Project plan+study diary

Jungle Hunt

version 1.1

|  |  |  |
| --- | --- | --- |
| TUT | Pervasive Computing | TIE-21106 Software Engineering Methodology |
| Author: Lassi Rintala | | Printed: - |
| Distribution: | | |
| Lassi Rintala 224895  Pinò Surace 262767  Samu Mäkinen 253582  Vili Saura 240264 | | |
| Document status: draft | | Modified: 11.2.2018 20:38 |

VersioN historY

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Authors | Explanation (modifications) |
| 1.0 | 18.01.2018 | Lassi R. | Initial version |
| 1.1 | 29.01.2018 | Lassi R. | Added tools & technologies, personnel information |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table Of CONTENTS

1. PROJECT RESOURCES 3

1.1 Personnel 3

1.2 Process description 3

1.3 Tools and technologies 4

2. StUDY DIARY 5

2.1 Sprint 1 (every sprint as a section) 5

2.1.1 What went well 5

2.1.2 What difficulties you had 5

2.1.3 What were the main learnings 5

2.1.4 What did you decide to change for the next sprint 5

2.2 Sprint 2 5

2.2.1 What went well 5

2.2.2 What difficulties you had 5

2.2.3 What were the main learnings 5

2.2.4 What did you decide to change for the next sprint 5

3. RISK MANAGEMENT PLAN 5

3.1 [example] Personnel risks 7

3.1.1 [example] Risk P1: short term absence of one person 7

3.2 [example] Technology risks 7

3.2.1 [example] Risk T1: hard disk failure 7

# PROJECT RESOURCES

This chapter holds the project resources.

## Personnel

For each person

* estimate contribution in person hours for each sprint
* travels or other known absences

Product owner:

**Tero Ahtee**

Development team:

**Lassi Rintala (Scrum master for sprints 0 and 1)**

* Email: [lassi.rintala@student.tut.fi](mailto:lassi.rintala@student.tut.fi)
* Previous experience: 3 years working as a software engineer
* Special skills: C/C++
* Specific fields of interest: Unity

**Samu Mäkinen**

* + - * Email: [samu.makinen@student.tut.fi](mailto:samu.makinen@student.tut.fi)
      * Previous experience: University coding, slight hobbyist coding for approx. 1 year.
      * Special skills: Jack of all trades, master of none.
      * Specific fields of interest: Game Design, C++, Unity

**Vili Saura**

* + - * Email: [vili.saura@student.tut.fi](mailto:vili.saura@student.tut.fi)
      * Previous experience: High School and University coding.
      * Special Skills: C++
      * Specific fields of interest: Game Design, Unity

**Pinò Surace**

* + - * Email: [pino.surace@student.tut.fi](mailto:pino.surace@student.tut.fi)
      * Previous experience: University study projects
      * Special skills: C, C++, Java, Python
      * Specific fields of interest: Unity

## Process description

Milestone 1: End of sprint 1, requirements 1-3 done

Milestone 2: End of sprint 2, requirements 4-6 done

Milestone 3: End of sprint 3, requirements 7-9 done

Milestone 4: End of sprint 4, requirements 10 done + polishing the game

Goals and success criteria:

* The product fulfills the given requirements
* The game developers are happy with the end result

Success measurement:

* Feedback from the customer
* Reaching milestones in time
* Assignment grade

Running the project:

* Bi-weekly meetings (Sunday evenings physical meeting and a shorter Skype meeting another day)
* Telegram chat group
* Project management with Agilefant
* Version control with Git
* Documentation available
* Scrum master responsibility is changed every sprint
  + Lassi is the first scrum master for sprints 0 and 1
  + Pino scrum master for sprint 2
* Responsibilities (who implements what, takes care of what, …) are mostly decided in the weekly meetings

*Check also the risks in chapter 3, as your process should make you at least robust to the risks. For example, how to avoid impact of absent key person? In addition, it is not enough to be robust, you should also learn. So, how do you get feedback from the team, assistant and so on? Daily Scrums, sprint autopsies and other such events?*

*KEEP THIS UPDATED AS YOU LEARN DURING THE COURSE.*

## 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) |  | 2010+ |
| Doxygen (comment notation and documentation generation) |  |  |
| Communication | Telegram |  |  |
| Skype (internet calls)  <http://www.skype.org> |  |  |
| Version management | Git | Lassi Rintala |  |
| Code implementation and compilation | Unity |  | 2017.3.0f3 |
| Visual Studio | Lassi Rintala | 2017 |

# 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 (every sprint as a section)

### What went well

* Work load distribution
* Meetings
* Project work in general
* Studying Unity

### What difficulties you had

* Modeling the rope
* Player catching the rope
* Difference between 2D and 3D game object components (tutorials in 3D)
* Scaling issues with GUI (canvas, camera)

### What were the main learnings

* How to use Git
* Unity basics
* Agilefant basics
* Task estimation
* Agile methods

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

* Scrum master changed to Pino
* Potentially changing the game theme from jungle to something else
* Make all the levels in some template / placeholder form

## Sprint 2

### 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

Consider risks for your project. **The most usual risks** that will affect projects are due to customer, the team itself and technology.

