive Summary

This hangman game is developed to use as a tool in English education in primary school in Swedish school. Hangman is an easy game when the player tries to recognize and guess different words, letter by letter. If the tries run out a picture of a hanged man grows step by step.

3. Vision

The vision about this project is to make an application of the game hangman played in the console. Is should be a good and funny tool to use in Swedish primary school, to motivate and inspire kids to learn and become familiar with new English words. It will also be a good way for young students to get contact with programming-tools when playing a game in a console. It should be really motivating for children who are not very motivated in learning in English words by books and talking. Some of us prefer playing a game and get new knowledge at the same time. The use of this game can with advantage be a part of a larger packages of mini-games.

Reflections about writing a vision

(In my fourth iteration I am updating my vision and my reflection about writing it.)

A vision is important because every person in a development-team should be on the same goal to reach. It should summarize the projects scope and inspire people in the develop-team to reach the goal. I think it is important to be clear and at the same time have high expectations of what it could become. During this course I have learnt a lot, and I now have a bit higher understanding about each part of the project and what is the difference between for example vision and next passage in report about the more objective explanation about how the project should look like.

4. Project Plan

4.1 Introduction

A hangman game usable in primary school should be made in Java-script.

4.2 Justification

It is a small developing project to be made and at the same time learn more about how to plan and work with a software project in a professional way. The game should be used in primary school as a part of package of games for training English language.

4.3 Stakeholders

Project manager is the person responsible for running the project from start to end.

Programmer/software developer is the person who writes the software-program, her interest is to produce a code with good structure for easier testing it or maintain it.

Tester is the person that writes tests and try out if the program is safe and run well, without any bugs.

Client is the company that ordered the assignment.

End users is the people who can use the program in the end. In this case it is primary school students.

4.4 Resources

Resources is money, time, software, hardware and staff who work with the project.

In this case there is no money involved because it is a university course, the time is set by the length of the course, 9 weeks, 20 hour/ week. Sum 180 hours.

The hardware to work with is my own Lenovo-computer.

The software in this project is Visual Studio Code, Node.js, gitbash and github.com. I am project manager, developer and tester, so the only employee in this project is me. I run every role in it.

4.5 Hard- and Software Requirements

The hardware to work with is my Lenovo-computer. It should also run the game in a terminal, on my computer or in another one.

The software in this project is Visual Studio Code, Node.js, gitbash and github.com. The requirements by these programs is

4.6 Overall Project Schedule

w. 6 Iteration 1, planning, sceletoncode, report (8/2)

w. 8 Iteration 2, planning, models, implemented standard code, report (21/2)

w. 11 Iteration 3, planning tests, testing code, report (8/3)

w. 12 Iteration 4, last planning, models, tests, implement, hand in full report (22/3)

4.7 Scope, Constraints and Assumptions

The vision about this project is to make an application of the game hangman played in the console. The user should be able to choose from a menu what to do from a list of numbers. The picture of a hanged man should be presented with ACII-signs. The game is about to guess a word from empty placeholders, try to guess letter for letter. If the letter is correct it will be put in the right place, and if wrong the picture of a hanged man should be shown step by step. The words are taken from a document of words (nouns) and will randomly be picked every time a new game is started. The user should be able to write in a username and save results in a high score list. The game should be built in JavaScript by separate modules for each functionality.

This project is about developing a console-based game, hangman. It is also about learning more around the professional process when developing software projects. The scope reaches in this case not so much about design, frameworks, and good-looking stuff, but it is not either unimportant. It is more about learning analysis, *what* to develop - w*hat* kind of application it should be and how to plan my work and follow my plan in a really good way. It is common to have opportunity to start new game, quit game and a button for reading “how to play”. In the scope I am also including opportunity to log in and log out with a user name and save results in a highscore-list.

Some constraints to take in account is my own lack of experience and of course, even if I make a good plan and risk analysis and stuff the time limit could always be a problem.

I have one assumption about users in this case. I think that they are not familiar with using consoles or terminals. But I also have in mind to build it very clearly so that every person with some knowledge about computers and computer-games have ability

to run the game. And it is also a goal with this application to give young primary school students new experience about using a terminal for playing.

