# Conway’s

## Game of life

## Technical Analysis

### Wael Almattar

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DOCUMENT METRIC** | | | | | | | | | |
| Project: | Conway’s game of life | | Company: | | | | WUT | | |
| Name: | Individual Project simulating Conway’s game of life. | | | | | | | | |
| Topic: | Conway’s game of life | | | | | | | | |
| Author: | Wael Almattar | | | | | | | | |
| File: | IPBusinessAnalysisWaelAlmattar.pdf | | | | | | | | |
| Version no: | 1.6 | Status: | | Working | | Opening Date: | | | 3/11/2015 |
| Summary: | The main purpose of this document is to provide a business analysis overview on ‘Conway’s game of life’ application. | | | | | | | | |
| Authorized by: | Prof. nzw. dr hab. inż. Lucjan Stapp | | | | Last Modification Date: | | | 3/15/2015 | |

|  |  |  |  |
| --- | --- | --- | --- |
| **History of changes** | | | |
| Version | Date | Who | Description |
| 1.0 | 3/11/2015 | Wael Almattar | Creating the document. |
| 1.1 | 3/11/2015 | Wael Almattar | Adding metrics. |
| 1.2 | 3/12/2015 | Wael Almattar | Adding Algorithm Used |
| 1.3 | 3/12/2015 | Wael Almattar | Adding Data Structure & methodology |
| 1.4 | 3/13/2015 | Wael Almattar | Adding Class Diagram |
| 1.5 | 3/13/2015 | Wael Almattar | Adding States Diagram |
| 1.6 | 3/15/2015 | Wael Almattar | Final editing |

# Contents

1. Abstract 4
2. Summary
   1. Overview 4
3. Requirements
   1. Functional Requirements 4
   2. Reliability 5
   3. Usability 5
   4. Performance 5
   5. Security 5
4. User Stories 6
5. Mock-up 7
6. End 8

# 1. Abstract

The aim of this project is to design and implement graphical simulator of well-known cellular automaton variable functions. It will perform complex mathematical calculations. In the business analysis we will limit ourselves to the modelling of dynamic behavior. We will make use of user stories to present main functionalities and priorities of the project. In the supplementary specification we will describe how the application should work. In the last section we will describe GUI design.

