Project plan+study diary

Pkoski Raid (Group 21)

version 1.3

TUT	Pervasive Computing	TIE-21106 Software Engineering Methodology
Author: Eino Itkonen		Printed:
Distribution: Tuomas Kaukoranta, tuomas.kaukoranta@student.tut.fi - 233733 Teemu Mähönen, teemu.mahonen@student.tut.fi - 229165 Eino Itkonen, eino.itkonen@student.tut.fi - 240373 Shaptarshi Basu, shaptarshi.basu@student.tut.fi - 267647		
Tero Ahtee, tero.ahtee@tut.fi		
Document status: sprint 3 version		Modified: 02.04.2017 22:10

VERSION HISTORY

Version	Date	Authors	Explanation (modifications)
1.0	29.01.2017	Group 21	Initial version of project plan

1.1	10.02.2017	Group 21	Sprint 1 version.
1.2	12.03.2017	Group 21	Sprint 2 version.
1.3	02.04.2017	Group 21	Sprint 3 version.
1.4			

TABLE OF CONTENTS

1. PROJECT RESOURCES

This chapter holds the project resources.

1.1 Personnel

Our group PPKoski consists of four members.
Here below are listed our personnel with their roles, contact information, previous experience, special skills and fields of interest.

Tuomas Kaukoranta - tuomas.kaukoranta@student.tut.fi
- **Product owner**
- Experienced with game development.
- Potentially the dedicated graphics artist.
- Capable of putting ~40 hours per sprint if necessary.

Teemu Mähönen - teemu.mahonen@student.tut.fi
- Scrum master
- Interested mainly in web development, solid programming skills.
- Able to contribute ~30-50 hours per sprint.

Eino Itkonen - eino.itkonen@student.tut.fi
- Programmer/developer
- Interested in web applications. Mediocre programmer.
- Able to contribute ~30-50 hours per sprint.

Shaptarshi Basu - shaptarshi.basu@student.tut.fi
- Programmer/developer
- Interested in game developing, solid programming skills.
- Able to contribute ~30-50 hours per sprint.

1.2 Process description

The process will rely heavily on using the Agilefant project management software. Product backlog will be handled there as stories that the product owner will manage. Optimally the team will have face-to-face meetings every friday, but these can be substituted with digital communications if absolutely necessary. Additional communication and meetings will be decided by product owner if deemed necessary.

The scrum master is responsible of making sure that the work is distributed well across the whole team and that no one strays too far behind in schedule. In general most of the work is done remotely, but if the situation requires more intense and focused work, the product owner or scrum master will arrange coding sessions at the university.

In case of any of the risks becoming reality, product owner and scrum manager are responsible for handling the redistribution of the workload and revising the backlogs.

1.2.1 Plans for Sprint 1:

-Learn the Processing software and get familiar with the tools needed.
-Implement at least customer requests 1 and 2.

1.2.2 Plans for Sprint 2:

-Get basic main menu and gameplay running with.
-Implement at least the customer requirements 3 and 4.

1.2.3 Plans for Sprint 3:

-Start forming proper gameplay with the elements built in the earlier sprints and make the thing look and feel more like a game.
-Implement at least the customer requirements 6, 7, 8. (5 was kinda done in sprint 2 already.)

1.3 Tools and technologies

In this chapter there are listed in the table all the tools and technologies that our group will be using during the project.

Table 1.1: Tools used in the project.

Purpose	Tool	Contact person	version
Documentation	Google Docs (Collaborative word processing) https://docs.google.com	S.B.	N/A
	draw.io (UML tool) https://www.draw.io/	T.K.	6.0.2.14
	OpenOffice Writer (word processing) http://www.openoffice.org/	T.K.	4.1.3
Communication	For quick access: WhatsApp group? Other purposes: TUT mail	E.I.	N/A
Version management	Git https://git-scm.com/ https://gitlab.rd.tut.fi/sweng-2017/g21---ppkoski/	T.M.	2.11.0
Artwork	GIMP (2D Graphics) https://www.gimp.org/	T.K.	2.8.18
Project management	Agilefant https://www.agilefant.com https://app.agilefant.com/TTY-TIE/product/196283/tree		N/A
Game engine / framework	Processing https://processing.org/		3.2.3

2. STUDY DIARY

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

2.1 Sprint 1 (every sprint as a section)

2.1.1 What went well

Overall knowledge of the Processing framework is now better, most of the sprint went into learning it and figuring out how to handle the 2d graphics in a way that's fast (with minimal stress to the garbage collector) and can be used somewhat intuitively by the programmers.

2.1.2 What difficulties you had

We had some problems getting started with how to divide any of the tasks, and how to use Agilefant in a way that actually helps with the development. Team communications and planning has also been hard.

2.1.3 What were the main learnings

Main learnings were mostly about the function of processing software and a little insight about Agilefant. Another thing learned was how hard it actually is to try and manage a 4 person project that's actually something that could be done by a single person.

