Project Plan and Study Diary

Version 1.18 from 03.04.2017

G22 – Yet Another Group

Yet Another Game (YAG)

TUT – Pervasive Computing – TIE-21106 Software Engineering Methodology

Distribution: Farshad Ahmadi, Tero Ahtee, Kari Systä

Document status: Draft

Authors:
Milla Mäkinen <milla.makinen@student.tut.fi>
Mengyang Chen <mengyang.chen@student.tut.fi>
Daniel Berezvai <daniel.berezvai@student.tut.fi>
Alexandre Kirszenberg <alexandre.kirszenberg@student.tut.fi>

Revision History

Revision	Date	Author(s)	Description
1.0	28.01.2014	Marko L.	Initial version
1.1	11.02.2014	Marko L.	Deleted finnish text
1.2	18.01.2017	Tensu	Sections 1.4.x, cosmetic tuning
1.3	26.1.2017	Marko L.	Final toucher
1.4	16.01.2017	Kari S.	Adaptation for 2017 needs
1.5	25.01.2017	Daniel B.	Initial Shared document.

1.6	29.01.2017	Alex K.	Expand on risks.
1.7	29.01.2017	Chen	Process description
1.8	29.01.2017	Daniel B.	Final touches, layout fix, merge
1.9	29.01.2017	Milla M.	Tools, technologies, fixes
1.10	14.02.2017	Alex K.	Sprint 1
1.11	14.02.2017	Daniel B.	Doubled the length of each paragraph in our sprint 1 review. Detail rich explanations, miscellaneous fixes, and spellcheck. Clever titles introduced (I remembered to update Table of Contents too!). New tool: LaTeX. (A. K.)
1.12	12.03.2017	Alex K.	LaTeX version.
1.13	12.03.2017	Alex K.	Sprint 2
1.14	12.03.2017	Milla M.	Sprint 2 learning reflections, modified personnel info
1.15	12.03.2017	Daniel B.	Add distribution (teachers names) to titlepage.tex, as authored in the docx version (from template)
1.16	01.04.2017	Daniel B.	More minor edits to snippets, resources, and titlepage.tex (I just joined the cool kid club with pdfLaTeX and TeX Live. Before this I was using just TeX source code.)
1.17	02.04.2017	Daniel B.	Sprint 3: Table of contents fixed so it shows up, long titles shortened, fits on one page. Some sections reordered.
1.18	03.04.2017	Daniel B.	Burndown graphs finished, added, and documented. Update effort left by hand when changing effort spent.

Contents

1 Project Resources

This chapter holds the project resources.

1.1 Personnel

- **Milla Mäkinen** <milla.makinen@student.tut.fi>
  Scrum Master, programmer and totally an artist. Industrial engineer, programmer at heart. Some five years of game dev experience mainly in C++ and Javascript. Specializes in having no life. Some commits are named Ymirdev because git author information switching (GitHub account).

- **Mengyang Chen** <mengyang.chen@student.tut.fi>
  Coder familiar with Javascript and C++, interested in web programming and game programming.

- **Daniel Berezvai** <daniel.berezvai@student.tut.fi>
  Product Owner, Game modder (Warcraft 3, etc.), studied 20+ programming languages, personal website: http://3ice.hu/

- **Alexandre Kirszenberg** <alexandre.kirszenberg@student.tut.fi>
  Programmer with previous experience as a Frontend Software Engineer. Interested in Software Development.

1.2 Process description

The goal of our project is to make an interesting game, and pass the course get the credits. To measure the success of the project, we will see whether the stories are fulfilled at the end.

Basically we are going to use Whatsapp for discussion with each other and allocate the works evenly to everyone in the team, and use email to inform some general information to everyone in the team.

To ensure the success of the project, one will inform other team members when facing some really tough tasks, so coding nights or jams will be held to solve the task together.

1.3 Tools and technology

The tools used in this project are listed in Table 2. Most of them are not affected by versioning as everyone is automatically updated to the latest version. If version difference issues arise, the contact person will decide which version everyone in the team will use.

Table 2: Tools used in the project

Purpose	**Tool**	**Contact Person**	**Version**
Documentation	LaTeX https://www.latex-project.org/ with pdflatex	A.K.	1.4
Communication	WhatsApp http://www.whatsapp.com	M.M.	–
	Skype http://www.skype.org	M.C.	–
Version management	GitLab http://rd.gitlab.tut.fi Accessible through the course selection or https://gitlab.rd.tut.fi//sweng-2017/g22—yet-another-group	M.M	–
Agile Management	AgileFant https://app.agilefant.com	D.B.	–
Programming language of choice	Processing https://processing.org/	D.B.	3.2.3

2 Study Diary

This chapter holds our journal of lessons learned during the course. Detailed analysis of each Sprints' contents shall follow.

