Project plan+study diary

SpaceHunt

version 1.5

|  |  |  |
| --- | --- | --- |
| TUT | Pervasive Computing | TIE-21106 Software Engineering Methodology |
| Author: TheGroup | | Printed: |
| Distribution: Anna Vaňková, Miloš Švaňa, Nejc Maček, Wladimir Hofmann, Ali Doruk Gezici | | |
|  | | |
|  | | |
|  | | |
| Document status: draft | | Modified: 11.03.2018 17:33 |

VersioN historY

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Authors | Explanation (modifications) |
| 1.0  1.1  1.2  1.3  1.4  1.5 | 28.01.2014  11.02.2014  18.01.2015  26.1.2015  16.01.2017  08.01.2018 | Marko L.  Marko L.  Tensu  Marko L.  Kari S.  Farshad A. | Initial version  Deleted finnish text  Sections 1.4.x, cosmetic tuning  Final toucher  Adaptation for 2017 needs  Adaptation for 2018 needs |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table Of CONTENTS

1. introduction 3

2. PROJECT RESOURCES 4

2.1 Personnel 4

2.1.1 Anna Vanková 4

2.1.2 Milos Svana 4

2.1.3 Wladimir Hofmann 5

2.1.4 Nejc Macek 5

2.1.5 Ali Doruk Gezici 6

2.2 Process description 7

2.2.1 Project roadmap 7

2.2.2 Communication 7

2.2.3 Responsibilities 7

2.3 Tools and technologies 8

3. StUDY DIARY 9

3.1 Sprint 1 (every sprint as a section) 9

3.1.1 What went well 9

3.1.2 What difficulties you had 9

3.1.3 What were the main learnings 9

3.1.4 What did you decide to change for the next sprint 9

3.2 Sprint 2 9

3.2.1 What went well 9

3.2.2 What difficulties you had 9

3.2.3 What were the main learnings 9

3.2.4 What did you decide to change for the next sprint 9

4. RISK MANAGEMENT PLAN 9

4.1 Personnel risks 11

4.1.1 Risk P1 - Short term absence 11

4.1.2 Risk P2 - Long term absence 11

4.1.3 Risk P3 – Busy with work/school assignment 11

4.2 Technology risks 12

4.2.1 Risk T1 – Library dependency issues 12

4.2.2 Risk T2 – Learning new technologies issues 12

4.2.3 Risk T3 – Framework malfunction 13

4.3 Customer risks 13

4.3.1 Risk C1 – Changing requirements 13

4.3.2 Risk C2 – Requirements unclear 13

4.4 Environmental risks 14

4.4.1 Risk E1 – HW failure 14

4.4.2 Risk E2 – Internet failure 14

4.4.3 Risk E3 – Change of SW environment 14

4.4.4 Risk E4 – AgileFant unavailability 15

4.4.5 Risk E5 – GitLab unavailibility 15

4.5 Project management risks 15

4.5.1 Risk PM1 – Forgetting things 15

4.5.2 Risk PM2 – Bad communication 16

4.5.3 Risk PM3 – Bad scheduling 16

# introduction

This project is developing within course Software Engineering Methodologies in TUT. Main topic of the project is to create a game which will be similar to the JungleHunt game. Unlike the original game, our game will be called SpaceHunt and it will take place on planet in another galaxy. We will manage our work by using Scrum framework and AgileFant software. Our work will be divided into 4 sprints. For implementing the game we will use JavaScript programming language.

# PROJECT RESOURCES

## Personnel

### Anna Vanková

* Role: Scrum Master
* Contact information:
  + [anna.vankova@student.tut.fi](mailto:anna.vankova@student.tut.fi)
  + WhatsApp and phone: +420 730 952 330
  + Skype: aniicka94
  + FB: Anna Vaňková
* Previous expirience
  + basics of Java
  + experiences with project management
  + design and implementation of project templates for Erasmus+ project – bachelor thesis
  + project manager in Erasmus+ mobilities
  + MS Project
* Specific skills
* Fields of interest
  + Project Management
  + Teamwork
* Absence: 8.2. – 12.2., 30.3. – 4.4.
* Working hours: 7 hours per week

