**System Design Document**

**<End-to-End C++ Application Development>**

**Nokia**

**team members:**

**emilian bogdan**

**radu borza**

**mihai budurean**

**roxana ciucioiu**

**Date**

**15 April 2022**

**Table of Contents**

INTRODUCTION ………………………………………………………………………………. 3

PURPOSE ……………………………………………………………………………………….. 3

SYSTEM OVERVIEW …………………………………………………………………………. 3

DESIGN CONSTRAINTS ……………………………………………………………………… 4

[ROLES](#_Toc36985638) AND RESPONSABILITIES ………………………………………………………… .. 4

PROJECT REFERENCES ……………………………………………………………………… 5

SYSTEM ARCHITECTURE …………………………………………………………………… 5

[Hardware and Software Detailed Design 5](#_Toc36985642)

# Introduction

The purpose of this document consists of the description of our team project, “End-to-End C++ Application Development”. For this theme, we have decided to implement Conway’s “Game of Life”, along with a system for automatic testing of the code. The Game of Life is a zero-player cellular automaton, meaning that its evolution is determined by its initial state, requiring no further input. The only interaction between an actor and the game is the creation of the initial state, followed by a simple observation of its evolution according to a predefined set of rules:

* We have an initial table made of a certain number of cells which can be either live or dead
* A live cell with less than two live neighbors will die
* A live cell with more than three live neighbors will die
* Any live cell with two or three live neighbors will live
* A dead cell with exactly three neighbors will become a live cell in the next generation

Our project consists of the C++ implementation of these rules, along with a testing system that will automatically verify whether the code runs correctly or not and it will be described in detail in the following chapters of this document.

# Purpose

The purpose of the System Design Document is, first of all, a better understanding of our project, “The Game of Life”, how it came to be, what is its purpose and what it can be used for. Through this document, we will reach a better understanding of what this game implies, its rules and the ways it can be implemented using the C++ programming language.

From an organizational point of view, this document will include a detailed description of our project, the elements it contains, the steps we need to follow in order to reach our final goal and different diagrams that will help us visualize the entire process.

# System Overview

As previously mentioned, our project consists of an automatically tested implementation of The Game of Life. The Game of Life is a zero-player cellular automaton, meaning that its evolution is determined by its initial state, requiring no further input. The only interaction between an actor and the game is the creation of the initial state, followed by a simple observation of its evolution according to a predefined set of rules:

* We have an initial table made of a certain number of cells which can be either live or dead
* A live cell with less than two live neighbors will die
* A live cell with more than three live neighbors will die
* Any live cell with two or three live neighbors will live
* A dead cell with exactly three neighbors will become a live cell in the next generation

The continuous implementation of these rules will lead to the appearance of different patterns. The most common ones are *still lifes* (they don’t change over the generations), *oscillators* (they return to their initial pattern after a finite number of generations) and *spaceships* (which translate themselves across the grid).

A picture containing crossword puzzle, text, indoor, shoji

Description automatically generated

A picture containing shoji, crossword puzzle, building, window

Description automatically generatedA picture containing shoji, crossword puzzle, tiled

Description automatically generated

*Still life Oscillator Spaceship*

To achieve these random simulations, we will create a C++ project in Visual Studio, followed by the automatic testing system, which will consist of a series of asserts created using *gtest.*

# Design Constraints

The only design constraint we managed to identify until now is the size of the matrix that will be used. This will be completely dependent on the IDE we will be using (Visual Studio). Therefore, we decided that our scope will only be covering a 500x500 matrix.

# Roles and Responsibilities

Ciucioiu Roxana – Team Leader, in charge of human resource administration and project documentation.

Budurean Mihai – responsible for the creation of the general functionalities of the Game of Life.

Borza Radu – responsible for the creation of additional functionalities for the Game of Life (such as the implementation of Gliders or other life forms).

Bogdan Emilian – responsible for the creation of the automatic testing system using gtest.

For the interface of the application and the bug fixing process, the entire team will work together.

# Project References

The first and most important reference we used is Conway’s ‘Game of Life’, as in the concept and rules that he created in 1970. The research on this subject has been done using mostly the information from the following webpage:

<https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life#Origins>

Another reference is the diagram created by the team that illustrates the main components of the code:

Application

Description automatically generated with low confidence

# System Architecture

The hardware requirements are very simple, the only thing needed being a basic computer.

The software requirements include an IDE, in this case Visual Studio, which will be used to write and run the code. After this, we will be using Gtest, a test framework, meaning a library for C++ programming and automated execution of test cases.

# Hardware and Software Detailed Design

As seen in the diagram above, the code will consist of the main class, GameOfLife, which has multiple functions that help with the creation of the initial map and then with its evolution. The *evolvingStage()* function will contain the application of the rules defined by Conway. This will mark the passing of every generation and will apply the changes to the cells according to the rules of the game. The set[…] functions will be used to create different life forms (patterns) that have been observed in the Game of Life in the past, the likes of which have been showed in the System Overview chapter. The associated class, Cell, will be used to keep the count of a cell’s number of neighbors and to verify whether the respective cell is alive or dead in the current generation.

After the implementation of the game, a testing system, consisting of a series of asserts, will be implemented using the Gtest framework in order to automatically check that the code is correct and the program runs smoothly.