|  |
| --- |
|  |
| Business Analysis  Individual Project: Cellular automaton |
| Michał Szklarski  Computer Science 7 March 2016 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Document metric | | | | | |
| Project: | Cellular Automaton | | | **Company:** | WUT |
| Name: | Business Analysis – Individual Project | | | | |
| Topics: | Requirements specification of Cellular Automaton | | | | |
| Author: | Michał Szklarski | | | | |
| File: | Business Analysis.docx | | | | |
| Version no: | 07 | **Status:** | FINAL | **Opening date:** | 2016-03-07 |
| Summary: | To define requirements of project from the client side and present them using business modeling and language. | | | | |
| Authorized by: |  | | | **Last modification date:** | 2016-03-10 |

|  |  |  |  |
| --- | --- | --- | --- |
| History of changes | | | |
| Version | **Date** | **Who** | **Description** |
| 01 | 2016-03-07 | Michał Szklarski | Initial version, definition |
| 02 | 2016-03-07 | Michał Szklarski | Added document metric and history of changes parts |
| 03 | 2016-03-07 | Michał Szklarski | Added summary – overview part. |
| 04 | 2016-03-08 | Michał Szklarski | Extended summary + notion description + general specification |
| 05 | 2016-03-09 | Michał Szklarski | User stories |
| 06 | 2016-03-10 | Michał Szklarski | User stories fixes + conclusion |
| 07 | 2016-03-10 | Michał Szklarski | Non-functional requirements + GUI description, finalizing. |

# Business Analysis

## Individual Project: Cellular automaton

## Requirements specification

### Summary – overview

Aim of this document is to model requirements specification from the business side of a Cellular Automaton application as provided for Individual Project subject for Computer Science classes. Specification contained here follows preparation of technical documentation, with all implementation details and decisions, as we will not discuss them certainly here. Document is divided into several parts, starting from notion description (dictionary/glossary), general specification description, user stories about the project, supplementary specification (user stories addition), and finally: conclusion and last summary about this document.

It was decided to use user stories as primary requirements, user paths and fundamental sources of data basis, due to their compactness, value, negotiability and testing properties. As supplement to them, non-functional requirements (as performance, usability, and security) are specified, along with description of Graphical User Interface (GUI).

### Notion description

Cellular automaton – a set of multi-state cells on a defined grid of decided shape that evolves during a number of steps, according to a collection of rules based on the states of neighboring cells.

GUI – Graphical User Interface, part that interacts with end-users during regular program run. Allows manipulating program data or input, and present results accordingly.

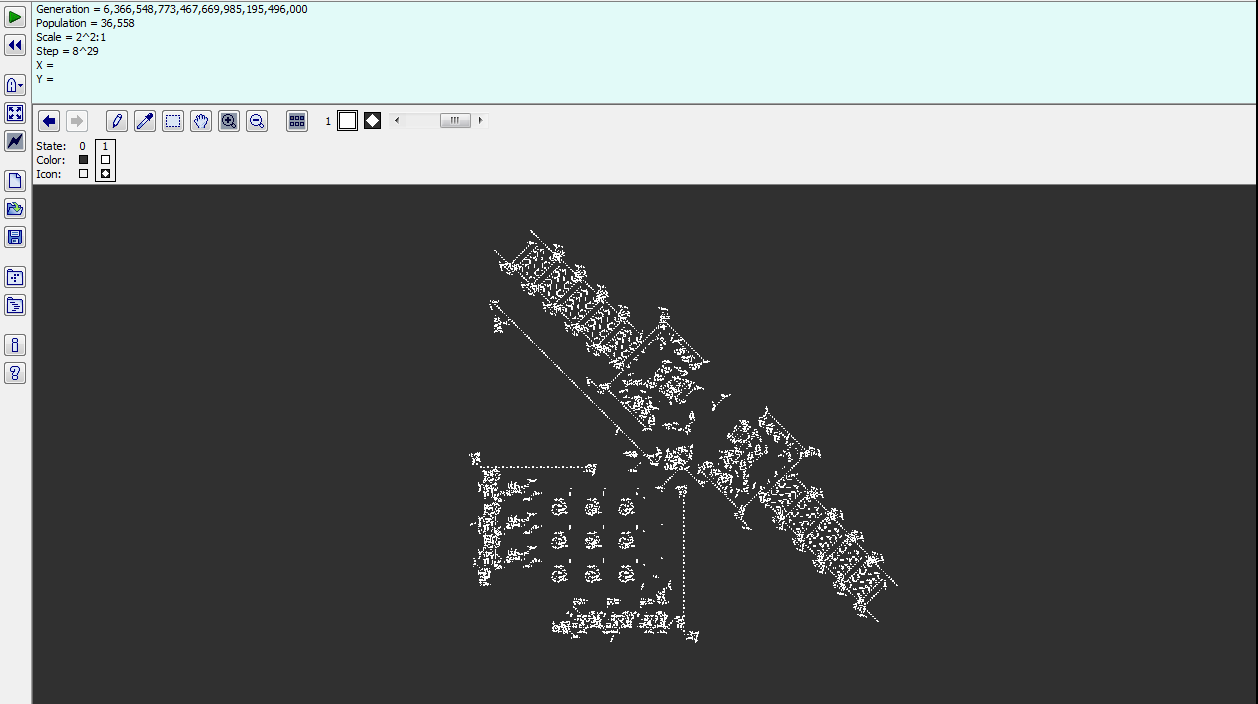
User story – basis of requirement, single or complex sentence, usually articulated with everyday language of user, with little detail, but it’s open to interpretation. One user story describes one requirement of developed software

Generation – current representation of cells and they state; can also represent a number – count of cycles from the start of simulation.