### Milos Svana

* Role: Developer
* Contact information:
  + [milos.svana@student.tut.fi](mailto:milos.svana@student.tut.fi),
  + WhatsApp and Slovak phone: +421 918 075 608
  + Finnish phone: +358 449 531 779
  + Skype: xauder1
  + LinkedIn: <https://goo.gl/zsAZLJ>
* Previous expirience:
  + many group projects during bachelor’s degree studies
  + developer for a small book social network project
  + developer and data analyst for heavy machinery business intelligence company
* Specific skills:
  + Python (Numpy, Sklearn),
  + advanced SQL,
  + PHP,
  + Javascript,
  + C,
  + C++,
  + basics of android applications development in Java
* Fields of interest: Data science, behavioral economy
* Absence: none
* Working hours: 7 hours per week, mostly on Friday

### Wladimir Hofmann

* Role: Product Owner, Developer
* Contact information:
  + Mail: [wladimir.hofmann@student.tut.fi](mailto:wladimir.hofmann@student.tut.fi)
  + Skype/FB: Fladdi.Mir
  + Phone/Whatsapp: +49 176 983 538 35
* Previous experience
  + Material flow simulation
  + Programming (C#, Java, C++ - Basic level)
* Specific skills: we will see :D
* Fields of interest: Programming, Teamwork, & Fun
* Absences: 08.-10.03.
* Working hours: 7h/week

### Nejc Macek

* Role: Developer
* Contact information:
  + [macek@student.tut.fi](mailto:macek@student.tut.fi)
  + [macek.nejc@gmail.com](mailto:macek.nejc@gmail.com)
  + [nejc.macek@student.um.si](mailto:nejc.macek@student.um.si)
  + WhatsApp and phone: +386 31 499 098
  + Skype: [nejcmacek](callto:nejcmacek)
  + Facebook: [Nejc Maček](https://www.facebook.com/macek.nejc)
  + LinkedIn: [Nejc Maček](https://www.linkedin.com/in/nejc-macek/)
  + Website: [nejo.si](https://nejo.si/)
* Previous experience:
  + Employment at PovioLabs
  + Employment at Agitavit
  + Voluntary services at Gimnazija Celje – Center
  + University projects:
    - c++ Variadic Templates research
    - [OpenScience Alternative Frontend](https://openscience.si/iskanjeNejcMacek/)
  + [Many personal projects](https://apps.nejo.si/)
* Specific skills:
  + Web develpment (JS, TS, CSS, HTML, React, Angular, jQuery)
  + Node.js develpment
  + C#, WinForms, .NET, .NET Core, ASP.NET
  + Java
  + c++
  + See also: [nejo.si/skills](https://nejo.si/skills/)
* Fields of interest: web development, web applications, frontend development
* Absence: 27.2. – 4.3., 15.3. – 30.3.
* Working hours: 7 hours per week

### Ali Doruk Gezici

* Role: Developer
* Contact information:
  + [ali.gezici@student.tut.fi](mailto:ali.gezici@student.tut.fi)
  + [doruk@gezici.me](mailto:doruk@gezici.me)
  + WhatsApp and Phone: +358 41 705 6467
  + Skype: doruk.gezici
  + Facebook: Doruk Gezici
  + LinkedIn: Doruk Gezici
  + Website: [dorukgezici.com](http://dorukgezici.com/)
* Previous experience:
  + iOS Developer at Kolay Randevu
  + IoT & Mobile Developer at ARDIC R&D
  + IT Coordinator at EESTEC LC Istanbul
  + IoT Line Fair ’17 Coordinator at EESTEC LC Istanbul
  + Django Developer at ITUscheduler
* Specific skills:
  + Web Development (Python/Django, HTML, CSS, Bootstrap, JS)
  + Python, C++, Software Development
  + Swift, iOS Mobile Application Development
  + Linux SysAdmin
  + Teamwork, Coordinator
* Fields of interest:
  + Entrepreneurship
  + Mobile & Back-End Web Development
  + Linux SysAdmin & Free Software
* Absence: 19.02-25.02, 30.03-03.04
* Working hours: 7 hours per week

## Process description

### Project roadmap

We will measure success by gathering feedback from our friends. After every sprint we will send the game to our friends, they will test it and give us a feedback.

#### Sprint 1

Environment setup and basic architecture. Our goal (milestone) will be that player can move in the environment.

