|  |
| --- |
|  |

System/Software Requirements Specification

for

Cellular Automaton Simulator

**Version 1.0 approved**

***Prepared by: Tran Quy Ban, Phan Minh Duong, Ha Vien Duong, Tran Thuy Kieu***

**USTH**

**Feb, 5 2020**

**Table of Contents**

[**Introduction**](#_heading=h.tppyqsp5ti9r) **1**

[Purpose](#_heading=h.2et92p0) 1

[System Purpose](#_heading=h.umpy1fyns6gr) 1

[Definitions, Acronyms and Abbreviations](#_heading=h.h6sz8y19owpz) 1

[Document Conventions](#_heading=h.x18q7kdkv99m) 1

[Intended Audience and Reading Suggestions](#_heading=h.i8xq1s7pmha1) 1

[Project Scope](#_heading=h.y4pbvyq5rwwg) 1

[References](#_heading=h.by0ysvw93sm) 2

[**Overall Description**](#_heading=h.svq80rj7w2he) **2**

[Product/System Perspective](#_heading=h.26in1rg) 2

[System/Product Features](#_heading=h.r4prjpx8y16z) 2

[User requirements](#_heading=h.j4oknd20mypy) 2

[Main Function](#_heading=h.5645u6ij525a) 2

[Main concept](#_heading=h.y683vs9st4ij) 2

[Normal User Function](#_heading=h.7timjofuq04i) 2

[Create/Edit parameter for the simulator](#_heading=h.rzmzzjgmcyjn) 2

[View result of simulator](#_heading=h.6bekzj6ari3c) 3

[User Classes and Characteristics](#_heading=h.ggfxp6v9ipat) 3

[Operating Environment](#_heading=h.nyl79zsui5rm) 3

[Design and Implementation Constraints](#_heading=h.lr3sgjnbjlcb) 3

[User Documentation](#_heading=h.yk231dv2c6ny) 3

[Assumptions and Dependencies](#_heading=h.by6el8h5d2o2) 3

[Apportioning of Requirements](#_heading=h.9vjbd2w0m1gr) 4

[**Specific Requirements**](#_heading=h.6worzw34g8vk) **4**

[Functional Requirements Specification](#_heading=h.1ci93xb) 4

[Function1 /Use-case 1](#_heading=h.mv1l0vnervhn) 4

[Function2 /Use-case 2](#_heading=h.hplefzkn0kfm) 5

[Function3 /Use-case 3](#_heading=h.xp77nx9cmjpb) 7

[Function4 /Use-case 4](#_heading=h.dobuppll61qx) 8

[Function5 /Use-case 5](#_heading=h.a6wey4ny4gdk) 10

[Non-Functional Requirements Specification](#_heading=h.3as4poj) 11

[External Interface Requirements](#_heading=h.jey72k2m63b3) 11

[User Interfaces](#_heading=h.4z93fa84s4zd) 11

[Hardware Interfaces](#_heading=h.3x5kx5y8yj1e) 12

[Software Interfaces](#_heading=h.ia1eth2grg46) 12

[Communications Interfaces](#_heading=h.arqy9flul9mb) 12

[Other Nonfunctional Requirements](#_heading=h.rvaa7owuun4a) 12

[Performance Requirements](#_heading=h.8gpb91k0sk1e) 12

[Design constraints](#_heading=h.kx9rtjnds5gd) 12

[Safety Requirements](#_heading=h.iyumcac7bays) 12

[Security Requirements](#_heading=h.9sai0l375nw2) 13

[Software Quality Attributes](#_heading=h.wn7gra130frj) 13

[**Other Requirements**](#_heading=h.3vgee3t6jkl6) **13**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Tran Quy Ban | Feb, 5 | Created | 1.0 |
| Ha Vien Duong | Feb, 6 | Created UCs 2 -5 | 1.0 |
| Tran Thuy Kieu | Feb, 6 | Created UCs 6 - 7 | 1.0 |
| Tran Quy Ban | Feb, 7 | Reviewed and updated URD and UCs | 1.0 |

# Introduction

## Purpose

*<Describe the document purposes; what information this document will contain; who are readers that this document aims at…>*

## System Purpose

<*Identify the product whose system requirements are specified in this document, including the revision or release number. Describe the system objectives, scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.*

## Definitions, Acronyms and Abbreviations

*<Describe all common business/technical definitions, acronyms, and abbreviations that are used in the whole document*

