**Department of Computer Science and Software Engineering**
**Concordia University**

**COMP 354 — Introduction to Software Engineering**
**Winter 2016**
**Term Project Description**
**1D-2D Puzzle GCHQ System**

# 1  Objective

The objective of this project is to apply software engineering best practices to develop a Puzzle application in an iterative and time-boxed manner. With each iteration, a new increment of the 1D-2D Puzzle GCHQ System must be specified, modeled, built, tested and presented.

The focus of the project is on learning the software process and the inter-relatedness of the activities in a project. The process emphasizes:

- test-driven development, to ensure good understanding of requirements, and working software;

- agile development, to encourage iteration, to divide the work into small chunks, and to allow students to improve their skills from one iteration to the next;

- object-oriented software development, in particular software systems as collections of collaborating objects, with a focus on responsibility-driven design, an emphasis of separating interfaces from implementations, and the use of patterns; and

- communication, by being team-based, and requiring basic documentation using UML for domain model, use cases, architecture and design, and testing.

# 2  Project Description and Expected Features

Your team has been hired By Britain's security and intelligence organization GCHQ to develop a 1D-2D GCHQ Puzzle application to help puzzle makers to create puzzles and to support puzzle solvers to solve the puzzles via a graphical user interface (GUI). A preliminary interaction with your client determined that the application shall support the following user stories and constraints:

**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).

**Constraints**:

- Standalone JAVA application
- JAVA SWING GUI
- SQLite DB

## A Small Sample 1D-2D GCHQ Puzzle
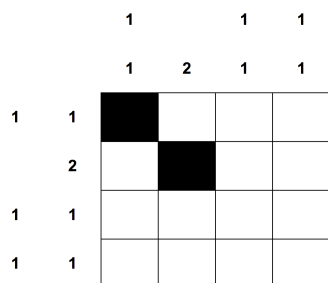
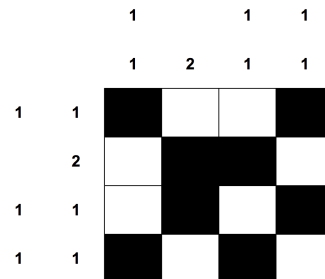Here is a small 4-by-4 example (A) with solution:



Figure 1: Puzzle (A)



Figure 2: Solution (A)

The puzzle consists of a grid of dimension R for rows, and C for columns. Here we use $R = 4$ and $C = 4$.

Think of it as a grid of pixels, which can be either black (B) or white (W).

The gird has some starting information of a few pixels that you know are black.

The puzzle also has constraints on the size of black regions for each row and column.

The numbers that are the headers of a row indicate the length of the black runs (or regions) that occur in the row. Each region is separated from the next region by one or more white spaces. So "1 1" requires two regions of length one, separated by one or more spaces: for a row of length 4 the possibilities are
(a) W B W B; (b) B W W B; or (c) B W B W.

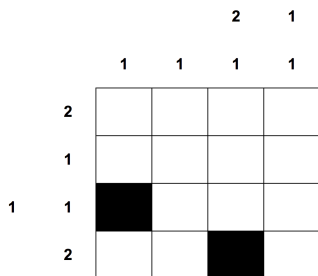Here is another small 4-by-4 example (B) with solution:
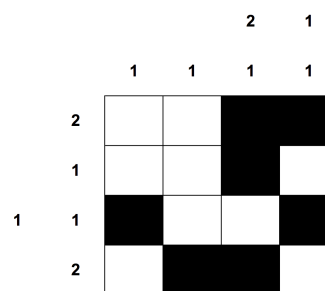


Figure 3: Puzzle (B)



Figure 4: Solution (B)

# GCHQ Christmas 2015 Puzzle

For the past Chistmas season the GCHQ released a 25-by-25 puzzle as follows:

**How to get started:** In this type of grid-shading puzzle, each square is either black or white. Some of the black squares have already been filled in for you. Each row or column is labelled with a string of numbers. The numbers indicate the length of all consecutive runs of black squares, and are displayed in the order that the runs appear in that line. For example, a label "216" indicates sets of two, one and six black squares, each of which will have at least one white square separating them.

# 3   Team Formation and Oragnization

The project is team-based. Each team consists of approximately 6 students and is organized into three subteams of size 2 to 3. There are three roles on the project:

**Developer:** should produce working source code

**Tester:** creates test cases, codes unit tests, and runs unit tests to check all is working

**Documenter:** should produce the documentation and specifications, but the content for the document is the joint responsibility of the team

Each subteam assumes the role of either developer, tester or documenter for the duration of an iteration. Team members rotate their roles such that each subteam will play each of the three roles.

It is important to understand that the project is a team effort and that it may be required to "loan" members from one subteam to another in order to meet the deadline. Note that there exist strong dependencies between the subteams. For example, testers need to talk to developers to know the class interfaces, and the expected results of each method.

Every team is responsible for establishing "ground rules" in order to minimize conflicts. In order to ensure on-time delivery, the internal delivery schedule should allow sufficient "slack" for mistakes and re-work. It is recommended to assign tasks as early as possible, so individuals can schedule their other work.

# 4   Development Environment

The following standard development tools shall be used:

- Java as the programming language;

- Eclipse as the integrated development environment (IDE);

- JUnit for unit testing in Java; and

- Rational Rose for UML modeling.

# 5   Project Plan and Deliverables

The 1D-2D Puzzle GCHQ System is to be developed in three iterations. Each iteration will produce a running system. Starting with a set of core features, with each iteration additional features will be added and/or the implementation of existing features will be modified. Each iteration will take three to four weeks. Each iteration will have a document as a deliverable: a requirements document, a design document, and a test plan and report

respectively for iteration one, two, and three. Each iteration will have source code as a deliverable including unit tests.

Submission dates for the three iterations are in week 5, week 9, and week 12 of semester. Exact deadlines for submission to the EAS (Electronic Assignment Submission) system are listed on the course web page.

Your team will be required to do a preliminary presentation of your system and document in the lab one week prior to submission. There will be a formal presentation of your submission in the lab following the submission deadline.

## Iteration 1

**User Stories**: US 1, US 2, US 3.

**Document**: Software requirements specification (for all user stories). Each team will be responsible to independently refine the user stories into software requirements. (Your team is your customer)

*Hints*: Create small puzzles of size 4-by-4, 6-by-8, and 8-by-8 by randomly filling in a 2D grid. It is the solution. Then calculate the headers from the 2D grid. Then remove most of the black pixels in the solution to leave you with a starting position for the grid.

Yes, you do want to make puzzles that are interesting to solve, and preferably a puzzle that has only one solution.

## Iteration 2

**User Stories**: US 4, US 5.

**Document**: Architectural and detailed design specification (for all user stories)

## Iteration 3

**User Stories**: US 6, US 7.

**Document**: Test plan and test report