# Cellular Automaton - Business Analysis

By Wojciech Zyskowski

## Document metric

| Document metric | | | | | |
|---|---|---|---|---|---|
| **Project** | Cellular Automaton | | | **Company:** | WUT |
| **Name:** | Cellular Automaton – Business Analysis | | | | |
| **Topics:** | Cellular Automaton | | | | |
| **Author:** | Wojciech Zyskowski | | | | |
| **File:** | Cellular Automaton – Business Analysis w zyskowski.pdf | | | | |
| **Version no:** | 1.1 | **Status:** | Working | **Opening date:** | 2015-03-01 |
| **Summary:** | The main purpose of this document is to provide business analysis for Cellular Automaton project. | | | | |
| **Authorized by** | | | | **Last modification date:** | 2015-03-11 |

## History of changes

| History of changes | | | |
|---|---|---|---|
| **Version** | **Date** | **Who** | **Description** |
| **0.1** | **2015-03-01** | **Wojciech Zyskowski** | **Initial draft of the document.** |
| **0.2** | **2015-03-03** | **Wojciech Zyskowski** | **Glossary added.** |
| **0.3** | **2015-03-04** | **Wojciech Zyskowski** | **Requirements set added, partial GUI mockups added.** |
| **0.6** | **2015-03-06** | **Wojciech Zyskowski** | **Requirement description added including full GUI mockup and user stories.** |
| **0.7** | **2015-03-08** | **Wojciech Zyskowski** | **Added summary, ending part and style guide.** |
| **1.0** | **2015-03-11** | **Wojciech Zyskowski** | **Modified images, corrected mistakes in few sections, enhanced requirement set and description sections.** |
| **1.1** | **2015-03-12** | **Wojciech Zyskowski** | **Correction of document. Glossary updated (Rule), Requirements Set updated, User Stories updated, GUI mockup descriptions (Main View, Rule Editor and Options).** |

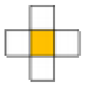
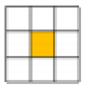
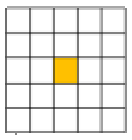
## Table of Contents

## 1. Summary

Cellular Automata is a tool simple enough to allow quite detailed mathematical analysis, being in the same time complex enough to represent multiple real-life dependencies. Cellular Automata is able to represent fractal growth of biological organisms, evolution processes, development of species and some chemical reactions.

## 2. Project Description

### 2.1 Glossary

- **Cellular Automaton.** A cellular automaton is a finite collection of cells located in a grid, each in one of a finite number of states; in case of this project equal only to two – “dead” or “alive” state. There is defined a rule set accordingly to which state transitions is defined.
- **Cell.** A single component of the automaton being also a cell of a grid.
- **Grid.** Most usually grid refers to two, or more infinite sets of parallel lines in a number of dimensions, intersecting each other at particular angles. In case of this project, we define grid as two perpendicular sets of evenly-spaced parallel lines intersecting in a single plane.
- **Simulation state.** It includes the grid size, state of all cells and current rule set.
- **Rule.** Generation of new cells and survival of existing ones is defined by rules. Each rule is defined as set of states of neighboring cells and the resulting state which is going to be applied to the specific cell. Every neighborhood size has its own rule sets, which cannot be combined. Rules for updating the state of cells are the same for each cell and are applied to the whole grid simultaneously. The rules continue to be applied repeatedly to create further generations. All defined rules are referred to as **rule set**. Each rule consists of list of **sub rules**, for the rule to apply all the sub rules conditions must be met.
- **Default rule.** Rule that applies to cell if no defined rules were applied.
- **Sub rule.** Is defined by three variables: set of selected cells in the neighborhood, condition (“exactly”, “at most”, “at least”) and number of cells associated with cell. For example: out of 7 selected cells at least 4 of them must be “alive”.
- **Neighborhood.** Set of nearby cells is called neighborhood, defined relatively to the specified cell. Depending on the option set, we consider three neighborhood sizes:
    - Four points neighborhood. 
    - Eight points neighborhood. 
    - Twenty four points neighborhood. 
- **Rule editor.** A popup window which contains tools for editing, deleting or adding rules in an easy matter.
- **Menu.** A popup window which contains options that are not relevant to current simulation, but allow data and rule management.