# 2. Algorithm Used

## a. Overview

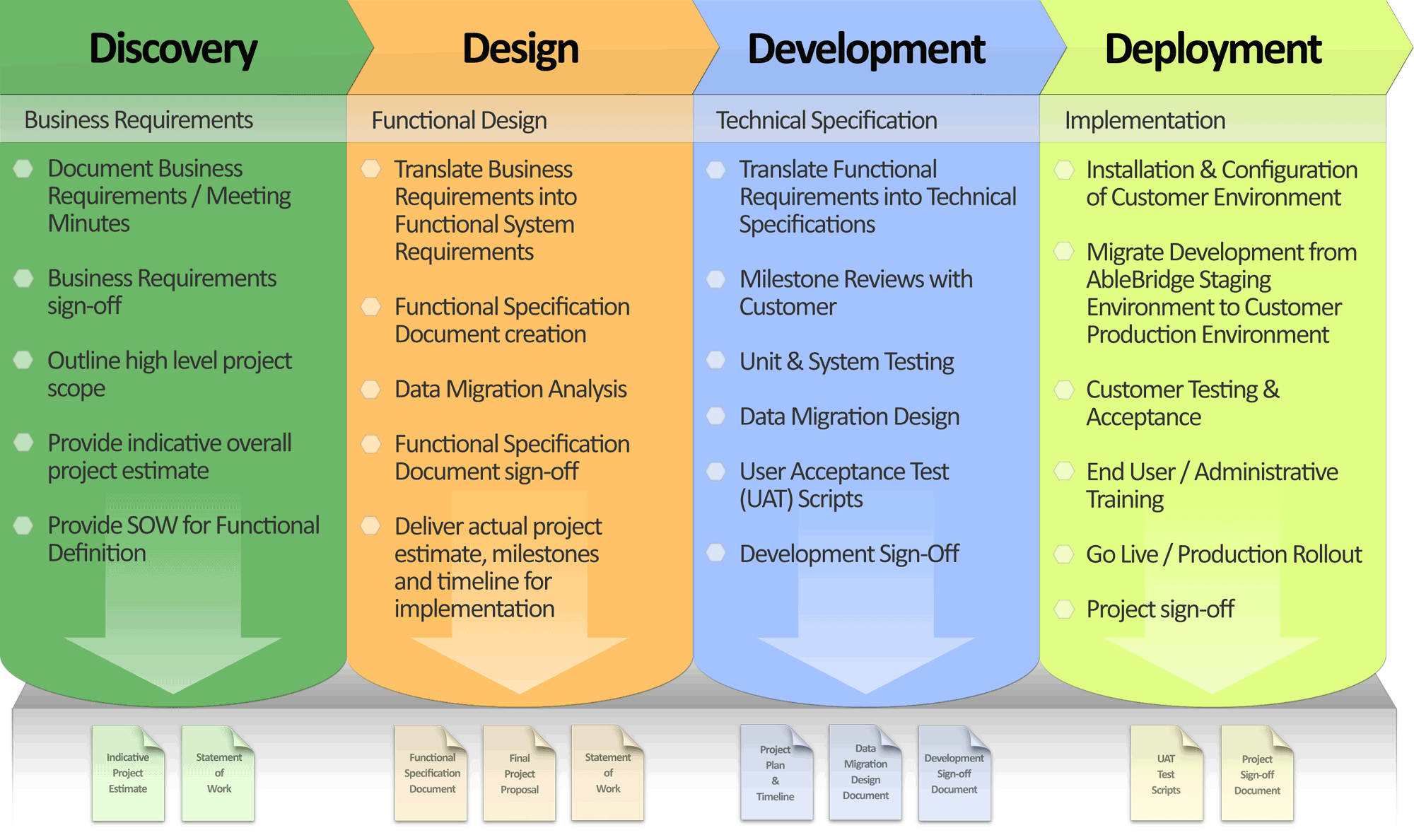
* Firstly Import the grid from a file and present it to be visible, where status are already settled by numbers, 1 for alive and 0 for dead cell.
* Load the rule set from second element file showing the rules, each rule on one line, rule set are exist between 'START' and 'END' words. The best solution is to load this rule set by diminution array.
* During the step, the program will 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.
* Each new grid state should be calculated asynchronously, by multiple threads/tasks, so that computation time will be minimized.
* We will go in a loop to pass by all cells row by row, for each row rules will be checks to show the next status of this cell:

1. Avoid getting stack overflow error by checking if the cell in the corner and and supposed its non­exist neighbors as dead.
2. Check each rule which will possibly make the dead cell alive.
3. Check each rules which will possibly make the alive cell dead.
4. Update the status of the cell in new array until we reach the last cell in the array.
5. Copy the final grid to our original data structure grid (array).

# 3. Implementation Methodology

Our implementation methodology consist of four steps:

* Discovery
* Design
* Development
* Deployment
* Testing



# 4. Data structures used

* Dimensional array will be implemented to present the grid of cells. This array will load the initial states and the last one to store on the hard drive data regarding the final state.
* For each rules will be denoted as class created consist of:

