28 March 2015

Lukasz Wójcik

Individual project
Cellular automaton
Technical project

Informations

Scheldue

Date	Stage
2015-03-12	requirement specification
2015-04-02	technical project
2015-04-23	code of modules
2015-04-30	version 0.98
2015-05-07	version 0.99
2015-05-14	version 1.0
2015-05-21	test report
2015-06-11	acceptation

Document metric

Document metric					
Project:	Cellular automaton		**Company:**	WUT	
Name:	Cellular Automata technical documentation				
Topics:	Technical aspect of project				
Author:	Lukasz Wójcik				
File:	cellular_automata_technical_documentation				
Version no:	0.3	**Status:**	Designing	**Opening date:**	2015-03-26
Summary:	Providing technical documentation.				
Authorized by:	Wladyslaw Homenda		**Last modification date:**		2015-04-08

Changes

History of changes			
Version	Date	Who	Description
0.1	2015-03-26	Lukasz Wójcik	Creation of technical documentation sections. Initial planning
0.2	2015-03-27	Lukasz Wójcik	Creation of technical documentation.
0.21	2015-03-27	Lukasz Wójcik	Correction of spelling mistakes.
0.3	2015-04-08	Lukasz Wójcik	Correction of algorithm for adding rules.

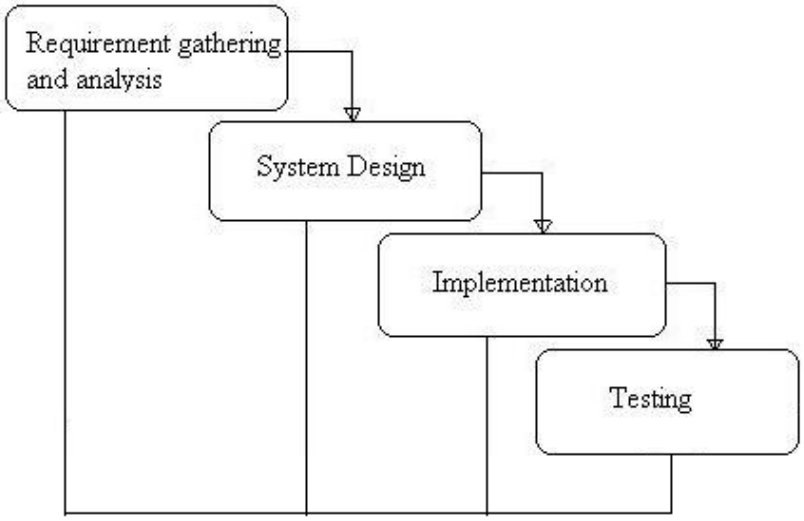
Contents

Short summary of documentation

This document is a technical documentation of Cellular automation program.

1 Production model

Methodology used for implementation is a "Waterfall Model"



Waterfall-model (In our case it end on testing)

This model is easy to understand and use. It is mostly used with relatively small project where all requirements are well understood (returning to earlier stages of project from e.g testing might be very expansive so it is better not to move to another stage without debugging). Each stage is completed one at a time so they are not overlapping.

2 Technologies

This application will be written in C# language.Visual Studio 2013 will be used for an app development. WPF (Windows Presentation Foundation) will be used as a graphic engine. WPF is based on .NET 3. It gives nice tools to divide project on GUI and logical parts what helps in creating clean code. This application is destined for Windows Vista and higher.

3 Used data structures

To store grid data we will use two nested lists of cells object(or just boolean).In case of N x M grid, a list storing other lists will have length of N and inner lists will have length of M (each inner list represents one column of grid). In C# exploring elements in List is as fast as in simple table but it gives way more tools. List is easy to expand/shrink and has many search functions already implemented.

4 Description of algorithms

- **Algorithm for defining a new rule**

I. Open "Add/Edit rule" window.

II. Click on "Add new" button to specify that a new rule will be added
"All subrules which are manually added to a rule set are determining that cell is alive. All other possible rules are making cell dead by default."

