Project plan+study diary

Cygnus 2022

version 1.7

TUT	Pervasive Computing	TIE-21106 Software Engineering Methodology
Author: Ignacio Laviña		Printed: 30.04.2017 22:39
Distribution: Team members, customer		
Document status: Done		Modified: 30.04.2017 22:39

VERSION HISTORY

Version	Date	Authors	Explanation (modifications)
1.0	29.01.2017	Ignacio L.	Initial version
1.1	10.02.2017	Victor G.	Added dependencies of stories
1.2	12.02.2017	Ignacio L.	Added review of sprint 1
1.3	06.03.2017	Victor G.	Corrections of sprint 1
1.4	12.03.2017	Ignacio L.	Added review of sprint 2
1.5	27.03.2017	Victor G.	Added version and configuration management
1.6	02.04.2017	Ignacio L.	Added review of sprint 3
1.7	30.04.2017	Victor G.	Added review of sprint 4

TABLE OF CONTENTS

1. PROJECT RESOURCES

On this part of the document, a brief presentation of the team background and skills is exposed, also describing the process of the work flow, tools and technologies used during the project.

1.1 Personnel

This chapter of the document exposes the team members and a short view of their capacities, roles and other documentation needed.

The team is composed by four members: Ignacio Laviña, Victor Garcia, Iaroslav Gridin, Likai Ren.

	Ignacio Laviña Faustmann (Product owner)
Contact	Ignacio.lavinafaustmann@student.tut.fi +34677804196
Experience	App developer and tester (February 2015 – April 2016)
Skills	Team work Active learning Java Creativity
Interests	New technologies, entrepreneurship

	Víctor García Zarco (Scrum master)
Contact	victor.garciazarco@student.tut.fi +34666740213
Experience	Project developer @ Demola Tampere (October 2016 – January 2017) Frontend & Backend lead developer @ Hightrack (August 2015 – February 2016) Talentum Startups @ Telefónica (December 2014 – May 2015)
Skills	Frontend development (HTML5, CSS3, Backbone.js)

	Backend development (Java, C, PHP) Photoshop Project management
Interests	New technologies, entrepreneurship

	Iaroslav Olegovich Gridin
Contact	iaroslav.gridin@student.tut.fi +358449165346
Experience	Self-employed freelancer (2009-2015) Research assistant @ TUT (March 2016 – December 2016)
Skills	Ruby on Rails backend development C++, C, Ruby, Haskell, Go
Interests	Data exchange networks, video games

	Likai Ren
Contact	likai.ren@student.tut.fi +358466143860
Experience	Web Designer Intern @EasyMarketing Finland Oy Ab (July 2015-September 2015)
Skills	Programming Language: Python, Java, JavaScript, C++, C, C#, PHP; Framework: Flask, Django, React, Bootstrap
Interests	Web development; Ethical hacking

1.1.1 Estimated contribution

Ignacio Laviña Faustmann	20h/sprint
Víctor García Zarco	20h/sprint
Iaroslav Gridin	20h/sprint
Likai Ren	20h/sprint

1.1.2 Team's absence

Ignacio Laviña Faustmann	27 February – 4 March
Víctor García Zarco	26 – 29 Jan, 27 Feb – 5 Mar
Iaroslav Gridin	
Likai Ren	

1.2 Process description

The team members will have a meeting before each sprint to review the previous sprint, define goals and achievements, define the next sprint and split the task and work according to the Agilefant plan.

During the process, there will be active communication between the team members through slack, and other ways if it's necessary. The team members are committed to be active in communication, and ask for others feedback or help if it becomes necessary.

The individual tasks are defined before each sprit in Agilefant, always with the team agreement. previous agreement. Some task will require a group meeting for developing together and solving problems.

1.3 Tools and technologies

Table 1.1: Tools used in the project.

Purpose	**Tool**	**Contact person**	**version**
Documentation	MS Word (word processing) office.microsoft.com		2015
	ArgoUML (UML tool) http://argouml.tigris.org/	V.G.Z	16.9
Communication	Slack http://slack.com		2.3.4
	Outlook https://outlook.live.com		2017
	Mutt http://www.mutt.org/		20170113 (1.7.2)
Version management	Git https://git-scm.com	V.G.Z	2.11.0
	GitLab https://gitlab.rd.tut.fi/		2017
Project management	Agilefant https://www.agilefant.com/	I.L.F	2017

	One Drive https://onedrive.live.com		2017
Development	Processing https://processing.org/		3.2.3
	ControlP5 library http://www.sojamo.de/libraries/controlP5/		2.2.6

As the duration of the project is less than 5 months if there is one new version of one tool/software we will ignore it, continuing with the current version (unless that version fixes security or important problems). The short duration of the project shouldn't be a problem for different versions.

Version control repository
The repository of the project is hosted in GitLab. The team have full access to it, while the customer will have only access to the *master* branch. Here, the customer will find the latest working version of the project.