## Document Conventions

*<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>*

## Intended Audience and Reading Suggestions

*<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>*

## Project Scope

*<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. An SRS that specifies the next release of an evolving product should contain its own scope statement as a subset of the long-term strategic product vision.>*

## References

*<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>*

# Overall Description

## Product/System Perspective

*<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>*

## System/Product Features

*<Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Details will be provided in Section 3, so only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, is often effective.>*

## User requirements

### Main Function

#### Main concept

A cellular automaton (CA) is a regular grid of cells, each one in one of a finite number of states, such as on and off. The grid can be in any finite number of dimensions. For each cell, a set of cells called its neighborhood is defined relative to the specified cell. An initial state (time t = 0) is selected by assigning a state for each cell. A new generation is created (advancing time t by 1), according to some fixed rule (a given mathematical function) that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighboring. Typically, the rule for updating the state of cells is the same for each cell and does not change over time, and is applied to the whole grid simultaneously, though exceptions are known, such as the [stochastic cellular automaton](https://en.wikipedia.org/wiki/Stochastic_cellular_automaton) and [asynchronous cellular automaton](https://en.wikipedia.org/wiki/Asynchronous_cellular_automaton).

A cellular automaton is a model of a system of “cell” objects with the following characteristics.

* The cells live on a grid. A cellular automaton can exist in any finite number of dimensions.
* Each cell has a state. The number of state possibilities is typically finite. The simplest example has the two possibilities of 1 and 0 (otherwise referred to as “on” and “off” or “alive” and “dead”).
* Each cell has a neighborhood. This can be defined in any number of ways, but it is typically a list of adjacent cells.

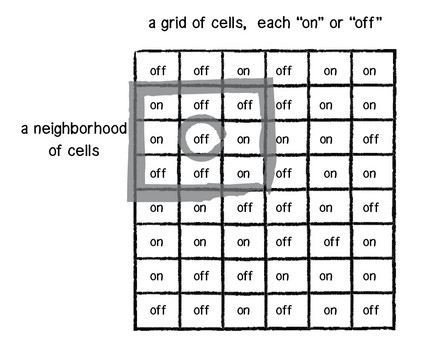


Figure 1: Example of cellular automata

1-Dimension

**Grid**. The simplest grid would be one-dimensional: a line of cells.

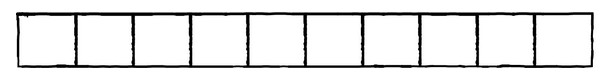


Figure 2: Example of 1-Dimension grid

**States**. The simplest set of states (beyond having only one state) would be two states: 0 or 1.

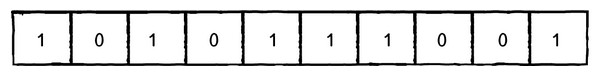


Figure 3: Example of 1-Dimension States

**Neighborhood**. The simplest neighborhood in one dimension for any given cell would be the cell itself and its two adjacent neighbors: one to the left and one to the right.

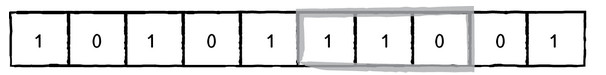


Figure 4: Example of 1-Dimension neighborhood

A cell’s new state is a function of all the states in the cell’s neighborhood at the previous moment in time (or during the previous generation). We calculate a new state value by looking at all the previous neighbor states.

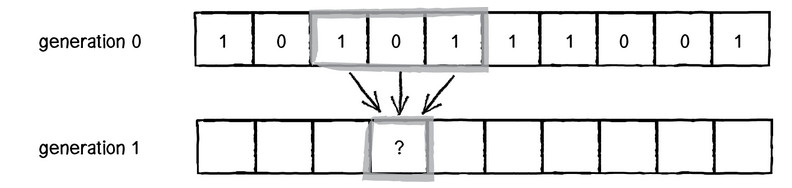


Figure 5:Example of change in next generation of 1-Dimension

We have three cells, each with a state of 0 or 1 so possible ways we can configure the states are 8. Let’s have a look.

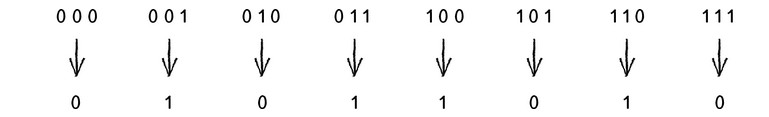


Figure 6: Examples of 1-Dimension possible states

At the end we have a rule set for example.