5. Iterations

5.1 Iteration 1

w.4

- Introduction in course (2 h)
- Reading trough material (2 h)
- Reading the book: Software Engineering, Ian Sommerville, (5 h)
- Starting to prepare the rapport in word (2 h)
- Work with questions to prepare before exam 1 (8 h)
- Listen to rerecorded lecture (1,5 h)

(Sum: 20,5 h)

w.5

- Reading the book: Software Engineering, Ian Sommerville, (10 h)
- Prepare for online-exam (8 h)
- Make online-exam 1, (0,5 h)
- Listen to rerecorded lecture (1,5 h)
- Listen to lecture (1,5 h)
- Working with rapport in word (4 h)

(Sum: 25,5 h)

w.6

- Reading the book: Software Engineering, Ian Sommerville, (5 h)
- Searching more information about the project (2 h)
- Working with rapport (10 h)
- Building skeleton-code and push it to github.com (1 h)

(Sum: 18 h)

5.2 Iteration 2

w.7 - 8

- Reading the book: Software Engineering, Ian Sommerville, (5 h)
- Listen to lectures, (8 h)
- Make online-exam 2, (5 + 0,5 h)
- Writing on rapport and searching for more knowledge (10 h)
  - build use case diagram
  - write use case text
  - create UMLs
- Building up code (10 h)
  - start up with skeleton-code, using UMLs and use case
  - build a functional game
  - build in a database with words

(Sum: 38 h)

5.3 Iteration 3

w.9–10

- Reading the book: Software Engineering, Ian Sommerville, kap 8 (2 h)
- Listen to lectures, (8 h)
- Make online-exam 3, (8 + 0,5 h)
- Writing on rapport and searching for more knowledge (8 h)
- Task 1 - test plan (2 h)
- Task 2 – Manual test cases (5 h)
- Task 3 – Unit tests (5 h)
- Look up Greeter Application, youtube, github (3h)

(Sum: 38 h)

5.4 Iteration 4

w.11–12

- Planning iteration (1 h)
- Working on report, make it better (2 h)
- Q & A with Tobias (4 h)
- Make models (3 h)
- Implement code (5 h)
- Plan tests (3 h)
- Make testing (3 h)
- Implement better interface ASCI (6 h)
- Finish report (4 h)

(Sum: 30 h)

6. Risk Analysis

Project managers must assess the risks that may affect a project, monitor these risks, and act when problems arise. In this project it is some sort of risks to deal with. Specially because I am the developer, and my experience is not so great yet. If you are more experienced, you have learnt from your earlier mistakes. It is more difficult to plan if you have not developed much before. It is difficult to deal with how much time it will take, and how to create the plan and vision. Because I don't really know everything about what you *can* do, and what is my highest level to reach at this moment?

6.1 List of risks

Project risks:

Risks that is about the project itself. I can be about the developer being sick or other unexpected things coming up during the project. About the hardware or software or about the employees. It is important that the project manager put right person in right place. Risk I identify at this moment:

- time-plan (high risk)
- knowledges/experience (moderate risk)
- failures/difficulties (high risk)

Product risks:

A product risk influences the quality of the software being developed or the hardware itself.

Risk I identify at this moment:

- bad solutions that slows the system down (moderate risk)

Business risks:

Business risks affect the organization developing or procuring the software. For example, a competitor introducing a new product is a business risk. The introduction of a competitive product may mean that the assumptions made about sales of existing software products may be unduly optimistic.

Risk I identify at this moment:

I should not say that there are any business risks in this project because it is a course at university. Of course, there are lots of competing hangman games out there, so it will probably not sell very well.

6.2 Strategies

I must manage risks and take control over them by avoiding it or by minimizing its effects on the project. If I make a good plan and thinking about every step to get through, it should be easier to not fail the time-plan. I must be one step ahead because I am not only student, also a mother and I can't plan if my kids or me get sick any day in the middle of project. But with hard work and good planning I minimize the risk about failing on this one.

The other point about my own knowledges, is to not set too high goals, because even small developing projects could be difficult to finish when you are not that experienced yet.

The third thing about failures that causes me as developer problems, both with time plan and maybe also affect the product, so the computer gets slow caused by bad solutions. I don't really know how to manage or minimize these effects, because I think it is a part of programming to struggle. Problem solving is what it is. And therefore, it also is a part of time planning, to set reasonable goals and put in lots of extra time in the project to get rid of troubleshooting process.

7. Time Log

7.1 Iteration 1

When:	What to do:	Time it will take:	Time it took:	Reflection:
w.4 22/2	Reading trough material		2 h	
23/2	Introduction in course		2 h	
	Reading the book: Software Engineering		5 h	
	Starting to prepare the report in word		2 h	
	Work with questions to prepare before exam 1		8 h	
	Listen to rerecorded lecture		1,5 h	
	Sum:	20 h	20,5 h	
w.5 28/2	Reading the book: Software Engineering		10 h	
	Prepare for online-exam		8 h	
	Make online-exam 1		0,5 h	
	Listen to rerecorded lecture		1,5 h	
	Listen to lecture		1,5 h	
	Working with report in word		4 h	
	Sum:	20 h	25,5 h	
w.6 4/2	Reading the book: Software Engineering		5 h	
4/2	Searching more information about the project		2 h	
5/2	Working with report		1 h	
6/2	Building skeleton-code and push it to github.com		1 h	
6/2	Write on report	2 h	4 h	It took longer time than I thought, to understand and think about what to write and how to describe it.

