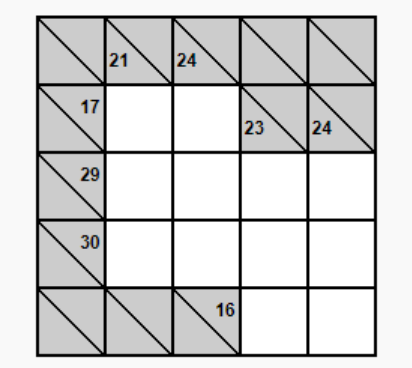
# Requirements Document

## Team PI-B

## 9 February 2020




**Kakuro**

Team members

| Name | ID Number |
|---|---|
| Sajib Ahmed | 40044867 |
| Yaroslav Bilodid | B |
| Jesse Desmarais | 4003576 |
| Antoine Farley | D |
| Marc Hegedus | 26242219 |
| Katerina Tambakis | 27010486 |
| Yingjie Zhou | 26433820 |

# 1 System

## 1.1 Introduction

The purpose of this document is to demonstrate the requirements needed to build the computer puzzle game Kakuro. Kakuro, typically a single player game, is a kind of logic puzzle that is often referred to as a mathematical transformation of the crossword puzzle. Kakuro is typically played on a 10x10 grid filled with black and white cells. The black cells contain a diagonal slash from upper-left to lower-right and a number in one or both halves, such that each horizontal entry has a number in the black half-cell to its immediate left and each vertical entry has a number in the black half-cell immediately above it. These numbers are commonly called clues. The objective of the puzzle is to insert a digit from 1 to 9 inclusive into each white cell such that the sum of the numbers in each entry matches the clue associated with it and that no digit is duplicated in any entry.

## 1.2 Purpose

The purpose of the Software Requirements Document describes the specification of the Kakuro puzzle game, which is in partial fulfillment of the requirements of COMP 354. This document will define the requirements of the user interface, the product functions, actors, non-functional constraints, data definition, and model for this application. The model will include use case diagrams and domain model UML diagrams. Furthermore, a detailed project plan will be provided, including the schedule of the upcoming phases. This document will serve as a basis for the upcoming phases for this project.

## 1.3 Context

This document addresses the requirements that will be used as a basis for the design phase. A number of figures will be provided to demonstrate how the game will appear at completion along with its special features such a difficulty level. The actors will include the target audience and environment required for this game. The model will demonstrate the use cases and its domain model UML diagram explaining the relationship of the actors with one another. Finally, a chart will include the breakdown of the amount of time logged into developing the project from all different tasks.

## 1.4 Business Goals

Our objective of developing kakuro is in relation to the growth of PI-B. We are thriving to fulfill the customer's needs so that we can expand our business portfolio in order to reach other customers' business. Moreover, we hope to achieve a high number of downloads in

order to catch the eye of advertisers that wish to participate in future projects that will be given to us.

# 2 Domain Concepts

## 2.1 Objectives

The project to be completed in COMP 354 of Winter 2020 is to create a functional replica of the puzzle game Kakuro. Team PIB's version in iteration one will consist of a singular solvable puzzle. Functionality will be limited to input from the client and feedback for the verification of a correct solution. The main objective is to apply software engineering techniques for the development process to be test-driven, agile, and object-oriented. There will be three iterations, each having their own deadlines. Efficient management and communication amongst our group is understood to be central in accomplishing the required tasks. The client wants the following as deliverables for the first iteration: a basic graphical user interface, a model-view-controller architecture coded in Java, and a categorized set of use cases.
The three iterations will each have a document to be handed in to the client. The information regarding their naming, deliverables, and dates are tabulated below:

| Iteration | Deliverable | Date |
|---|---|---|
| Requirement | Requirements Document | 2020/02/09 |
| Design | Design Document | 2020/03/15 |
| Implementation | Final Document | 2020/04/5 |

