# User Requirements Document

## Team T

## February 3rd 2016

| Name | ID Number |
|---|---|
| Lenmor Larroza Dimanalata | 27699727 |
| Andrea Julia Biondi | 26456251 |
| Jonathan Roux | 26978185 |
| Andres Kebe | 26638422 |
| Matin Nafekh | 27423993 |
| Sebastien Seguier | 27010699 |
| Erick Hernandez | 27083181 |
| Haochen Wang | 26414249 |

Table 1: Team

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 19/01/2016 | 1.0 | Document Start | Jonathan Roux |
| 23/01/2016 | 1.1 | Use Cases added | Jonathan Roux |
| 24/01/2016 | 1.2 | Use Case Diagram | Jonathan Roux |
| 27/01/2016 | 1.3 | Domain Concepts | Jonathan Roux |
| 27/01/2016 | 1.4 | User Groups, Functional Req. | Andrea Julia Biondi |
| 28/01/2016 | 1.5 | Functional Req., System Overview | Andrea Julia Biondi |
| 01/02/2016 | 1.6 | Brief modifications (Use Cases, Requirements) | Andrea Julia Biondi |
| 02/02/2016 | 1.7 | Modification in functional req., References | Andrea Julia Biondi |
| 02/02/2016 | 1.8 | 2 new use cases and update use case diagram | Jonathan Roux |

# Contents

# 1 Purpose of the document

The purpose of this document is to describe all the necessary requirements for the development of a 1D-2D GCHQ Puzzle Application. It is intended for review by the Britain's security and intelligence organization GCHQ

| Group | Intended purposes |
|---|---|
| Puzzle makers and solvers | To help coders review the application |
| Coders | To know the different components for the application |
| Testers | To test whether the application meets the requirements |
| Writers of related documents | To use as a reference for future deliverable |

Table 2: Intended audience of this document

# 2 Business Goals

The 1D-2D GCHQ Puzzle is simple, but requires a fair amount of computation to obtain and solve. The main goals of this project can be separated in two categories: a tool to make puzzles with different settings, and a GUI with many options for puzzle solvers.

A puzzle is meant to be challenging and since different users prefer different challenges, a puzzle maker will be able to create puzzles of different sizes and level of difficulty. Also, the assignment of priorities can be applied to rows and columns to help the puzzle solvers. Moreover, the puzzle maker will be able to inform the puzzle solvers when some row or column has all its conditions met.

From the perspective of a puzzle solver, the application will be an intuitive GUI that will allow easy visualisation, manipulation and modification of the puzzle. One of the principal functions of this program is that it will determine if the current grid is a solution or not. Also, given some state of the grid the puzzle solver will be able to consider different solutions simultaneously.

# 3 Domain Concepts

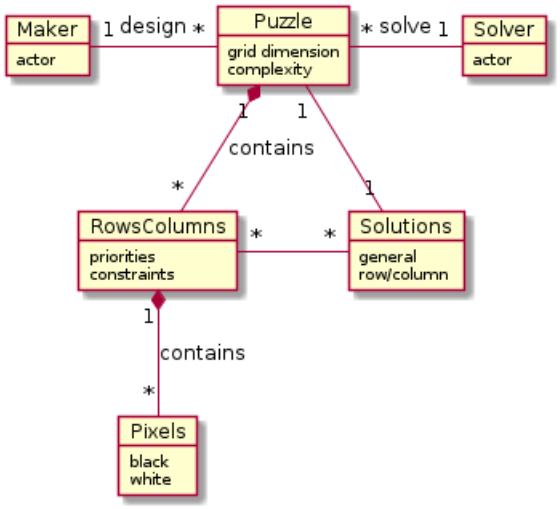


Figure 1: UML Class Diagram

| **Concept** | **Description** |
|---|---|
| Puzzle | This is the central concept of this project. All other domain concepts are related in some way to the puzzle concept. It is composed of rows and columns which are determined by the grid dimension attribute. The complexity attribute describe how many starting pixels will be available to the puzzle solver. |
| Rows and Columns | The puzzle can be decomposed into rows and columns, each one with its own constraints and priorities. The constraints determine the number and length of black pixel runs. The priorities is extra information about which row or column to solve first to help the puzzle solver. |
| Pixels | This is the most basic component of the puzzle and is either black or white. The goal is to determine which pixels are black so all the constraints are satisfied. |
| Solution | This is the configuration of the puzzle when all the constraints for each rows and columns are satisfied. There is also intermediary solution, when a configuration is satisfied only for a specific row or column. More than one solution is possible for a given puzzle. |
| Puzzle Maker | This is the actor who will determine the configuration of the puzzle: dimension, complexity, constraints and priorities on rows and columns. |
| Puzzle Solver | This is the actor who will try to find the initial configuration of the puzzle given some initial black pixels and respecting the constraints. Some extra information about priorities on some rows or columns may also be available. |

Table 3: Main Domain concept of the 1D-2D Puzzle

# 4 System overview

This system allows users to create or solve puzzles. These users will be using the systems' GUI in order to display the features they can use.