7/2	Write on report	3h	2 h	
8/2	Write on report, complete the first iteration	2h	2, 5 h	It took longer time than I thought, to understand and think about what to write and how to describe it.
	Sum:	20 h	17,5 h	
	Sum Iteration 1 (w.4-6)		**63 h**	I have put in more time then the course requirement says, it is a 50 %-course, so it is fair to put in approximately 20 hours a week. And it is about what I did.

In this first part I have done estimates afterwards (first 2 weeks in my time log) because I did not pay attention at this in a early time. So next time, I should be more attentive and read through all parts before I start my work.

7.2 Iteration 2

When:	**What to do:**	**Time it will take:**	**Time it took:**	**Reflection:**
w.7 11/2	Reading trough material on course site	1 h	1 h	
11/ 2	See tutorial from last week	2 h	1 h	Split it up I two parts, and continue later.
11/2	See tutorial from last week	1 h	1 h	
11/2	Reading chapter 4 in book	1,5 h	1,25 h	
12/2	Reading chapter 5 in book	0,5 h	0,5h	
13/2	Reading chapter 5 in book	1 h	1 h	
13/2	Build a use case diagram	0,5 h	0,5 h	
13/2	Start coding something	1 h	1 h	
13/2	Search more knowledge about UML	1 h	1 h	
13/2	Listen to rerecorded lecture 4	1,5 h	1, 5 h	
14/2	Reading the book chap 5	1, 5 h	1 h	
14/2	Listen to rerecorded lecture 5	1, 5 h	1, 5 h	
14/2	Building UMLs	4 h	0 h	Did not reach my goal, tomorrow new take.
14/2	Reading material, preparing UMLs etc	1 h	1,5 h	
17/2	Task 2 Write Use Cases	1 h	0,75h	It went faster than I thought now because I have read and thought about it quite a lot.

17/2	Task 3 Modeling Behavior del 1	8 h	3,75h	
	Sum:	20 h	18,25h	
18/2	Task 3 Implementing code	4 h	2 h	
	Lecture 6	1, 5 h	1 h	Jesper is good in 1,5 speed on youtube ;)
18/2	Reading the book: Software Engineering	3 h	1,5h	Chapter 6, chapter 7
18/2	Lecture Hobbe	1, 5 h	1 h	
19/2	Lecture 7	1, 5 h	1,25h	
19/2	Working with questions/study material	1,5 h	1,5h	
19/2	Reading the book: Software Engineering	3 h	1 h	Chapter 15
19/2	Task 3 Implementing code del 2	2 h	1 h	
20/2	Task 4 Modeling Structure, Class diagram	2,5 h	0 h	Did not feel very well, wasn't able to finish what I supposed to.
20/2	Lecture 8	1,5 h	0 h	Did not feel very well, wasn't able to finish what I supposed to.
20/2	Lecture Hobbe	1,5h	2 h	
20/2	Implementing code	2 h	2 h	
21/2	Reading the book: Software Engineering. Chapter 20, study before onlinetest	3h	3h	
21/2	Task 4 Modeling Structure, Class diagram	2 h	2 h	
21/2	Write on report, complete the second iteration	2h	2 h	
21/2	Make online-exam 1	0,5 h	0,5 h	Did not pass the test, it was several questions with alternatives where I missed one thing or boxed one to much ☹

	Sum:	20 h	18,25h	
	Sum Iteration 2 (w.4-6)	**40 h**	**36,5h**	

Reflections

I think I am quite good at planning and using the time I need to do what I should. I study hard and not slacking, and I always do my best. In this course it is often information about what to do and what to learn is a bit all over the place, so it takes a lot of time to structure material and feel comfortable about it.

In my planning I did put colours to each part of the tasks to easier discuss and see what I put time in. In my planning I did some assumptions, and I can now afterwards see that I overall. I can see that I used reading in book a lot less than I first planned and thought, so I can learn to next iteration to plan less time in that moment. I did a miscalculation about how many lectures I should listen to, but I also noticed that it was a good thing to speed lectures at youtube a bit, because I still had the opportunity to understand and get the substance out of it. It was a way to save some time in that. Working on code and studying and preparing models was probably quite good in my assumptions about time to put in it. It is improving doing these estimates and reflections about it. It also helps me to focus more on one task at a time, and not mix and match. Multitasking is not very effective, it is more effective to make a plan, and then follow it as good as possible.