Agilefant
Project management is done using Agilefant. Customer requirements are made into user stories, then they are converted to backlogs and distributed between sprints based on difficulty, dependencies and value. Then tasks based on backlogs are distributed among team members based on their capabilities and preferences. Team velocity is tracked and allows better time allocation in future.

Processing
The main development work is done by Processing, which is an integrated development environment (IDE) and a programming language for visual arts. Processing is open source and free to use in multiple platform, including Linux, Mac OS X and Windows. Processing can be used to create interactive programs with 2D, 3D or PDF output. With OpenGL integrated for accelerated 2D and 3D, Processing even has more than 100 libraries extending the core software.

1.4 SPRINT BACKLOGS

After analyzing the requirements given by the customer, some user stories have been made to manage them easily. Also, as Processing is a new technology for the team, the learning curve will increase exponentially from the beginning (personnel with experience in different fields). Because of that, the first phases of the projects will contain less workload than the final ones.

The picture below shows the dependencies between the different user stories. These dependencies are required to know when a new story can be started (based on the previous ones).

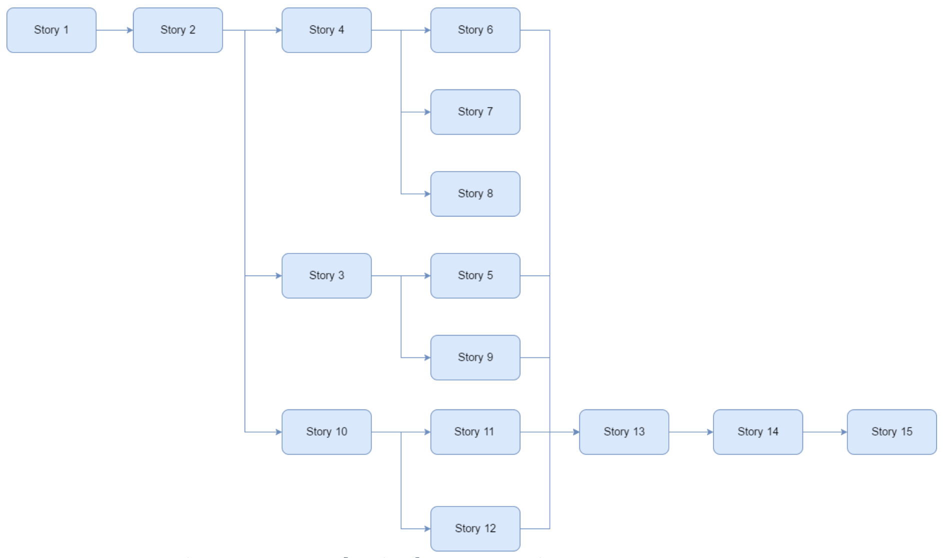


Figure 1 - Dependencies between stories

NOTE: More detailed information about the stories (with the linked requirements and tasks) can be found in Agilefant.

2. STUDY DIARY

2.1 Sprint 1

2.1.1 What went well

In this first sprint there were many things that went well for the team:

The group started move on, the game has the welcome screen and part of the story game. Also the user can introduce the name that would be used for the story and the score of the game.

All the user stories were submited to agilefant and the work for the rest of the project was splited in the different sprints to have an initial idea about the work distribution.

Some creatives ideas come to the story of the game, and great drawings were implemented for the stories.

The team had continuos communication through the communication channels.

2.1.2 What difficulties you had

The main difficulties were related to the processing learning, despite the team has coding skills, when facing a new programming environment some difficulties appear.

Thanks to some examples provided on the own processing environment and other tutorials found on the internet the team started learning and developing the first tasks.

2.1.3 What were the main learnings

During this sprint the team has achieved some learnings related to:

Scrum methodology: In contrast to the task defined in the first sprint, on the next sprints the task are much more specific and concretes. So splitting the work will be easier for the next parts of the project.

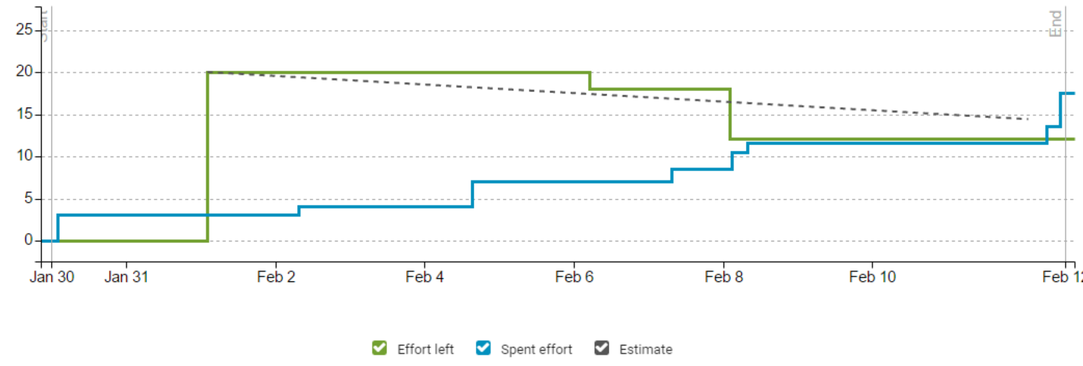
Git methodology: A part of the team wasn't used to Git methodology, so thanks to the team support and some tutorials all the team has now the knowledge of Git.

