HANGMAN

CAMILLA HEIDING

Linnéuniversitetet

2019-03-22

Contents

1. Revision History

Date	Version	Description	Author
7/2-2019	1	Vision, project plan and risk analysis added. Iteration 1 completed.	*Camilla Heiding*
22/2-2019	2	Iteration 2 and time log was updated.	*Camilla Heiding*
8/3-2019	3	Iteration 3 and time log was updated.	*Camilla Heiding*
22/3	4	All parts except risk analysis was updated, iteration 4 was added.	*Camilla Heiding*

2. General Information

Project summary	
Project Name	Project ID
Hangman	1DV600_ch223cd
Project Manager	Main Clients
Camilla Heiding	English speaking players wanting a simple word guessing game for the console.
Key Stakeholders	
• Customer • Project manager • Development team • End users	
Executive Summary	
This project is going to develop a text-based word guessing game called Hangman that can be played in the console. The player gets to guess one letter at the time and need to guess all the letters before he run out of guesses otherwise he loses. The game is developed because there are players looking for a game like this which means there is a market for it. The goal is not to earn a lot of money but to provide a fun playable game for hangman enthusiasts.	

3. Vision

In this project a game of hangman is going to be developed. The game should be able to be played in the console window and will therefore be a text-based version where the hanged man will be constructed by available characters on the keyboard.

There should be a regular version where, as in all hangman games, a word will be picked from a predefined list of nouns and the player get to guess letters. In this game though if the player guesses the correct word he gets a point and get to guess on another word and earn a high-score for each correct word. Apart from that there should also be the alternative to play a multiplayer version where one player enters a word and the other player get to guess on the entered word. This will make hangman players looking for a game they can play with their friends interested in this game.

Another addition in this game is that the player should be able to leave the game he is currently playing and then return to it as long as he did not initialize a new single player game. For example, if he is currently playing a really tricky game by himself but then his friend comes along and want to play a multiplayer game with him and do not want to wait for him to finish his current game. In this situation he should be able to return to the menu and initiate a multiplayer version with his friend and then continue to his own game when they are done.

In the beginning there should be a long list of predefined nouns in English, so the player does not have to enter words by himself, this to increase the game friendliness for single players. But if the player finds a word that he really does not like while playing he should also be able to remove it from the list of nouns.

These additional features could really increase the interest in this hangman game in particular. The ability for the player to adapt the game after his own needs should make more people open to play this particular game.

The game should be developed using a plan-driven project plan and the documentation of the project is very significant.

3.1 Reflection

It was a bit tricky to determine how much details should be included in the vision. I know that there should not be anything about the code in the vision, but it was hard to know how many features should be presented and so on. Also, the vision tended to feel a bit short since the hangman game is not that big of a project. The text in the vision feels kind of similar to the one under the headline “scope” in the project plan.

Iteration 4: Changed the vision to be more inspiring and less describing. It now focuses more on the additions to an ordinary hangman game and why these additions could be useful. Removed some text describing a basic hangman.

4. Project Plan

4.1 Introduction

This project is about creating a hangman game that can be played in the console. The project will be plan-driven beginning with planning, then modeling and finally testing. Features could be added at a later stage, but it is important that the different deadlines are kept.

4.2 Justification

The reason for the project is mainly an educational purpose. To learn more about project managing, learn how to construct a plan that should be followed and divide a bigger task into smaller tasks that can be dealt with step by step. To get used to planning and documentation instead of just diving directly in to coding. It will also be an opportunity to understand how modelling and testing works. During development software and new techniques will be used which also will deepen knowledge in the field.

It should also result in a game friendly to players who want to both be able to play a multiplayer version with their friends but also a single player version where they get random words from a predefined list of nouns.

4.3 Stakeholders

The most important stakeholder in the project is the customer who already defined the main requirements. The project manager however gets to decide how ambiguous the game should be, how many additional features should be implemented. The knowledge and experience of the project team members also determines the limits of the project.

Customer – In this case the customer has predefined the main task, namely to create a game of hangman and have stated certain deadlines for subtasks and then given the project manager the responsibility to decide on design and implementation.

