3 March 2015

Jakub Ciecierski

Cellular automaton
Requirement specification

Contents

1 Schedule

Date	Asset
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 1.00
2015-05-28	Test report
2015-06-11	Acceptation

2 Document metric

Document metric					
Project:	Cellular Automaton	Company:	WUT		
Name:	Requirement specification				
Topics:	Business analysis of the product				
Author:	Jakub Ciecierski				
File:	requirement_specification.pdf				
Version no:	0.1	Status:	Under development	Opening date:	2015-03-03
Summary:	Business analysis of application that allows for creating a cellular automaton				
Authorized by:	Wadysaw Homenda Lucjan Stapp	Last modification date:			2015-03-03

3 History of changes

History of Changes			
Version	Date	Who	Description
0.1	2015-03-03	Jakub Ciecierski	Definition of the main purpose of the document

4 Glossary

- **Pattern Recognition** In broad terms, pattern recognition is science of making assumptions about data using various tools from statistics, machine learning and many others fields. Focuses on designing and building machines that can recognize patterns. Such patterns can be found in speech, fingerprint, optical characters etc.

  Feature is defined as a quality or characteristic of an element. Such feature can be a symbolic measure (e.g. color) or numeric (e.g. width). Collection of d features is called a d-dimensional feature vector.

  In classification a pattern can be represented by a pair (x, w) where x is the feature vector and w is label. A label tells the computer to which class a given element belongs to. Elements from the same class should have similar features, while elements belonging to different classes should have relatively different features.

- **Alphabet** Is a finite, non empty set of symbols, commonly denoted by Σ. Examples of common alphabets:

  1. $\Sigma = \{0, 1\}$ - binary alphabet.
  2. $\Sigma = \{a, b, ..., z\}$ - small letters of latin alphabet.

- **Word over Alphabet** Also called a *string*, is a sequence of symbols over some alphabet Σ. Examples of words:

  1. A sequence $'01010'$ is a word over binary alphabet $\Sigma = \{0, 1\}$.
  2. A word $'lorem'$ is a word over the latin alphabet $\Sigma = \{a, b, ..., z\}$.

  An empty word is a word with no symbols. Commonly denoted by ε. Such word can be taken from any alphabet.

  A set of all words over alphabet Σ is denoted by Σ^*, where

  $$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ... \qquad (1)$$

  If $\Sigma = \{0, 1\}$ then $\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \{0, 1\}$, $\Sigma^2 = \{00, 01, 10, 11\}$, $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$ and so on

  It is important to note that, Σ^* is infinite countable set.

- **Language** Language over alphabet Σ will be denoted by L. Language L is a subset of all words Σ^*, $L \subseteq \Sigma^*$.

  Examples of languages:

  1. $L = \{\varepsilon, 01, 10, 0011, 0101, 0110, ...\}$ - a set of all binary words that have the same number of occurrences of $0's$ and $1's$
  2. $L = \emptyset$ - language is an empty set, contains no words
  3. $L = \{\varepsilon\}$ - language containing only empty word.

- **Deterministic Finite Automaton (DFA)** Automaton is a very simply computability model. It can be thought of as a physical machine containing a *tape* with input word, a *head* reading a single symbol from the tape and finally a steering mechanism which can change its state during the computations based on current state and a symbol that is being read. DFA computes a word in order to check if a given word belongs to a language accepted by this machine.

  Formally DFA is a system of five fields:

  $$A = (Q, \Sigma, \delta, q_0, F) \tag{2}$$

  where
  Q - finite set of states.
  Σ - Finite input alphabet.
  δ - transition function. $\delta : Q \times \Sigma \to Q$
  q_0 - the initial state. $q_0 \in Q$
  F - Set of accepting states. $F \subseteq Q$

  The computations of DFA is a sequence of transitions based on transition function. Depending on a state q and symbol x read by the head the machine:

  1. changes its state to $p \in Q$
  2. moves the head one cell to the right.

  Automaton finished computations when all symbols were read. It accepts input if computations end in accepting state, otherwise the input is rejected.