Processing: Learning processing while developing the game is something that will appear during all the parts of the project, as no one of the members had worked in processing before.

2.1.4 What did you decide to change for the next sprint

As is said above, for the next sprint the task are more specific, so will be easier to split the work and define responsibilities.

2.1.5 Burndown analysis



The burndown graphic shows the evolution of the sprint in terms of productivity. At the beginning, the team forgot to update the spent time so the effort left was not modified while the spent one was increasing. The distribution of the graphic is balanced, having at the end more work than the expected.

2.2 Sprint 2

2.2.1 What went well

During this Sprint we decided to develop fifteen user stories, the increase is quite significant from the first Sprint. This was possible thanks to the individual learning time that we took on the first Sprint. This made us more confident to develop much more stories and tasks.

In this Sprint we were able to include the map, the jet, enemies, islands and fuel repostages, and also implement the movement of all of this elements. This was great because is the main part of the game, and now the game has most of the difficult elements that we have to implement.

This also generates in the team sensation of success and motivates us to keep learning and developing

2.2.2 What difficulties you had

While it is true that we manage to implement the most diffcult parts, we found some problems that we needed to solve:

All the new elements included needed interaction between them, splittig the taks made neccessay to explain each part of the implementation in order to facilite the others members interact with all the parts.

When merging the individual branch into the main branch on Git called development, some merge conflicts appeared and it took some time to solve all of them in a succesfull way.

Also, in the graffic is possible to see that the main effort of the Sprint is spent at the beginning and at the end of the Sprint.

The team didn't work on the project during the exams week, part of the team couldn't work because of a trip, as it was programmed in the Project resources/team abscence. Others because of other courses and exams.

Also it is possible to notice that in the next week after exams the team restablish the work flow on Thursday, four days before the deadline with the most time consuming tasks. This could be a bad time management of the team.

2.2.3 What were the main learnings

This Sprint has allowed the team learn much more about processing. New functionalities of processing where implemented in this part.

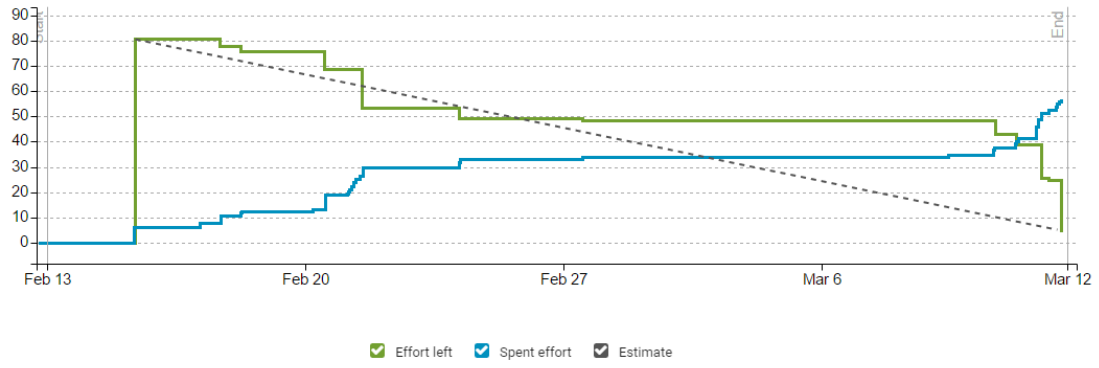
With the time, the team keep improving the team work methodology and the continuous communication through slack.

2.2.4 What did you decide to change for the next Sprint

For the next Sprint the team is going to take some time at the beginning to review the current code and give it a clear structure.

Also the team should try to redistribute the own way of spent effort in a more regular way, avoiding accumulate to much work on the end of the Sprint.

2.2.5 Burndown graphic



The burndown graphic shows the evolution of the project during the second sprint. At the begining, the results obtained were in line with the estimation of workload. Due to the exams week, after 27th of February there is a break in terms of work. Finally, the last days before the deadline the team had to work to finish on time, spending more effort than the estimated (due to some complications with the code). Also, there effort left at the end doesn't have a value of zero: there are two stories deferred, because the team agreed on to fix and refactor the current code before including new features.

2.3 Sprint 3

2.3.1 What went well

This Sprint was probably the most successful for the team. We accomplished a high number of stories, and we managed to refactor and solve some problems from the previous Sprint.

The main goals of this Sprint has been:

Collisions refactor: Now all the collisions work perfectly and the team has developed a new logic for the map and level creation.