Project manager – The manager will have overview of the project and see that dead-lines are kept, assign different task to staff and see that there is always staff available. The manager will define the requirements for the project after communicating with other stakeholders. The project manager is able to see if a certain requirement is causing delay and come up with a solution.

Programmer – The programmer wants structured code for better maintainability. The knowledge and experience of the programmer will also affect which features is possible and how much time is needed to implement these features.

Tester – The tester also wants structured code preferably with distinct hierarchy to make automated testing of separate classes possible.

Designer – The designer wants clear state charts and diagrams and also a good-looking, user-friendly game.

End user – The end user will be able to come with some input about which features would be fun or useful in the game. Towards the end of the development the end user could tell about

the experience of playing the game and minor or major sources of irritation that should be handled.

4.4 Resources

Resources to the project is a project manager, programmer, tester, designer. None of these roles can work at the same time and they are available about 20 hours per week.

There are teachers from the university that the team can ask for help and literature about both the planning processes as well as the programming. The literature that will mainly be used is *Software Engineering 10th ed. by Ian Sommerville.*

Considering time there is about two weeks for making a project plan, two weeks to design and implement the game and two weeks to test the program. Then there is about another two weeks to finish the final details of the program, so after eight weeks in total the game should be finished.

4.5 Hard- and Software Requirements

Required hardware is a **working computer** that can handle the software. Software that is going to be used for coding is **java 11**. Java is going to be coded in the development environment **Eclipse, Oxygen edition**. Also, the documents of the project should be stored in a repository in a **GitHub-account** shared by different stakeholders. To draw state charts and diagrams **draw.io** will be used. The project plan and other documents should be stored as an pdf produced by **Word**.

4.6 Overall Project Schedule

Project plan – 8/2-2019
An overall plan of the project containing a vision, project plan, risk analysis and some skeleton code. Detailed plan in iteration 1.

Modelling – 21/2-2019
Use case diagram, fully dressed use cases, state machine diagram, implementation of a somewhat playable version of the game and a class diagram. Most features should be implemented at this stage. Detailed plan in iteration 2.

Testing – 8/3-2019
A test plan for both manual and automated tests and a test report after performing these tests. The game should be carefully tested, and problems should be fixed. Detailed plan in iteration 3.

Complete project – 22/3-2019
The deadline of the entire project. At this point a finished game of hangman should be able to be delivered to the customer. Detailed plan in iteration 4.

4.7 Scope, Constraints and Assumptions

4.7.1 Scope

This program shall result in a working hangman game. It must be able to randomize a word from a list of nouns and represent it as underscores. It must also be able to present a part of a

stickman for each wrong guess and tell the player that the game is over when the entire man is drawn or tell the player that he has won when all letters in the game is correctly guessed. While guessing, the letters should also appear in the correct positions of the word and if not part of the word they should be presented as guessed letters. Player should not be able to guess the same letter twice.

If time permits the game should also be able to present a menu with different versions of the game such as a multiplayer and single player version.

Single player game should be extended so that if the player guesses the correct word he should earn a point and get to guess another word. When he fails to guess a word his high-score should be presented.

Multiplayer version means that one player enters a word and another player guess the letters of that word. If the second player manage to guess the word that means he wins the game, otherwise the player who entered the word wins. The player should always be able to return to the menu or terminate the program during games.

There should be a long, predefined list of nouns, but the player should be able to remove words from this list through the game menu.

Note:

- *Requirement of a high-score list was removed due to lack of time and that other things was higher prioritized.*
- *Requirement to register a username was also removed since it was mainly meant to be used in the high score list.*

4.7.2 Constraints

Constraints of the project would mainly be the time. There are four deadlines that can not be changed and if features take to much time to design and implement then they should be excluded from the game.

The experience and knowledge of the development team is also a constraint. Programmer only knows of the java language and do not have a lot of experience. Therefore, only a certain number of features will be possible to implement during this project. The programmer will have to learn along the development process which will consume time.

