# Individual project

Cellular automaton

---

Technical project

Andrzej Frankowski

| **Document Metric** | | | | | |
|---|---|---|---|---|---|
| **Project** | Cellular automaton | | | **Company** | WUT |
| **Name** | Technical project | | | | |
| **Topics** | Technical project of a program simulating Cellular automaton | | | | |
| **Author** | Andrzej Frankowski | | | | |
| **File** | IP-TP-Frankowski.pdf | | | | |
| **Summary** | Technical d of system's functional requirements. | | | | |
| **Version no** | 1.0 | **Status** | Working | **Opening date** | 2015-03-18 |
| **Authorized by** | | | | **Last modification date** | 2015-03-18 |

| **History of changes** | | | |
|---|---|---|---|
| Version | Date | Author | Description. |
| 0.1 | 2015-03-18 | Andrzej Frankowski | Document setup and initial layout. |
| 0.2 | 2015-03-19 | Andrzej Frankowski | Added overview, technology, methodology, architecture: use-case diagram, *sample class diagram*, GUI implementation |
| 1.0 | 2015-04-01 | Andrzej Frankowski | Added ClassDiagram and Statechart diagram |
| 1.1 | 2015-04-02 | Andrzej Frankowski | Changed final conclusions |
| 1.2 | 2015-04-13 | Andrzej Frankowski | Algorithms |
| 1.3 | 2015-04-16 | Andrzej Frankowski | Detailed algorithm description |
| 1.4 | 2015-04-16 | Andrzej Frankowski | Considering number of neighbours as a part of a rule - changing business analysis |

# Table of content

# 1. Summary

a. Document overview

This document is about to describe in detail all technical aspects of the project. Further in this document used technology, methodology and detailed architecture of the program will be described.

# 2. Technology

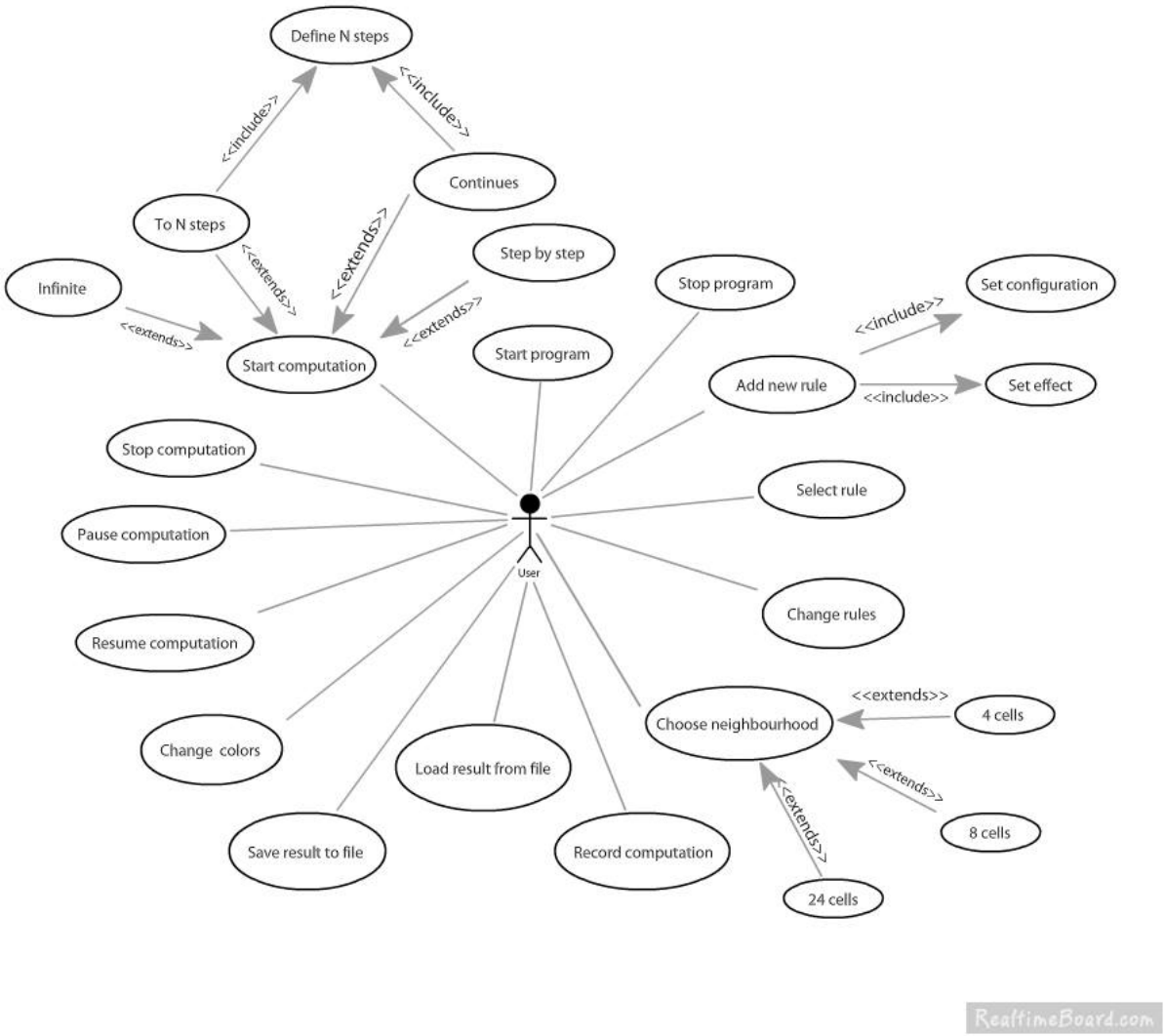
Program will be written C# language in Windows Presentation Foundation because they are one of the leading technologies in computer science nowadays. Main advantage of WPF and C# is that it is Window's native environment, the give a lot of possibilities, build-in functions and what is the most important, they are well known for the programmer.