5 User stories

Grid editor

- As a user, I want to open grid editor, in order to change the grid size.
- As a user, I want to open grid editor, in order to change color of each state of a cell.
- As a user, I want to open grid editor, in order to enable/disable wrapping option.
- As a user, I want to scroll my mouse roll over the grid, in order to adjust the scale of the grid.
- As a user, I want to select the brush, in order to draw cells on the grid.

Rule editor

- As a user, I want to open rule editor, in order to create new rule.
- As a user, I want to choose neighborhood environment, in order to add new rule.
- As a user, I want to define specific transition for a given state of cell, in order to generate new state.
- As a user, I want to click save/save as button in rule editor, in order to save current rule.
- As a user, I want to click load button in rule editor, in order to load current rule and possible edit it.

Application option

- As a user, I want to move View components (e.g. rule editor / grid editor / browser), in order to position them in different location.
- As a user, I want to click next generation button to compute next generation
- As a user, I want to click next N generations button, to compute next N generations.
- As a user, I want to set the number of generation to skip by clicking next N generations button, in order to compute next N generations.
- As a user, I want set the speed of computation of next generation in running mode, to customize the speed of which the automaton is transitioning.

Pattern editor

- As a user, I want to save a current state of grid into a pattern, so that later I can load it into the program.
- As a user, I want to browse for my patterns in the browser window, in order to load it to the pattern editor.
- As a user, I want to click change rule in the pattern editor, in order to add a rule of my choosing to that pattern.

6 Functional Requirements

Priority

- 1 - must be implemented
- 2 - can be implemented optionally
- 3 - is a nice addition, but not needed.

ID	Requirement	Comments	Priority
1	The system provides a Grid options allowing for changing the size, colour of cells and enabling/disabling wrapping option	The colour of cells represent a state of the cell. In other words the user can choose in what state to put a cell into.	1
1.1	The system should allow grid maneuvers, zooming in/out and if the entire grid is not visible in one screen, possibility of moving around the grid		1
2	The system provides a Rule editor in which the user can create, edit and save rules.		1
2.1	By clicking create rule button in Rule editor, the application will open a fresh rule creation window		1
2.2	By clicking load button in Rule editor, the application will open a browser which will allow the user to find saved rules		1
2.3	By clicking save button in Rule editor, the application will make sure that name for the rule is provided and then will save the rule in specified by the user location		1

ID	Requirement	Comments	Priority
2.4	The system provides three different neighborhood environments in which the user can create rules, 4-point, 8-point, 24-point	See Glossary / Neighborhood for more information	1
2.5	The application provides special file extension for saving and keeping rules		1
2.6	For 4-point and 8-point environments the system should provide a way to create rules in which positions of neighbors relative to the cell are considered. If a transition is not defined then this transition does not change the state of current cell	The user can choose to what state current cell transitions, based on this cell's state and states of his neighbors	1
2.7	For 24-point environment system should provide a may of creating rules in which the user specifies number of neighbors in each column	This environment can be represented as a 5 by 5 matrix with the current cell in the middle	1
2.8	For 4-point, 8-point and 24-point environments the system should provide a simplified mode of creating rules in which the user inputs only number of neighbors in given state in the neighborhood for a current cell state.	The user inputs number of neighbors with given state which should appear for the cell to transition to another specified state	2
3.1	The system provides a step-by-step button which computes next generation		1
3.2	The system provides next-N button which computes next N generations, the N must be easily chosen by the user		1
3.3	The system provides a run button which will start the animation of consecutive generations		1
3.4	The system provides way to change speed of which the animation is drawn in the 'run' mode		1
4.1	The application allow user to draw cells on the grid		1
4.2	The application provides a way for user to save grid state into patterns, additionally the pattern can have a rule attached to it, which later can be loaded into the grid.	A pattern editor view component should be created. The grid state consists of its size, states of cells and other grid options.	2
4.3	The application provides Browser window in which the user can browse saved rules and patterns		1
4.4	The application allows the user to have multiple grids opened.		2
4.5	The application should have example of simple game of Life called Conway's Game of Life		2

7 Non Functional Requirements