Software Requirements Specification

“Strange Multiplayer Game”
version 1.0

Fabio Bove | 216219@studenti.unimore.it

1.0. Introduction

1.1. Specific Requests

All'esame deve essere mostrata l'applicazione in esecuzione e deve essere presentata una relazione che descrive il progetto realizzato e deve comprendere:

- *Descrizione dei requisiti, ed in particolare delle funzionalità messe a disposizione (ad es. tramite **SRS**);*
- *Descrizione dell'architettura (ad es. tramite diagramma a blocchi o **UML**);*
- *Descrizione dei **protocolli** usati (client-server o peer-to-peer, ad es. tramite diagrammi UML).*

1.2. Purpose

The purpose of this document is to present a detailed description of the system, purpose and features, interfaces of the "**Strange Multiplayer Game**".

What the system will do, the constraints under which it must operate and how the system will react to external stimuli.

1.2. Scope of Project

This software will be a **Strange Multiplayer Game**, where people around the world can connect and challenge their opponents in a nonsense fight.

Each ***player*** connects to the main ***server*** and is asked to provide an unique ***username*** to join the game. Players are identified by their names, stored in a queue.

A new game is started when two players join the server.

Whenever a player leaves the game is considered closed and its ***opponent*** who's still connected can start a new game when ready, maintaining its original username.

For a game: A long string (***long strange word***) is given to two players, and they need to find the exact number of ***occurrences*** for a given character - randomly picked up in the string. The first player who finds out the correct number wins, and can continue to play, the previous ***match*** is closed, and a new one is started with the first opponent available.

1.3. Glossary

Term	Definition
player	Person connected to the server.
username	Unique String that identifies a user or player
opponents	Two players that challenge each other in a given game
match	Consist of one round, each player needs to identify as fast as he can the occurrence of a given char in a string.
long strange word	String of n-chars (n can be set as the complexity of the game) randomly chosen and shuffled.