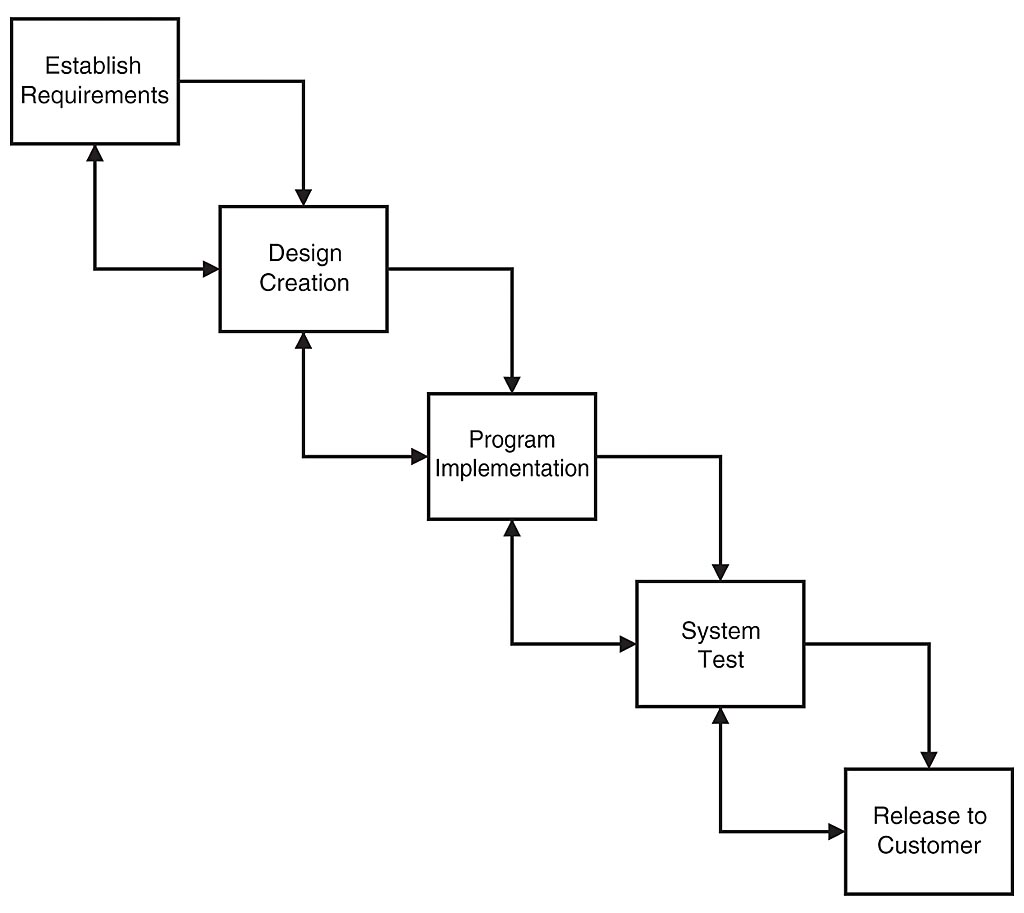
1. Dimensional array present the sub grid for the rules implemented.
2. Boolean variable to denoted 'death' or 'alive' state for cell after checking the rules.
3. String 'At most’, 'at least' or 'exactly'.

* which means for N sub rules we have N sub class created.