4.7.3 Assumptions

The game will assume that the player understands the basics of hangman or at least is able to figure them out during the game. There will not be a descriptive text with rules before them game start. Although there should be a line saying “Enter a letter” or similar, to make it clear what the user is supposed to do in that specific step.

If different versions of the game are implemented, then there should be a short description of the differences between the versions.

4.8 Reflection

The project plan contains a lot of details about the project like deadlines and which resources are available and so on. It also tells about features that must be included and features that could be included if time permits. Although it does not feel like something you can follow to get a finished product. Requirements is going to be established in iteration 2 but it feels like they should be included to be allowed to call it a project plan. The iterations feel more like a project plan, but they are under a separate headline?

5. Iterations

5.1 Iteration 1

Deadline: 8/2-2019

To be well prepared when writing the vision, project plan and risk analysis literature on the subject should be read plus lectures offered by the teacher should be read. For iteration one chapter 2, 3, 22 and 23 in *Software Engineering 10th ed. by Ian Sommerville* should be read which is a little over 100 pages which should take about 4 hours. Apart from that three given lectures on the subject should be attended which in total is about 6 hours.

Write vision: Project manager in association with other stakeholders should write a vision for the project. This should take about 30 minutes.

Write project plan: The project manager should then make a project plan. In this stage there should not be details about coding or modeling but more of a higher level of planning. For some parts of the project plan the manager must communicate with other resources and stakeholders.

- Write a justification why the game should be made – *After writing the vision the manager should have a view of why the game should be developed. This should take about 15 minutes.*

- List stakeholders – *Make a list of stakeholders in the project and specify which part of the project the different stakeholders should influence most. Also try to grade the different stakeholders influence. This should take about 30 minutes.*

- List resources and hard- and software requirements – *Which resources are available; how much time is there for the project? Which hard- and software requirements are there to develop the game? For this the manager will need help from the development team. This should take about 30 minutes.*

- List deadlines - *These are already decided by the customer, list them and specify shortly what is included in each deadline. This should take about 15 minutes.*

- Scope, Constraints and Assumptions – *List what must be included in the project and what should be included if possible. Also specify constraints that may affect the development of the project. To do this the manager will need help from the development team as well as the customer. This should take about 30 minutes.*

Write risk analysis: The manager then must make a risk analysis for the project.

- What are the risks?
- How to prevent risks?
- How to minimize the impact of risks?
- How to deal with risks?

To determine these different things the manager, again, will need help from the development team. This should take about one to two hours.

Prepare GitHub account – An account must be created, and different stakeholders must be invited to the repository. Since implementation of the program have not yet started the repository will still be fairly empty but some skeleton code should be uploaded along whit the project plan. This should take about an hour.

After completing all steps in iteration one, about two hours should be used to proofread the vision, project plan and risk analysis to make sure that everything is correct.

5.2 Iteration 2

Deadline: 22/2-2019

Read chapters: 4, 5, 20, 6, 7 and 15 in *Software Engineering 10th ed. by Ian Sommerville.* This is 200 pages and should take in total 7-8 hours. This should not be read in a streak but distributed through iteration 2. Lectures on this subject should also be attended. There are 7 lectures which is estimated 2 hours each which result in 14 hours.

Basic:

- Write a fully dressed use case for a *basic* version of requirement Play Game. This should take 4 hours.
- Create a state machine for playing the game. This should take 2 hours.
- Implement the models. This should take 3 hours.
- Create a class diagram. This should take an hour.

Additional:

- Create extended use case diagram, this should take 30min.
- Write fully dressed use case for single player game, this should take an hour.
- Write fully dressed use case for multiplayer version. This should take an hour.
- Create state machine for single player version.
- Create state machine for multiplayer version. This should take an hour.
- Implement as much as possible of the extended models. This should take 4 hours.
- Create a class diagram. This should take an hour.

5.3 Iteration 3

Deadline: 8/3-2019

Read chapter 8 in *Software Engineering 10th ed. by Ian Sommerville.* This should take about an hour. Lectures on this subject should also be attended. There are five lectures which is estimated 2 hours each which result in 10 hours.