7.3 Iteration 3

When:	**What to do:**	**Time it will take:**	**Time it took:**	**Reflection:**
1/3	Read Instructions Plan my work	1 h	0,75 h	
1/3	Reading the book: Software Engineering. Chapter 8	1,5 h	1 h 20 min	
1/3	Q&A Daniel	1,5 h	1,5 h	
2/3	Se Jesper lecture 9	1,25 h	1,5 h	
3/3	Se Jesper lecture 10 and 10.5	1,75 h	1,5 h	Did not finish lecture 10.5
4/3	Lecture 10.5	0.5 h	0,5 h	

4/3	Greeter application	3 h	0,5 h	Seen the tutorial on youtube
4/3	Youtube, searching info about mocha	1 h	1 h	
5/3	Task 1 - Test Plan	2 h	0,5 h	
5/3	Reading the book: Software Engineering. Chapter 8	2 h	1,5 h	
6/3	Task 2 - Manual test cases, start up	2, 5	1,5 h	Finished
6/3	Q&A Daniel	2 h	1,25 h	
6/3	Task 3 - Unit test, start up	2 h	0,75 h	
7/3	Task 3 - Unit test, finish	2,5h	1 h	
7/3	Working with code structure	1,5h	1,5 h	
7/3	Working with ass 3 report	1,5 h	1,5 h	
8/3	Study + onlineexam 3	6 h	6	
	Sum Iteration 3 (w.9-10)	**40 h**	**23 h**	

Reflections

I think I am getting a little bit better in estimating. But it was difficult after last iteration that was very much to do, and in this it was a smaller scope to complete. So, I planned after a 50 % course and estimated hours in every bit. But I didn't need all time this time.

7.4 Iteration 4

When:	**What to do:**	**Time it will take:**	**Time it took:**	**Reflection:**
11/3	Read Instructions Plan my work	1 h	1 h	

11/3	Read response from iteration 1 and work with report	2 h	2 h	
11/3	Make new class diagram	1 h	1,5 h	A little bit tricky to solve it, take a bit longer than I estimated for.
14/3	Make use cases + update model	1 h	1, 5 h	A little bit tricky to solve it, take a bit longer than I estimated for.
14/3	Make state chart	1 h	0,5 h	
14/3	Implement code, start up	1 h	1,5 h	- created a DB-model - Mongo DB-connection - new user saved at DB
14/3	Q & A Tobias, Questions	2 h	0,5 h	There was a short Q&A.
16/3	Implement code	4 h	1 h	- fixed a problem with DB - fixed a bit with interface
22/3	Working with code	0 h	2 h	/ 14,5 h hittills
26/3	Working with code	0 h	4 h	Fixed the code for menu, so it is possible to play again. Deleted the user-inlog-functionality and added a functionality to choose level instead.
27/3	Get Feedback from Iteration2 and must make new state machine and class-diagram.	3 h	3,5 h	1,5 h new state machine 1 h new class diagram 1 h update use cases
28/3	Make tests	3 h	2,5 h	Written updated manual tests. Written reflections and corrugated code.
28/3	Finish report	1 h	1 h	
	Sum iteration 4 (w.11-12)	**30 h**	25,5 h	

Reflection about time log

It has been easier afterwards to make more specific plans and estimate time to use. I think is a very good practice to do and I am going to use this more in future. Even if it is not forced to do, I take many advantages about doing it.

8. Reflection

It is time to hand in the project and to sum the working process.

8.1 Vision and Scope

After iteration 1 I reworked a bit about vision and scope. I got an idea about using hangman as a part of more mini games where school pupils could use to learn English words. Summary I feel that the game is not finished yet to use in that way. But it is possible to develop the game in more ways to use it in the way I thought about in my vision. I think I could have used more time in the beginning to build a better code, instead of just make it work. Because of my limited experience in coding, I should have put in more time in the code. It is learning I been made afterwards. It got its consequences at the end, because I didn't understand my code and why I done as I done. The result of it was that I had to change plan and limit myself about my vision. So, I had to change my extra features to something easier to deal with than username/login and a point-system that I thought of earlier in process.

8.2 Risks

In the risk-chapter I have made some estimations and reflections before the project started what possible could be a problem or a risk. I think that my reflections in that part, was quite good. Because it was both my own knowledge/limited experience and the time estimations, that was a problem. Even if I have learned a lot to plan, and now have made my first developing project in a structured and kind of professional way with iterations and all around. There has been a hard work to finish each iteration at time and deliver good quality. The risk in my own experience became a problem when I closer to the end get the idea to implement username/login/points-system and didn't really understood how mush work was need about that and that I was not able to finish it in time. So, I possible learnt something about this to.