The puzzle maker is allowed to create a puzzle which can vary of size and difficulty. While the puzzle is being created, the system can allow different priorities to be set for the puzzle. In addition to these features, the system can identify different possibilities for a puzzle solution.

The puzzle solver has features that include assigning black or white pixels in the grid of the puzzle, verifying if the grid is a correct solution, and manipulating certain regions to maximize their ability to find a solution.

# 5 User groups

| **Users** | **Description** | **Number of Users** |
|---|---|---|
| Puzzle Maker | Those who create the puzzles | 1 |
| Puzzle Solver | Those who solve the puzzles | 1 |

Table 4: Users of the system

# 6 Functional requirements

| ID | Feature | Requirement | Rationale | Priority | Use Case |
|---|---|---|---|---|---|
| 01 | Opening | Any user shall be able to open the application | There is no login required | Must | 1 |
| 02 | Opening | User shall be able to choose between creating and solving puzzle | User can be both maker and solver, but only one at a time | Must | 1 |
| 03 | Creating | Puzzle maker shall be able to set number of rows and columns | User adjusts the size based on level of difficulty | Must | 2 |
| 04 | Creating | Puzzle maker shall be able to set black pixels in puzzle | The pixels are placed in order to complete the puzzle | Must | 2 |
| 05 | Creating | Puzzle maker shall be able to modify the size of black regions | Constraints will be added according to the black regions | Must | 3 |
| 06 | Confirmation | The system must be able to confirm if a puzzle has been solved quickly | An optional feature that is used by the puzzle maker | Must | 3 |
| 07 | Creating | User shall be able to add priorities to rows/columns | The priorities guide the puzzle solver | Must | 4 |
| 08 | Solving | Puzzle solver shall be able to identify different possibilities of solutions | Allows the solver to find new solutions | Must | 5 |
| 09 | Solving | Puzzle solver shall be able to manipulate specific regions in solution | User can drag regions in different directions to change solution | Must | 6 |

Table 5: Functional

# 7 Non-functional requirements (quality requirements, properties)

| ID | Requirement | Rationale | Priority | Use Case |
|---|---|---|---|---|
| 01 | Portability | As many users as possible must be able to play the game | Must | - |
| 02 | Response Time | Each time the puzzle solver click a pixels the system has to check if the constraints are satisfied | Must | 3 |
| 03 | Accessibility | Puzzles of different sizes and complexity must be available to the puzzle solver | Must | 2 |
| 04 | Performance | Regardless of the size of the puzzle, the system must check the solution under a second | Must | 3 |
| 05 | Reliability | Neither the solver or the maker want the application to crash | Must | All |
| 06 | Extensibility | New ideas from users must be able to be easily implemented to the system | Must | - |