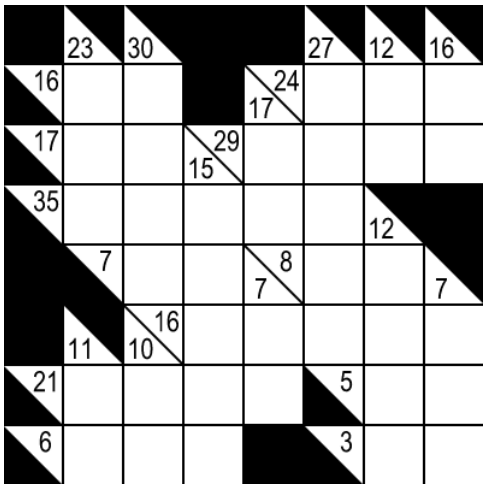
*Project Timeline*

### 2.1.1 Kakuro Puzzle Game Specifications

Kakuro is a japanese mathematical logic puzzle with rules similar to the crossword and sudoku. To develop a successful graphical user interface, the game's rules and characteristics are to be clearly detailed. This will allow clients and various actors to recognize the product for an intuitive feel. Due to the simplistic nature of the game, development will be focused on limiting user interaction and increasing response. This will be realized in the form of input validation. Users will interact with a single interface where numerals from one to nine are to be entered. If incorrect, a red translucent overlay will be superimposed upon the respective tile. A thorough description of Kakuro's mechanics is provided below.

#### 2.1.1.1 Gameboard

Kakuro's classic form is composed of a monochromatic square matrix where three different tiles are displayed. They are the following: blank, input, and clue. Blank tiles are black and cannot contain any value. Clients are to interact only with the input tile. Numerical values will be inserted to complete the sums found within the clue tiles. Clue tiles are bisected diagonally where the lower and higher sums must correspond to the addition of all the vertically and horizontally aligned numerals in the input tiles, respectively. Input tiles along a specified direction can only have unique values from 1 to 9. Their intersections must then contain the same numeral. As such, a solution is obtained when every sum has been rightfully achieved throughout the gameboard matrix. The figure below shows a Kakuro puzzle at the easy difficulty.




*Easy Kakuro gameboard*

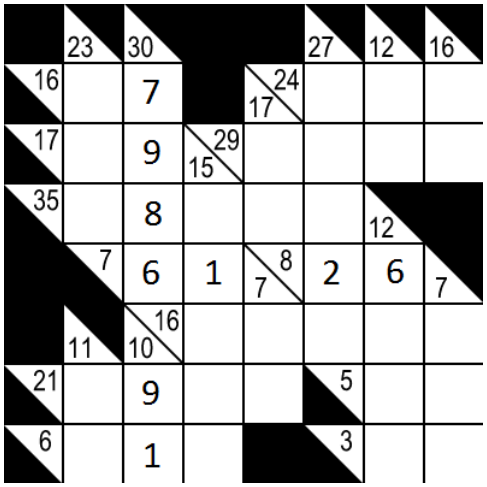
#### 2.1.1.2 Tiles

The three distinct types of tiles are shown in the figure below in the given order: blank, input, and clue. Blank tiles are always filled with black and no numeral is ever shown within its confines. Input tiles are white and users are to fill them with a number from one to nine. Clue tiles are bisected from the top left to the bottom right and come in three separate formats. If a corner is white a sum will shown, whereas if it's black none will appear. As such, both corners may be hold a sum, or only one of the two might.

*Tile variation as seen in Kakuro*

### 2.1.1.3 Sum and intersections

A vertical and horizontal column has been filled out with a possible solution to the gameboard shown in section 2.1.1.1. The top of the vertical column suggests a sum of 30 is needed from the input tiles vertically below. A combination of 9-8-7-6 must be placed within the four allotted slots. This vertical column had two clues where the second formed a break. Clue tiles that appear sequentially along a particular column represent a new sum to acquire from the input tiles. The second reads a sum of 10. A combination of 9-1, 8-2, 7-3, or 6-4 is allowed to be placed. The horizontal that has been filled shares a tile with the vertical column. This is called an intersection. The number in that input tile must be correct for both columns. The horizontal's first number is 7. Since a there lies a 6, only a 1 can be placed within the adjacent tile. The flow of the game concerns itself by correctly forming the proper sums and accommodating the many intersections between columns.




*Partially completed Kakuro gameboard*

#### 2.1.1.4 Solution

The final solution to the gameboard shown in section 2.1.1.1 is given below. Each of the following requirements has been met:

- NUMERALS Numbers used within the input tiles are limited from 1 to 9.
- SUM Every input tile along a column corresponds to its vertical and/or horizontal clue tile.
- INTERSECTION Every intersection between vertical and horizontal columns had a common numeral that worked towards the respective sum.
- COMPLETION Every input tile has been correctly filled out in the gameboard matrix.

