LINNAEUS UNIVERSITY

Software Project

Hangman

Christoffer Lundström

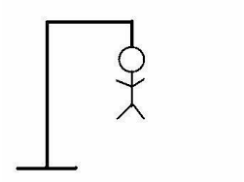
28/01-2019

Software Technology – IDV600

Teachers: Tobias Andersson Gidlund

Daniel Toll

Tobias Olsson



Contents

1. | Revision History

Date	Version	Description	Author
050219	0.1.1a	First iteration, planning and structure.	C.L
200219	0.1.2a	Design & implementation. Added UML, Use cases etc.	C.L
050319	0.1.3a	Unit testing	C.L
170319	0.1.4	Iteration 4 added to project plan. Updated timeline and deadlines.	C.L

2. | General Information

Project Summary	
Project Name	Project ID
Hangman	V 0.1.2a
Project Manager	Main Client
Christoffer Lundström	IDV600 @ LNU
Key Stakeholders	
The product owner will be the sole stakeholder of this product.	
Executive Summary	
Current requirements of the course IDV600 at Linnaeus University include a new open source Hangman game single-handedly developed and organized by specified project manager. The project will exclusively involve software development through an iterative software process and the goal is to design and implement a rudimentary console-based UI and basic game logic to meet the demand. Estimations suggest that the project will be completed and unit tested by March 2019 prior to its due date currently *t.b.d.	

3. | Vision

The vision of this project include creating a text-based game of Hangman. This will be a simple but stable game with basic functionality such as *play*, *highscore* and *exit*. It will be written in Java to ensure platform independence and to possibly reach a broader audience.

User Interface
Each respective category will be enumerated and be navigated through by an easily comprehensible user interaction. Each UI element will in turn lead to its designated target logic.

Play will initiate a round of Hangman by first prompting the user for a name and a difficulty. Preferably constrained to 20 characters or so for formatting. After a name has been entered a player object will be created and a Scene object will be loaded from the game directory.

Highscore will lead to a list of the top ranking players and the score will be determined by an algorithm.

Game Logic
After the Scene is set up a blank field will be generated of the same size as a randomly fetched word from a dictionary and a first round is ready to begin. A user will be asked to guess a character.

Should the character be wrong the next *Scene* of hangman will be rendered. If the man is eventually hanged the correct word and the amount of guesses will be printed to console.

Scenes & I/O
The idea is to have Scenes drawn in standard ANSI or UTF-8 format and saved to a text-file. They will represent the progression of the Hangman scenes of events, much like frames in a movie. They will be loaded to memory and printed onto the console window.

Reflection
To ensure that the management, dev-team are working towards the same end goal it is essential to reflect upon these ideas and share the same vision.

This not only creates a foundation for the software process but also facilitates efficiency and in extension longevity and quality of the end product. Having a clear and concise vision arguably, is a form of risk minimization strategy which will not only mitigate confusion, but also establish faith in the product and management.

4. | Project Plan

4.1 Introduction

This is the project plan for the console based game Hangman. The project will be written in Java SE 11 using IntelliJ 2.0 and GitHub for version control.

The project will start off with the creation of this project plan as a first iteration out of four, following an agile task based method.

4.2 Justification

This product will be produced by demand of the course Software Technology IDV600 of Linnaeus University.

4.3 Stakeholders

Stakeholders in this project will be management (Christoffer Lundström) and end users of IDV600 at Linnaeus University.