2.1 Sprint 1

For this first sprint the product owner proposed a very minimalist approach as we familiarized ourselves with the several new tools at our disposal. The official requirement was to fulfill two user stories per sprint, and conveniently enough, the first two tasks were as simple as; one: reading input from the user (their name) and then two: printing out a personalized backstory for them.

2.1.1 Everything went well (almost everything)

Right from the get go we had a very good understanding of how to achieve our goals. As such, it was easy to start working on the project and rapidly deliver outstanding results for the first sprint. In fact we completed our intentionally very light sprint 1 workload before it even officially began.

As to not fall into despair from lack of work, we selected additional tasks to complete. In the end, we have four additional user stories more or less in a completed state, in addition to the first two proposed solutions that were refactored several times until they met everyone's ever higher standards for good code.

2.1.2 We had difficulties

Communication Working in a team can be difficult, especially when it comes to communicating between team members. None of us were very familiar with Agilefant, so we did most of the communication on WhatsApp and in person.

Missed deadline One failure was that we missed the delivery deadline. All our tasks were complete weeks ago, but we never submitted them to Repolainen because we constantly kept adding new things. In the future, we will

create calendar entries for each deadline with automatic email notifications set up to give us a reminder two days in advance.

2.1.3 What were the main learnings

We should use Agilefant more often, in order to make sure that we're all on the same page. It also helps in defining clear objectives for individuals and for the team as a whole.

2.1.4 For the next sprint, we decided to change...

our work habits and one tool We decided to try and take more advantage of the tools at our disposal. Since our team works mostly in quick and very productive iterations, it might be interesting for us to synchronize our efforts in real-life sprints.

As we have had difficulties taming Microsoft Office, a proposed move to TeX was initiated. It will improve our work process greatly. No more crashes due to faulty edit conflict resolution, nor unnecessary struggling with the laggy online version of Word that performs especially poorly on MacBooks.

2.2 Sprint 2

The goal of this sprint was to implement the core features of the game. Indeed, in order for our product to actually *feel* like a game, we needed the notions of an objective and obstacles.

2.2.1 All is well in the world

It's been smooth sailing on this sprint. With our experience working as a team with Agilefant and Processing, a lot of features we'd planned came together nicely.

2.2.2 Difficulties?

The difficulties are really starting to show on the road that lies ahead. Some questions were brought up during this sprint that will only be answered during Sprint 3. Hopefully, the answers will not be too painful for us.

One such question is the issue of performance. While performance is very dependent on the computer the game runs on, we noticed some frames dropped on the current version of the game. A situation that could get worse as we add more and more logic and objects on each game scene. Thus we will have to pay special attention to the related parts of the program to make sure the game still runs smoothly in the future.

Some questions arose also about the exact satisfaction of the customer requirements. Will we fill the requirements exactly how they are presented, or change them slightly to create a better game?

2.2.3 What we learned

Compared to Sprint 1, the learnings were much lighter this time around. Our skills in both Processing and Agilefant continued improving during the sprint. Additionally, we learned that we need to better take into account the future directions of the game.

2.2.4 Moving on

While some features are best left to be implemented later, it is sometimes useful to prepare the road for future work. Some big architecture changes on this sprint should really ease the implementation of some of Sprint 3's user stories.

However, more architecture changes will probably be needed later on.

2.3 Sprint 3

This sprint focused on further refining the game, implementing almost all of the remaining user stories. Extra thought was put into our git workflow and the definition of done.

Git workflow We are using a centralized workflow. This means we only have one master branch, which developers clone and commit to it. Centralized workflow is simple for the developers as long as conflicts are avoided. Since this is a small project and everyone is usually working at different times

and different features, these problems rarely occur. Additionally, new features are simple enough that there is no need to commit non-working code and thus branches are generally not needed. Releases are marked with tags for Repolainen returns. Of course, some problems are possible: If two people try pushing at once, conflict happens. Also; always pull first, and then push. Resolve conflicts, if any. (Edit conflicts will arise if working on the same feature.)

Definition of Done Although definition of done is not static, we have produced a list of features which indicate a user story is "Done". First of all, the code is completed, as in, all of the features which were to be done are produced in code. The code is pushed into the version control system and tested generally with the already existing features. The code is documented if necessary for the rest of the team to utilize it. Naturally, this means the code builds without errors and works as depicted from the story. The whole team approves of the addition, and the task hours are up to date and the estimated time left is set to zero in Agilefant.

2.3.1 General feelings

A lot of time was spent discussing the features and the workflow. This pushed the actual implementation of the features quite late, but with teamwork everything is possible, and we managed to get them done just like we wanted in this sprint.