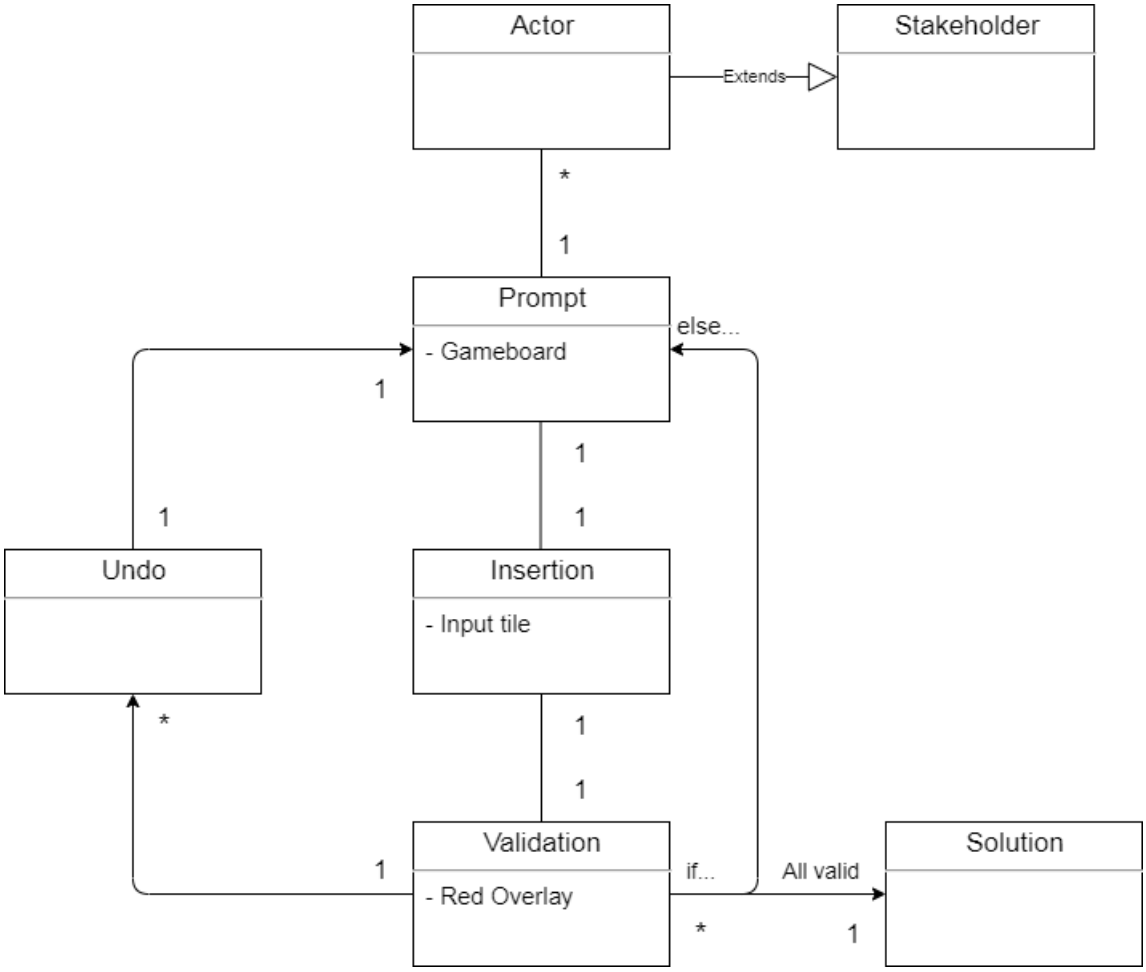
The difficulty is set to easy due to the clues being too telling of the numbers needing to be entered. Taking a look at the top left, there's only one variation of numbers which can be inserted. This forms a good base to complete the rest of the puzzle.

| | 23 | 30 | | | 27 | 12 | 16 |
|---|---|---|---|---|---|---|---|
| 16 | 9 | 7 | | 24 17 | 8 | 7 | 9 |
| 17 | 8 | 9 | 29 15 | 8 | 9 | 5 | 7 |
| 35 | 6 | 8 | 5 | 9 | 7 | 12 | |
| | 7 | 6 | 1 | 8 7 | 2 | 6 | 7 |
| | 11 | 16 10 | 4 | 6 | 1 | 3 | 2 |
| 21 | 8 | 9 | 3 | 1 | 5 | 1 | 4 |
| 6 | 3 | 1 | 2 | | 3 | 2 | 1 |