# 3. Methodology

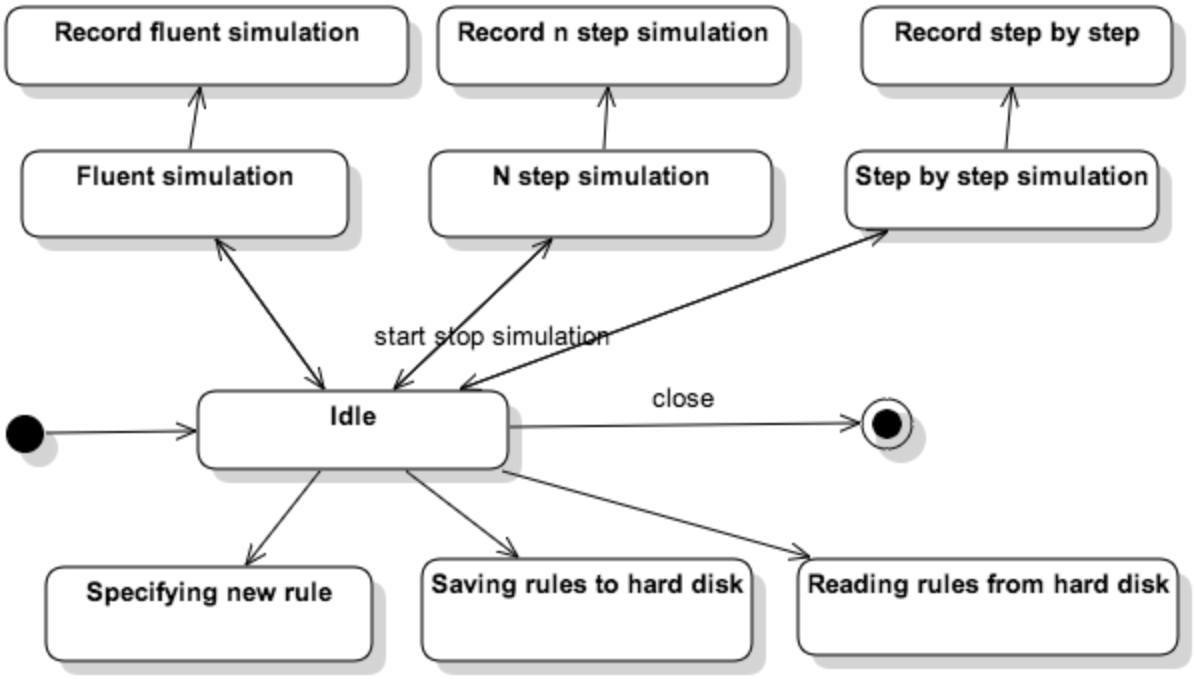
The project will be developed in the waterfall model. It's main advantage is that the project is easy to manage due to the sequentiality of the model. Each phase has specific deliverables and a review process. In this model phases of development are completed on at a time and do not overlap what introduces great simplicity for smaller project. Waterfall model is also great for projects, where the requirements are very well known, which makes this pattern an obvious choice.