2.3.2 What we learned

We learned to use Agilefant even better than before, and to define our workflow and the "Done". Our skills in Processing improved further as we implemented new features to our game.

Meta-learning The learnings on sprints have largely been retrospective based on the things that went well and the things that did not go so well. The learnings have been discussed in the group to find the biggest problems in each sprint and how to improve the work in the future. We have so far managed to greatly improve our Agilefant performance based on the feedback we have received in each sprint as well as having a better grasp on the project in general.

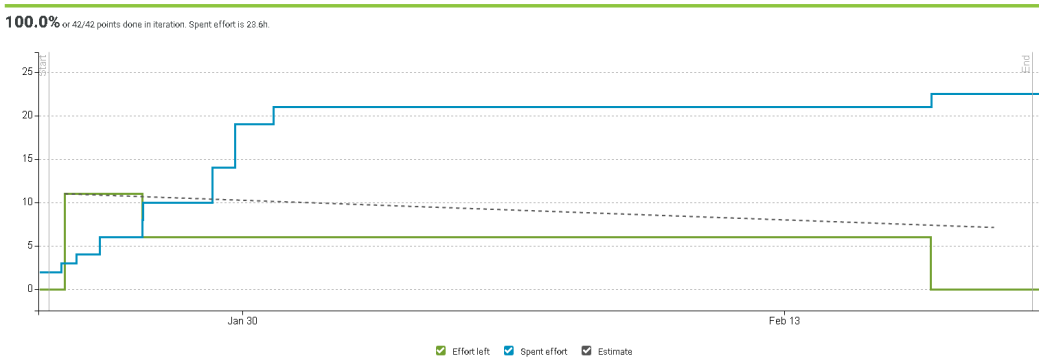
2.3.3 For the next sprint...

For the next sprint, we should again pay attention to distributing the work involving the decisions in the game and development process and the actual development more evenly with each other, and not one part first, second part later -style. We are continually improving our Agilefant workflow.

2.3.4 Burndown graphs comparison

After learning how to log our spent effort properly in sprint 3, our graphs became less jagged. Yes, Agilefant does not decrease remaining effort automatically. We have to do it by hand. And the moment we mark a task as done, all "effort left" on it is lost — reset to zero. Something to keep in mind for Sprint 4. *Log effort first, mark as done second.* But I digress; the following images depict how we enhanced our reporting technique:

Figure 1: At first, we were quite ”blocky”.



And we were finished with everything before the sprint even officially began.

Figure 2: Then we fine-grained the effort spent portion of the graph (blue).

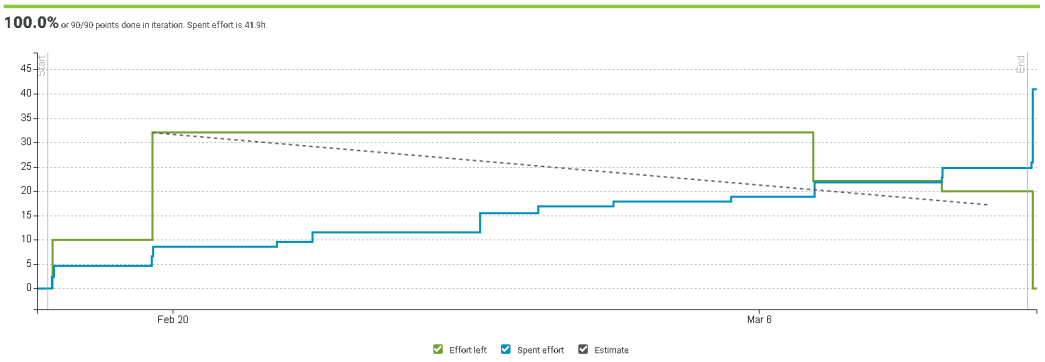


Figure 3: Finally, even the effort left portion is starting to look more detailed (green).