Cycle – one run of applying the set of given rules on all cells in grid.

Population – count of cells in current generation.

### General specification

 Software developed on the basis of this document is called professionally Cellular Automation. Generally speaking, it is a computer program, that simulates a life of requested number of multi-state cells population on a grid plane. A good example of such program is, as an instance, software called “Golly” that is based on Conway's Game of Life and other cellular automata.

As there are many cellular automata following some different sets of rules, here user should define its own. Thus, main window should present such grid. Users are able to specify size of the grid (dimensions), and of course rules that indicate how cells will grow or die, in separate, chosen ways to do so. Neighborhood of any cell on this grid is strictly relative, as it only detects surrounding other cells. Rule sets are applied to given cell in generation by going from the top to the bottom of grid, with defined number of cycles. The basic source of data for this program comes only from user input to appropriate fields.

### User stories

As described by the expected user of created software:

* As a user [of this project software] I want to have a main screen with grid of cells of size defined by me, before the start of simulation, because it is such simulation basis.
* As a user I want to have a fast and most convenient way of placement of all important actions or buttons, so that I can focus mostly on simulation run, not searching for program options.
* As a user starting the simulation I want to have a way of removing previous simulation run from the grid, because I need a clear screen to start new one.
* As a user running the simulation I want to have a possibility to move through the grid using a mouse, so that I can customize the current view to accommodate my needs.
* As a user running the simulation I want the one, defined by me beforehand, set of rules to be applied on whole grid simultaneously, so that I can observe fluent changes in the full system.
* As a user I want to have the possibility to zoom in and zoom out the grid, so that I can inspect selected cell groups with convenience.
* As a user I want to have displayed short statistics about current simulation run which should contain generation number, population count and scale (zoom) level, so that I can view this data at any point and compare it with visual output.
* As a user I want to have cells in three states – dead, alive, and empty, colored properly: red, dark green, white, so that I can introduce diversity in simulation and distinguish them.
* As a user I want to run the simulation at any moment, pause it, or stop, so that I can observe generation in any cycle.
* As a user running the simulation I want to have the possibility of defining total count of cycles after which simulation will stop, because I want to observe grid after such number of cycles specified by me and run only strict, partial simulations.
* As a user I want to define neighborhood of any cell to fixed number of 24 surrounding cells, because that suits my needs at best, along with simulation ideas.
* As a user observing the simulation I want to always have a possibility to observe whole population of cells, so that I can perform observations on full system.
* As a user defining simulation details I want to have predefined initial suite of rules with examples of starting state (partially filled grids of cells in various states) in the program on how the cell will react to various situations during simulation run, because this will allow me to perform quick simulation runs.
* As a user defining simulation details I want to introduce my own rules with examples using some kind of editor in the program on how the cell will react to various situations during simulation run, in a most convenient and representative way for me, because it will allow me to fully manipulate the simulation execution.
* As a user defining simulation details I want the editor of custom rules to have mechanism that will resolve contradicting rules, and warns me about them, so that I cannot create nonsense or irrelevant systems.
* As a user running the simulation I want to have a possibility to dynamically change the set of rules on how a single cell react to its neighbors, because I want the program and simulation to be more flexible.

### Supplements

#### Main flow of the application

From the start of software, user should be able to customize simulation details through specific fields and editor, apply them, start the simulation, and be able to observe and stop it any time or view current state after some number of cycles. During simulation run, user can change rules deciding on cells behavior. Then the user can remove finished simulation from the screen and start the flow once again.

Editing rules

Editing rules

#### Non-functional requirements

Requirements that are not a part of user-defined stories, but defines additional common or specific issues (either performance, safety/security or usability based):

* There shouldn’t be too many pop-ups with warnings or errors at once.
* Application should work without stuttering seen by human eye.
* Application shouldn’t overload client system resources with computations during simulation run.
* Application should not depend on any data located in client’s computer.

#### Graphical User Interface description

As a supplement for user stories, in this part proposed layout of project Graphical User Interface is presented, however it needs to be noted, that here contained only is a proposition of such GUI, therefore final design might differ vastly from this version, due to technical limitations or hints based on User Experience (UX) for such project:

* Main window
  + Contains scrollable grid of size where main simulation runs. The grid consists of cells that can be filled with alive or dead cells, if not – they are empty.
  + Size of one cell is relative – depends on scale/zoom factor.
  + By default, whole population should be observable on main simulation grid at once.
  + Contains also toolbar with general options: run, stop, pause/resume simulation, simulation options, and run with cycle count n (with field to enter them), so that user can observe system after n cycles.
  + Also on toolbar: zoom in, zoom out, move options.
  + Below the simulation grid: place for statistics: generation number, population count and scale (zoom) level.
* Simulation options editor window
  + Should contain fields to determine: max. number of cycles/generations.
  + Should contain initial sets of rules chooser, with presented examples in graphical form.
  + Should link to or contain rules editor, in any best for user form: either graphical or text-based.
    - Rules editor should have many fields for rules to enter.
    - Contradicting rules should be detected and displayed as error with option to fix them manually or automatically by program.
    - Should also be possible to access that window during simulation run, as specified by user.

From proposition described here, qualified graphic designer should create various mockups that will later be used as real, 1:1 GUI guidelines.

### Conclusion

As described in this document, business requirements model and analysis is done. Following remaining part is to prepare technical documentation that will allow determining general overview and details of implementation for such cellular automaton simulation software. All points considered here should be easily translatable into technical language of specific requirements. From there, development processes can finally and properly move on into implementation phase, which is most important in every such project.