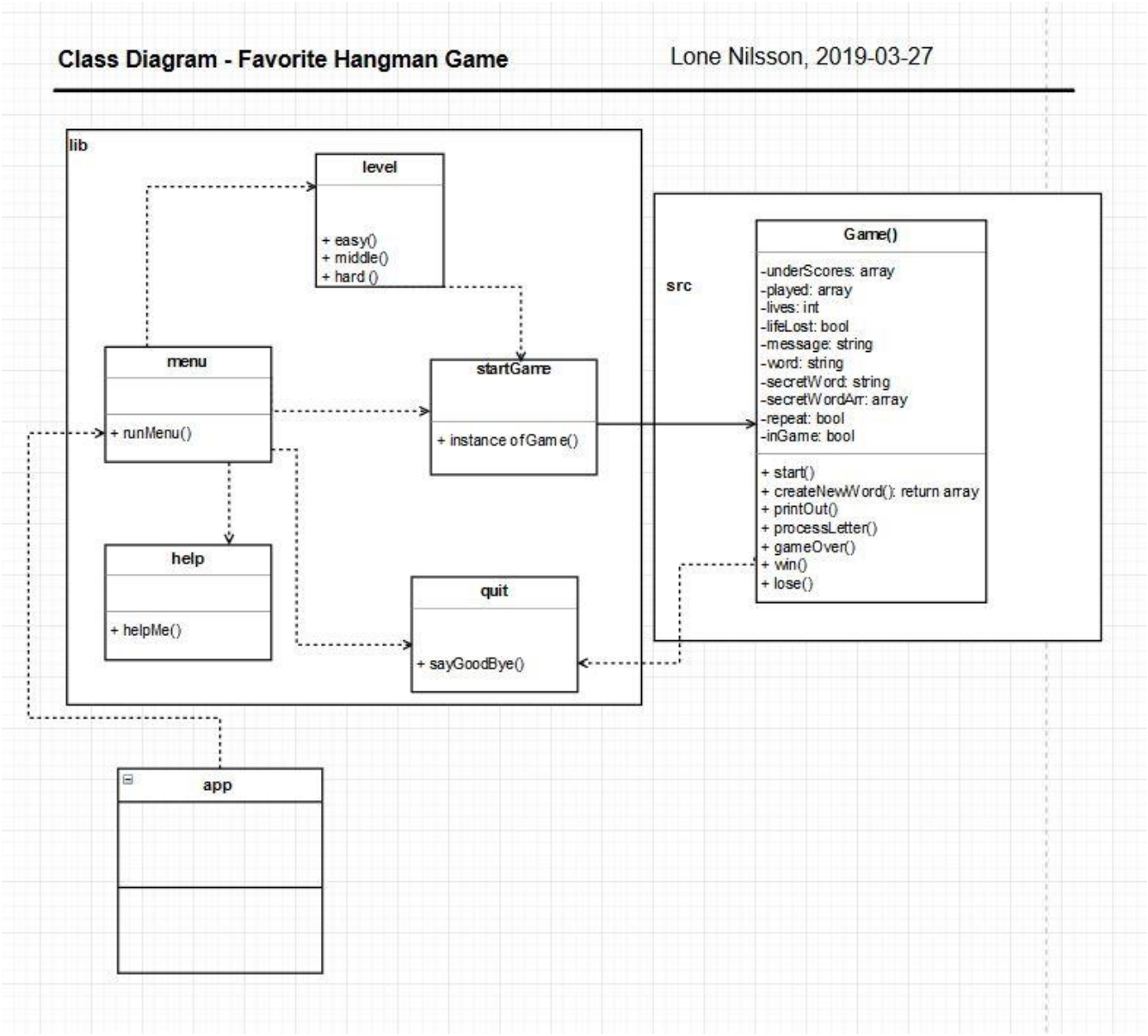
8.3 Time Log

It was more interesting to write time-log than I first though. Because it is very good and a learning process to make these estimations and to divide different parts into smaller pieces. At first, I planned I quite big parts, and at the end I became better to specify what exactly to do and it also got easier to estimate time then. It is also less stress when I have a plan to follow and when everything to do is written down, it is easier to work it through part after part instead of became fragmented and do something here and something there.

8.4 My work

Lastly, I want to sum my own insert in project. I think that I have learned a lot in this course. But I am not going to hymn with that it has been hard work and bitwise very difficult to understand what I am supposed to do and how I can handle it. I am very happy if this work is good enough for passing approved grade. I have not higher expectations than that, and that is because I have a life beside the university, and it is not always easy to handle it both beside each other. It is my wish to become a web developer and I need knowledge about this kind of stuff in my future work. I can also understand that it comes more chances to improve and make it better and better every time. There is always something new to learn.

Attachment 1 – Class Diagram over Hangman Game



Attachment 2 – Use Cases for Hangman Game

UC 1 Start Game

Precondition: none.