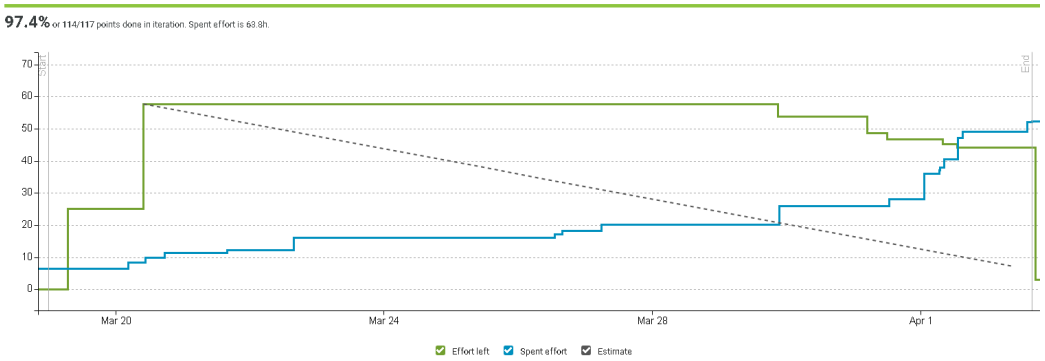
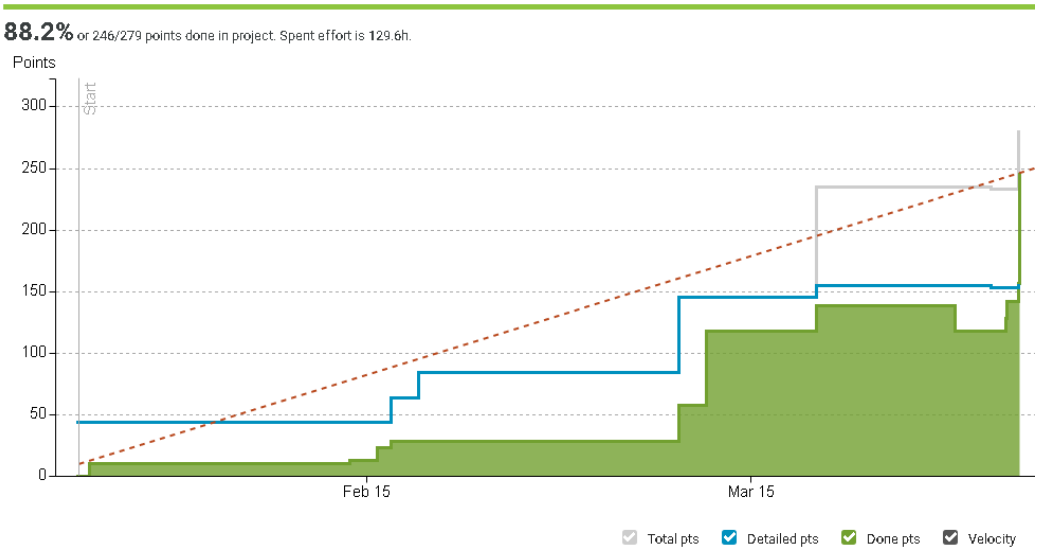


Figure 4: And here is an overview of the entire project. Significant improvement in Sprint 3 as we assign points to tasks more accurately.



3 Risk Management Plan

Table 3: Project risks

Risk ID	Description	Probability	Impact
P1	Short term absence	3	1
P2	Long term absence	2	1
P3	Someone drops out	2	2
T1	Someone force-pushes to Gitlab	2	1
T2	Processing turns out to be a very bad choice	1	3
C1	Customer changes requirements	3	2
M1	Divide between management and personnel	1	3

3.1 Personnel risks

3.1.1 Risk P1: short term absence of one person

Root cause: A member will be absent for several days.
Importance: Little importance, one of us can manage the project on their own anyway.
Avoidance: It would still be nice to warn the project members so that we don't rely on the concerned person to do any work.
Response: Someone else takes responsibility for the person's work.

3.1.2 Risk P2: long term absence of one person

Root cause: A member will be absent for several weeks.
Importance: Similarly to a short term absence, we can manage without one person for a prolonged period of time.
Avoidance: A warning will do.
Response: Someone else takes responsibility for the person's work.

3.1.3 Risk P3: someone drops out

Root cause: A member drops out of the course
Importance: This is slightly more impactful than a prolonged absence.
Response: We will have to reorganize the project around the three or fewer remaining members.

3.2 Technology risks

3.2.1 Risk T1: someone force-pushes to Gitlab

Root cause: Lack of knowledge in the technology brings someone to erase all progress on Gitlab.
Importance: Little importance, other members will have a copy of the project's history.
Response: Find out whoever has the most recent copy and push again.

3.2.2 Risk T2: Processing turns out to be a very bad choice

Root cause: As we iterate over the project, we realize Processing will severely hinder our progress moving forward.
Importance: Extremely unlikely given our constraints.

3.3 Customer risks

3.3.1 Risk C1: customer changes requirements

Root cause: The customer changes their mind on a part of the project.
Importance: Will depend on the size of the change.
Response: Create or modify user stories, rework and refactor the concerned parts of the project.

3.4 Management risks

3.4.1 Risk M1: divide between management and personnel

Root cause: The management and the personnel do not see eye-to-eye.
Importance: Very unlikely considering the management and the personnel are one and the same on this project.