#### Sprint 2

Implementing 2 levels of the game. We will have a playable game with 2 levels.

#### Sprint 3

Implementing another 2 levels of the game. We will have a playable game with 4 levels.

#### Sprint 4

Implementing score system, storytelling, testing, fixing bugs and improving graphics.

### Communication

We will have weekly meetings each Friday from 2pm to 4pm. After this meeting we will have an optional coding session. The first one will be held on 26.1. with the aim of setting up the develoment envirnoment and preparing the application structure. We plan to use WhatsApp and Skype for remote communication.

### Responsibilities

Implementation of individual user requirements/stories will be delegated on the start of each sprint to one of the developers.

#### Anna

Storytelling, project management – Agilefant

#### Milos

Code review, accepting/declining merge requests to master branch, implementation of user requirements.

#### Ali Doruk

Development Environment, Server Management, Implementation of User Stories

#### Wladimir

Implementation of user stories

#### Nejc

Software and project architecture. Implementation of user stories.

## Tools and technologies

Table 1.1: Tools used in the project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Purpose** | **Tool** | **Contact person** | **version** |
| Documentation | MS Word (word processing)  [office.microsoft.com](file:///\\intra.tut.fi\..\..\..\..\..\Local%20Settings\Temp\office.microsoft.com) | A.V. | 2010 |
| Communication | WhatsApp  <https://www.whatsapp.com/> | A.V. | 7.22 |
| Skype (internet calls)  <http://www.skype.org> | A.V. | 4.64 |
| Version management | Git  <https://git-scm.com/>  Repository:  <https://course-gitlab.tut.fi/sweng_2018/g10---thegroup> | M.S. | 1.4.6 |
| Development platform | Javascript |  | Chrome 63 runtime or later |
| Project Management | AgileFant  <https://app.agilefant.com/login> | A.V. |  |
| Editor/IDE | Visual Studio Code  <https://code.visualstudio.com/> | N.M. | latest |
| IntelliJ platform  <https://www.jetbrains.com/idea/> | M.S. | 2018.3 or later |

# StUDY DIARY

This chapter holds your journal of lessons learned during the course. That is, **more detailed analysis of previous Sprint’s contents**.

## Sprint 1

### What went well

* We handled to do everything we planned.
* Everyone has became familiar with a framework withouth any bigger difficulties.
* We were able to meet every week in predetermined times. We comunicated without any probems for all the time.

### What difficulties we had

* Our description of tasks wasn’t concreete enough and we had problem to understand it later.
* Tasks weren't split equally.
* Our palnning of the sprint took too much time.
* We estimated inaccurately the efforts of the tasks.

### What were the main learnings

* We get to know how the Scrum works.
* We get to know pitfalls and benefits of teamwork.
* We found out that graphics could takes much more time than we expected.

### What did we decide to change for the next sprint

* We should do better description of tasks.
* We will divide our work more equally.

## Sprint 2

### What went well

* Creation of swinging wines were easier than we expected.
* Tasks were better distributed.
* Our sprint planning was shorter and better organized.
* We weren’t forgetting to update an Agilefant so much.

### What difficulties we had

* Because of exams and assignment we didn’t have as much time as we wanted.
* We have found some bugs in framework.
* Issue with collision detection in level 1 – underestimation of this task.
* We couldn’t find suitable sprites.
* Broken arm.

### What were the main learnings

* The risks are more real (serious) then we expected.
* New findings about the framework.

### What did we decide to change for the next sprint

* Better presentation of our product.

## Sprint 3

### What went well

* Basic and most importatn things for this sprint were implemented on time.
* We had better work distribution.
* Basic things are done and we have enough time to finish everytnig in last sprint.

### What difficulties we had

* Members of the team were traveling a lot within this sprint.
* Bad delegation of Döruks' work while he was gone.
* Git merging – we didn’t merge older branches on time which caused some problems.

### What were the main learnings

* We need to delegate the work when somebody is missing.

### What did we decide to change for the next sprint

* Be more careful about the time.

## Sprint 4

### What went well

### What difficulties you had

### What were the main learnings

### What did you decide to change for the next sprint

# RISK MANAGEMENT PLAN

Our risks are divided into 5 categories – personnel, technology, customer, environmental and project management risks.

We put risks into a table 4.1 and for every risk we estimated probability and impact by using a scale of 1 to 3. Then we calculated seriousness for each risk.