Graphics: The game now has improved graphics, that creates a better game experience, and make all the different elements of the game more coherent in sense of graphics.

Defeating enemies: A new feature has been included according to the requirements, the jet can defeat enemies. Each time the player press the space bar, the jet shoot a rocket that can defeat the enemies.

To finalize, other important part of this Sprint was trying to clarify the continuous increment of code. We added comments and separated some part of the code in different functions.

2.3.2 What difficulties you had

In order to keep the proportion of all the graphics in any kind of screens, the team designed some functions that control the width and the height of the full screen and keep always the correct proportion. This create some little problems in the past Sprint with the collisions, that were not so much precise. In this Sprint we solved this problem and the map collisions that we couldn't in the past.

2.3.3 What were the main learnings

The main learning in this Sprint are related to the efficiency of the code. Until now we didn't wonder that much if the code was efficient enough if it was working. In this Sprint some changes has been made to make the code efficient for the computer.

Also the team has learnt how to work with the computer time, adding some real time delays in different functions, like timing between each shoot.

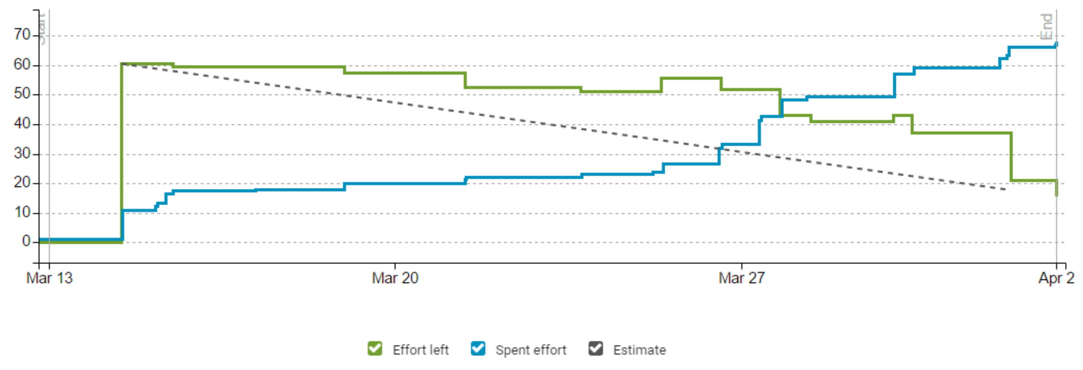
2.3.4 What did you decide to change for the next Sprint

At the end of this Sprint, all the main functionality and the most important details are done, so for the next sprint the team will try to adjust some values, like speed, fuel consumption or number of enemies to find the best play experience.

In addition to this, the customer has asked for a new unpredict requirement 'The enemies can shoot to the jet'. The first part of the last Sprint will be mainly for developing this task.

2.3.5 Burndown graphic

In this Sprint, the burndown graphic of the effort is the following:



The main part of the effort spent has been made in the second half of the Sprint. This happened because at the beginning of the Sprint, the Team focused on thinking solutions to solve the problems of the previous Sprint, and thinking a good algorithm and logic for the implementation of the map.

Once we had the logic, in the second part the team has spent more effort on the implementation and the problems that has been appearing during the process.

2.4 Sprint 4

2.4.1 What went well

The main goal of this sprint for the team was being able to finish the game succesfuly and following the course schedulle.

Thanks to the work done in the previous sprints, the team managed to accomplish almost all the initial requirements, so on the initial meeting of the sprint 4, we decided to include some new requirements to improve the game:

- Customer requirement 13: The enemy is shooting at the player. When certain hits are taken, the jet crashes.
- Customer requirement 14: " Possibility of mute the sound"
- Customer requirement 16: "The user can destroy the fuel depots by shooting them"
- Customer requirement 17: "The user can pause the game and resume it by pressing a button"
- Customer requirement 18: "The user can choose between different jets at the beginning of the game"
- Customer requirement 19: "Two players can play at the same time"

2.4.2 What difficulties you had

Since the first sprint, the number of lines of code and the complextity has increased a lot. This make the code more hard to maintain and find the errors.

While the continuous communication and work process followed for the team solved many of the possibles errors thet could appear, sometimes a little change of one part of the code could affect many of the other parts, and a few times was required to involve all the team to solve this little errors.

During the project the team has also tasks for coding refactor and review. Again, as the code was increasing, the reviews and refatoring were more complex and time consuming.