# 4. Architecture

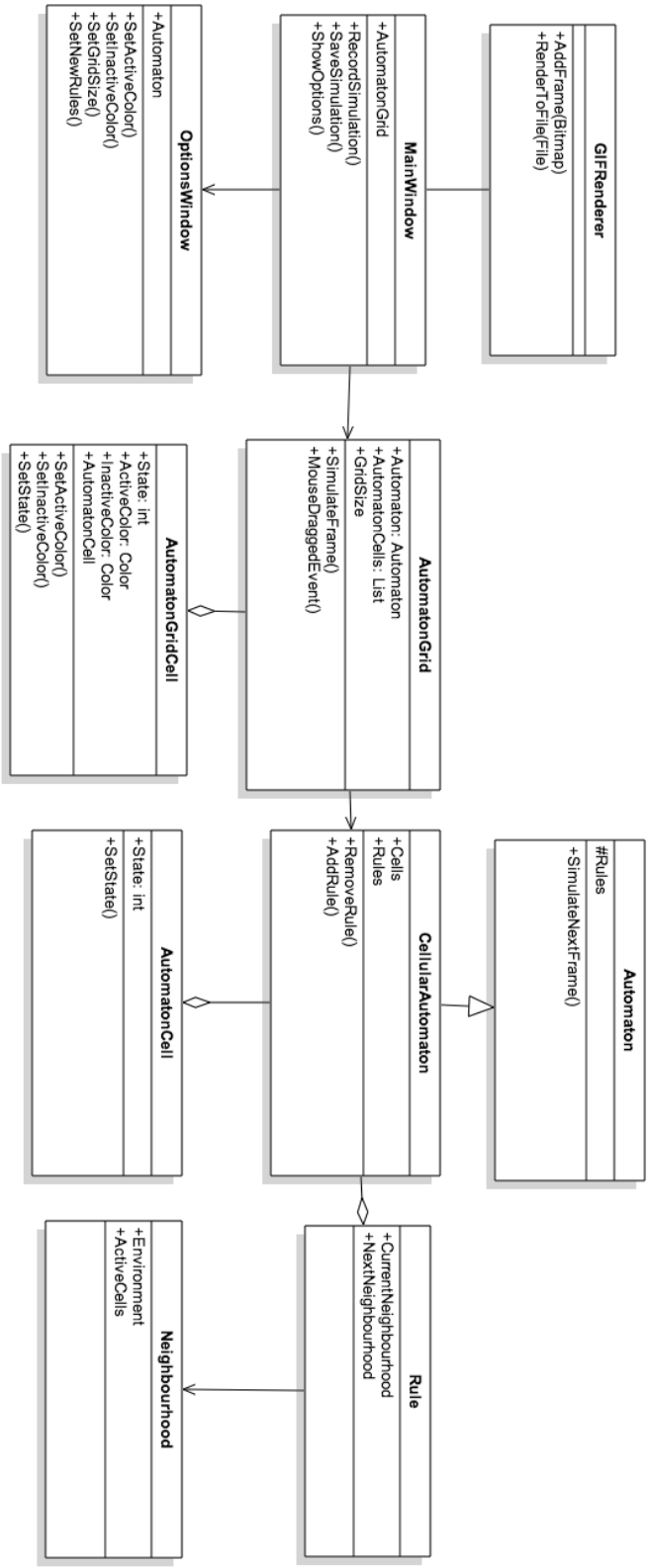
a. Use case diagram



b. State diagram



c. Class diagram



# 5. Data structures

No special complex data structures will be used. In model classes simple types. For storing cells lists and 2-dimensional lists will be used (actually an array of array, as in C# it is more efficient than multidimensional arrays).