III. Click on cells in a neighbors grid to choose those which will be included in counting active neighbors count, which will be refereed to as an "important" cell (It is also possible to select or deselect all rules by clicking on "Select all/Deselect all" button).

IV. Chose a number from 0 to N (number of cells in neighborhood included in process of determining a state of a particular cell) to define how many "imporatant" cells have to be alive to change state of a cell.

V. Click on button indicating set of sub-rules to add to it a newly created sub-rule.

VI. Continue adding new sub-rules or click apply button to use newly created rule.

- **Algorithm for grid space wrapping**

  "Each cell in a matrix has its own index which represents its position in 2D space (coordinates [x,y]). In case where a cell from a neighborhood exceeds possible grid index we assume that top border and bottom border are connected and so are left and right border. After such assumptions we can imagine our matrix as a doughnut.All new values are saved to a copy of a grid."

  I. If coordinate X of a neighborhood cell exceeds matrix from right side, calculate the difference between X coordinate and grid width and set a X coordinate to the value of a difference.

  II. If coordinate X of a neighborhood cell exceeds matrix from left side, set a value of an X coordinate to a sum of its starting value and grid width.

  III. If coordinate Y of a neighborhood cell exceeds matrix from bottom side, calculate the difference between Y coordinate and grid height and set Y coordinate to the value of a difference.

  IV. If coordinate Y of a neighborhood cell exceeds matrix from the upper side, set a value of an y coordinate to a sum of its starting value and grid height.

  "Neighborhood cell may exceed both X and Y axis of the grid. In this situation both coordinates should be recalculated."

- **Algorithm for determining state of a cell by a rule.**

  I. Do following procedures until sub-rule for living cell will be found or until the and of sub-rule list:

     a) Iterate trough all cells in neighborhood and count a number of "important", living ones ("important" cells are defined by sub-rule).

     b) Check if obtained number is equal to the one expected by sub-rule.

     c) If obtained number is equal to needed value than return information that sub-rule for living cell can be applied and stop checking other sub-rules.

     d) If obtained number is not equal to needed value than check next sub-rule.

  II. If previous procedure returned information that there is a sub-rule which can be applied to a cell than change its state to living, otherwise change it to dead.

- **Algorithm for creating new generation.**

"For this procedure two matrices will be needed. One storing state values from current generation and second for a new generation. After computation of each generation they are switching places"

I. For each cell in a current generation matrix do following procedures:
   a) Calculate a new state of cell following algorithm determining a state of a cell.
   b) save a new state to a matrix storing states for a new generation

II. Set new generation matrix to a current generation matrix and redraw visualization of a grid.

5 Functionality

- Display visualization of cell movements - every step of components behavior must be shown on a grid.

- Change environment - by selecting them; there must be at least 3 types of environments: 4, 8, 24 points neighborhood.

- Introduce new rules - by adding new rules determining new behavior of cells.

- Change rules - by dynamically changing existing rules.

- Create new rules - by selecting neighborhood cells and how many of them should be alive in order to make the target cell either alive or dead.

- Select number of steps .

- Control simulation - by choosing options play, pause, stop or reset simulation.

- Observe movements step by step - by rewinding cells movement on click, depending on the selected number of steps.