1.4. References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

2.0. Overall Description

2.1. System Environment

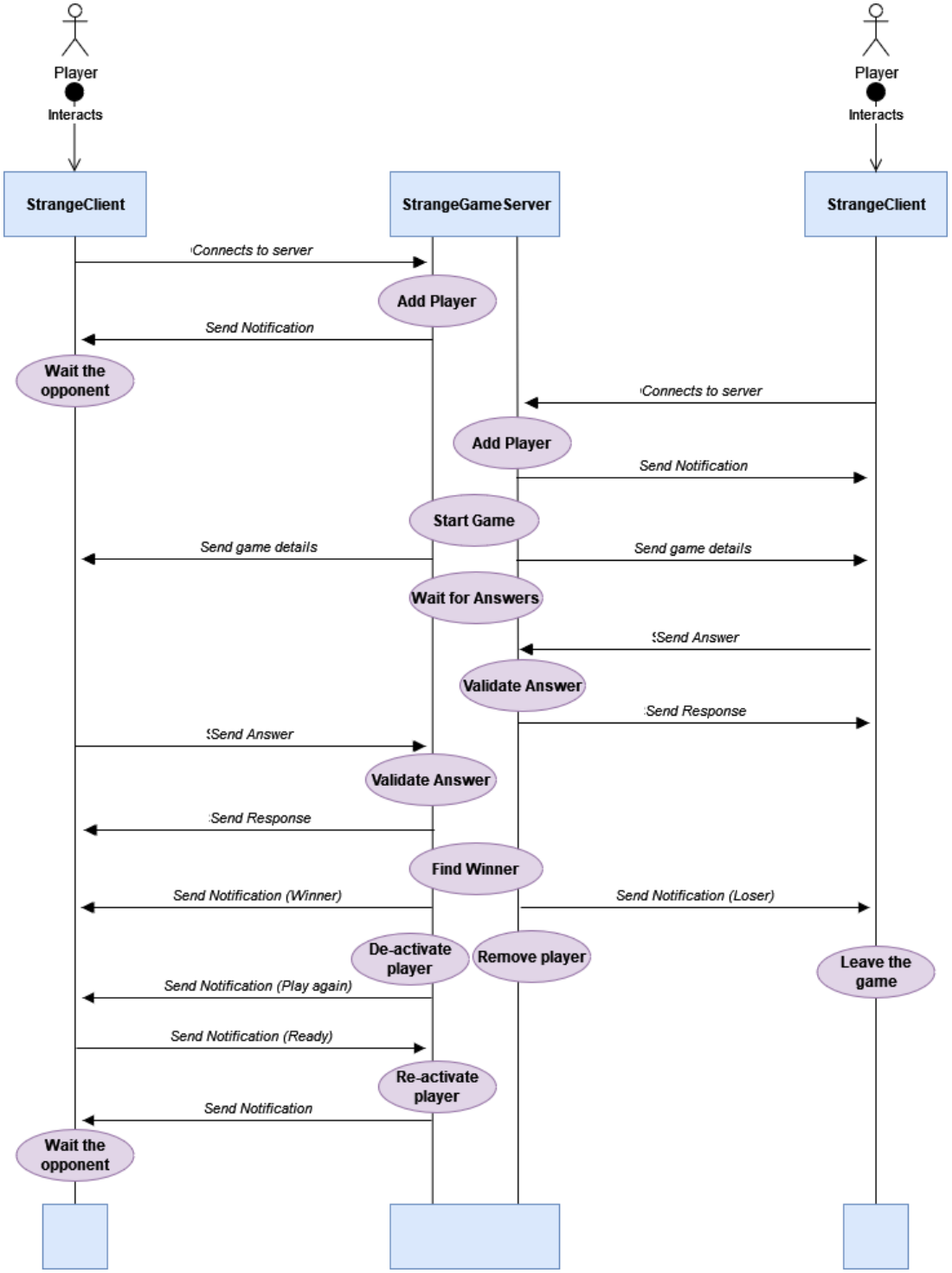

Figure 1 - *System Environment*

2.2. Functional Requirements

The system is based on a client - server structure organized as follows.

2.2.1. Client program

<class: StrangeClient> *in Client.py module*

This is the part of the system that sends requests to the server program. In our system a client identifies a player and can perform a set of actions:

- Connect to server
- Set its own name
- Search for games
- Send Answer to the server - as part of the game process
- Quit the game
- Join a new game

The client's code actually uses the remote objects by calling their methods. Thanks to **Pyro** we are enabled to build distributed object systems, with minimal effort, as it makes the use of remote objects (almost) transparent.

Anyways the Client has to perform some initialization and setup steps.

- Find the location identifier of the required object, by using the Pyro Name Server.

  ```
  client.connect_to_server(server_uri="PYRONAME:StrangeGameServer")
  ```

- Create a special object (*proxy)* that actually calls the remote object. Once it has this proxy object, the client can call it just as if it were a regular -local- Python object.

  ```
  self.server = Pyro4.Proxy(server_uri)
  ```

2.2.2 Server program

<class: StrangeGameServer> *in Server.py module*

The server stores the remote objects that are accessed by the clients. The server has to do several things:

- Create object instances (using Pyro) and give them names, register these with the Name Server
- Tell Pyro to sit idle in a loop waiting for incoming method calls, this is done with Deamon
    - A server just creates one of those and tells it to sit waiting for incoming requests. The Daemon takes care of everything from that moment on. Every server has one Daemon that knows about all the Pyro objects the server provides.

```
Pyro4.Daemon.serveSimple( # Finds the name server
      {StrangeGameServer: "StrangeGameServer"},
      ns = Pyro4.locateNS(),
      verbose = False
)
```

The Server provides the current functionalities:

- Allow player to register using a name - that should be unique
- Keeps track of active games and players
- Manages games:
    - Gets the details for a new game (Long strange word, Occurrences)
    - Check the validity of a player answer
    - Finds the winner of match
- Allow player to leave / start a new match