- Write detailed planning for this iteration which should take about 30min.
- Design manual testcases for the use cases which was modeled in iteration two. This should take about 5 hours.
- Create automated unit-tests which also should take about 5 hours.
- Perform the manual testcases that was created and write down any problems. This should take about 30minutes.
- Inspect the code that is tested and look for improvements. This should take an hour.
- Write a test report after performing the tests that was designed in this iteration including any problems that came up. This should take about 30 minutes.

5.4 Iteration 4

Deadline: 22/3-2019

- Write detailed planning for iteration 4 which should take an hour.

Handle comments about iteration 1-3

- Clarify main client and executive summary which should take 20 minutes.
- Update vision which should take 30min.
- Update project plan which should take 30min.
- Fix details in state machines which should take 30min.
- Increase readability by adding more line-breaks and bold text to guide the tester which should take 20min.

Implement removing word from predefined list of word

- Write fully dressed use case for removing a word to the predefined list which should take 30min.
- Create state machine for removing a word to the predefined list which should take 30min.

- Implement removing a word from the predefined list which should take two hours.
- Update the class diagram which should take 15min.
- Design testcases for removing a word from the predefined list which should take an hour.
- Perform the manual testcases that was designed which should take 20min.
- Update the test report if any problems with the new implementation arise which should take 15min.

Implement possibility to play games until failure and earning a high-score

- Change the use case single player game to continue guessing words until player fails to guess a word and add a high score which is being enumerated for each win. This should take an hour.
- Update state chart for single player game which should take 30min.
- Implement the updated use case and state chart for single player game which should take 4 hours.
- Update test cases for use case single player game which should take an hour.
- Re-iterate testcases which should take 20min
- Update class diagram which should take 15min.

Finish project

- Add manual testcase for user case 6 Quit game which should take 30min.
- Clean up code and check methods documentation which should take 45min.
- Check project plan, diagrams and state charts which should take an hour.
- Check manual and automated tests which should take 30min.
- Perform all tests one final time which should take 30min.
- Write reflection with motivations for the diagrams, state machines tests and so on that was used during iteration 4 which should take 20min.
- Make a playable version of the game that can be played in terminal which should take an hour.

5.4.1 Reflection iteration 4

I have chosen to implement two new features in this iteration since they seemed reasonably hard to implement. I decided to exclude for example username and high-score list since this seemed to complicate to implement and I prioritized planning and testing the features I do have instead of number of features.

I have chosen to use similar diagrams as I did for the features implemented in iteration 2 to have a unity in the project. The new features are similar in size and complexity to the previous features and needed about as much planning. Therefor I made use case diagrams and state machines. More diagrams would not result in a better result but mainly take time from my perspective. I also updated the class diagram.

I managed to implement a few more unit tests for the new features, mainly the remove word feature to see that the word really is removed. Otherwise I have mainly made manual test cases. I realize that this is not optional since manual test cases take a lot of time and is risky since tester might get sloppy, but since a lot of the game involve user input and prints to the screen I did not have the knowledge on how to automate this but still wanted coverage.

There is some problem with my implementation since it is hard to automate and also have some double directed dependencies. I am aware of this but was not able to correct it during this course due to lack of time and knowledge.

6. Risk analysis

6.1 List of risks

In every project there is a number of risks, here is a list of the most relevant risks in this project.

Numbering	Risk	Probability	Effects
1	The time required to develop the software is underestimated	High	Serious
2	Staff is not available because of other task assignments	High	Serious
3	Hardware crashes	Moderate	Catastrophic
4	The size of the software is underestimated	Moderate	Serious
5	Key staff are ill at critical times in the project	Moderate	Serious
6	Required training for staff is not available	Low	Tolerable

Chart of possible risks and the probability of them happening and seriousness of the effects

1. This risk is relevant since it is one of the first software development the project manager is managing, and it is hard to estimate how long each part of the project will take. There is very strict dead-lines so extra time can not be added during the development.