Just listing some risks at the beginning of the project doesn’t help you much… if anything at all.

You can try to come up with **Plan Bs** for the risks. However, remember that the things you won’t expect, will hurt you the most. Thus, focus on the generalities, not on specifics.

Try not to underestimate the probability of small and common risks, and not to overestimate the probability of rare and remarkable events. For example, people usually get 1-2 flus during a year, so in 4 months, it is quite probable that one of the team will be sick and may infect others, too. An average flu lasts for more than one week. So, be prepared. On the other hand, getting hurt in traffic so that it will take a week to recover happens to only for 15000 people yearly in Finland (less than 3 permille of population).

Be sensitive for weak signals, such as difficulties with new technology or runny noses.

**You should think of risks in all categories:**

* customer (ending the project, changing requirements, requirements remain unclear,…)
* technologies (hw/sw; hard to acquire, learning new technologies takes time, suitable library is not found,…)
* environment (network connections and servers fail,…)
* personnel (getting ill, changing jobs, busy with work,…)
* project management (bad scheduling, bad communication, forgetting things,…).

Usually we calculate risk’s **seriousness = severity \* probability**.

Table 4.1: Project risks.

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk ID** | **Description** | **Probability** | **Impact** |
| P1 | Short term absence | 3 | 2 |
| T1 | Hard disk failure | 2 | 2 |
| M1 | Too low task time estimations causing tight schedule | 2 | 2 |
| M2 | Confusion in task assignment (overlapping implementations etc.) | 1 | 1 |
| S1 | Huge refactoring of current implementation | 3 | 3 |
| S2 | Customer changes or adds requirements | 2 | 2 |
| S3 | Minor bugs in the final release | 3 | 1 |
| S4 | Major bugs in the final release | 1 | 3 |

## Personnel risks

Try to estimate risk probability, use a scale of **1 to 3** (or 1..5) or Small, Medium, Large.

Other criterion will be the impact or severity. So, how the risk will harm you, if realized. Use similar scaling as in probability.

### Risk P1: short term absence of one person

**Root cause (source):** A key person will be absent for several days.

**Importance (seriousness):** from the table, basically probability and impact, possibly combined with frequency.

**Avoidance:** Avoid being near people who have a flu

**Response (prevention):** Redistribute the work load and share all relevant information, so that the team will be able to carry on.

**Recovery (survival):** Redistribute the workload; focus on the most important features.

## Technology risks

### Risk T1: hard disk failure

**Symptom, early warning sign:** disk makes noise, arbitrary reading errors occur more often than before.

**Source or reason:** hard disk is at the end of its lifespan, or hard hit

on computer while disk was running.

**Probability:** 2 medium (on scale 1-3)

**Seriousness:** 2 medium (on scale 1-3)

**How to avoid:** buy a new disk when starting a project.

**How to prevent:** when first symptoms occur, take additional back-ups and change the disk as soon as possible.

**How to survive:** back-ups, and a replacement disk or whole computer.

## Management risks

### Risk M1: Too low task time estimations causing tight schedule

**Reason:** Tasks take longer to complete than originally estimated

**How to avoid:** Make estimations always bigger than expected time used

**How to prevent:** Really concentrate on roughly implementing the feature rather than for example paying too much attention to small details

**How to survive:** Implement tasks in the order of priority

### Risk M2: Confusion in task assignment (overlapping implementations etc.)

**Reason:** Task status is not updated correctly in Agilefant or otherwise absence of communication between team members about which tasks they are working on

**How to avoid:** Always keep Agilefant updated

**How to prevent:** Ask team members if someone is already working on the task you are about to start

**How to survive:** Choose one of the parallel implementations to be used, discard others

## Software risks

### Risk S1: Huge refactoring of current implementation

**Reason:** Some software component has been first poorly designed and needs reimplementation in a new way to support further development

**How to avoid:** When designing feature implementations, think about them in their context far ahead

**How to survive:** Coordinate development so that the refactoring won’t cause too much interference in other developers work

### Risk S2: Customer changes or adds requirements

**Reason:** Customer changes existing requirements or adds some more

**How to avoid:** Can’t be avoided

**How to survive:** Have the software implemented so that adding more things in it is easy enough

### Risk S3: Minor bugs in the final release

**Reason:** Due to implementation not careful enough, bugs are still existing in the final product release

**How to avoid:** Extensive testing and careful design and implementation throughout the development process

**How to survive:** Nothing to do after final release

### Risk S4: Major bugs in the final release

**Reason:** Due to implementation not careful enough, bugs are still existing in the final product release

**How to avoid:** Extensive testing and careful design and implementation throughout the development process

**How to survive:** Nothing to do after final release