2.2.3. Finally - The Game

<class: StrangeGame> *in Game.py module*

This class allows you to create a “StrangeGame” object.
Every game has different details, such as:

- **Complexity** “k”, which is a parameter set when a new game is started, and is simply the number of characters that compose the “Long Strange Word”. The higher k (the string length) the harder the game.
- **Long Strange Word**, a string of k characters randomly chosen from an ascii set of uppercase letters and digits. The seed of the random choice is based on the game number and the time of its creation. Once generated, the word characters are shuffled.
- **Character to find**, is a character randomly chosen from the ones within the Long Strange Word.
- **Occurrences** of the characters to find. It's simply the count of the times that the chosen character appears in the Long Strange Word.

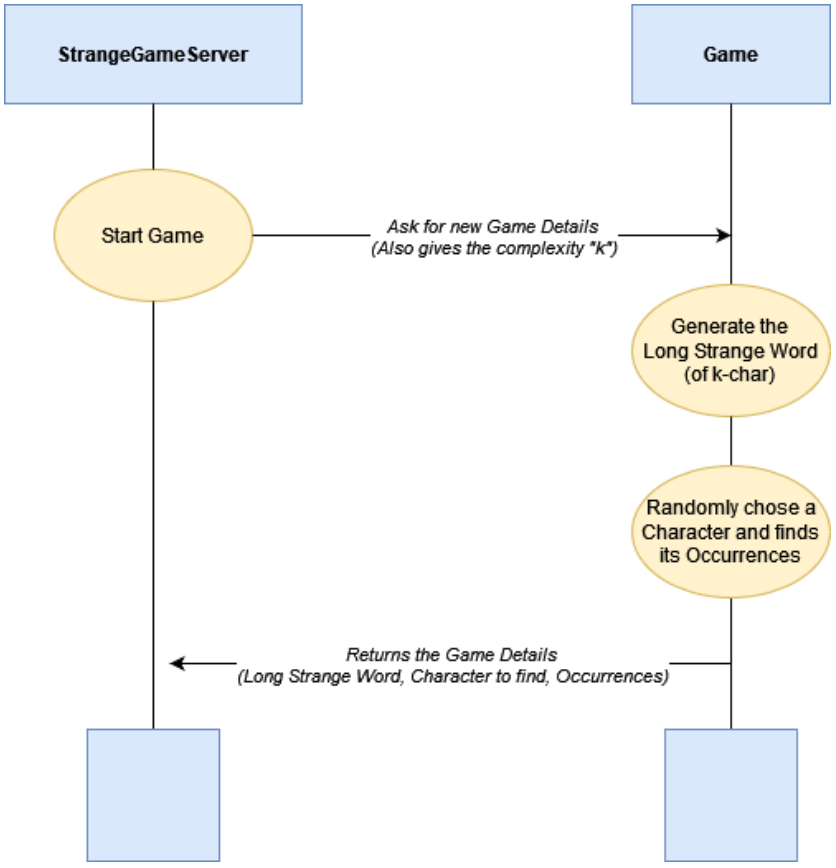

Figure 2 *- How a “StrangeGame” is created*

3.0 Software Requirements

Windows Operating System with python and pip, recommended versions:

- python 3.10.4
- pip 22.3.1

If you like you can set a virtual environment for the project - to create/activate one go to the main directory and type:.

```
>> virtualenv pad-venv

>> pad-venv\Scripts\activate
```

Then import all the needed packages from the **"requirements.txt"** file:

```
>> pip install -r requirements.txt
```

Below you can find the full list of the requirements for the “**StrangeGame**”

```
Package     Version
----------- -------
pip         22.3.1
Pyro4       4.82
pywin32     305
serpent     1.41
setuptools  65.5.1
wheel       0.38.4
```

4.0 References

GitHub | https://github.com/fabiobove-dr

Contacts | fabio.bove.dr@gmail.com - 216219@studenti.unimore.it