2. Since almost every role in the project team is assigned to the same person there will be a serious effect if this person is not available.

3. The computer being used is a few years old and the risk of malfunction is not irrelevant. If the computer where the whole project is being stored and developed this would be very bad. It could also take some time to get a new computer to work on if the old one does not work which could be very bad, especially right before a dead-line.

4. The staff is not very experienced and therefore it is hard to estimate how complicated the game will be to develop. There is features that the developers do not yet know how to implement, and time will be spent studying and finding a solution for this.

5. Similar to risk number 2.

6. As mentioned in risk number 4 there is features that the developer do not know how to implement yet. Information should be available for this but could be time consuming.

6.2 Strategies

The most optimate solution would be to avoid the risks all together, here are a list of strategies to minimize the chance of the risks happening and also reduce the impact it would occur.

Risk	Strategy
The time required to develop the software is underestimated	• Including extra time in schedule when making planning. • Not saving tasks for the last minute.

Risk	Strategy
Staff is not available because of other task assignments	• Making schedule for when this project must be prioritized. • Doing as much as possible when there is time available. • Not save critical tasks for last minute.
Hardware crashes	• Store files in different places such as GitHub, computer and possible even on a flash drive. • Maintain the computer and use antivirus program.
The size of the software is underestimated	• Make sure that essential parts of the program are prioritized and implemented first. • Update the priority during implementation when realizing something take more time than estimated.
Key staff are ill at critical times in the project	• Try to make it possible to work from home. • Doing as much as possible when staff is available. • Not save critical tasks for last minute.
Required training for staff is not available	• Make time for replanning. • Get an overview of what training will be available before deciding on requirements.

Chart of possible risk and strategy to prevent them from happening and minimize the effect if they do.

If there was no way to avoid the risk, here is a list of strategies for what to do.

Risk	Strategy
The time required to develop the software is underestimated	• Replan the project, remove features with lowest priority.
Staff is not available because of other task assignments	• Try to catch up when staff is back. • If this happens right before a dead-line, see if missing features can be added before next deadline instead. • Prioritize and try to get staff to do most essential tasks.
Hardware crashes	• Try to get new hardware as soon as possible. • See if it is possible to borrow software until new arrives.
The size of the software is underestimated	• Reprioritize, do most essential parts.
Key staff are ill at critical times in the project	• See if possible that staff work from home. • Try to catch up when staff is back.
Required training for staff is not available	• Remove features that is impossible to implement. • Try to find other solutions to implement the feature.

Chart of possible risk and what to do if they occur.

6.3 Reflection

The hangman project is not that big, and it is obviously possible to do since it is an assignment, so it was a bit hard to come up with a large amount of risks. But it was still possible to come up with different strategies to prevent them and make the impact of them as little as possible. Many risks are things that feel kind of obvious and you usually just deal with them as they occur, so it was educational to write them down and analyze them.

7. Time log

Iteration 1

Task	Date	Estimated time	Actual time	Reflection
Reading literature	23-30/1-19	4hours	4-5 hours	
Attend lectures	23-30/1-19	6hours	6 hours	
Prepare GitHub account	24/1-19 7/2-19	1hour	2hours	Never used GitHub and needed to watch a lot of videos on how to use it.
Write a vision for the project	4-7/2-19	30min	60min	Time consumed finding out what should be included in vision
Write justification	4/2-19	15min	15min	
List and define stakeholders	4/2-19	30min	30min	
List resources and hard- and software requirements	4/2-19	30min	30min	
List deadlines	4/2-19	15min	15min	
Scope, Constraints and Assumptions	4-7/2-19	30min	30min	
Make risk analysis	4-7/2-19	1-2h	2h	
Begin planning of iteration 2 and 3	7/2-19	15min	15min	
Proofread the vision, project plan and risk analysis	7/2-19	2h	2h	

Iteration 2

Task	Date	Estimated time	Actual time	Reflection
Reading literature	5-18/2	7-8 hours	8 hours	
Attend lectures	6-20/2	14 hours	14 hours	
Write a fully dressed use case for a *basic* version of requirement UC 2 Play Game	13-20/2	4 hours	4 hours	Had to go back after starting the extended version so this actual time may not be exact.
Create a state chart for playing the game.	13-20/2	2 hours	6 hours	Took a lot of time to understand. Find and learn a program to draw state chart. Some time was wasted doing unnecessary work.
Implement the models.	16-19/2	3 hours	4 hours	Time went faster than expected
Create extended use case diagram	13-18/2	30min	45 min	Fairly ok estimation. Changed this diagram several times.