Table 6: Non-Functional Requirements

# 8 Use Cases

## 8.1 Overview

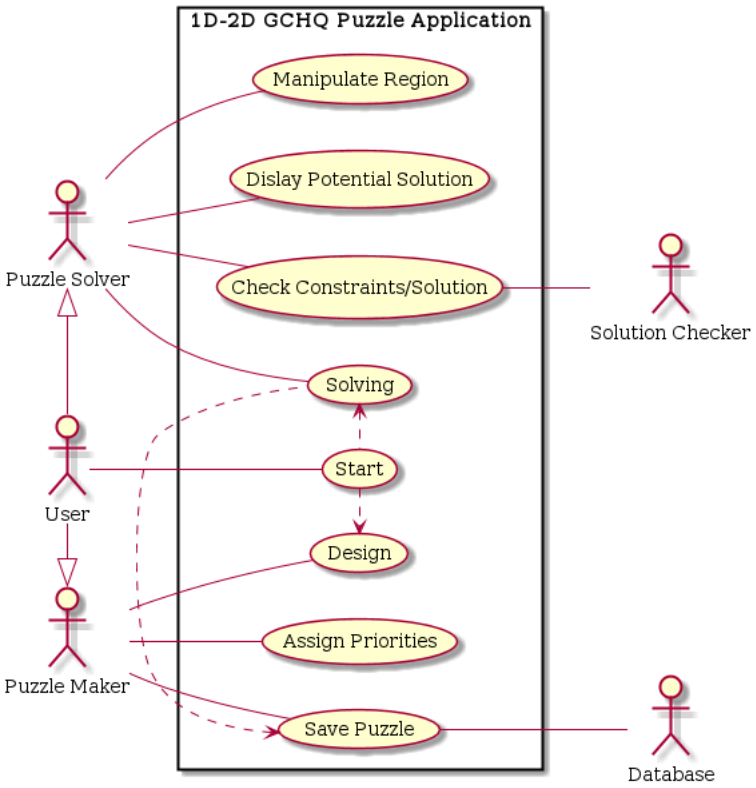


Figure 2: Use Case Diagram

## 8.2 Use Case 1

| | |
|---|---|
| **Name** | Start |
| **Summary** | The user open the application. |
| **Actors** | Any user. |
| **Preconditions** | - |
| **Basic Sequence** | 1. A user starts the application.<br>2. The user decides whether he wants to make or solve puzzles by clicking on the right button. |
| **Exceptions** | - |
| **Post Conditions** | A user will be either a puzzle maker or puzzle solver. |
| **Priority** | Must |
| **Status** | Implemented |
| **Traces to requirements** | - |
| **Traces to test cases** | reference to test cases |

## 8.3 Use Case 2

| | |
|---|---|
| **Name** | Design |
| **Summary** | The puzzle maker creates a puzzle. |
| **Actors** | Puzzle maker |
| **Preconditions** | The user is a puzzle maker starting the creation of a new puzzle. |
| **Basic Sequence** | 1. The puzzle maker is requested to enter the number of rows and columns for its next puzzle.<br>2. The system generates an empty grid using the specified number of rows and columns.<br>3. The puzzle maker chooses which pixels are black or white by clicking them.<br>4. When done, the puzzle maker is requested to determine the black pixels that will be available to the puzzle solver as hints. |
| **Exceptions** | 1a. If the number of rows or columns is too large, the message "Number of Rows/Columns too large" will be displayed<br>1b. If the number of rows of columns is too small, the message "Number of Rows/Coumns must be equal or greater than 4" will be displayed |
| **Post Conditions** | A new puzzle was created. |
| **Priority** | Must |
| **Status** | Implemented |
| **Traces to requirements** | User Story #1 |
| **Traces to test cases** | reference to test cases |

## 8.4 Use Case 3

| | |
|---|---|
| **Name** | Solving |
| **Summary** | The puzzle solver solves a puzzle. |
| **Actors** | The puzzle solver, Solution Checker |
| **Preconditions** | The user is a puzzle solver. |
| **Basic Sequence** | 1. The puzzle solver chooses a puzzle by size and complexity.<br>2. The puzzle solver can modify the color of the pixels to either black or white by clicking them.<br>3. The puzzle solver can click the button "Check", when he thinks he has the right answer.<br>4. The solution checker has to check if the configuration of the grid satisfies all the constraints of the puzzle.<br>5. The system congratulates the puzzle solver. |
| **Exceptions** | 3a. If the configuration of the grid is not right, the puzzle solver can keep on trying and re-submit later. |
| **Post Conditions** | The puzzle solver solved a puzzle. |
| **Priority** | Must |
| **Status** | Implemented |
| **Traces to requirements** | User Story 2,3 |
| **Traces to test cases** | reference to test cases |

## 8.5 Use Case 4

| | |
|---|---|
| **Name** | Assign Priorities |
| **Summary** | The puzzle maker can decide to assign priorities to rows/columns to guide the puzzle solver. |
| **Actors** | Puzzle Solver, Solution Checker |
| **Preconditions** | The puzzle maker designs a puzzle. |
| **Basic Sequence** | 1. While designing the puzzle, the puzzle maker can decide to give priorities to some rows/columns to be solved by the puzzle solver.<br>2. The solution checker will analyse the puzzle and determine which rows/columns should be solved first by the puzzle solver. |
| **Exceptions** | 2a. If the puzzle maker updates the puzzle, he should reassign priorities to rows/columns according to the new configuation. |
| **Post Conditions** | The puzzle maker can keep on designing the puzzle. |
| **Priority** | Must |
| **Status** | In progress |
| **Traces to requirements** | User Story #4 |
| **Traces to test cases** | reference to test cases |