- **Options.** A popup window accessible from menu which contains options regarding simulation size, default rule and GUI.
- **Side menu.** A static window which contains options that are relevant to current simulation which is visible at all times at the top right corner.
- **Step.** Singular application of all appropriate rules to all of the cells present in the grid. On each step, for each cell is checked what rules apply and conduct appropriate rules, if there are any conflicts the user should be asked to resolve them.

## 3. Requirements Set

### 3.1 Obligatory

- Simulation of current rule set with a graphical representation:
  - Step by step
  - Automatic, based on timer, with possibility to pause/resume.
- Edition of rule set.
- Resizing of the grid.
- Default rule definition.
- Changing timer for the automatic simulation.
- Clearing grid.
- Logo.

### 3.2 Optional

Optional requirements are listed in decreasing order according to the priority of implementation.

- Save/Load rule set.
- Save/Load simulation state.
- Loading examples (i.e. Life).
- Zooming in/out from the grid and moving around grid.
- Simulation statistics (Grid size, percentage of “alive” cells, simulation speed etc.).
- Multiple color themes.
- Hide/Show side menu and/or simulation statistics.
- Auto save of current simulation state and rule set on every change.
- Options persistency (between multiple launches of application).
- Hotkeys support

### 3.3 Security and Performance

As the program doesn’t use any confidential type of data, it only stores simulation state, thus security measures are not required. On the other hand performance might become an issue if the user will want to conduct automatic simulation with small step interval. Either the code should be optimized or interval set to high enough levels so that the simulation will be smooth and non-stuttering.

# 4. Requirements Description

## 4.1 User Stories

As a user I want to:

- Be able to conduct simulation on step by step basis, so I can thoroughly analyze the simulation.
- Be able to conduct simulation on modifiable timer, so I can analyze the simulation in longer terms without unnecessary effort.
- Have a rule editor allowing me to create rules in an easy matter, i.e. not only allowing to specify exact neighbors positions, but also number of neighbors in row, column or in whole neighborhood on 'exactly X', 'at least X' or 'at most X', where X is some arbitrary number of cells present, so that I can easily generate new rules.
- Be able to select in the rule editor size of the neighborhood, so that I can conduct more interesting simulation scenarios.
- Be able to iterate through and remove currently existing rules, so that I can better manage them.
- Be able to modify (i.e. add or remove) rules after pausing, so that I can influence simulation process.
- As a user, I want to be able to pause/resume the simulation using appropriate buttons on GUI, so that I can easily analyze interesting moments of the simulation.
- Optionally, be able to save and load rule sets and simulation states, so that I can save time setting up environment.
- Be able to change size of available map, so that I can conduct simulations in more differenced environments.
- Optionally, be able to move the part of visible grid by dragging view, so that I can easily move around whole available grid, without zooming out.
- Optionally, be able to change the color theme of the application, so I can adapt it to my visual needs.
- A new popup window to appear if two rules are exclusive, so that I can choose which one of them should be preferred if another conflict emerges.
- Optionally, be able to launch some exemplary simulations, so that I can easily get familiar with software.
- Optionally, be able to use hotkeys for functions like pause/resume, open menu etc., so that I can access those functions in easier matter.
- A new popup window to appear, if I will try to change size of the grid after starting simulation (so introduce a change which will clear the simulation grid), so that I won't lose it by accident.
- A new popup window to appear, if I would try to exit rule editor after implementing changes without saving them, so I don't lose changes introduced.
- Be able to clear current simulation, so that I can restart it without the need to restart the application.
- Optionally, be able to introduce auto saves, so that I am sure I will not lose any data from the simulation step in case of some component failure.

- Optionally, be able to show/hide various components of the GUI so that I can focus easier on currently important aspects of the simulation.
- Be able to define default rule, so that I can better manage the simulation.
- A new popup window to appear if user will want to introduce impossible to apply sub rule (for example that 4 out of 2 cells will be in “alive” state).