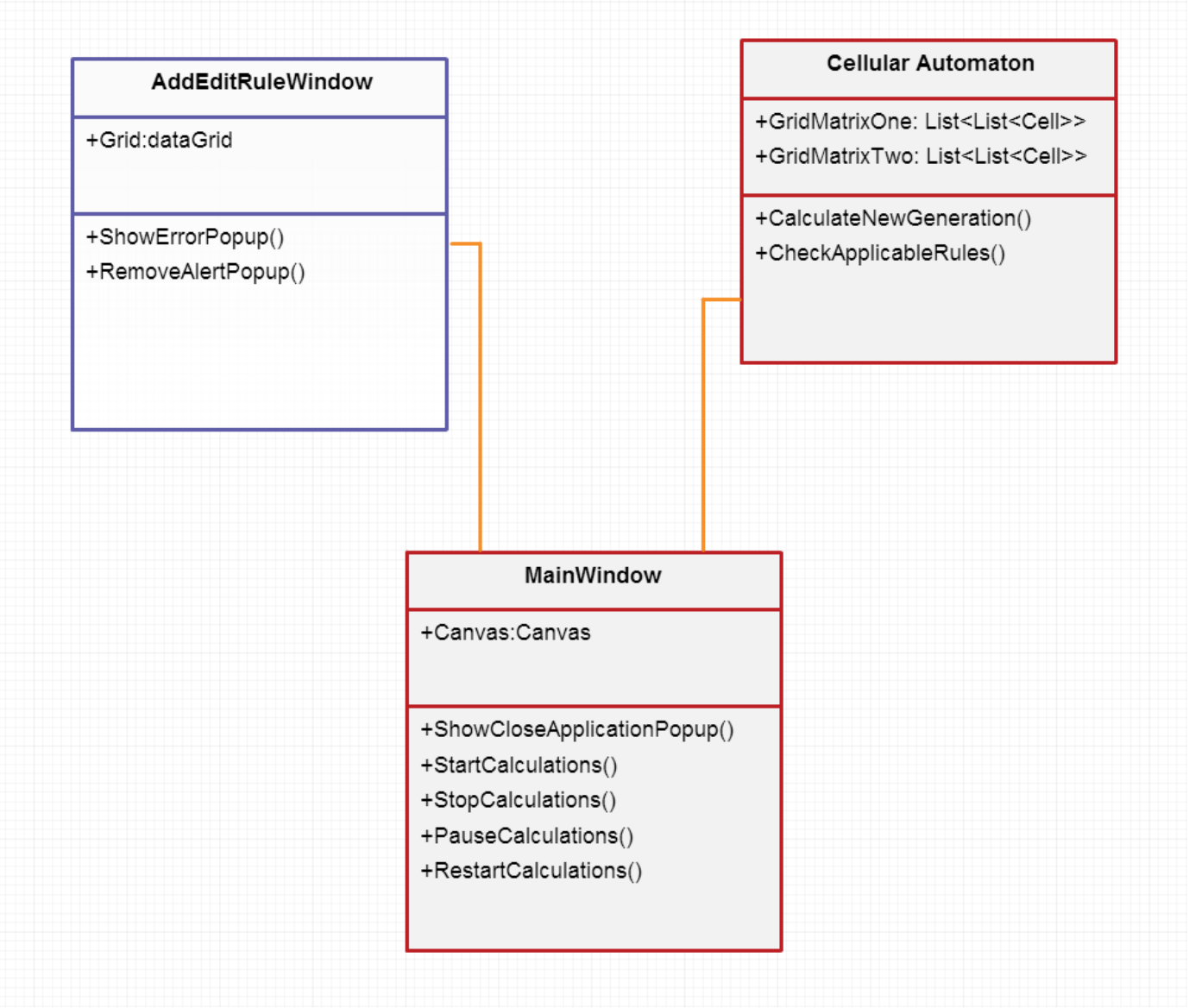
- Change grid's size - by option to zoom in/zoom out the visualisation.

- Load known automaton - for example Langton's ant.

- Display information - by showing for example time of simulation (in seconds), number of cells alive.