2.1.4 What did you decide to change for the next sprint

More in depth analysis of the stories to divide them better into tasks, more team communication after stories are added to backlog to talk about the tasks before anyone starts anything, so the whole team has a better idea of who's gonna do what.

2.2 Sprint 2

2.2.1 What went well

The project is progressing somewhat, few new implementations of the customer requirements are made. Project management and communication worked better than the last sprint, with actual meetings and usage of the Agilefant platform.

2.2.2 What difficulties you had

The exam week messed up schedules and provided an unwelcome workload, as well as assignments from other courses taking up a lot of our time that should've been used for planning/organizing. Using of the agilefant is still not at a level where it should be, it still feels too clunky to use for what it's worth compared to similar platforms like Trello.

2.2.3 What were the main learnings

Getting the hang of Agilefant in general is one, realizing how the amount of available time for everyone varies from week to week with other courses taking priority.

2.2.4 What did you decide to change for the next sprint

Perhaps some live programming sessions would be useful to get everyone on the same page not just about tasks and what needs to be done but also what has been done and how the implementations work.

2.3 Sprint 3

2.3.1 What went well

Game progressed further and needed requirements should be all done by next sprint review. Working with the current codebase is fairly smooth.

2.3.2 What difficulties you had

Starting actual work early in the sprint.

2.3.3 What were the main learnings

2.3.4 What did you decide to change for the next sprint

Apart from starting working earlier, not much, seems like we can work this way to finish the project and our schedules do not allow us to push much further than this.

3. RISK MANAGEMENT PLAN

In this chapter different risks are listed and explained. Probabilities and impacts are valuated from 1 to 3, where 1 is least probable and has the least impact and 3 is the most probable and has the greatest impact.

Table 3.1: Project risks.

Risk ID	Description	Probability	Impact
P1	Short term absence	3	1
P2	Group member leaves	1	3
T1	Developer computer failure	1	2
T2	Isuues with Processing	2	2
C1	Customer not approving product	1	2
E1	Git or web tool downtime	2	2
M1	Starting working too late in a sprint	2	3

3.1 Personnel risks

3.1.1 Risk P1: Short term absence of one person

Source or reason: A group member is absent for a few days because of sickness or being temporarily busy with other things.
Probability: 3 high (on scale 1-3)
Seriousness: 1 low (on scale 1-3)
How to avoid: No reasonable way to avoid sickness but for other busyness team members have to manage their own time well.
How to prevent: Arranging workload distribution as early as possible to not let schedule slip.
How to survive: Other team members have to work more and harder to reach the goals, or goals have to be moved closer if everyone is already overworked.

3.1.2 Risk P2: Group member leaves

Source or reason: A group member can't continue with the course.

Probability: 1 low (on scale 1-3)
Seriousness: 3 high (on scale 1-3)
How to avoid: No one in the group leaves without a very good reason.
How to prevent: Inform others early if it looks like it could happen, divide work to other members.
How to survive: Distribute workload to the rest of the team and revise plan to match the lesser manpower.

3.2 Technology risks

3.2.1 Risk T1: Developer computer failure

Source or reason: Any hardware component failing, OS getting messed up, or losing a laptop.
Probability: 1 low (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)
How to avoid: Treat your computers with respect.
How to prevent: Commit often to git.
How to survive: Version control to counter data-loss, and a secondary computer (workstations at university will do) to resume work.

3.2.2 Risk T2: Issues with Processing

Source or reason: Not being able to learn the Processing software well enough.
Probability: 2 medium (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)
How to avoid: Be sure to go through the documentation early to get a good idea of the workflow and capabilities of the software before serious work.
How to prevent: Have documentation handy and plan alternative approaches for tasks that seem tricky.
How to survive: Ask course personnel for help or revise plan to get around the problematic parts.

3.3 Customer risks

3.3.1 Risk C1: Customer not approving product

Source or reason: Not following the requirements closely enough, or the product is lackluster in quality.
Probability: 1 low (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)

How to avoid: Do not stray from the requirements, stay on schedule and do not write sloppy code.
How to prevent: Confirm product quality and direction with the customer often enough.
How to survive: If it's in the middle of the development, revise plan to include fixing issues to get the product on an acceptable level.

3.4 Environment risks

3.4.1 Risk E1: Git or web tool downtime

Source or reason: Git or some other online tool is temporarily unavailable.
Probability: 2 medium (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)
How to avoid: Out of our hands, can't be avoided.
How to prevent: Make sure to not leave things until the last day where services not working properly would become major issue.
How to survive: Find alternative tools for the same tasks if possible, hope that the service resumes soon.

3.5 Management risks

3.5.1 Risk M1: Starting working too late in a sprint

Source or reason: Internal time management failing.
Probability: 2 medium (on scale 1-3)
Seriousness: 3 high (on scale 1-3)
How to avoid: Get a habit of always starting as soon as possible.
How to prevent: Keep asking the team members about the status of their tasks often to make sure they don't forget and push each other to work before it's too late.
How to survive: Distribute workload if possible, cut features if necessary.