## 4.2 GUI mockup

Graphical representation of the project can easily be divided into four independent parts:

### Main View

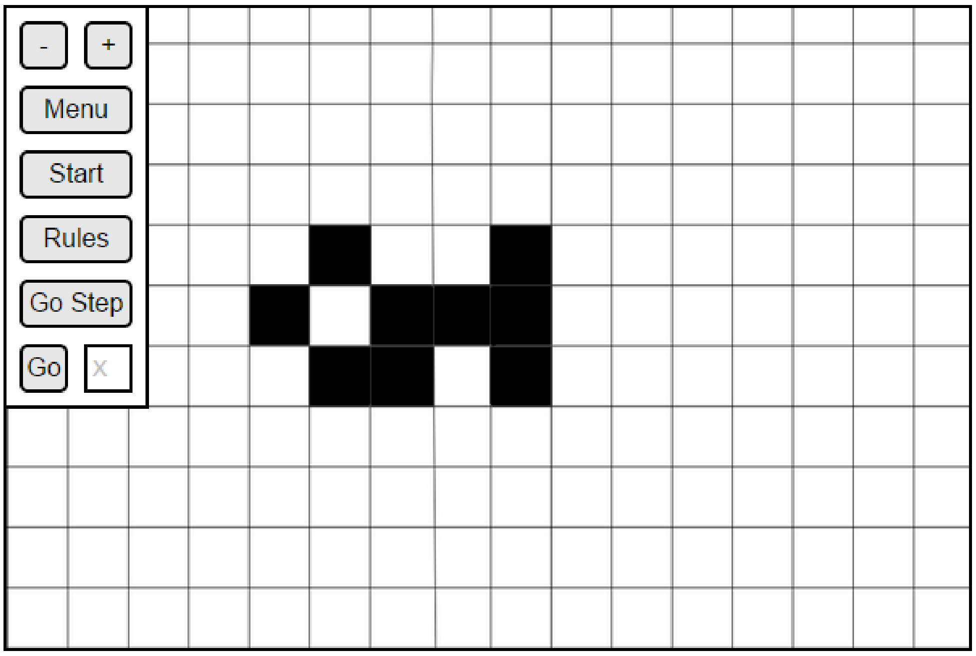


Figure 1: Main view mockup

The main view should contain grid on which the cells are visible and side menu, which allows for simulation management. During the step, the program should apply all rules to all cells simultaneously, but obviously there is no need for that to conduct proper simulation. If the current rule set would be divided into two sub groups - “survival” and “generation”, one could limit number of applied rules to each cell, as cells that are “alive” don’t have to make check for generation, and cells that are “dead” don’t have to make survival checks. Additionally, each new grid state should be calculated asynchronously, by multiple threads/tasks, so that computation time will be minimized. Implementation should follow such criteria:

- Cells that are “alive” should be marked in non-white color (black on exemplary mockup), cells that are “dead” should be marked in white color.
- When user hovers on button a context menu should be visible explaining what the button does.
- If user right-clicks on a cell before the first step in current simulation it should change state (from “dead” to “alive” and from “alive” to “dead”).
- The user should be able to move around the grid by dragging (i.e. it should be movable only after left-click). Additionally, left-click should change the mouse pointer to holding hand.

- The “+” and “-“ buttons should respectively increase the size of each cell and decrease the size of each cell, so that it is possible to zoom in and out from the grid.
- The “menu” button should show the menu popup window and pause the simulation if it was running.
- The “start” button should, based on a timer, conduct some number of steps per second, and change the button to “pause” button. The speed of simulation (i.e. number of steps per second), should be modifiable in the options menu.
- The “pause” button should stop the automatic simulation and change the button to “start” button.
- The “rules” button should show the rule editor popup window and pause the simulation if it was running.
- The “go step” button should conduct a single step.
- The “go” button should conduct number of steps written in the textbox to the right.

## Menu

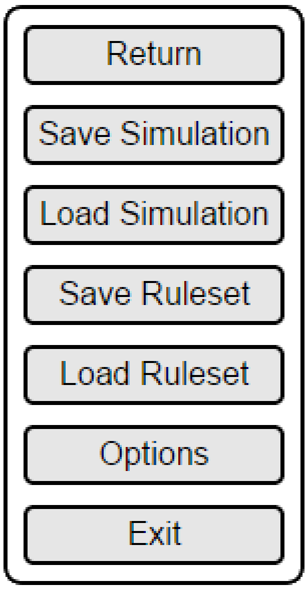


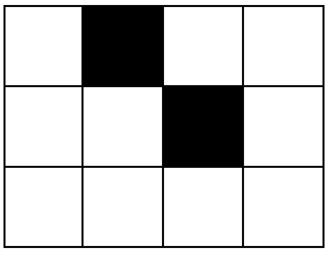
Figure 2: Menu mockup

The menu popup should contain functionalities that do not directly influence the simulation, hence they are not needed in the main view. Implementation should follow such criteria:

- The “return” button should close the menu popup window and resume the simulation if it was previously paused by opening of the menu popup window.
- The “save simulation” button should save the simulation state.
- The “load simulation” button should open new popup window, allowing user to choose previously saved simulation state.
- The “save ruleset” button should save current rule set.
- The “load ruleset” button should open new popup window, allowing user to choose previously saved rule set.
- The “options” button should hide menu popup and show the options popup window.
- The “exit” button should exit the application.

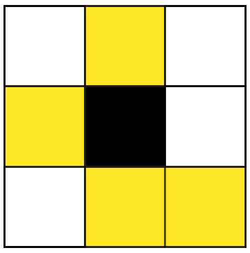
Saving simulation state consists of storing on the hard drive data regarding two elements – the grid state and the current ruleset. Grid state should be saved with size of the grid in the first row, and then array representing the grid with “0”s in the place of “dead” cells and “1”s in the place of “alive” cells. Exemplary grid save file would look then like this:

```
3 4
0 1 0 0
0 0 1 0
0 0 0 0
```

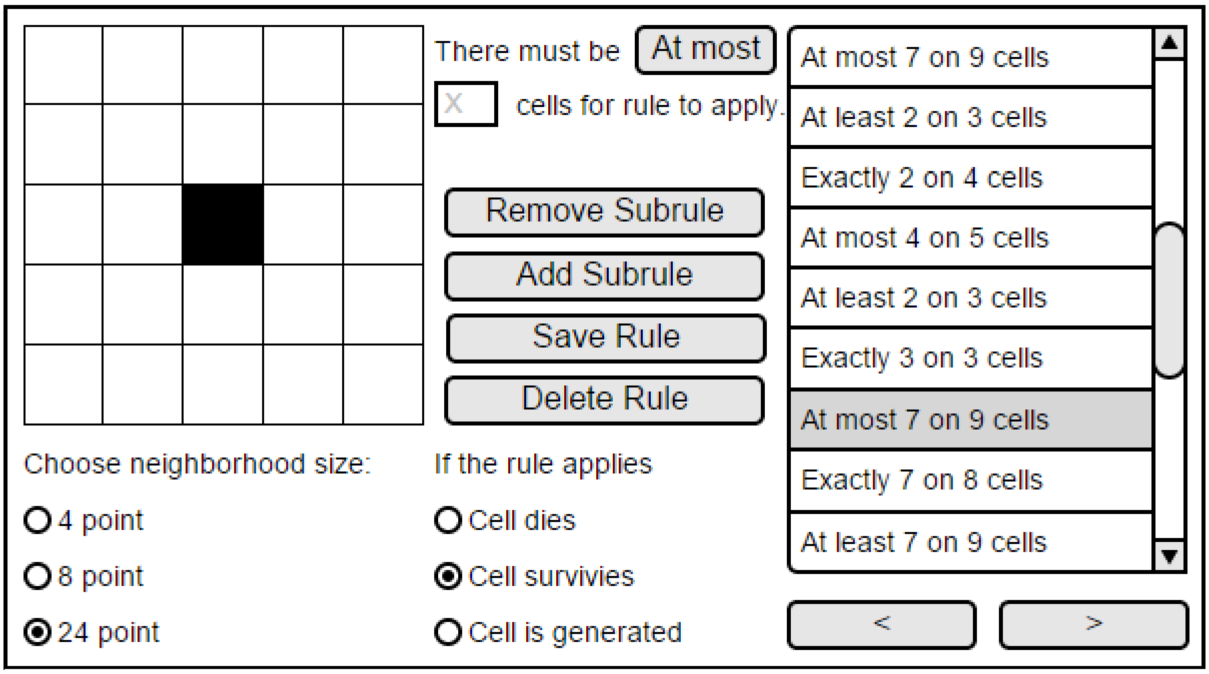


Current ruleset save file should firstly contain default rule defined in options popup, where default survival of the cell rule would be denoted as “1” and default death of the cell would be denoted as “0” and the size of neighborhood. From then on, each rule would be enclosed in “START” and “END” markers, with sub rules in different rows, represented as in Figure 3 and an array representing selected cells. Exemplary ruleset save would look like this:

```
1 8
START
At most 3 on 4 cells 01010011
END
```



## Rule Editor



There must be At most

X cells for rule to apply.