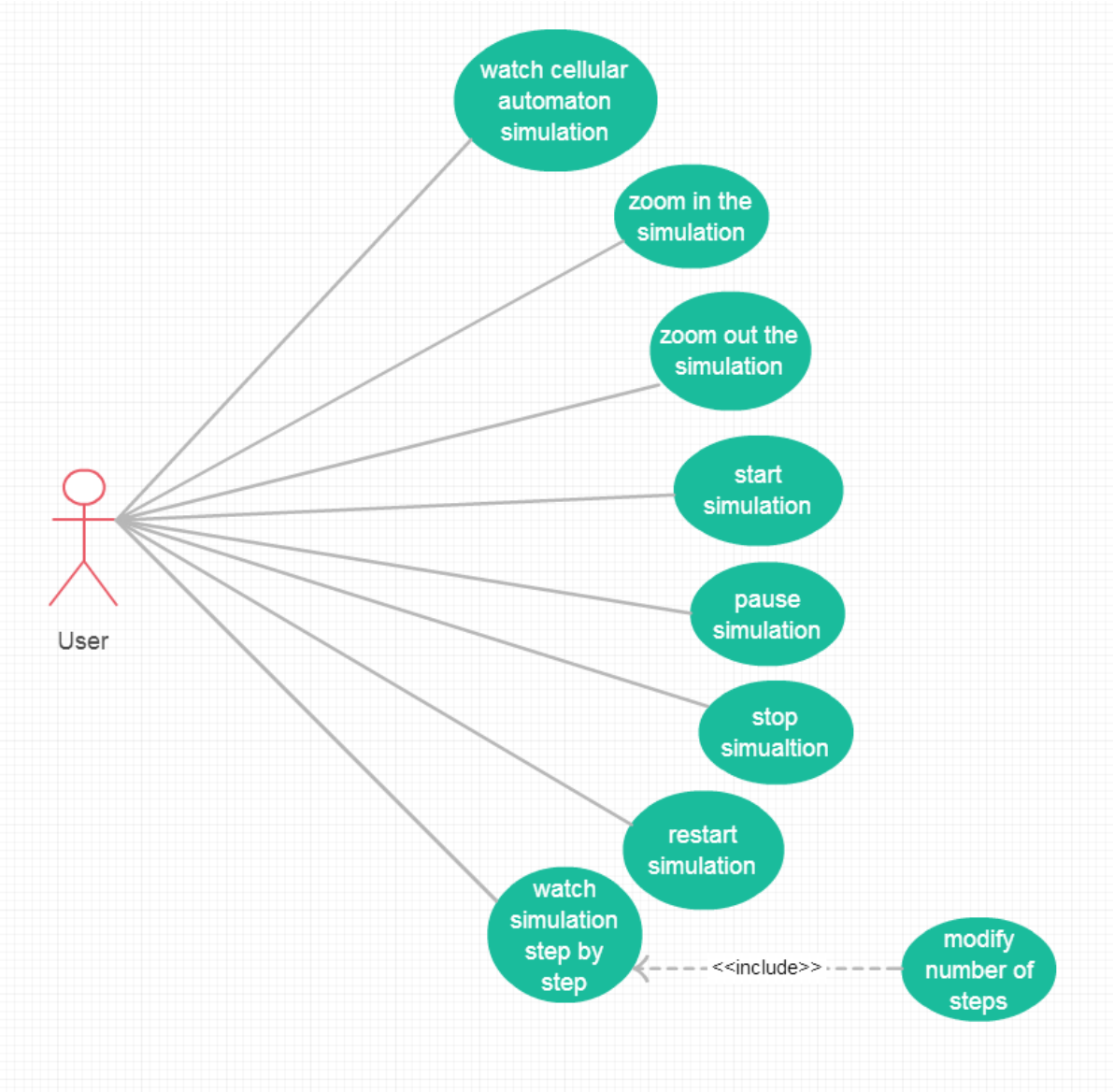
6 Class diagram / program structure

This class diagram is a simple visualization of classes in our program. Person implementing these classes is free to divide them to smaller ones.