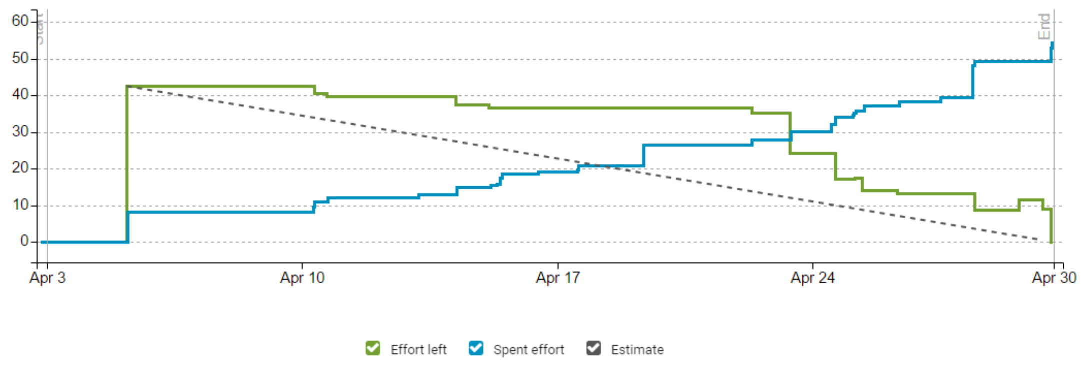
2.4.3 What were the main learnings

Including buttons and sound to the game bring to the project the necessity of learning new libraries and get familiar to them.

To make the game more realistic and for improving the player experience, some animations have been included when the jet is damaged or crash. This make necessary to get and control the time, in order to achieve the animation effect.

2.4.4 Burndown graphic

The burndown graphic shows the regular work that the team have done along all the sprint.



At some points the team has spent effort, but the left effort didn’t go down. This is explained because we estimated initial time for complete the minimun required task, but as we saw that we had time for making improvements, the improved task required a bit more time than just a simple task.

Another important point is that between the 15th and 22th of April, the effort left doesn’t change while the spent effort is increasing. This is because the team was trying to add new features but the results were not good, so the code and try outs were not merged.

2.5 General project overview

2.5.1 What went well

Along the project the team has working

Good distribution of the work, some sprints part of the team work more than other, because other projects etc. All the team agreed? With that.

Also the team has demostrated an active attitude, proposing ideas and sharing them on the weekly meetings or on the communication channels.

If any of the team members at some point had any problem, other member was allways ready to help or cover the task.

2.5.2 What difficulties you had

Along the project, the team has faced some difficulties that have been explained in each sprint.

Most of them were related to the increasing number classes, methods and lines of code that increase the difficulty of understand and solve any error when it appears. The goals set by the team were much further than the initial requirements, so even when the team agree with including more requirements and go further it brings also some coding difficulties and increase the possibility of new errors. In addition to this, the more lines of code and complexity make the manteinance a bit harder.

2.5.3 What were the main learnings

Could be noticed clearly how the team has been learning the scrum process and the use of the project tools.

At the first sprint the use of the Agilefant tool was not than efficient than in the followings sprints. The team learned how to use the tool properly and get used to follow the plan and spent correctly the effort.

The team now is able to make a better estimation of the time of the each task, including also the fact that we are now more skilled in coding than at the beginning.

Related to the tasks, it could be noticed how on the first sprint the tasks was more oppened and less detailed than at the end. Also the used more the part 'tasks without story' on Agilefant.

Also could be noticed a great change on the capability, skills and efficience of the team from the first sprint until the final of the last sprint, not just on the coding part, but also in all the aspects related to the project as communication, problem managemet and team work between others.

At any point the team have had any problems related to the communication, we used slack channels for all the issues related to this. But it could be said that thanks to the continuos use of the slack we manage to be much more efficients and solve many problems at early stages. The team is very aware of the importance of the communication.

3. RISK MANAGEMENT PLAN

The ID of the risks is defined by the pattern XY, where:

- X refers to the category of the risk.
    - P: Project management
    - T: Technologies
    - C: Customer
    - E: Environment
    - Pe: Personnel

Table 4.1: Project risks.

Risk ID	Description	Proba-bility	Im-pact
P1	Bad scheduling of the project/sprints	2	3
T1	Online tool not available	2	2
T2	Learning of new technologies	1	2
T3	HW problems with the equipment	1	2
C1	Bad communication with the customer	3	2
C2	Unclear requirements	3	2
C3	Number of requirements increased	2	1
E1	External attack to own systems	1	3
E2	Internet connection lost	1	1
Pe1	Short term absence	2	3
Pe2	Long term absence	1	3
Pe3	Bad communication within the team	1	2
Pe4	Overload of work	1	1
Pe5	Change of job (leaving the team)	1	2

3.1 Project management risks

3.1.1 Risk P1: Bad scheduling of the project/sprints

Symptom, early warning sign: not enough time to finish on time.
Source or reason: bad scheduling of the times of the project/sprints due to historical data.
Probability: 2 medium (on scale 1-3)
Seriousness: 3 high (on scale 1-3)
How to avoid: think well about what is able to do each member of the team in the scheduled time.
How to prevent: reorganize the tasks and workload for each member.
How to survive: good relationship with the customer, allowing small changes on the plan.