Our basic risk-prevention policy is shared code ownership, by which we mean, that everybody should at least at a very basic level know about code written by others. Anybody can view and modify any part of the codebase. This allows others to finish other person’s task in case of sudden absence.

Table 4.1: Project risks.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk ID** | **Description** | **Probability** | **Impact** | **Seriousness** |
| P1 | Short term absence | 3 | 2 | 6 |
| P2 | Long term absence | 1 | 3 | 3 |
| P3 | Busy with work/school assignment | 3 | 2 | 6 |
| T1 | Library dependency issues | 1 | 2 | 2 |
| T2 | Learning new technologies issues | 3 | 2 | 6 |
| T3 | Framework malfunction | 1 | 3 | 3 |
| C1 | Changing requirements | 3 | 1 | 3 |
| C2 | Requirements unclear | 3 | 2 | 6 |
| E1 | HW failure | 1 | 2 | 2 |
| E2 | Internet failure | 1 | 3 | 3 |
| E3 | Change of SW environment | 1 | 1 | 1 |
| E4 | AgileFant unavailability | 1 | 3 | 3 |
| E5 | GitLab unavailability | 1 | 1 | 1 |
| PM1 | Forgetting things | 2 | 2 | 4 |
| PM2 | Bad communication | 2 | 2 | 4 |
| PM3 | Bad scheduling | 3 | 2 | 5 |

## Personnel risks

### Risk P1 - Short term absence

**Root cause:** It can be flu or absence caused by travelling.

**Seriousness:** Probabilityof this risk is high because most of us plan some travelling and we are not accustomed to such a low temperature, co we can easily get ill. However, impact is medium and we are prepared for this.

**How to avoid:** At least in case of illness, we should take care about our health and wear warm clothes.

**How to prevent:** Everything is shared and everyone can modify it, even when the key person is missing.

**How to survive:** We will divide a missing person’s work and we will try to keep in touch, also with ill person throw Skype and Whatsapp.

### Risk P2 - Long term absence

**Root cause:** Something more serious could happen and someone would have to stay in hospital or leave the Finland. It can be bad injury or some family tragedy.

**Seriousness:** Probability in this case is quite low, but on the other hand impact would be very high. Everyone is important for our teamwork.

**How to avoid:** Don’t do any risky sports or other dangerous activities.

**How to prevent:** Everything is shared and everyone can modify it, even when the key person is missing.

**How to survive:** We would have extra meeting where we would divide a missing person’s work and do more things in shorter time.

### Risk P3 – Busy with work/school assignment

**Root cause:** We have signed up for many courses, some of us are working and it can happen that we will need to finish some other important work.

**Seriousness:** Chance that this will happen is very big and it would have medium impact on our work. This risk is one of the most serious.

**How to avoid:** To avoid this risk is important to prepare everything, make some schedule of work and try to follow it.

**How to prevent:** It is good to do difficult tasks as soon as possible to have some time reserve at the end of the project.

**How to survive:** We have some time reserves and we are doing the most work we can when we have a time.

## Technology risks

### Risk T1 – Library dependency issues

**Root cause:** Installing npm packages causes libraries of different version to be installed on different machines, causing functionality mismatch, resulting in misbehaving or malfunctioning application componets.

**Seriousness:** Probability is low as npm handles versioning well. Severity is medium as this would result in wasted hours trying to solve the issue.

**How to avoid:** Anticipate which libraries might change in future and specify their definite version.

**How to prevent:** Discuss the issue with other team members to find most appropriate solution. Changing version on your own and not informing others about the issue may worsen the problem for future development.

**How to survive:** Find which library and its version caused the issue. Resolve the issue by updating libraries or specifying their definite version. Inform other team members who should update their libraries as well.

### Risk T2 – Learning new technologies issues

**Root cause:** Some technologies are new for most of us and learn it can be harder than we expected.

**Seriousness:** It will happen with high probability, but impact is not so high.

**How to avoid:** We should agree on technologies we will use in advance.

**How to prevent:** At least one team member knows the technology.

**How to survive:** We will have meeting every week, where we will discuss our problems, we will help each other and explaining unclear things.