7 Use cases

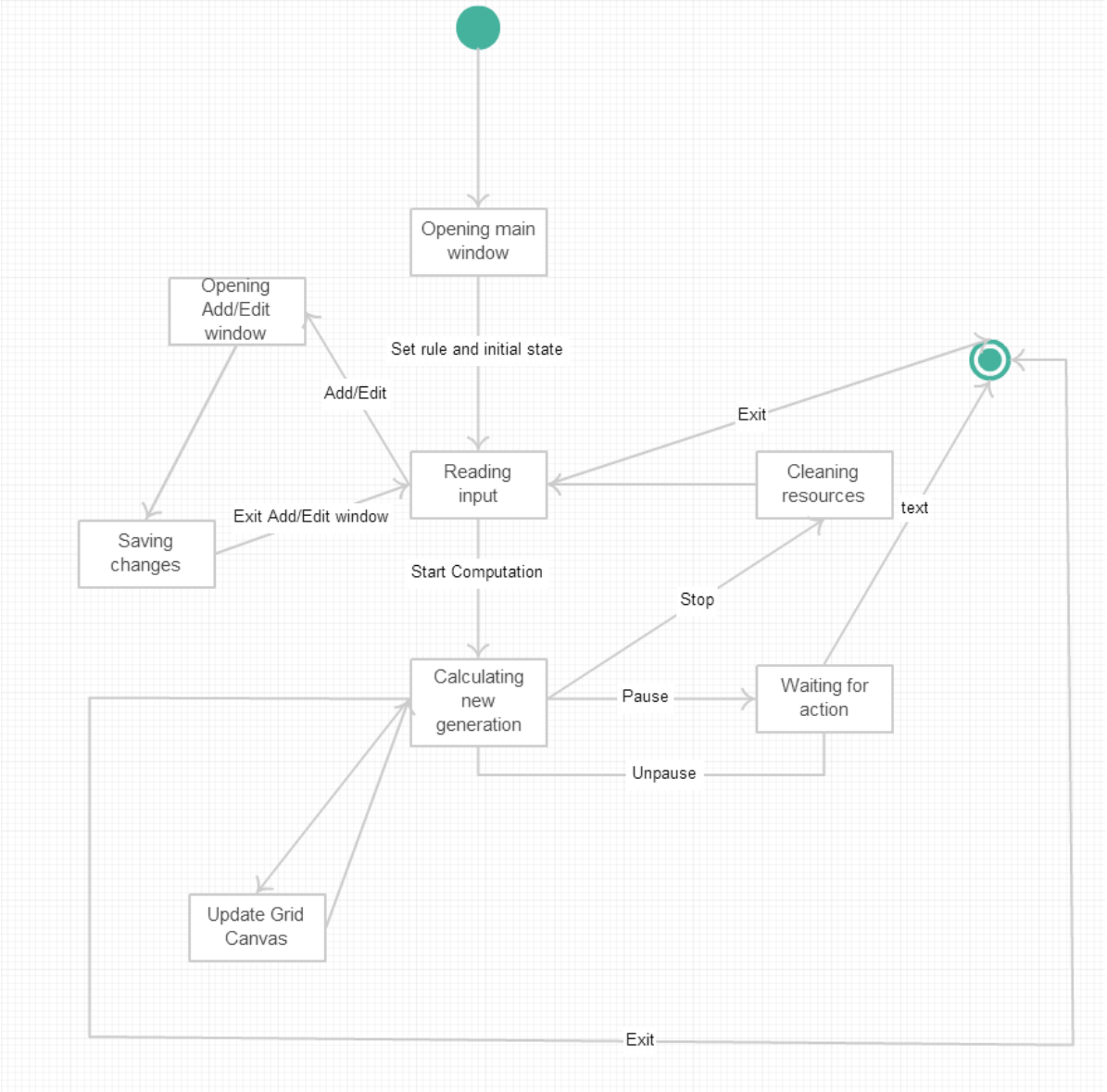
Use case for user in main menu.



Use case for user in Add/Edit menu.

8 States diagram

Simple state diagram describing behavior of program in general.



9 Graphical user interface mock-up

"Graphical user interface has been slightly modified since there won't be any possibility to manually determine rules for death of cell, hence there won't be any contradiction in applying rules."