3.2 Technology risks

3.2.1 Risk T1: Online tool not available

Symptom, early warning sign: delays or no access to the online tool.
Source or reason: external problem with the provider.
Probability: 2 medium (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)
How to avoid: selection of the best tool provider.
How to prevent: premium accounts use to have preferences for this situation.
How to survive: existing alternative to work (other online platform or offline work).

3.2.2 Risk T2: Learning of new technologies

Symptom, early warning sign: the speed of the users working is not efficient.
Source or reason: the team doesn't know the new technology.
Probability: 1 low (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)
How to avoid: provide a solution for the customer with known technologies.
How to prevent: give all the necessary information to improve the skills with the technology.
How to survive: temporary hiring of a new member who knows the technology.

3.2.3 Risk T3: HW problems with the equipment

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: 1 low (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.

3.3 Customer risks

3.3.1 Risk C1: Bad communication with the customer

Symptom, early warning sign: the customer doesn't receive the product expected.
Source or reason: lack of communication with the client
Probability: 3 high (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)
How to avoid: define regular meetings and ways of communication.
How to prevent: increase the number of meetings with the customer.
How to survive: redefine the ways of communication and have a meeting with the customer to review all the work.

3.3.2 Risk C2: Unclear requirements

Symptom, early warning sign: the customer is not receiving the product expected.
Source or reason: bad or lack of communication with the customer. Maybe the customer doesn't really know what he wants.
Probability: 3 high (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)
How to avoid: define regular meetings and help the customer to decide.
How to prevent: stop the project and redefine the requirements.
How to survive: meeting to redefine the requirements and the project.

3.3.3 Risk C3: Number of requirements increased

Symptom, early warning sign: the customer asks for new functionalities.
Source or reason: the preferences of the customer have changed-
Probability: 2 medium (on scale 1-3)
Seriousness: 1 low (on scale 1-3)
How to avoid: closed budget and requirements list before starting the project.
How to prevent: meeting with the customer to decide if it is possible to increase the number of the requirements.
How to survive: re-schedule the work.

3.4 Environment risks

3.4.1 Risk E1: External attack to own systems

Symptom, early warning sign: alarms in the firewall and other systems.
Source or reason: external attack (DDoS, for example)
Probability: 1 low (on scale 1-3)
Seriousness: 3 high (on scale 1-3)
How to avoid: define and implement a good security plan
How to prevent: isolate the infected device
How to survive: backups of all the important data.

3.4.2 Risk E2: Internet connection lost

Symptom, early warning sign: delays and lack of connectivity to internet.
Source or reason: external attack, failure in internal network, failure in network cards.
Probability: 1 low (on scale 1-3)
Seriousness: 1 low (on scale 1-3)
How to avoid: review of systems and devices. Reliable network provider.
How to prevent: identify the problem (HW or network provider) and look for an alternative.
How to survive: possibility of working in offline mode.

3.5 Personnel risks

3.5.1 Risk Pe1: Short term absence

Symptom, early warning sign: one team member is missing for some days.
Source or reason: illness, personal reasons
Probability: 2 medium (on scale 1-3)
Seriousness: 3 high (on scale 1-3)
How to avoid: defined schedule and days when the members are not available.
How to prevent: to have access to the data of the member (other one can replace him easily).
How to survive: re-distribute the workload between the rest of the member, giving incentives for doing it.

3.5.2 Risk Pe2: Long term absence

Symptom, early warning sign: one team member is missing for a lot of time.
Source or reason: illness, personal reasons.
Probability: 1 low (on scale 1-3)
Seriousness: 3 high (on scale 1-3)

How to avoid: defined schedule and days when the members are not available.
How to prevent: to have access to the data of the member (other one can replace him easily).
How to survive: re-distribute the workload between the rest of the member, giving incentives for doing it.

3.5.3 Risk Pe3: Bad communication within the team

Symptom, early warning sign: the team is not synchronized in the work.
Source or reason: lack of communication, bad relationships.
Probability: 1 low (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)
How to avoid: define ways of communication within the team. Promote good relationships with different activities.
How to prevent: meeting of the team to update the information of everybody.
How to survive: meeting of the team to solve the problem.

3.5.4 Risk Pe4: Task overload

Symptom, early warning sign: the team is not accomplishing the deadlines.
Source or reason: bad distribution of the work
Probability: 1 low (on scale 1-3)
Seriousness: 1 low (on scale 1-3)
How to avoid: necessary to know the capabilities of each member of the team.
How to prevent: redistribute the workload for that member.
How to survive: redistribute the excess of workload between the rest of the team.

3.5.5 Risk Pe5: Change of job

Symptom, early warning sign: one member of the team receive offers from different companies
Source or reason: bad conditions in the current job (or worse than the new offered)
Probability: 1 low (on scale 1-3)
Seriousness: 2 medium (on scale 1-3)
How to avoid: good treat to the employees with the best conditions as possible.
How to prevent: re-negotiation of the current conditions.
How to survive: replace the employee as soon as possible. If not, redistribute the workload between the rest of the team.

