# TDA367 Requirements Analysis Document

## Group 9

Smurfs vs. Gargamel is a strategic tower defense game inspired by Plants vs. Zombies, where players must strategically defend their smurf villages against waves of Gargamel's forces. Players place different smurf units, each with its own unique ability, to attack the advancing gargamels. The game combines resource management, strategic planning, and good action to create an engaging and challenging gameplay experience.

Here follows some of our epic user stories:

*As a Player, I want to be able to choose where I place Smurfs on the battlefield, because I need to strategically defend my base.*

*As a Player, I want an intuitive resource system used to deploy Smurfs, so I can manage resources effectively and prioritize which smurfs to buy.*

*As a Player, I want to enjoy an interactive UI with engaging graphics and sounds, so I feel immersed in the Smurf universe.*

Below is the domain model for Smurfs vs. Gargamel:

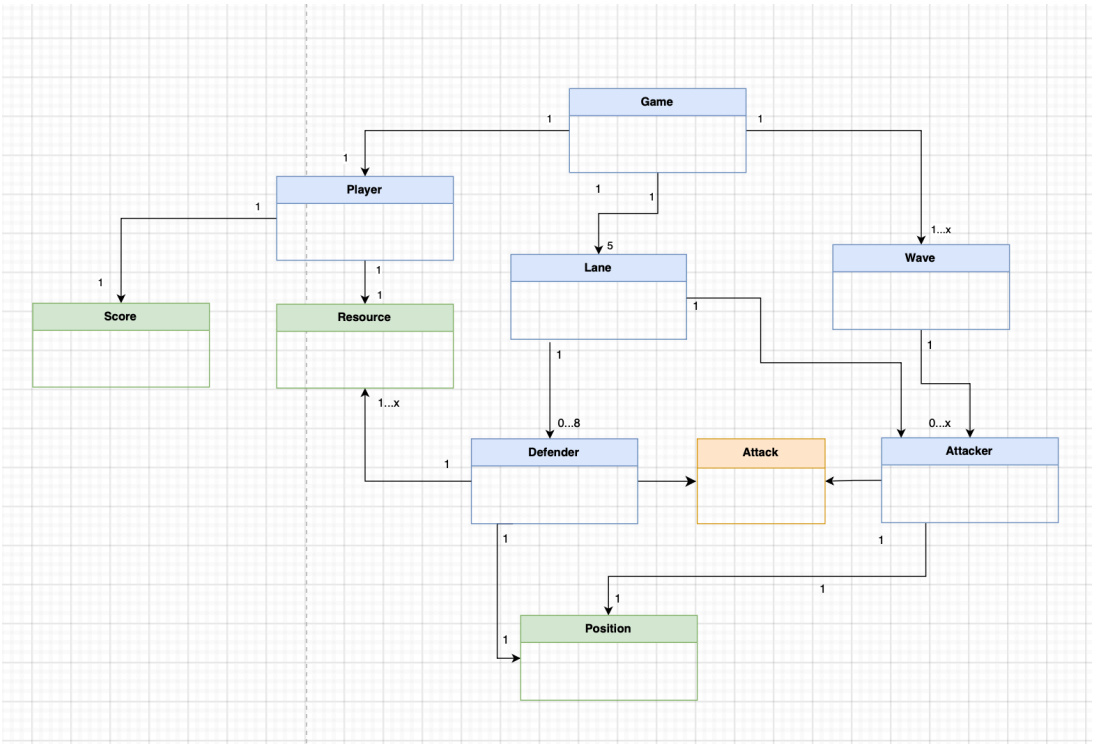


Figure 1: Domain model for Smurfs vs. Gargamel

Here is a mockup of the game, together with an early rendering:

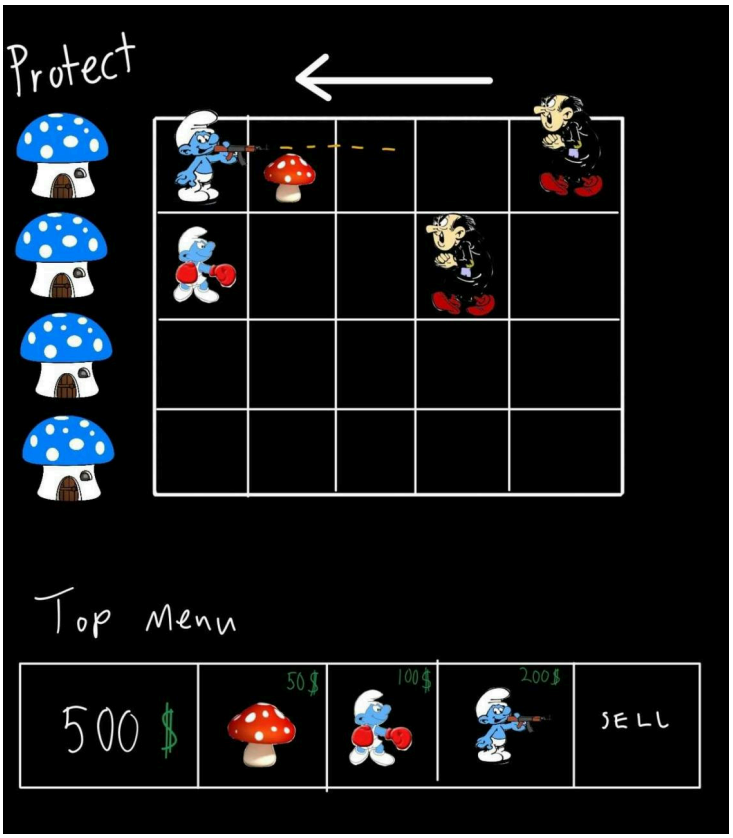


Figure 2: GUI mockup

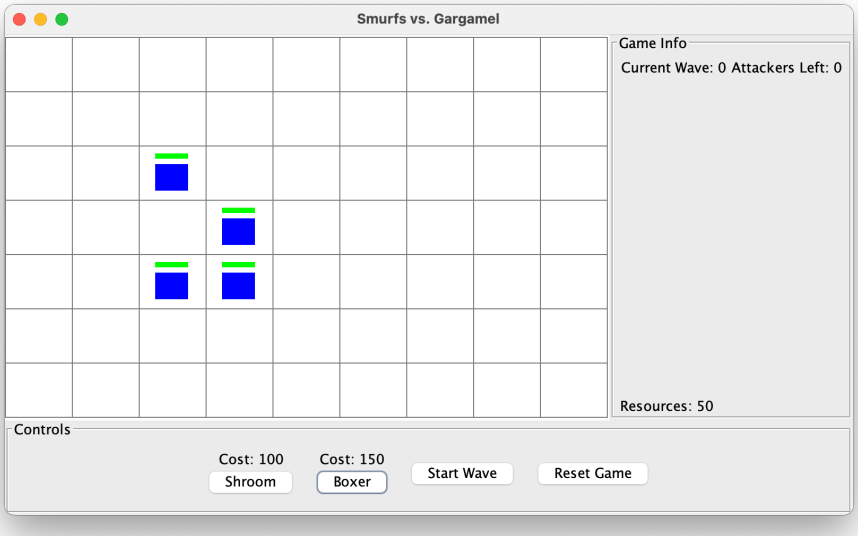


Figure 3: Early rendering of the game

## Requirements Overview

To define our requirements, we first start by defining the scope. We want to highlight these four key features of our game. The game should include:

- A grid-based board divided into multiple lanes.
- Defenders with ranged and melee attacks.
- Waves of attackers with varying properties and behaviours.
- Resource management and rewards for defeating attackers and completing waves.

The game only interacts with a user as its business context: The table below showcases this relation:

| Communication Partner | Inputs | Outputs |
|---|---|---|
| **Player (User)** | • Mouse input for placing defenders.<br>• Commands to start and reset the game.<br>• Selection of defender types | • Visual feedback for the game state.<br>• Resource updates.<br>• Notifications for winning and losing. |

Table 1: Business Context

There are also some technical interfaces, as the game is designed as a desktop-based application. Here follows the technical context:

| Channel | Inputs | Outputs |
|---|---|---|
| **Input Devices** | • Mouse and keyboard events for interaction. | • |
| **Display** | • | Graphical updates for game visuals. |

Table 2: Technical Context

Based on the our current user feedback, we represent the following stakeholders:

| Role | Contact | Expectations |
|---|---|---|
| **Product Owner** | stefan@gargasmurf.com | • A high-level overview of the architecture. |
| **Development Team** | devs@gargasmurf.com | • Access to detailed class diagrams and data flows. |
| **End Users** | support@gargasmurf.com | • A fun, engaging game, running smooth without unexpected behaviour. |

Now comes the main requirements needed by our game:

**Functional Requirements:**

| Type of Requirement | Requirements |
|---|---|
| **Game Setup** | • The game must have a board consisting of multiple lanes.<br>• Each lane must contain a number of grid cells.<br>• Each lane must track attackers. |
| **Defender Management** | • Defenders must be placeable on the grid at specific positions.<br>• Each defender must have attributes for health, attack damage, range, and cost. |
| **Attacker Management** | • Attackers must move forward in their respective lanes.<br>• Each attacker must have attributes for health, attack damage, speed, and a resource reward. |
| **Game Progression** | • The game must support waves of attackers with increasing difficulty.<br>• The game must end once an attacker reaches the end of a lane. |
| **Resource Management** | • The game must track resources available to the player. |

Table 3: Functional Requirements

**Non-Functional Requirements**

| Type of Requirement | Requirements |
|---|---|
| **Performance** | • The game must process updates at a consistant rate. |
| **Scalability** | • The game must support configurable lane and grid sizes. |
| **Maintainability** | • The game must adhere to object-oriented design, with responsibilities devided among all classes. |

Table 4: Non-Functional Requirements