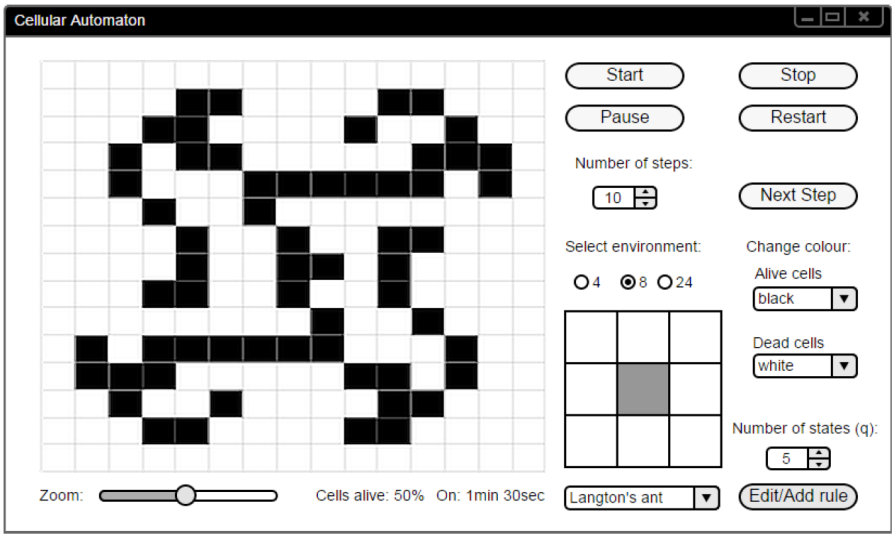


Figure 1. Main window of the application

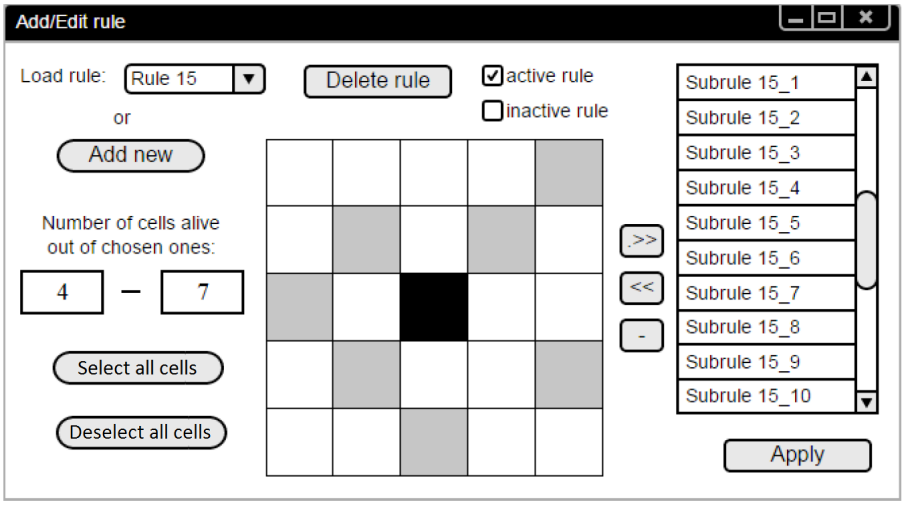


Figure 2. Edit/Add new rule window

10 Summary

This technical documentation is based on Business analysis made by Mai Viet Ba. I have made small changes in terms of functionality. User won't be able to manually create rules for death of a cell. User will be able to create rules which will change status of a cell to living and the rest of rules from rule set will cause death of a cell. Such changes will prevent any contradiction and will save a lot of computational power since it won't be needed to check all rules in a rule set. "Functionality" and "Graphical from business documentation has been modified and included in this document.

Application created basing on this documentation should be very fast, user friendly and eye pleasing. The biggest drawback will be support for only one computer system (it also might be open through "Wine" on Linux but it's not an official solution so it might not work properly).