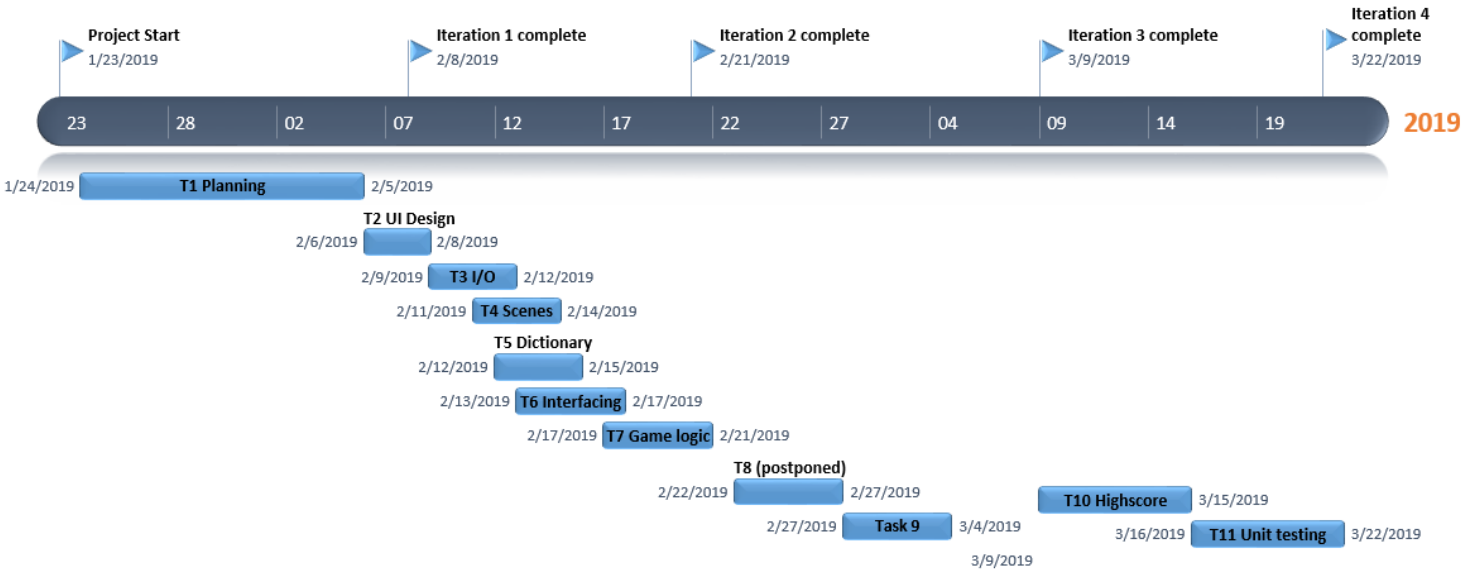
4.4 Resources

Resources consist of a sole project worker and an IDE. In this case IntelliJ 2.0 community edition with JDK SE 11 will be utilized.

4.5 Hard- and Software Requirements

Software requirements in this project is Java Virtual Machine version 11 or later and a computer compatible with JVM and IntelliJ 2.0.

4.6 Overall Project Schedule



Deadlines:

Iteration 1 - v0.1.1a – 8th of February

Complete project scheduling and basic UI implementation.

Iteration 2, v0.1.2a – 22th of February (revised from 21 th)

Complete I/O file operations.
Create scenes for Hangman.
Provide a dictionary.
Design interfaces between the software components.
Design and implementation of game logic.

Iteration 3, v0.1.3a - 8th of March (revised from 9th)

Unit tests complete.

Iteration 4, v0.1.4a – 22th of March.

Extra *Highscore* functionality added.
Ready for beta testing and release.

4.7 Scope, Constraints and Assumptions

The scope of this project will be within the limits of the strict deadlines. This will be the most constraining factor to the scale of the project. Another one is staff resource and budget.

Assuming the project only require a single threaded application with basic knowledge of Java the scope will perfectly within reach.

4.8 Reflection

During this scheduling phase I found it particularly difficult estimating the time each task will take. A lot of them are pure guesses based on own coding speed.

Even though these guesses might be off target they provide a foundation and structure which seem easy to comprehend. Dividing the project into tasks seems less overwhelming than starting from nowhere. The importance of these plans must be substantial in a larger scoped projects.

Creating the project schedule and reviewing it draws me to make a parallel with my recent construction work where the same type of schedules are used for dividing tasks to sub-contractors in large enterprises.