*Kakuro gameboard solution*

## 2.2 Domain Model UML

In test driven development progression is measured through the completion of use cases. The methodology is also aimed to be agile, favouring short cycles or iterations. For the first iteration base functionality is to be implemented. The following UML depicts several use cases that are currently part of the problem domain:




*Problem Domain UML*

# 3 Actors

Stakeholders are those who would show interest in a product, for example, through marketing. Actors form a subset from those that directly interact with the product. In the first iteration of the Kakuro puzzle, actors come in the given forms:

- Coders In a test driven design, code is tested in segments to implement functionality.
- Documenters Through communication and verification, a requirements document can be written as a deliverable to the client.
- Organizers To keep a steady purposeful work flow, one must keep track and organize the trajectory of the project by interacting with the coder's work.
- Quality Assurance The improvement of the code's structure and design is validated by an interaction.
- Players Once base functionality is attained, users may willing play through a puzzle.

# 4 Use Cases

In the first iteration, the user and system need only interact to the point where base functionality is achieved. The use case diagram below depicts the requirements and scope of the system.

Use cases 1 and 2 are system functions relayed to the player and should not need validation to operate. Whereas with use cases 3 and 4, information from the player is relayed to the system. For example, the former sends feedback to the player if the value entered is incorrect.

## 4.1 Overview

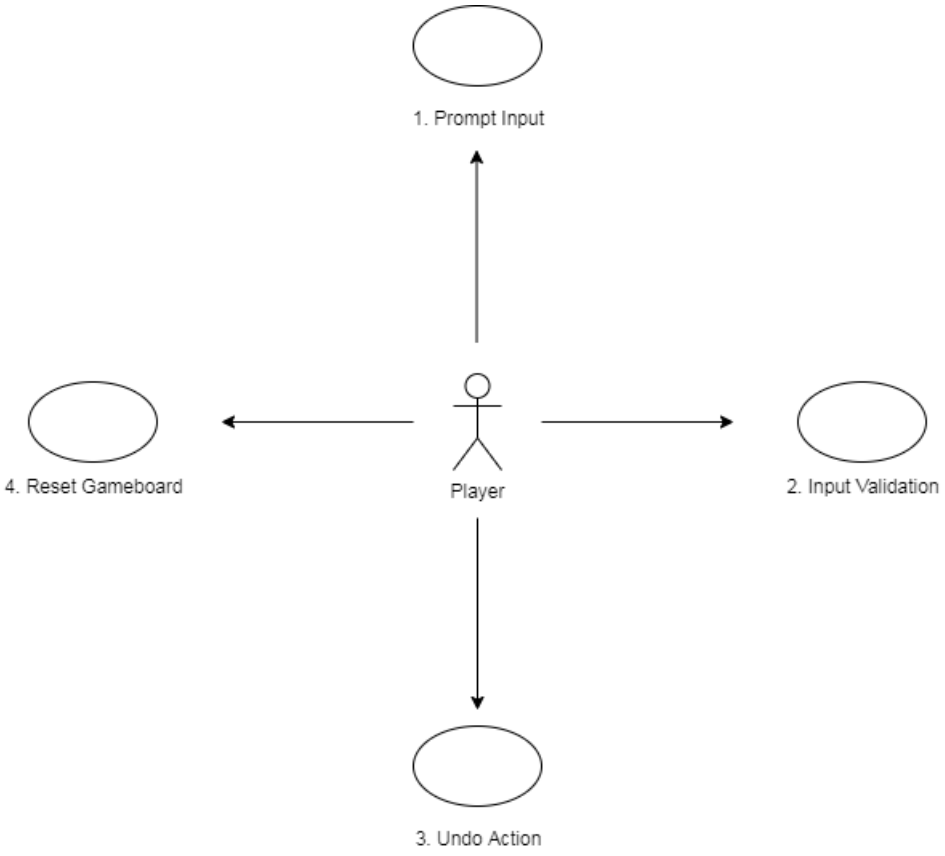



Diagram.png Diagram.png

*Use Case Diagram*

### 4.1.1 Use Case 1: Prompt Input

| | |
|---|---|
| Description | Signals user to enter values. |
| Summary | A Kakuro puzzle is displayed. |
| Actors | Player |
| Pre-conditions | • Player starts a game. |
| Main Scenario | **1.** Player starts a game.<br>**2.** A Kakuro puzzle is displayed.<br>**3.** Input tiles represent a location where numerals may be accepted. |
| Exceptions | None |
| Post-Conditions | • The Kakuro puzzle prompts input until a solution is found. |
| Priority | Must have |
| Traces to Test Cases | None |