Postcondition: a choice of levels is shown.

Main scenario

1. Gamer wants to begin a session of the hangman game.
2. System presents the main menu with a title, the option to play, the option to choose level, option to help, or option to quit the game.
3. Gamer makes the choice to start the game.

Alternative scenarios

3.1 Gamer makes the choice to quit the game.

1. The system quits the game (UC4)

4.1 Invalid menu choice

1. The system presents an error message.
2. Go to (UC1.2)

UC 2 Play Game

Postcondition: Gamer wins and message is shown

Alt Postcondition: Gamer loses, and a message is shown.

Main scenarios

1. System picks a word and show one underscore for each letter, it also shows a message with how many guesses left.

2. Gamer pick a letter on keyboard to make a guess.

3. System put the letter in right place and change an underscore to the picked letter.

4. Go to 2

5. System shows a message when all correct letters is found. “You won! Go Back to menu”.

6. Go to UC1.2

Alternative scenarios:

2.1 The Gamer wants to quit game and uses the quit-button before tried any guesses. (UC 4)

1. The system terminates.

3.1 The system says it is wrong letter and there is one guess less.

1. The guessed letter shows in a box of used letters.

2. Go to 2.

3.1 The Gamer wants to quit game and uses the quit-button during the game. (UC 4)

1. The system terminates.

5.1 The system shows a message when all guesses is tried. “You are hanged. Go back to menu”.

1. Go to UC1.2

UC 3 Choose Level

Precondition: System is running, gamer want to play game.

Postconditions: Game starts.

Main scenario

1. Gamer wants play level.
2. System presents three levels: easy, middle, hard.
3. Gamer writes the command for the level to choose.
4. System presents a new word. (UC2.1)

Alternative scenarios:

3.1 Gamer picks a wrong command.

1. System gives gamer a new try (UC3.2)

UC 4 Quit Game

Precondition: The game is running.

Postcondition: The game is terminated.

Main scenario

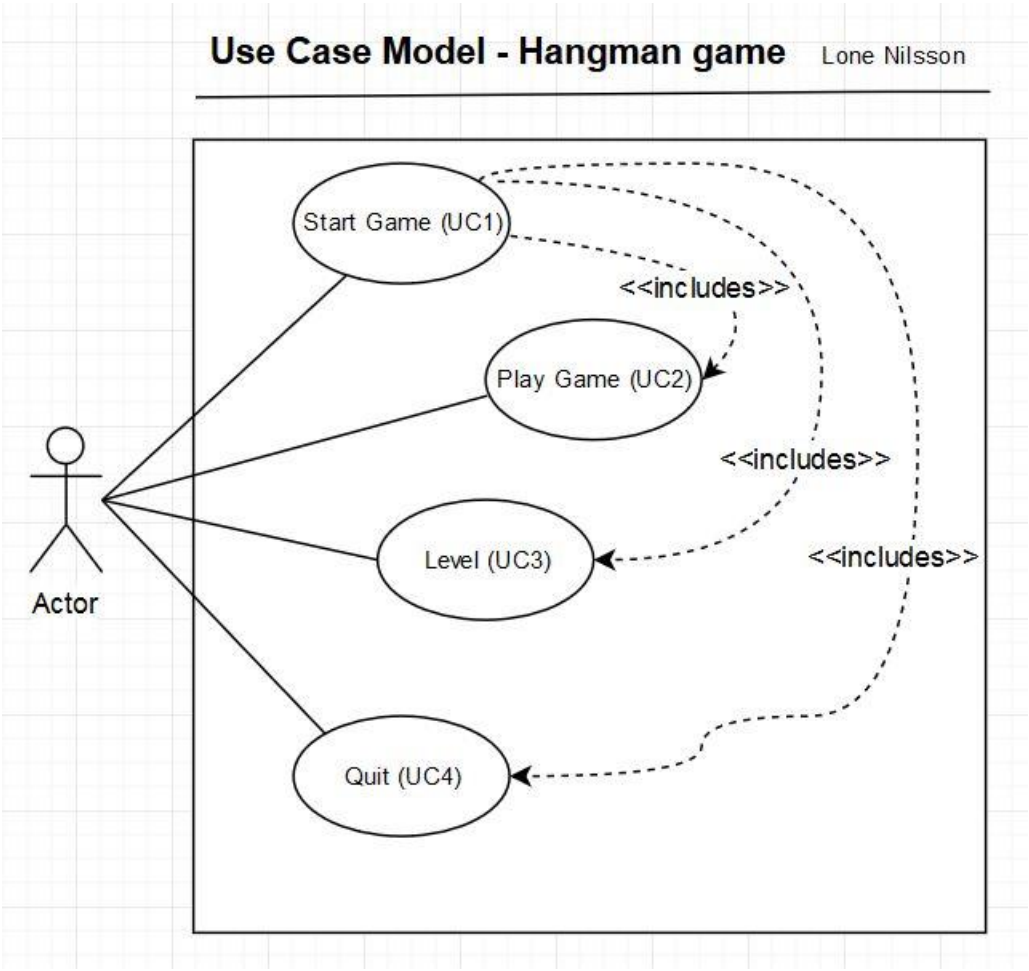
1. Starts when the user wants to quit the game.
2. System prompts for confirmation.
3. Gamer confirms.
4. System terminates and logging out.