5. | Iterations

5.1 Iteration 1

Summary: The goal of the first iteration is to complete Task 1 and 2 and have the basic structure for the game designed and implemented.
Total time estimated: 25h.

Task 1:

The first task will be to do some research regarding the general concept of Hangman, finding influences and inspiration. Which will be followed by analysing the requirements and writing a project plan, breaking down each requirement to a task.

GitHub repository will be set up and provided for version control. Will also include creation a basic game directory structure where assets will be divided into respective category together with a simple markdown readme.
Estimated time: 20h.

Task 2:

Rudimentary console design. Will provide basic functionality and type-safety for the application. Implementation of console-based menu using numbers to walk through menus and sub-menus.
Estimated time: 5h.

5.2 Iteration 2

Summary: The goal of the second iteration is to have a bare minimum functional game.
Total time estimated: 25h

Task 3:
Design and implement the necessary I/O and read-functions which will load each frame from file, will later be used when implementing highscore (T8).
Estimated time: 8h.

Task 4:
Draw each *scene* of *Hangman* in separate read-only text-files. This will be done in notepad or notepad++.
Estimated time: 2h.

Task 5: Implementing and integration of a list of words which will be used in later tasks. Words will be saved in notepad and encoded in UTF-8 or ANSI using notepad++. All values should be separated by semi-colons in the format “comma-separated value” .csv.
Estimated time: 2h

Task 6: Designing and implementation of the interface between game scenes and menus using the read functions written in Task 4.
Estimated time: 5h

Task 7: Design and implement game logic. Will include designing the interaction between the user and the Scenes.
Estimated time: 8h

5. 3 Iteration 3

Summary: By the end of this iteration the application should be functional and bug-tested. Should time allow this is the iteration for additional implementations.
Total time estimated: 10h (revised from 14)

~~**Task 8**~~: Implementation of a basic *Highscore* list. It will provide a top 10 list of players sorted by score. Score will in turn be defined by the number of victories by a player.
~~**Estimated time: 4h**~~ **(postponed)**

Task 9: Unit testing and fine tuning.
Estimated time: 10h

5.4 Iteration 4

Summary: The goal is for all tasks to be completed and the game should be unit tested and ready for release by the end of this iteration. Due to the rather small feature addition and somewhat tight schedule I have decided to regard Planning, design and implementation and finally testing as one final iteration.
Total time estimated: 10h

Task 10: Previous task 8 was first intended to be implemented in iteration three but was discontinued from iteration three and moved to this iteration due to time management and new project requirements of the course IDV600.

Description
Design and implementation of a basic *Highscore* list. It will provide a top 10 list of players sorted by score. Score will in turn be defined by the number of victories by a player.
Estimated time: 8h

Task 11: Final unit testing of project together with new Highscore component. Will include regression-testing with unit tests and manual tests on newly implemented Highscore list.
Estimated time: 2h

Final Reflection

For this final iteration I decided to use an Activity diagram as a foundation for the new Highscore feature. I found that it would be most helpful because it would describe all steps necessary for my component without specifying too many details regarding its implementation.

Because this component is more of a technical detail rather than a Use case or a description of different states it seemed like the most logical diagram for my purpose. I also decided to update the current class diagram to include the Highscore class and the updated player class to have a complete overview over the classes.

During this project I think I have had quite accurate estimations of time. There are a couple of hours miscalculated, but there has also been a few times where I've had to refactor quite a bit of code. During this last iteration my intention was to implement the Highscore list as a Binary heap to keep it sorted at all times using parts from another project but it proved to be difficult to integrate, and thus had to be scrapped.

There has also been a few changes to the requirements throughout the project which have meant moving around and reprioritizing a few Tasks. As stated in the risk analysis the effects of these changes were expected to be moderate, which they have been.

