|  |
| --- |
|  |
| Technical Analysis  Individual Project: Cellular automaton |
| Michał Szklarski  Computer Science 4 April 2016 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Document metric | | | | | |
| Project: | Cellular Automaton | | | **Company:** | WUT |
| Name: | Technical Analysis – Individual Project | | | | |
| Topics: | Technical specification of Cellular Automaton | | | | |
| Author: | Michał Szklarski | | | | |
| File: | Technical Analysis.docx | | | | |
| Version no: | 07 | **Status:** | FINAL | **Opening date:** | 2016-04-04 |
| Summary: | To extend and concretize requirements described in Business Analysis for the Cellular Automaton project. | | | | |
| Authorized by: |  | | | **Last modification date:** | 2016-04-07 |

|  |  |  |  |
| --- | --- | --- | --- |
| History of changes | | | |
| Version | **Date** | **Who** | **Description** |
| 01 | 2016-04-04 | Michał Szklarski | Initial version, definition |
| 02 | 2016-04-04 | Michał Szklarski | Added document metric and history of changes parts |
| 03 | 2016-04-05 | Michał Szklarski | Added summary – overview part. |
| 04 | 2016-04-05 | Michał Szklarski | Extended summary + general specification + conclusion |
| 05 | 2016-04-06 | Michał Szklarski | Technologies + methodology + dev. process flow |
| 06 | 2016-04-07 | Michał Szklarski | Similar solutions analysis |
| 07 | 2016-04-07 | Michał Szklarski | Algorithm and other elements description, diagrams, GUI |

# Technical Analysis

## Individual Project: Cellular automaton

## Technical specification

Table of Contents

[Technical Analysis 1](#_Toc447752567)

[Individual Project: Cellular automaton 1](#_Toc447752568)

[Technical specification 1](#_Toc447752569)

[Summary – overview 2](#_Toc447752570)

[Notion description 3](#_Toc447752571)

[General specification 3](#_Toc447752572)

[Technologies 3](#_Toc447752573)

[Methodology (development model) 3](#_Toc447752574)

[Development process flow 5](#_Toc447752575)

[Similar solutions analysis 5](#_Toc447752576)

[Algorithm description 6](#_Toc447752577)

[Other program elements and structure description 7](#_Toc447752578)

[State diagram 7](#_Toc447752579)

[Class diagram 7](#_Toc447752580)

[GUI prototype 7](#_Toc447752581)

[Conclusion 7](#_Toc447752582)

### Summary – overview

Aim of this document is to present requirements from the business analysis of a Cellular Automaton application as a technical specification concretization. All implementation details and decisions, such as chosen language, technologies, frameworks, libraries and algorithms will be described in this document. Document is divided into several parts, starting from general specification description, planned technologies, methodology description, development process flow, similar solutions analysis, algorithm description, other program elements description, diagrams (state, class), UI prototypes and finally: conclusion and last summary about this document.

This document opens path to the implementation process by giving exact directions and decisions, which will be followed by testing phase.

### 

### General specification

A Cellular Automation software, as described in previous document from the business point of view is a software emulating cellular life.

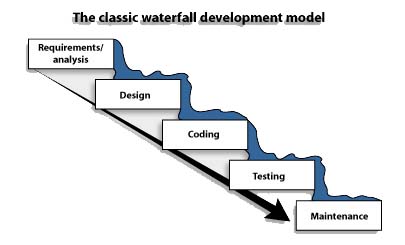
Generally speaking, program will be composed of three main parts: main grid with cells, menus, inputs and buttons (both on main window) for operations and custom rules editor as a separate window. Algorithm handling grid and rules check & enforce policies will be present in the background of whole solution.

### Technologies

As a best suited option for development of such project, **C#** programming language was chosen. Best choice for presentation layer was estimated as a **Windows Presentation Foundation** (WPF) solution. WPF provides a rich and stable API for simple and complex solution, as the one described in this document. It is also well documented, which is an important factor for development process. Main IDE for the project was appointed to be **Visual Studio 2015**, as it is most powerful C# development environment, with integrated **Visual Studio Unit Testing Framework**, that also will be uses here, for testing phase.

### Methodology (development model)

Business specification is finished now, chosen methodology for this project is **Waterfall**. Due to project complexity, clarity of the requirements, individuality, and course requirements described in initial presentation it is exactly the perfect solution for this assignment. As presented here:



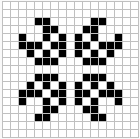
One modification to that diagram is such, that we won’t be handling maintenance phase. Everything up to Testing remains according to the official Waterfall methodology rules. Current phase is determined as Design on the diagram. Each stage has clearly defined goal therefore it is possible to efficiently control project flow, and detect any impediment.

### Development process flow

### Similar solutions analysis

As supposed, our project isn’t the pioneer one. There exist similar solutions, like, for example: Conway’s Life, Wireworld, Langton’s Ant, Brian’s Brain. Let’s discuss and analyze most popular two of them - Conway’s Game of Life and Langton’s Ant cellular automaton.

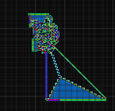
##### Conway’s Game of Life

 Conway's Game of Life is a cellular automaton invented by the British mathematician John Horton Conway in 1970. It implements infinite, two-dimensional and orthogonal grid of cells, each on it in two possible states: dead or alive. Neighborhood of interaction for one cell is set as an eight direct adjacent cells. For each iteration of lifecycle (a tick) following rules are applied:

* Any live cell with two or three live neighbors lives on to the next generation.
* Any live cell with fewer than two live neighbors dies, as if caused by under-population.
* Any live cell with more than three live neighbors dies, as if by over-population.
* Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

These rules are applied repeatedly in order to create further generations. Initial one is created by applying all of the rules on every cell field.

##### Langton’s Ant

 Langton’s Ant is a Cellular Automaton with a very simple set of rules but complex resulting behavior. It was invented by Chris Langton in 1986. It also implements two-dimensional grid of cells, each on it in two possible states: black or white. One cell is designated to be an “ant”, which can travel in any of the four basic directions (N, S, W, E) according to the rules:

* At a white square, turn 90° right, flip the color of the square, move forward one unit
* At a black square, turn 90° left, flip the color of the square, move forward one unit

These simple rules are proven to create complex behavior, described either as a simple, chaotic, or emerging order.

With similar solutions analyze completed, we can now focus on delivering our own one, knowing about advantages and disadvantages of other implementations.

### Algorithm description

### Other program elements and structure description

### State diagram

### Class diagram

### GUI prototype

### Conclusion

As described in this document, technical analysis is done. Following remaining part is to implement desired solution. All points considered here should be thoroughly translatable into chosen programming language. From there, development processes can move on into testing phase, which is very important in every such project.