Remove Subrule

Add Subrule

Save Rule

Delete Rule

At most 7 on 9 cells
At least 2 on 3 cells
Exactly 2 on 4 cells
At most 4 on 5 cells
At least 2 on 3 cells
Exactly 3 on 3 cells
At most 7 on 9 cells
Exactly 7 on 8 cells
At least 7 on 9 cells

Choose neighborhood size:
4 point 8 point [✓ 24 point]

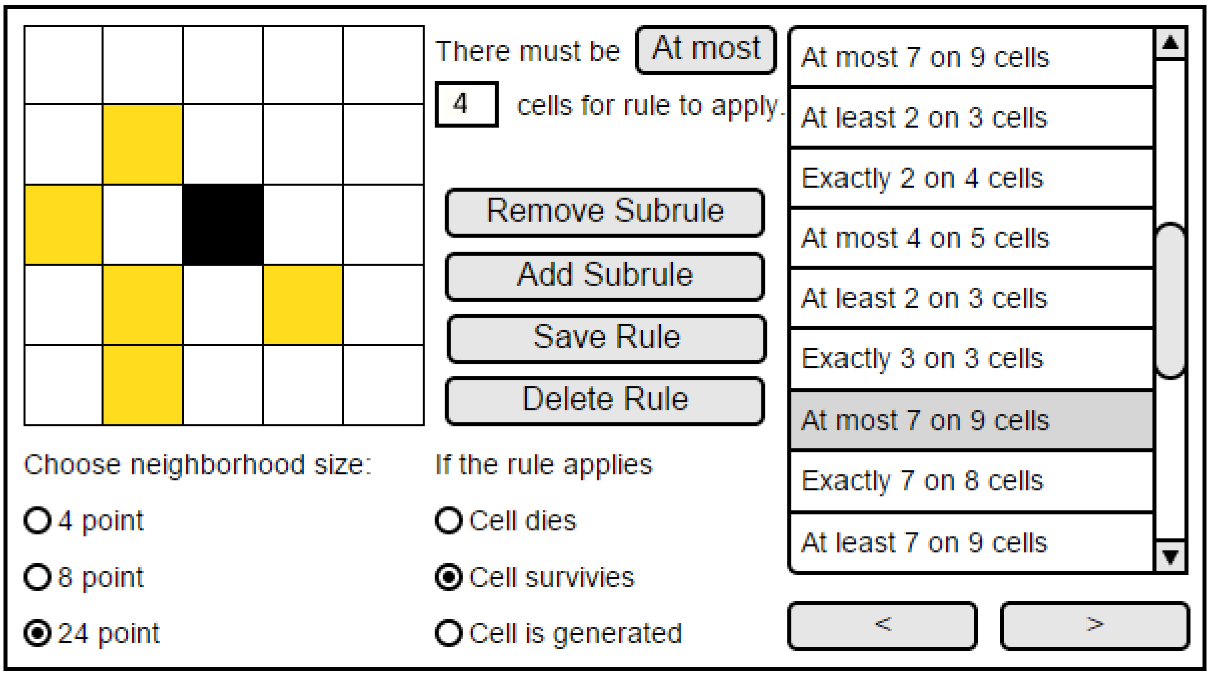
If the rule applies
Cell dies [✓ Cell survivies] Cell is generated

< >

Figure 3: Rule editor mockup

The rule editor popup allows for easy management of the current rule set. On the left side there is visible grid representing neighborhood in currently chosen size. On this grid user selects cells that sub rule will apply to. “There must be ‘at most’ ‘x’ cells for rule to apply” means the sub rule condition. This example says that if from selected on grid cells, there are at most x alive, then sub rule is applied. Additionally, the application should create two main lists of rules in current rule set, one containing all rules that result in death or survival of a cell and the other that determine cell generation, so that step computations can conduct as described in Main View description. Implementation should follow such criteria:

- The “at most” button on click should change to “exactly”, then to “at least” and then again to “at most”, allowing for changing the condition for the sub rule.
- The “x” field means number of “alive” cells in selected on grid cells that fulfill the condition for the rule.
- The “choose neighborhood size” allows for changing the grid size. If it is changed the current rule set should be cleared.
- The “if the rule applies” determines what happens to the cell after application of the rule.
- List on the right represent set of sub rules creating currently viewed rule, with selected sub rule.
- The “<” and “>”button sallow for changing view between separate rules.
- The “remove subrule” button removes currently selected sub rule.
- The “add subrule” button adds a new, clear sub rule and selects it on the list.
- The “save rule” button saves changes introduced to the rule in the rule editor.
- The “delete rule” button deletes currently selected rule.
- Exiting from rule editor should be possible by closing the popup window through the default close button in popup window frame.


There must be At most 4 cells for rule to apply.

Choose neighborhood size:
- 4 point
- 8 point
- [✓ 24 point]

If the rule applies
- Cell dies
- [✓ Cell survivies]
- Cell is generated

At most 7 on 9 cells
At least 2 on 3 cells
Exactly 2 on 4 cells
At most 4 on 5 cells
At least 2 on 3 cells
Exactly 3 on 3 cells
At most 7 on 9 cells
Exactly 7 on 8 cells
At least 7 on 9 cells

Figure 4: Rule editor example

In the example above five cells have been selected. If at most four of them is in “alive” state ( i.e. not all of them are “alive”), then the cell that the rule is applied to (black one) will survive.

## Options

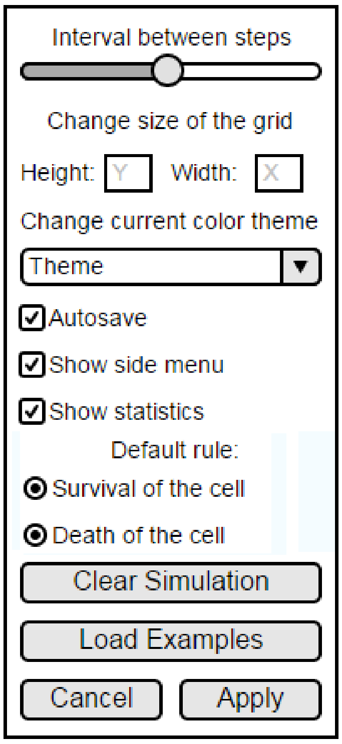


Figure 5: Options mockup

The options popup contain much of optional functionality and allow for their management. Implementation should follow such criteria:

- The “interval between steps” slider represents time between the subsequent steps. Suggested range is between 50 milliseconds to 2 seconds, though those values can be changed if deemed appropriate.
- The “height” and “width” fields set size of the available grid for simulation. Changing those values should reset current simulation.
- The “change current color theme” dropdown menu, allows for changing color theme in the application.
- If the “autosave” checkbox is checked, the application saves the state of the simulation and the rule set on every change.
- The “show side menu” and “show statistics” are define whether respective objects are visible in the main view.
- The “clear simulation” button clears current simulation state.
- The “load examples” button opens new popup window in which user is allowed to choose from one of examples to be loaded.
- The “cancel” button disregards all changes done in the options menu, hides options popup and shows menu popup again.
- The “apply” button saves all changes done in the options menu, hides options popup and shows menu popup again.

- The “default rule” specifies what happens if there is no rule applies for specific configuration.

### 4.3 Style Guide

The final product should contain the following logo. The choosing of place is non-relevant, only requirement is that it should be clearly visible during the usage of the program.

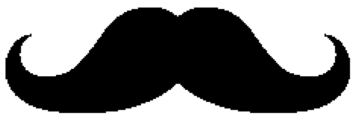


Figure 6: Logo

It is additionally available at http://imgur.com/XPlknTI.

## 5. Ending Part

- Optional requirements can be implemented by the developer in any way found fitting.
- If no optional requirements will be implemented the developer may choose to resign from implementation of options popup and provide other means of changing simulation timer, default rule and grid size, though all other GUI elements must be implemented as are.
- Any additional questions regarding the business analysis can be sent to w.zyskows@gmail.com, most probably they will be answered within 24 hours.