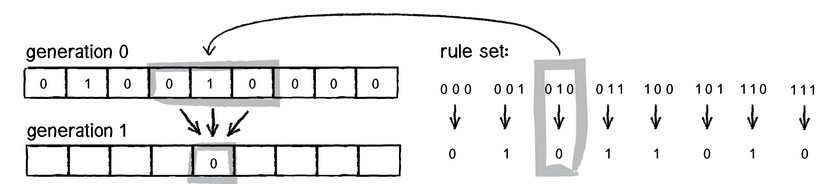


Figure 7: Example of 1-Dimension rule sets

2-Dimension

One way to simulate a two-dimensional cellular automaton is with an infinite sheet of [graph paper](https://en.wikipedia.org/wiki/Graph_paper) along with a set of rules for the cells to follow. Each square is called a "cell" and each cell has two possible states, black and white. The neighborhood of a cell is the nearby, usually adjacent, cells. The two most common types of neighborhoods are the [von Neumann neighborhood](https://en.wikipedia.org/wiki/Von_Neumann_neighborhood) and the[.](https://en.wikipedia.org/wiki/Moore_neighborhood) The former, named after the founding cellular automaton theorist, consists of the four [orthogonally](https://en.wikipedia.org/wiki/Orthogonal) adjacent cells. The latter includes the von Neumann neighborhood as well as the four diagonally adjacent cells. For such a cell and its Moore neighborhood, there are 512 (= 29) possible patterns. For each of the 512 possible patterns, the rule table would state whether the center cell will be black or white on the next time interval. [Conway's Game of Life](https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life) is a popular version of this model. Another common neighborhood type is the extended von Neumann neighborhood, which includes the two closest cells in each orthogonal direction, for a total of eight. The general equation for such a system of rules is ks, where k is the number of possible states for a cell, and s is the number of neighboring cells (including the cell to be calculated itself) used to determine the cell's next state. Thus, in the two-dimensional system with a Moore neighborhood, the total number of automata possible would be 29, or 1.34×10154

It is usually assumed that every cell in the universe starts in the same state, except for a finite number of cells in other states; the assignment of state values is called a configuration. More generally, it is sometimes assumed that the universe starts out covered with a periodic pattern, and only a finite number of cells violate that pattern. The latter assumption is common in one-dimensional cellular automata.

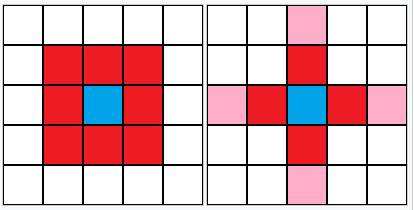


Figure 8: Example of 2-Dimension cellular automata

3-Dimension and further

When the dimension is increased, the number of rules is risen too. For such a cell with 2 states and if we take all the neighborhood cells, then there are 227 possible patterns. However, in this simulator, we only focus on 2-Dimension.

#### Stochastic Cellular Automata

Stochastic cellular automata or probabilistic cellular automata are an important extension of [cellular automaton](https://en.wikipedia.org/wiki/Cellular_automaton). Cellular automata are a discrete-time [dynamical system](https://en.wikipedia.org/wiki/Dynamical_system) of interacting entities, whose state is discrete.

The state of the collection of entities is updated at each discrete time according to some simple homogeneous rule. All entities' states are updated in parallel or synchronously. Stochastic Cellular Automata are CA whose updating rule is a [stochastic](https://en.wikipedia.org/wiki/Stochastic) one, which means the new entities' states are chosen according to some probability distributions. It is a discrete-time [random dynamical system](https://en.wikipedia.org/wiki/Random_dynamical_system). From the spatial interaction between the entities, despite the simplicity of the updating rules, [complex behaviour](https://en.wikipedia.org/wiki/Complex_system) may [emerge](https://en.wikipedia.org/wiki/Emergence) like [self-organization](https://en.wikipedia.org/wiki/Self-organization). As mathematical object, it may be considered in the framework of [stochastic processes](https://en.wikipedia.org/wiki/Stochastic_processes) as an [interacting particle system](https://en.wikipedia.org/wiki/Interacting_particle_system) in discrete-time.