### Risk T3 – Framework malfunction

**Root cause:** Chosen framework won’t have required functions, or they are malfunctioning.

**Seriousness:** Impact of this would be high, but probability of this is very low.

**How to avoid:** Learn as much as possible about the framework before project starts.

**How to prevent:** Use different features, which have been tested to work.

**How to survive:** Find a workaround, implement custom error handling logic, or, at worst, change the framework used. The latter should only be considered in events of apocalyptic severity.

## Customer risks

### Risk C1 – Changing requirements

**Root cause:** Customer will change his/her mind about requirements.

**Seriousness:** Because we are learning it, probability that some requirement will be changed is high, anyway we are prepare for this situation that’s why impact is low.

**How to avoid:** Ask customer about every detail about the product and his requirements in advance and write it down.

**How to prevent:** Meet with customer often to approve our work.

**How to survive:** Divide the work on small tasks which will be easy to change.

### Risk C2 – Requirements unclear

**Root cause:** Requirements are not as clear as they could be.

**Seriousness:** This risk is very serious, because there is high probability of this and it could have medium impact on our work.

**How to avoid:** Ask customer about every detail about the product and his requirements in advance and write it down.

**How to prevent:** Ask about every requirement more times and let explain his/her the unclear things.

**How to survive:** Divide the work on small tasks which will be easy to change.

## Environmental risks

### Risk E1 – HW failure

**Root cause:** Bad condition of HW, it is old or broken.

**Seriousness:** There is a low chance this will happen and it would have just medium impact.

**How to avoid:** Regularly check out HW condition.

**How to prevent:** Save a done work into the cloud.

**How to survive:** Use school HW.

### Risk E2 – Internet failure

**Root cause:** It is not possible to connect to the Internet either in school, or in our dormitory.

**Seriousness:** This has very low possibility, but high impact.

**How to avoid:** Have a prepaid mobile internet.

**How to prevent:** Call Internet service provider.

**How to survive:** Everyone has prepaid mobile internet and saved work also in the computer, not just cloud.

### Risk E3 – Change of SW environment

**Root cause:** Personal reasons for changing development environment (e.g. change of operating system, change of IDE or text editor).

**Seriousness:** As the reason for the change would be our own preferences and not a certain threat, we can consider this issue unserious. It may lead to problems such as the need to set up the development environment again, which is time consuming.

**How to avoid:** Try to anticipate which SW is better and use it from start.

**How to prevent:** Try to cope with existing software until an appropriate time to make the change appears.

**How to survive:** Make sure to reserve enough time for changing the SW environment. Risk of arising problems should be considered. Backup you work first.

### Risk E4 – AgileFant unavailability

**Root cause:** It is not possible to use AgileFant, there is some problem with the app.

**Seriousness:** Probability of this is very low, but it would have high impact on the project.

**How to avoid:** Use some alternative SW.

**How to prevent:** Inform a teacher about this issue.

**How to survive:** We would plane our activities just in a paper, or some text document.

### Risk E5 – GitLab unavailibility

**Root cause:** Server malfunction, system crash.

**Seriousness:** The issue is considered unserious as local copies of remote repository still exist on our computer.

**How to avoid:** Keep a copy of the remote repository locally.

**How to prevent:** Inform staff about system malfunction.

**How to survive:** Setup new git repository on a different git system.

## Project management risks

### Risk PM1 – Forgetting things

**Root cause:** We could overlook something within planning project.

**Seriousness:** This risk has medium probability and medium impact.

**How to avoid:** We can avoid it by careful planning of the project and all team should have participated.

**How to prevent:** Consult the project process with a teacher after planning process.

**How to survive:** Divide the project into small tasks, that will be easy to add something.

### Risk PM2 – Bad communication

**Root cause:** Team members are not aware what the others are doing.

**Seriousness:** This risk has medium probability and medium impact.

**How to avoid:** Agree on communication channels and meet personally every week.

**How to prevent:** Try to initiate the conversation.

**How to survive:** We have meeting every Friday, where everyone can ask what is unclear and divide the work.

### Risk PM3 – Bad scheduling

**Root cause:** We are not able to meet our plans on time.

**Seriousness:** The probability here is very high, but it does not have such an impact because we have some time reserves.

**How to avoid:** Include in a plan just what the customer wants.

**How to prevent:** We should have time reserves.

**How to survive:** Implement just the most important features.