6. | Risk Analysis

6.1 List of risks

Risks	Affect	Probability	Effects
Sudden hardware failure	Project	Low	Low
Staff illness	Project	Low	Catastrophic
Changes to software requirements	Project	High	Moderate
Wrong time estimations	Project	High	Moderate
Training of staff need for new requirements	Project	Moderate	Moderate
Entire software remodelling	Project	Low	Catastrophic
Misestimating scope and size of project	Project	Medium	Moderate
Organizational restructure	Project	Negligible	Low
Relocation of Linnaeus university.	Client	In Effect	Low

6.2 Strategies

As a first step in risk avoidance all work is saved on cloud based Google One (formerly known as Drive). Both the Java repositories and project plans. Google One is in turn synced to a work laptop and a stationary PC. This should render the risk of losing all work through a hardware crash extremely unlikely.

Sudden staff illness is always a huge risk for tiny projects, especially for a one man project like this. Should this happen, the project may or may not fail completely, all depending on the situation. In a real work scenario the damage caused by said event might be minimized by changing shifts or personnel.

Other risks as such as changes to the software requirements or wrong time-estimations are extremely likely due to the fact that this is a learning exercise and the effects of these changes could delay the project substantially.

To minimize the effects of changing software requirements the project has been chosen to be simple, agile and loosely coupled. Misestimating time is almost inevitable given the lack of software planning experience and changing course material.

Currently the client Linnaeus University is relocating. This might have an impact on the client and in extension the project albeit minimal, since most of the technology faculty is already up and running as regular.

Contingency & Monitoring

Following the project and re-evaluating the risks and time required of each iteration might decrease the likelihood of project failure. This will be done each iteration.

Should there be a catastrophic failure due to staff illness or the need to remodel the entire project there are extended second chance deadlines to aim for.

Reflection

The importance of proper risk analysis should not be underestimated. Depending on the business, project and product the funds, integrity or even safety of customers and clients might be compromised by failure in this software process.

Furthermore the risk minimization strategies will reduce the already calculated risks and hopefully provide security and stability for the project and its affiliates. Should the risk factor still occur there ought to be thorough contingency plans ready to deploy.

Given the simulative nature and scope of this project, this might be the only exception to the rule.

7. | Time log

Below is a time log spanning the time estimated and taken to complete each task.

Date	Task	Estimated	Actual	Diff +/-
280119	T1: Fill in basics of project template	1h	1h	0
290119	T1: Game idea research	1h	1h	0
300119	T1: Setting up GitHub repository	1h	0.5h	-0.5
310119	T1: Sketch game logic design	2h	1h	-1
020219	Rewrite project template	2h	2h	0
040219	T1: Project summary & vision	2h	4h	+2
050219	T1: Project plan & iterations	4h	6h	+2
060219	T1: Risk analysis	3h	5h	+2
070219	T1: Risk analysis cont.	2h	1h	-1
070219	T2: Implement basic UI	2h	1h	-1
>>>>>>>>>>	**Iteration 1**	20h	22	+2
120219	T4: Hangman drawing	2h	1h	-1
130219	T5: Word list implemented	2h	1h	-1
140219	T3: Design & implement	8h	2h	-1
150219	T3 cont.	-	2h	x
180219	T3 cont.	-	3h	x
190219	Task 7: Game logic	8	4h	+4
200219	T7 cont.	-	8h	x
210219	Task 6: Interfacing	5h	6h	+1
>>>>>>>>>>	**Iteration 2**	25h	27h	+2
-	~~T8: Highscore list~~ (postponed)	4h	-	
040319	T8: Test plan	3h	4h	+1
040319	T8: Manual tests	2h	1h	-1
050319	T8: Automated tests write & run	3h	2h	-1
060319	T8: Code revision	1h	1h	0
070319	T8: Automated test rerun	1h	1h	0
>>>>>>>>>>	**Iteration 3**	10h	9h	-1
160319	T10: Planning & design.	-	3h	-
170319	T10: Implementation	-	4h	-
180319	T10: Implementation	8h	3h	+2
190319	T11: Tests written and run	2h	3h	+1
190319	**Iteration 4**	10h	11h	+3