The probabilistic automaton (PA) is a generalization of the [nondeterministic finite automaton](https://en.wikipedia.org/wiki/Nondeterministic_finite_automaton); it includes the probability of a given transition into the [transition function](https://en.wikipedia.org/wiki/Finite_state_machine), turning it into a [transition matrix](https://en.wikipedia.org/wiki/Stochastic_matrix). Thus, the probabilistic automaton generalizes the concept of a [Markov chain](https://en.wikipedia.org/wiki/Markov_chain) or [subshift of finite type](https://en.wikipedia.org/wiki/Subshift_of_finite_type). The [languages](https://en.wikipedia.org/wiki/Formal_language) recognized by probabilistic automata are called stochastic languages; these include the [regular languages](https://en.wikipedia.org/wiki/Regular_language) as a subset. The number of stochastic languages is [uncountable](https://en.wikipedia.org/wiki/Uncountable).

The probabilistic automaton may be defined as an extension of a [non-deterministic finite automaton](https://en.wikipedia.org/wiki/Non-deterministic_finite_automaton) (Q , Σ , δ , q0, F), together with two probabilities: the probability P of a particular state transition taking place, and with the initial state q0 replaced by a [stochastic vector](https://en.wikipedia.org/wiki/Stochastic_vector) giving the probability of the automaton being in a given initial state.

For the ordinary non-deterministic finite automaton, one has

* a finite [set](https://en.wikipedia.org/wiki/Set_(mathematics)) of states Q
* a finite set of [input symbols](https://en.wikipedia.org/wiki/Input_symbol) Σ
* transition function δ : Q × Σ → P (Q)
* a set of states F distinguished as accepting (or final) states F ⊂ Q.

Here, P (Q) denotes the [power set](https://en.wikipedia.org/wiki/Power_set) of Q.

#### Simulator

Based on the main concepts we define the scenario for the SCA as follows:

Simulator is based on 2-Dimension

There are several input values for SCA

Cell: containing size of cell (Height, Width)

State: containing the number of the states that user want to implement

Probability: That parameter used to verify the condition for the cell to know whether the cell is changed or not.

Neighbors: This parameter used to help user can express their rules

Rule: Denote the matrix as R, Rijk with i is the state of cell center, j is the state of the neighbor cells and k is destination state of the cell center which user want to implement. is a value that satisfy if the numbers of neighbor cells have the same state equate j when the cell center has state is equal i, then the cell center’s state changed to k.

Initialize: The button that initialize the first condition for the simulator

Start: button that kicks off the implementation

Stop: button that stops the implementation

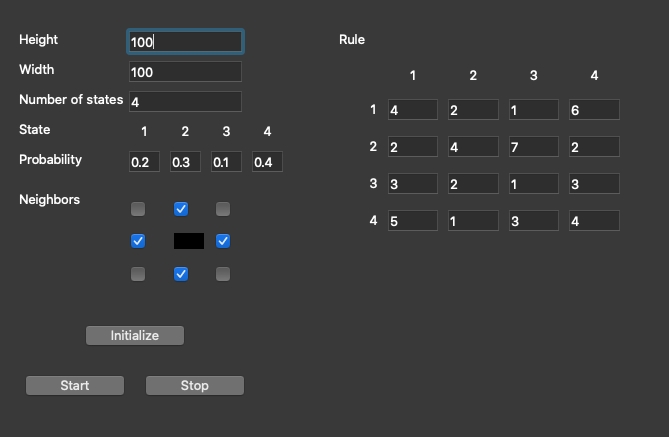


Figure 9: User Interface of the simulator

When user click on “Start” button on Rule Management screen. The system checks all cells, at each cell the system calls rule function (for example: rules(cell, neighbors, [probability]) and return of this function is the next state.

Denotes i is state of a cell, j is state of the neighbors and k is total state j of the neighbors. Rule(i,j,k) is a value of next state of i.

With each state of the cell at the time t-1, the rule function check all the states of the neighbors at time t-1 which has the state j, value of j from 1 to maximum of state (j is expected value of the neighbor cell that user wants in rule). When total of the neighbors having state j is k, then the Rule(i,j,k) is valid so the state of the cell at time t is value of Rule(i,j,k).

When the system has state i at the time t, it must verify state i with probability P, if it is valid then the next state will change at the time t, otherwise, the system keep the state the same as original.

#### Normal User Function

##### Create/Edit parameter for the simulator

* User can change the dimension of the simulator. The default dimension of the simulator is 2.
* User can change the size of the grid. The cell lives on the grid. The grid has maximum and minimum value and if user do not change, the grid takes the default value.
* Each cell has a state and user can change the state of the cell. The minimum value of the state is 2 (on or off, dead or alive, etc…)
* User can express and change the rule of the simulator.
* User can change the iteration of the simulator
* User can clear the graph to run a new rule.