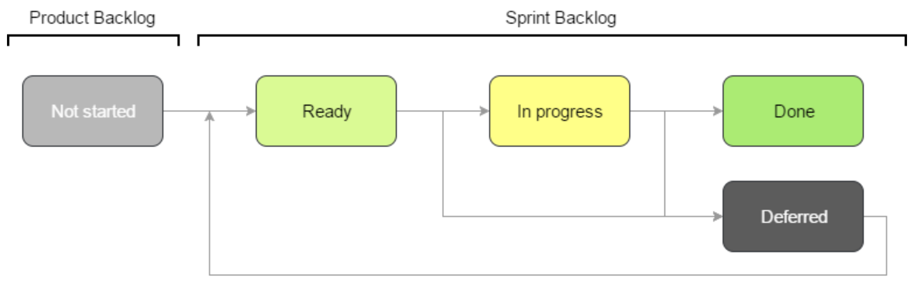
4. VERSION AND CONFIGURATION MANAGEMENT

This chapter resumes the version and configuration management of the project. It contains first, the management of the different states of the tasks and stories included in Agilefant. After that, the workflow used with Git is described.

4.1 States management

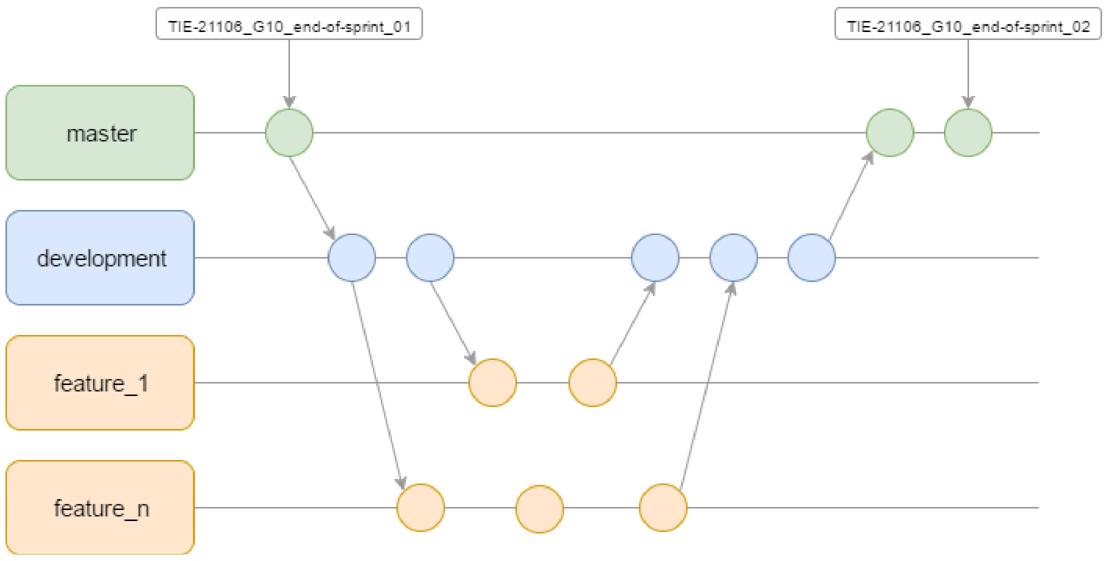
The tasks of the projects are managed using Agilefant. In the platform, all the user stories are included with the specific tasks for each one.
The state that each task and story can have are described below:

- **Not started.** All the tasks and stories included in the platform have this state by default.
- **Ready.** This state determines which tasks and stories are ready to be implemented in the current sprint. This reflects in the product backlog which stories are ready to be developed.
- **In progress.** When an user starts to perform one task or develop one story, the state of this one is changed to ”In progress”. This shows that the task or story is on work.
- **Done.** When a task or story is completed, it is marked as done. As the team has a really active communication, it is not necessary to set a state which shows that the task or story is completed but still need the supervision of the rest of the team to be integrated with the application. When an user completes one task or story, he notifies the rest of the team. After receiving a good feedback, the task is mark as done.
- **Deferred.** The tasks or stories that finally are not going to be completed during the current sprint, are marked as deferred so they will be done the next sprint.



4.2 Git workflow

The graphic below describes the workflow used by the team when working with git. It is a simplified version of Gitflow, because the project and the team are small.


TIE-21106_G10_end-of-sprint_01
TIE-21106_G10_end-of-sprint_02
master
development
feature_1
feature_n

The branches and tags are also created following a specific process described below-

4.2.1 Branches

The team is working following a simple structure of branches:

- **Master.** It contains the versions of the application that have been submitted. In other works, it contains the lastest working version of the code. In this branch, only modifications of the document are allowed (not modifications of code).
- **Development.** In this branch, all the new development for each sprint will be included. It is the main development space, so all the new characteristics and features developed will merged to this branch. At the end of the sprint, this branch will be merged into master.
- **Feature.** For each feature (task or story) that is developed in each sprint, the person responsible creates a new branch from development to work only on that feature. At the end of its development, if the feature is working as expected or the task has been completed, it will be merged into development.