## 8.6 Use Case 5

| | |
|---|---|
| **Name** | Display Potential Solutions |
| **Summary** | The puzzle solver can see the different solutions for a given row or column. |
| **Actors** | Puzzle solver |
| **Preconditions** | The puzzle solver is working out the puzzle. |
| **Basic Sequence** | 1. The puzzle solver can select an entire row or column.<br>2. The system identifies and displays the different possibilities for black pixels for this particular row or column according to the current configuation of the puzzle. |
| **Exceptions** | 1a. |
| **Post Conditions** | The puzzle solver may try one of displayed solutions for the given row/column. |
| **Priority** | Must |
| **Status** | In progress |
| **Traces to requirements** | User Story #5 |
| **Traces to test cases** | reference to test cases |

## 8.7 Use Case 6

| | |
|---|---|
| **Name** | Manipulate Region |
| **Summary** | The puzzle solver can move entire regions of black pixels horizontally for a row or vertically for a column. |
| **Actors** | Puzzle solver |
| **Preconditions** | The puzzle solver is working out the puzzle. |
| **Basic Sequence** | 1. The puzzle solver can click on a black pixel.<br>2. The system can recognize if the clicked pixels has other adjacent black pixels forming a block.<br>3. The puzzle solver can drag this block either horizontally or vertically and permanently change the position of the whole block. |
| **Exceptions** | 1a. |
| **Post Conditions** | The puzzle solver can keep on working out the puzzle. |
| **Priority** | Must |
| **Status** | In progress |
| **Traces to requirements** | User Story #7 |
| **Traces to test cases** | reference to test cases |

## 8.8 Use Case 7

| | |
|---|---|
| **Name** | Check Constraints |
| **Summary** | The Solution Checker checks in real-time if some row or column constraints are satisfied. |
| **Actors** | The puzzle solver, Solution Checker |
| **Preconditions** | The user is a puzzle solver solving a puzzle. |
| **Basic Sequence** | 1. The puzzle solver clicks on a pixel, changing the configuration of the grid.<br>2. The Solution Checker highlights the constraints that are satisfied.<br>3. If some row or column have all their constraints satisfied, the entire row/column will be highlighted. |
| **Exceptions** | 2a. If a modification in the configuration of the grid loses a previously satisfied constraint, then the Solution Checker will remove the highlight on the constraint.<br>3a. If a modification in the configuration of the grid loses a previously satisfied row/column, then the Solution Checker will remove the highlight on the entire row/column. |
| **Post Conditions** | The puzzle solver can visualize the current status of its puzzle. |
| **Priority** | Optional |
| **Status** | In progress |
| **Traces to requirements** | User Story #6 |
| **Traces to test cases** | reference to test cases |

## 8.9 Use Case 8

| | |
|---|---|
| **Name** | Save Puzzle |
| **Summary** | The puzzle maker saves a puzzle to a database. |
| **Actors** | Puzzle maker, Database |
| **Preconditions** | The puzzle maker finished the creation of a new puzzle. |
| **Basic Sequence** | 1. The puzzle maker can save the configuration of the new puzzle.<br>2. The system will assign and id to the puzzle and save it in a database. |
| **Exceptions** | - |
| **Post Conditions** | A new puzzle was created and saved to a database. |
| **Priority** | Must |
| **Status** | In progress |
| **Traces to requirements** | User Story #1 |
| **Traces to test cases** | reference to test cases |

# 9 References

**User Stories:**

- **US 1:** As a puzzle maker, I want to create puzzles of different sizes and complexity with the application.
- **US 2:** As a puzzle solver, I want to assign black-white values to pixels in the grid of the puzzle.
- **US 3:** As a puzzle solver, I want to check whether my current grid is a solution or not.
- **US 4:** As a puzzle maker, I want to assign priorities to rows and columns of the puzzle grid to guide the puzzle solver
- **US 5:** As a puzzle solver, I want to simultaneously consider several potential solutions for a row (or a column).
- **US 6:** As a puzzle maker, I want to inform the puzzle solver when there is a row or column that fully determined by the current state of the puzzle grid.
- **US 7:** As a puzzle solver, I want to be able to manipulate entire regions in a potential solutions for a row (or a column) by dragging them horizontally (or vertically).