##### View result of simulator

* User can view the result of the simulator on the graph.
* User can close the simulator.

## User Classes and Characteristics

*<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.>*

## Operating Environment

*<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>*

## Design and Implementation Constraints

*<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>*

## User Documentation

*<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>*

## Assumptions and Dependencies

*<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>*

## Apportioning of Requirements

<List of a*ll the requirements is optional. They can be deferred to the future or not necessary to implement* >

# Specific Requirements

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

## Functional Requirements Specification

### Function1 /Use-case 1

|  |  |
| --- | --- |
| **Use Case ID** | UC-1 |
| **Use Case Name** | Initialize parameters |
| **Actors** | User |
| **Description** | Start the simulator |
| **Pre-condition** | The simulator is started |
| **Post-condition** | N/A |
| **Trigger** | N/A |
| **Business Rules** | N/A |
| **Main flow** | |  |  |  | | --- | --- | --- | | **No.** | **Actor** | **Action** | | 1 | User | Click icon to open the simulator | | 2 | System | Display an user interface of the simulator with the default values | |  |  |  | |  |  |  | |
| **Alternative / Exception flow** | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | |  | | | |  |  |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |
| **Uses** | N/A |
| **Extends** | N/A |
| **Frequency** | High |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Mandatory** | **Read only** | **Control type** | | **Data Type** | **Length** |
| **Main** | | | | | | | |
|  |  | - | - |  | - | | - |
|  |  | - | - |  | - | | - |
|  |  | - | - |  | - | | - |
|  |  | - | - |  | - | | - |
|  |  | - | - |  | - | | - |

### Function2 /Use-case 2

|  |  |
| --- | --- |
| **Use Case ID** | UC-2 |
| **Use Case Name** | Express a new rule |
| **Actors** | User |
| **Description** | The system allows user to express a new rule |
| **Pre-condition** | N/A |
| **Post-condition** | N/A |
| **Trigger** | N/A |
| **Business Rules** | N/A |
| **Main flow** | |  |  |  | | --- | --- | --- | | **No.** | **Actor** | **Action** | | 1 | User |  | | 2 | System |  | | 3 |  |  | | 4 |  |  | |
| **Alternative / Exception flow** | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | |  | | | | 1 | User |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |
| **Uses** | N/A |
| **Extends** | N/A |
| **Frequency** | High |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Mandatory** | **Read only** | **Control type** | | **Data Type** | **Length** |
| **Main** | | | | | | | |
|  |  | - | - |  |  | | - |
|  |  | - | - |  | - | | - |
|  |  |  |  |  |  | |  |
|  |  |  |  |  |  | |  |

### Function3 /Use-case 3

|  |  |
| --- | --- |
| **Use Case ID** | UC-3 |
| **Use Case Name** | Edit rule (advanced function) |
| **Actors** | User |
| **Description** | The system allows user to edit a rule |
| **Pre-condition** | N/A |
| **Post-condition** | N/A |
| **Trigger** | N/A |
| **Business Rules** | N/A |
| **Main flow** | |  |  |  | | --- | --- | --- | | **No.** | **Actor** | **Action** | | 1 | User |  | | 2 | System |  | | 3 | User |  | | 4 | System |  | |
| **Alternative / Exception flow** | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | |  | | | | 1 | User |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |
| **Uses** | N/A |
| **Extends** | N/A |
| **Frequency** | High |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Mandatory** | **Read only** | **Control type** | | **Data Type** | **Length** |
| **Main** | | | | | | | |
| Rule name | Name of rule | - | - | Label | String | | - |
| Save |  | - | - | Button | - | | - |
| Reset |  |  |  | Button |  | |  |
| Edit |  |  |  | Button |  | |  |

### Function4 /Use-case 4

|  |  |
| --- | --- |
| **Use Case ID** | UC-4 |
| **Use Case Name** | Delete rule (advanced function) |
| **Actors** | User |
| **Description** | The system allows user to delete a rule |
| **Pre-condition** | N/A |
| **Post-condition** | N/A |
| **Trigger** | N/A |
| **Business Rules** | N/A |
| **Main flow** | |  |  |  | | --- | --- | --- | | **No.** | **Actor** | **Action** | | 1 | User |  | | 2 | System |  | | 3 | User |  | | 4 | System |  | |
| **Alternative / Exception flow** | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | |  | | | |  |  |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |
| **Uses** | N/A |
| **Extends** | N/A |
| **Frequency** | High |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Mandatory** | **Read only** | **Control type** | | **Data Type** | **Length** |
| **Main** | | | | | | | |
| Delete |  |  |  | Button |  | |  |
|  |  |  |  |  |  | |  |
|  |  |  |  |  |  | |  |
|  |  |  |  |  |  | |  |
|  |  |  |  |  |  | |  |
|  |  |  |  |  |  | |  |
|  |  |  |  |  |  | |  |