Following this workflow, the team guarantee that the latest version of the working code will be always on development. Also, if there is any problem when developing a new feature or task, as it is separated by its own branch there won't be problems with the rest of the code in development branch (the rest of the users can continue working with their tasks).

Note: As most of the times fast-forward merging is used, it can be that the graphic of branches sometimes shows commits included directly on development (instead of showing a branch of feature) if there are no commits in between.

4.2.2 Tags

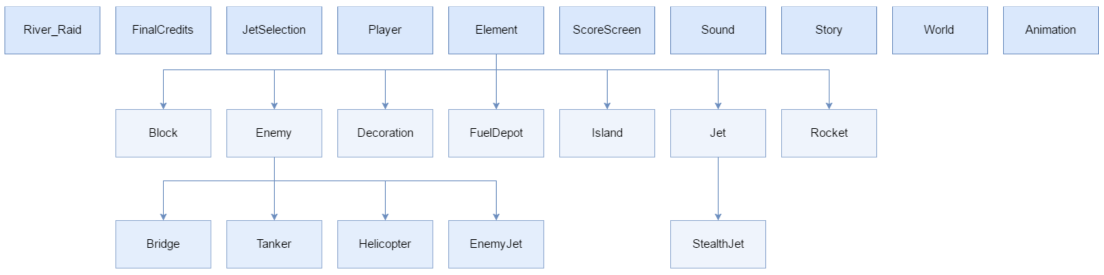
The creation of tags is related with the end of each sprint. As we need to define a specific tag for the submission of the project, we are using this tags to refer to the latest version of the working code delivered.
The structure of the tags is:

```
TIE-21106_G10_end-of-sprint_0X
```

- **TIE-21106** refers to the course code.
- **G10** refers to the group number.
- **end-of-sprint_0X** , where X refers to the number of the sprint

5. SOFTWARE ARCHITECTURE

The game is divided into some classes, trying to keep independance between the different entities. The graphic below shows the class structure and hierarchy:



The structure shows the different hierarchy between some classes. This allows to extend new functionalities based on the current code.

Also, it is important to mention the class River_Raid. It is the main class of the game and it is the controller of the game. Here, the setup of the different entities and values is done. The drawing functionality for the graphics is controlled also by this class, which is responsible of calling the other class when an element is required on the screen.

5.1 Maintainability and scalability

One example of this extension can be seen with the enemies. Extending from the class Enemy, it is possible to add new kind of enemies without spending too much effort.

The independency of classes is also a good practice for splitting responsibilities. One example of this is about the Story. It can be changed without modifying the rest of the objects.

The team has tried to take care of these things for ensure a good maintainability of the code and potential scalability with new features. At the beginning (first spring), everything was included in only one class (River_Raid). But at the end of the sprint, the team noticed that this was impossible to maintain because the game was going to increase exponentially in terms of lines of code. Because of that we decided to define the different objects necessary for the game, including at the same time the relations of hierarchy needed.

6. QUALITY ASSURANCE

During the first sprint, the team worked on one simple class called River Raid (as it is said in the point above). Everything was merged in that class, without the feedback of the rest of the team. A lot of conflicts and new bugs were appearing very often.

Reggarding this, the team decided to follow a methodology of work to ensure the code quality. Each time one important feature was going to be included in the code, it was tested on a separated branch on git by the team members. After that, the feedback was given to the main responsible of that feature who made the oportunes changes. It was in that moment with the team approbation when it was merged in the development branch.

On the second and third sprints, it was necessary to externalize some parts of the code to other new classes, in order to maintain the main class as cleaner as possible. At the end of each sprint and the beginning of the next one, the team spent some time on refactoring the code and makings reviews.

Thanks to the code reviews, the team obtain some feedback for improvements about the quality of the code. Some examples where:

- Name of the variables. At some points it was necessary to change the names of the variables for others that are more cohesionate with their function.

- Different class. Externalize the code of the main class in other sublcasses, acording acording to the software architecture explained before.

- Comments. Thanks to the code review, the team agreed to include more comments, to make the review easier and help to understand the code.

- Game logic. The team meetings and the code review, gave as result at some points of the project changes on the game logic. For instance, the way of creating the map was completely redesigned, at the beginning the team agreed that the efficience and quality of the code part related to the map didn't achieve the minimum expected quality. After that, the team created another new class called World with the expected quality.

- Code efficience. Thanks to the code tests with the task manager on the computer the team realised that the game was consuming to much memory and sources, after a code inspection we detected that there were some data that was being charged unnecesary many times each second (some images and sounds). The team solved this problem reducing the game memory consumption.