

# Software Engineering – specification

ANTOINE GRIMOD, ARNAUD DABY-SEESARAM, GABRIEL JANTET and ZIBO YANG

September 2021

## 1 Project introduction

The project is to build a online multiplayer Real Time Strategy game (RTS). In this project, we intend to write two main applications: a server and a client.

The RTS will take place in a fantasy world in which each player evolves. Their main goal is to build and continuously improve a village or small town, and raise and interact with other villages and towns. The players controls and influence the number of inhabitants, their professions and resources usage<sup>1</sup>.

The players will be able to interact with one another through:

**military campaigns** a player can raise an army and use it to attack another player

**trades** two players can agree on a commercial trade

## 2 Project management and technical choices

In order to keep track of the project advancement, we use a plain bare repository (with no web-GUI) available at `git.sofamaniac.xyz:rts.git`. The repository is accessible for reading to anyone (everyone can clone it using `git clone git://git.sofamaniac.xyz/rts`).

We will meet at least once a week during Software Engineering sessions and have a Discord channel in order to stay in touch during the week.

We will probably<sup>2</sup> use:

- Rust for the server-side
- Go for the client-side

None of us have ever used these languages in whole projects in the past, but we are willing to discover them. This said, if the group decide that one of the above language no longer suits the project aim or that another known language would be a better solution, we would switch to the latest.

---

<sup>1</sup>An inhabitant will cost resources. Thus, the player has to keep a balance between developing its place and increasing its size.

<sup>2</sup>We might have a change of heart after the first two weeks.

There exist static analysers for both these languages. We intend to try some of them on our code.

Each distinct implementation task will be developed in separate git-branches. These will be merged as soon as the features have been improved and stabilised. An implementation will be considered over if it matches its specification and passes the corresponding tests (which could be created for this implementation).

The final work will be packaged:

- in an archive for a manual installation
- for several GNU/Linux-based OS (at least ArchLinux, Debian and GuixOs will be included)

## 3 Game description

### 3.1 Species

Each player will choose a species for its character at the beginning of the game. They will choose among:

**elves** Elves have access to magic. They will be able to develop new powers, nurture them and use them. These powers would help the player kick start the rebuilding of its village at first (if any damage was to be caused by attacks), and eventually help the player defend the village, using defensive spells.

An elf player will start its constructions slowly, but would store up a capacity to start better in case of destruction, as magic would then kick start the player.

**humans** The humans will have access to technology. This technology would bring them stability as they will be able to both defend themselves from enemies and computerise their blue collar jobs such as mining or manufacturing clothes.

As for the elfic magic, the human technology would take time to unlock completely, as the use of a piece of machinery for a certain amount of time might unlock a new system.

**orcs** The orcs will be more resilient than the above species, allowing them to build heavier structures and faster than the other players. This strength will allow a player to start the game faster.

Nonetheless, the orcs will not have tools to automatise the defence or the manufactures, which means that such tasks will always monopolise inhabitants.

### 3.2 Town specification

This section aims to give specifics on the different building types and the way a player can unlock new and better ones. It will be divided in, four sections, three of which will cover the species specific structures a player can build and use.

### 3.2.1 Common structures

**town hall** to build any other structure and give order to the population (*i.e.* controlling how many citizen works in such and such area).

In case of an attack, if the town hall is destroyed, it needs to be rebuilt before any other building.

**mines, forests, ...** in order to gain resources, which are required to build the structures and possibly trade with other players.

At the beginning of the game, each player will discover basic resources and inhabitants from and with which build a town hall and start exploiting those resources.

The forest will slowly grow as long as there exists a living tree. Thus, the player needs to be careful on how they use the resources available.

### 3.2.2 Elf village specific structures

**Mana Tower** Automagically gather mana

**Academy of Magic** Allocating inhabitants there allows the player to unlock new spells / abilities

### 3.2.3 Human village specific structures

**Research centre** By allocating inhabitants to the research centre, the player can collect technopoints which can be used to unlock new technologies

**Automation centre ?** same as research centre but used to unlock automation upgrades (could be a building unlocked by research)

### 3.2.4 Orc village specific structures

**Orc breeding centre** Orcs inhabitants and soldier should be fast to produce, maybe add a building that automatically create an orc every few seconds

## 3.3 Military specification

This section aims to give specifics on the different soldier types and the way a player can unlock new and better ones. It will be divided in, four sections, three of which will cover the species specific military soldiers and structures a player can use.

### 3.3.1 Soldiers

We might specialize the units depending on their race, giving them bonuses and maluses For example for the first basic unit Men : att: +0; speed:+0; def:+0 Elves: att:+0; speed:+1; def:-1 Orcs: att:+1; speed: -1; def:+0

## 4 Other

The first tasks of the group is to define the elements of the game:

1. The different building types, professions, ... for the every day virtual life,
2. The specification of the army and its evolution,
3. The protocol to use and the repartition of the information between the server and the client *i.e.* decide what will the server and the client each store.

Each of these three points lead to two implementations: one on the server-side and one on the client-side.

The group members are requested to write and complete the documentation related to their code, which is a way to split the work and ensures the correction of what is written.

Constraints:

- Provide a specification of the project: explain the main goal of the project, present the tasks distribution within the team, the temporary schedule, ...
- Produce a documentation
- Provide testing tools
- Perform static analysis on the code
- Package the final work
- The controls should be fully customisable
- The client must be resource efficient

## 5 Planning

- October 1: Git tutorial
- October 5: Give our teachers an access to the git repository (which should contain the description / specification of the project: 5 pages)
- October 8: Presentation of the project to the other groups