### 

### Function5 /Use-case 5

|  |  |
| --- | --- |
| **Use Case ID** | UC-5 |
| **Use Case Name** | View rule (advanced function) |
| **Actors** | User |
| **Description** | The system allows user to view content of a specific rule |
| **Pre-condition** | N/A |
| **Post-condition** | N/A |
| **Trigger** | N/A |
| **Business Rules** | N/A |
| **Main flow** | |  |  |  | | --- | --- | --- | | **No.** | **Actor** | **Action** | | 1 | User |  | | 2 | System |  | |  |  |  | |  |  |  | |
| **Alternative / Exception flow** | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | |  | | | |  |  |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |  | | | |  |  |  | |  |  |  | |
| **Uses** | N/A |
| **Extends** | N/A |
| **Frequency** | High |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Mandatory** | **Read only** | **Control type** | | **Data Type** | **Length** |
| **Main** | | | | | | | |
| Rule name | Name of rule | - | - | Label | String | | - |
| View |  |  |  | Button |  | |  |
|  |  |  |  |  |  | |  |
|  |  |  |  |  |  | |  |

* + 1. **Function6 /Use-case 6**

|  |  |
| --- | --- |
| **Use Case ID** | UC-6 |
| **Use Case Name** |  |
| **Actors** | User |
| **Description** |  |
| **Pre-condition** |  |
| **Post-condition** | N/A |
| **Trigger** | N/A |
| **Business Rules** | N/A |
| **Main flow** | |  |  |  | | --- | --- | --- | | **No.** | **Actor** | **Action** | | 1 | User |  | | 2 | System |  | |  |  |  | |  |  |  | |
| **Alternative / Exception flow** | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | |  | | | |  |  |  | |  |  |  | |
| **Uses** | N/A |
| **Extends** | N/A |
| **Frequency** | High |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Mandatory** | **Read only** | **Control type** | | **Data Type** | **Length** |
| **Main** | | | | | | | |
| Start |  |  |  | Button |  | |  |

* + 1. **Function7 /Use-case 7**

|  |  |
| --- | --- |
| **Use Case ID** | UC-7 |
| **Use Case Name** |  |
| **Actors** | User |
| **Description** |  |
| **Pre-condition** |  |
| **Post-condition** | N/A |
| **Trigger** | N/A |
| **Business Rules** | N/A |
| **Main flow** | |  |  |  | | --- | --- | --- | | **No.** | **Actor** | **Action** | | 1 | User |  | | 2 | System |  | |
| **Alternative / Exception flow** | |  |  |  | | --- | --- | --- | | **No** | **Actor** | **Action** | |  | | | |  |  |  | |  |  |  | |
| **Uses** | N/A |
| **Extends** | N/A |
| **Frequency** | High |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Mandatory** | **Read only** | **Control type** | | **Data Type** | **Length** |
| **Main** | | | | | | | |
| Stop |  |  |  | Button |  | |  |

## Non-Functional Requirements Specification

### External Interface Requirements

#### User Interfaces

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

#### Hardware Interfaces

*<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>*

#### Software Interfaces

*<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>*

#### Communications Interfaces

*<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>*

### Other Nonfunctional Requirements

#### Performance Requirements

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.*

*This* section lists all requirements whose ***Category*** has been stated in the goal model to be ‘**performance’***>*

#### Design constraints

*<This section lists all requirements whose* **Category** *has been stated in the goal model to be ‘****development’*** *or ‘****architecture’*** *(see Figure 7.5)*

#### Safety Requirements

*<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.*

*This section* lists all requirements whose ***Category*** has another value e.g. ‘**safety’**

*>*

#### Security Requirements

*<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>*

#### Software Quality Attributes

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.*

*This section* lists all requirements whose ***Category*** has another value e.g. **‘accuracy**’ *>*

# Other Requirements

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

**Appendix A: Glossary**

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: Issues List**

*< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>*