### 4.1.2 Use Case 2: Input Value

| | |
|---|---|
| Description | Allows user to enter a numeral in an input tile. |
| Summary | User may input numerals in input tiles to form a solution. |
| Actors | Player |
| Pre-conditions | • Player's turn to act.<br>• Tile selected is an input tile. |
| Main Scenario | **1.** Player starts a game.<br>**2.** Player selects an input tile.<br>**3.** Player enters a value. |
| Exceptions | None |
| Post-Conditions | • A value now shows within a played input tile. |
| Priority | Must have |
| Traces to Test Cases | None |

### 4.1.3 Use Case 3: Input Validation

| | |
|---|---|
| Description | Allows user to receive feedback on a number entered in an input tile. |
| Summary | A number is entered from 1 to 9 in an input tile and a red overlay is presented if incorrect. |
| Actors | Player |
| Pre-conditions | • Player's turn to act.<br>• Input tile is playable.<br>• The gameboard has not been solved. |
| Main Scenario | **1.** Player selects an input tile.<br>**2.** The player enters an incorrect number.<br>**3.** A red overlay on the input tile is displayed. |
| Exceptions | None |
| Post-Conditions | • A red overlay is now showing.<br>• Player has feedback that the number is incorrect. |
| Priority | Must have |
| Traces to Test Cases | None |

### 4.1.4 Use Case 4: Undo Input

| | |
|---|---|
| Description | Allows user to undo a numeral entered in an input tile. |
| Summary | User enters a numeral in an input tile. Undo removes the entered numeral from the gameboard matrix. |
| Actors | Player |
| Pre-conditions | • Player's turn to act.<br>• A numeral has been entered on the gameboard matrix<br>• The gameboard has not been solved. |
| Main Scenario | **1.** User has entered a numeral within an input tile.<br>**2.** The numeral has the option to be undone.<br>**3.** The numeral is removed from the current input tile. |
| Exceptions | None |
| Post-Conditions | • Last played numeral is no longer on the gameboard matrix.<br>• A previously entered numeral is present on the gameboard matrix.<br>• Numerals may be removed until all input tiles are cleared. |
| Priority | Good to have |
| Traces to Test Cases | None |

# 5 Non-Functional Constraints

There are several non-functional constraints that comes along with this project. Considering that the majority of PC's of the players run either on Windows or Mac OS, one constraint is that the application Kakuro must be built in a language that is supported by these operating systems. The programming language chosen to create this game is Java. Another constraint is that the game will only be functional on desktops and laptops as it has not been adjusted to function on mobile devices. Lastly, the application must be updated time to time if there are updates to the operating systems to make sure the game is still compatible.

# 6 Data Dictionary

| Name | Definition |
|---|---|
| Application | Start of the application. Creates an AppView and AppModel. Controls access to both creating the puzzle |
| AppView | Handles the adding of views to the application window. Has 2 layers appMain and appPopup |
| AppModel | Requests database for needed data. For example, it finds a puzzle with matching id and returns the PuzzleModel |
| Puzzle | The controller of the class requests a model from AppModel and uses it to create a puzzle view, it checks each box in it's view to see if they have the correct answer |
| PuzzleView: | Creates a gridPanel and a validate button |
| PuzzleModel | GUI format of the solution array |
| SolutionArray | Contains identity of each square. Whether if it is playable, contains top and bottom digit, can take an input |
| GridPanel | The collection of 100 10x10 squares |
| GridSquare | Contains the GUI of the squares. Input square is white, sum square is grey, unplayable square is black |
| InputSquare | A square with a text box |
| ValidateButton | Onclick will call the validate function in the Puzzle class |

# 7 References

Kakuro. (2019, December 18). Retrieved from https://en.wikipedia.org/wiki/Kakuro

# A Description of File Format: Tasks

Not Applicable.

# B Description of File Format: Persons

Not Applicable.