Alternative scenarios

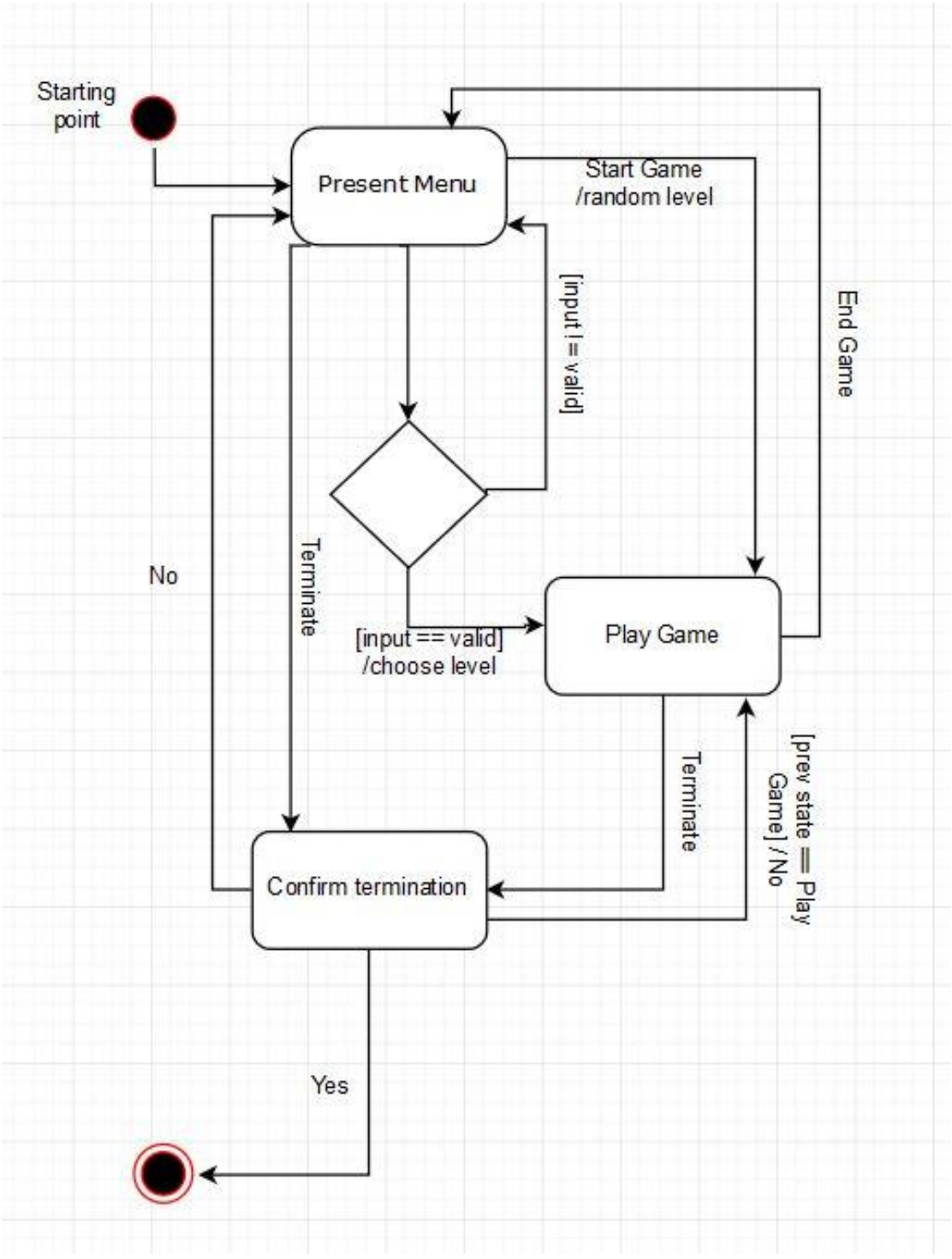
3.1. Gamer does not confirm

1. System returns to its previous state

Attachment 3 – Use Case Model



Attachment 4 – State Chart Model for Hangman Game



Attachment 5 - TEST REPORT FOR HANGMAN GAME

The application should be tested in manual and automatic tests to see that it does what it is supposed to do. First, I show the manual tests and a table over the results. I have written two test cases for each use case (1-3). I am testing the use cases manually to se that the implementation is correct to expected. Since iteration 3 I have tried to make the expected output more specific so the tester can be sure to se the correct expected output.