Write fully dressed use case for single player	18-20/2	1 hour	40min	
Write fully dressed use case for multiplayer version	18-20/2	1 hour	1 hour	
Create state machine for single player version	18-19/2	1hour	50min	
Create state machine for multiplayer version	18-19/2	1hour	1hour	
Implement the extended models	18-22/2	4 hours	5-6 hours	Not all extended model functions are not implemented. Some functions was harder to implement than expected.
Updating models and user cases	16-22/2	Unexpected	3 hours	A lot of time was consumed changing things since I changed my mind on how to do them.
Create a class diagram.	19-20/2	1 hour	1 hour	

It was very hard to measure time even though toggle was used. Especially when I was stressed and went back and forth between different tasks. The actual time may have been slightly underestimated but is as accurate as possible.

Iteration 3

Task	Estimated time	Actual time	Reflection
Reading literature	1 hour	1h 15min	
Attend lectures	10 hours	10 hours	
Write planning	30 min	20 min	
Write manual test cases	5 hours	6 hours 30 min	Some functions from the orginal UC was removed. But before this was done some time was consumed trying to solve such problems.
Create Unit Tests	5 hours	5 hours 30 min	The implementation was not well adjusted for testing, so time was consumed figuring out what was testable.
Running manual test cases	30 min	25 min	
Code inspection	1 hour	1hour 30 min	Parts of code had to be reconstructed to be testable
Test report	30 min	20 min	

Iteration 4

Task	Estimated time	Actual time	Reflection
Write detailed planning for iteration 4	1hour	40 min	
Handle comments about iteration 1-3			
Clarify main client and executive summary	20min	15 min	
Update vision.	30min	40 min	
Update project plan.	30min	1 hour	
Fix details in state machines	30min	30min	
Increase readability by adding more line-breaks and bold text to guide the tester.	20min	25 min	
Implement removing word from predefined list of word			
Write fully dressed use case for removing a word from the predefined list.	30min	35 min	
Create state machine for removing a word from the predefined list.	30min	40 min	
Implement removing a word from the predefined list.	2h	2 hours 15min	
Update the class diagram.	15min	20 min	
Design testcases for removing a word from the predefined list.	1h	2h 25 min	Waisted a lot of time on screenshots of test code for documentation which turned out not to be a demand for iteration 4.
Perform the manual testcases that was designed.	20min	25min	
Update the test report if any problems with the new implementation arise.	15min	0 min	No problems to report
Implement possibility to play games until failure and earning a high-score			
Change the use case single player game to continue guessing words until player fails to guess a word and add a high score	1h	30 min	Was not as hard to plan the change as it was to actually change it in implementation.

which is being enumerated for each win.			
Update state chart for single player game.	30min	35 min	
Implement the updated use case and state chart for single player game.	4hours	3h 45min	
Update test cases for use case single player game.	1hour	25 min	Manual testcases was not written from scratch only changed where needed. Automated testcases was har since most changes involved input.
Re-iterate testcases	20min	25 min	
Update class diagram.	15min	10 min	
Finish project			
Add manual testcase for user case 6 Quit game	30 min	20 min	
Clean code and methods-documentation.	45 min	35 min	
Check project plan, diagrams and state charts.	1 hour	45 min	
Check manual and automated tests.	30 min	30 min	
Perform all tests one final time.	30 min	30 min	
Write reflection for iteration 4.	20 min	15 min	
Make a playable version of the game that can be played in terminal.	1 hour	2 hours	Problems with finding text-file with nouns when exporting code to a jar-file.