# 6. Algorithms

The process of evaluation of a cell's state needs closer look. When calculating next generation of the automaton, all cells will be evaluated one after another. It is not expected to change the cells on the go, because it would change the way the rules apply to other cells. Thus it is needed to calculate the new state from the current grid in an atomic way and then apply the changes. Precalculating cell states will happen in a temporary copy of the grid model.

All the evaluation process happens in the model classes, not on the UI elements. Cell states are changed according to the 'before' and 'after' configuration in the current specified rule and comparing it with current neighbourhood cells (property Neighbourhood). UI classes hold references to models and they are both independent. The communication happens automatically thanks to data binding through INotifiyPropertyChanged.

Implementation of evaluation of a cell states require changes in business analysis. Document did not consider number of neighbours as a part of a rule. This change influences class diagram. In the program, the Rule class will be an abstract class with as for now 2 subclasses - CountRule and MatchRule.

*Algorithm 1* - specifying next generation of the cellular automaton:

1. Create a copy of the CellularAutomaton model
2. For each *row* in Cells<2d array> in the copy of CellularAutomaton model
    a. For each *cell* in a *row*
        i. Find corresponding rule (see Algorithm 2)
        ii. If corresponding rule was found - get the new state of the *cell*
            1. If new state differs, replace the *cell's* state in original CellularAutomaton model
        iii. If corresponding not found, do nothing (do not change the *cell's* state)
3. End

*Algorithm 2* - finds corresponding rule. Takes a cell with it's neighbours as an argument

1. For each rule in CellularAutomaton rules
    a. If Rule is a CountRule
        i. Count active neighbouring cells
        ii. If count matches
            1. return the rule
        iii. Else
            1. continue with next rule
    b. Else if Rule is MatchRule
        i. For each *row* in Neighbourhood
            1. For each *cell* in a *row*
                a. If neighbouring *cell* state differs
                    i. continue with next rule
                b. Else if *cell* state matches
                    i. If it was the last neighbouring cell to consider
                        1. return the rule
                    ii. Else
                        1. continue with next neighbouring cells
2. End (no corresponding rule found)

# 7. GUI implementation

User interface will be implemented in Windows Presentation Foundation. All elements will be implemented using native components.

Business analysis changes:

1. New rules view changed to a scrollable view. In the previous version it would hard for user to preview current rules of automaton.
2. There will be 1 Automaton Grid. Business analysis suggests to display initial state of the Automaton. In my opinion, more space for displaying automaton is more important. Especially if user desires to simulate a automaton of a big size.

Non-trivial components to implement are automaton grid and views for specifying new rules, hence the detailed description of these 2 UI elements:

- Automaton grid - details
    - Subclass of WPF's Grid, type of Rectangle
    - Bounded to Automaton model class
    - Number of Grid.RowDefinitions and Grid.ColumnDefinitions will correspond to size of the Automaton
    - Each Row and Column space in the grid will be filled with Rectangle object representing single cell
    - Each Rectangle will be filled accordingly to it's state in corresponding model
- New rules view
    - Scrollable view with  pairs mini versions of Automaton grids, of size corresponding to neighbourhood environment.
    - Each pair will represent before and after configuration of the cell and its neighbourhood
    - Cell's will respond to MouseClick events which will fill selected neighbours, specifying its states

# 8. Final conclusions

The program will be able to implement all features specified in the business analysis. Some minor changes were applied into graphical user interface. Classes are designed so that this project may be a basis for future projects willing to extend it to simulate other automatons or adding new features.