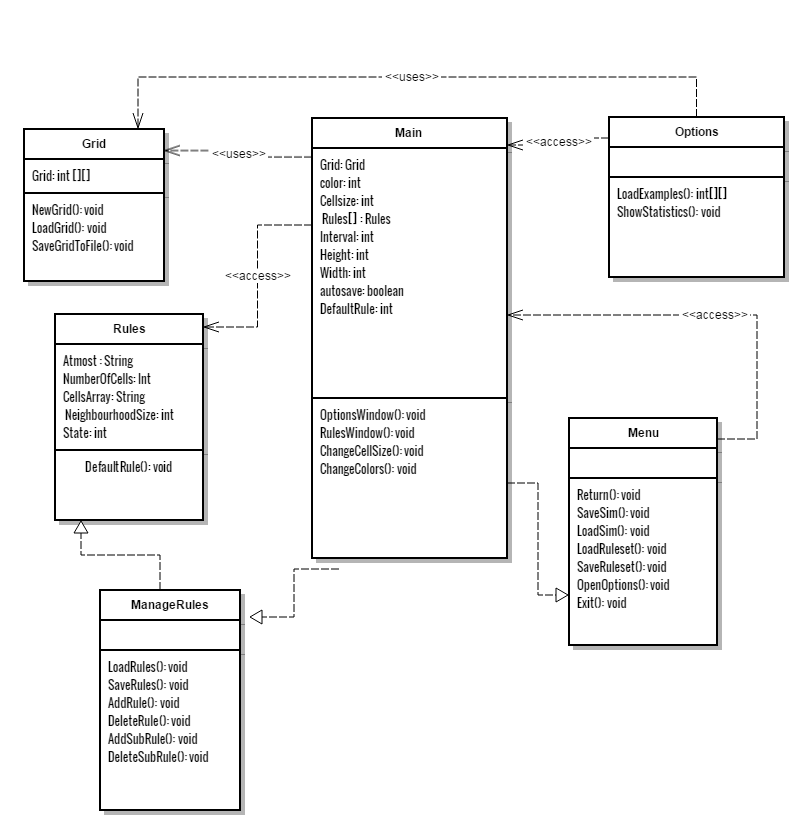
# 5. Development strategy

## a. Waterfall Model

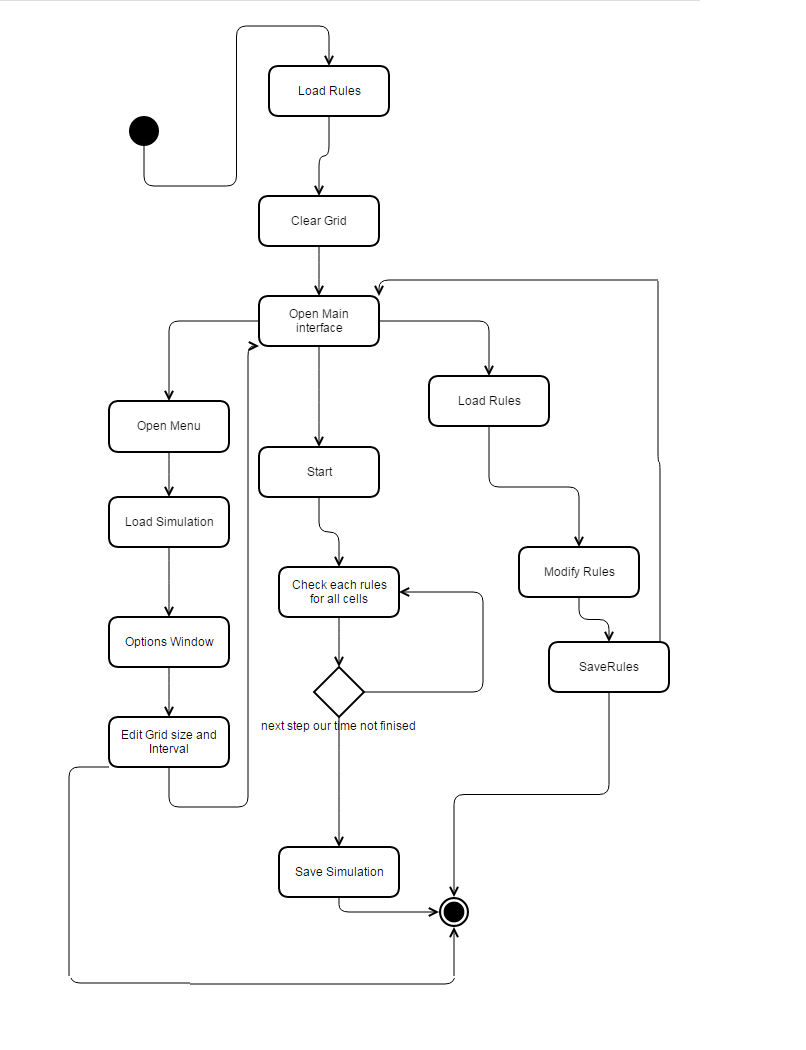
**Definition Study / Analysis:** During this phase research is being conducted which includes brainstorming about the software, what it is going to be and what purpose is it going to fulfill.  
**Basic Design:** If the first phase gets successfully completed and a well thought out plan for the software development has been laid then the next step involves formulating the basic design of the software on paper.  
**Technical Design:**  After the basic design gets approved, then a more elaborated technical design can be planned. Here the functions of each of the part are decided and the engineering units are placed for example modules, programs etc.  
**Construction / Implementation:** In this phase the source code of the programs is written.  
**Testing:** At this phase, the whole design and its construction is put under a test to check its functionality. If there are any errors then they will surface at this point of the process.



# 6. Class Diagram



# 7. States Diagram



# 6. End

Conway’s game of life is a user friendly light weighted offline application designed to provide the easiest way to simulate the life of cells and observe it each step. Conway’s game of life needs no high performance PCs to operate. The previously shown mockup is just a preview of how its GUI may look, can be changed or updated according to what may be more comfortable to the user.