The automated tests I have handled so that they are actual to the code. In iteration 3 I had a thought about implementing functionality for adding nickname in a database and so on. In iteration 4 I hade some problems to finish it all and had to decide on back on that thing. So because of that I have get rid of the tests about adding nicknames. I also decided about not implementing more automate tests. That is because my code is not easy to test with automatic tests, and that is because my methods is to complex, they are not just easy methods with one thing in return. Maybe I must learn to write methods in another way? Or is JavaScript a language with more complex methods and need more of other test methods? I have a lot to learn more about in these things. And my time in this project is running out, so I can't do everything from scratch in a new take. I also had a comment on iteration 3 that my tests is not orthogonal, and I agree on that, but I don't know what to test and how in my complex method-code. So, because of that, I leave it as I have write before.

Manually tests

TC1.1 Start Game successful

Use Case: UC1 Start Game

Scenario: Start game successful

The main scenario of UC1 is tested where a user starts a new game.

Precondition: Run node app.js in terminal

Test steps

• System shows a menu with four alternatives

• Enter P for play

Expected

• System should show the text: Guess a letter:, five underscores and the text lives: 6.

Result:

As expected.

TC1.2 Start Game unsuccessful

Use Case: UC1 Start Game

Scenario: Start game unsuccessful

The main scenario of UC1 is tested where a user starts a new game.

Precondition: Run node app.js in terminal

Test steps

- System shows a menu with four alternatives
- Enter O for mistake

Expected

- System should show the message: Try again! and the menu again

Result:

As expected.

TC2.1 Play Game successful

Use Case: UC2 Play Game

Scenario: Play game successful

The main scenario of UC2 is tested where a user plays a game.

Precondition: Choose test-version of word-bank, so you can be sure about what word it is. Run node app.js in terminal, pick P for play.

Test steps

- System shows five underscores for the word PUPPY, and also 6 tries left
- Enter P
- Enter U
- Enter P
- Enter P
- Enter Y

Expected

- System should show a message about winning game and show the start menu again

Result:

As expected.

TC2.2 Play Game unsuccessful

Use Case: UC2 Play Game

Scenario: Play game unsuccessful

The main scenario of UC2 is tested where a user plays a game.

Precondition: Choose test-version of word-bank, so you can be sure about what word it is. Run node app.js in terminal, pick P for play.

Test steps

- System shows five underscores for the word PUPPY, and lives: 6.
- Enter A
- System shows five underscores, and lives: 5.
- Enter B
- System shows five underscores, and lives: 4.
- Enter C
- System shows five underscores, and lives: 3.
- Enter D
- System shows five underscores, and lives: 2.
- Enter E
- System shows five underscores, and lives: 1.
- Enter F

Expected

- System should show the message: Sorry you are out of lives! Game over and shows the start menu again.

Result:

As expected.

TC3.1 Choose easy level successful

Use Case: UC3 Choose level

Scenario: Choose level successful

The main scenario of UC3 is tested where a user chooses a level.

Precondition: Run node app.js in terminal.

Test steps

- System shows menu with Play(P) Level (L) Help(H) and Quit(q)
- Enter L for level.
- System shows the text: “What level do you want to play? Choose between easy(1), middle(2) or hard(3):” and a writeline.
- Enter 1 for easy

Expected

- System should empty terminal and show the message “Guess a word:”, underscores for a easy word and “6 lives”.

Result

As expected.

TC3.2 Choose easy level unsuccessful

Use Case: UC3 Choose level

Scenario: Choose level unsuccessful

The scenario of UC3 is tested where a user choose a level with wrong command.

Precondition: Run node app.js in terminal.

Test steps

- System shows menu with Play(P) Level (L) Help(H) and Quit(q)
- Enter L for level.
- System shows the text: “What level do you want to play? Choose between easy(1), middle(2) or hard(3):” and a writeline.
- Enter ‘e’ for easy

Expected

- System should show a message “Try again!” and the menu again.

Result

As expected.

Test Report Table

Test	UC1	UC2	UC3
TC1.1	1/OK	0	0
TC1.2	1/OK	0	0
TC2.1	0	1/OK	0
TC2.2	0	1/OK	0
TC3.1	0	0	1/OK
TC3.2	0	0	1/OK
COVERAGE & SUCCESS	2/OK	2/OK	2/OK