### **SOLVEDOKU**

(SDK)

#### CS 3337 Software Engineering

#### Functional Requirements and Design Document

Prepared by

Kevin Trang

Allen Ruan

Nikita Aggarwal

Brandon Lee

APR 16, 2018

**Table of Contents**

# List of Figures

Figure 1-1 Context Diagram (Level 0 Data Flow Diagram)

Figure 2-1 Level 1 Data Flow Diagram

# List of Tables

None.**1.0 INTRODUCTION**

**1.1 Purpose**

The purpose of this document is four-fold:

a) Define a full set of requirements for SOLVEDOKU.

b) Define the design for SOLVEDOKU.

c) Define the implementation of SOLVEDOKU.

d) Define the Test Plan for SOLVEDOKU.

The complete definition of all SOLVEDOKU requirements provides the requirements to be used in the subsequent software subsystem documents.

**1.2 Scope**

This documentation was developed as part of a Software Engineering class, CS 3337.

The scope of this document includes the following:

* All functional and non-functional requirements. These requirements are organized by key SOLVEDOKU functional units shown in DFD 1
* Functional descriptions of each of the modules of SOLVEDOKU.
* General descriptions of hardware necessary for implementation of SOLVEDOKU.

**1.2.1 Document Organization**

The organization of this document provides a natural 'flow' or allocation of requirements to each succeeding section. Details regarding the overall document structure are discussed in sub-section 1.4.

**1.2.2 Relationship to Other Documents**

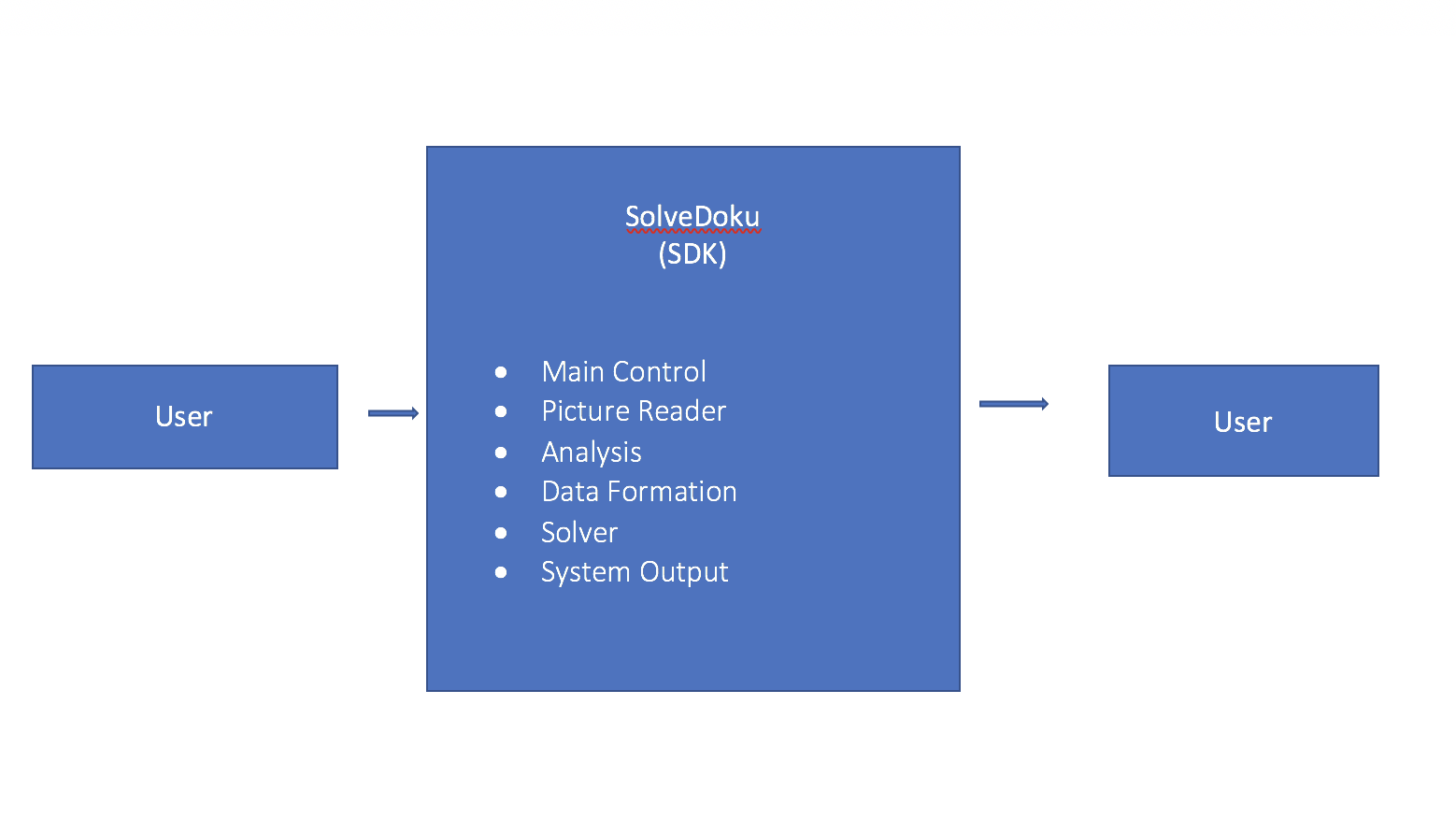
The SOLVEDOKU SRD/SDD/STP/SID is a complete self-contained document. However, Java code is included in separate files. Some relationships to other documents in the literature are indicated below in sub-section 1.5.

**1.3 Project Name Architecture**

**1.3.1 Context Diagram (DFD Level 0)**

SOLVEDOKU’s architecture is summarized in the Context Diagram (DFD Level 0) given below. A more detailed Functional Description is given in Section 2 of this document.

**Figure 1-1:** **Level 0 DFD**

****

**1.3.2 Description and Major Functions of SOLVEDOKU**

SOLVEDOKU automates the solving process of a Sudoku puzzle. The system offers the ability to upload an image and have it solved using image recognition. SOLVEDOKU recognizes the image and stores all the digits and uses backtracking to solve the 9x9 puzzle. SOLVEDOKU is controlled by a GUI. The name SOLVEDOKU is a combination between the famous Sudoku puzzle and the word solve.

**1.3.3 Hardware and Software Considerations**

SOLVEDOKU requires standard computer hardware. Other software requirements includes the latest build of Oracle’s Java Runtime Environment.

**1.4 Documentation of the Development Process**

SOLVEDOKU’s detailed functional description is documented in section 2.0. Section 2 is a succinct software description document. The overall detailed functional description is based on higher level DFDs (above level 1). All major functional units are described in detail in this part of the document.

Requirements for SOLVEDOKU are captured in Section 3.0 of this document. This section includes both functional and non-functional software requirements, supplemented with more detailed information when necessary.

**1.5 References**

All references used in the creation of this document are listed below.

1.5.1

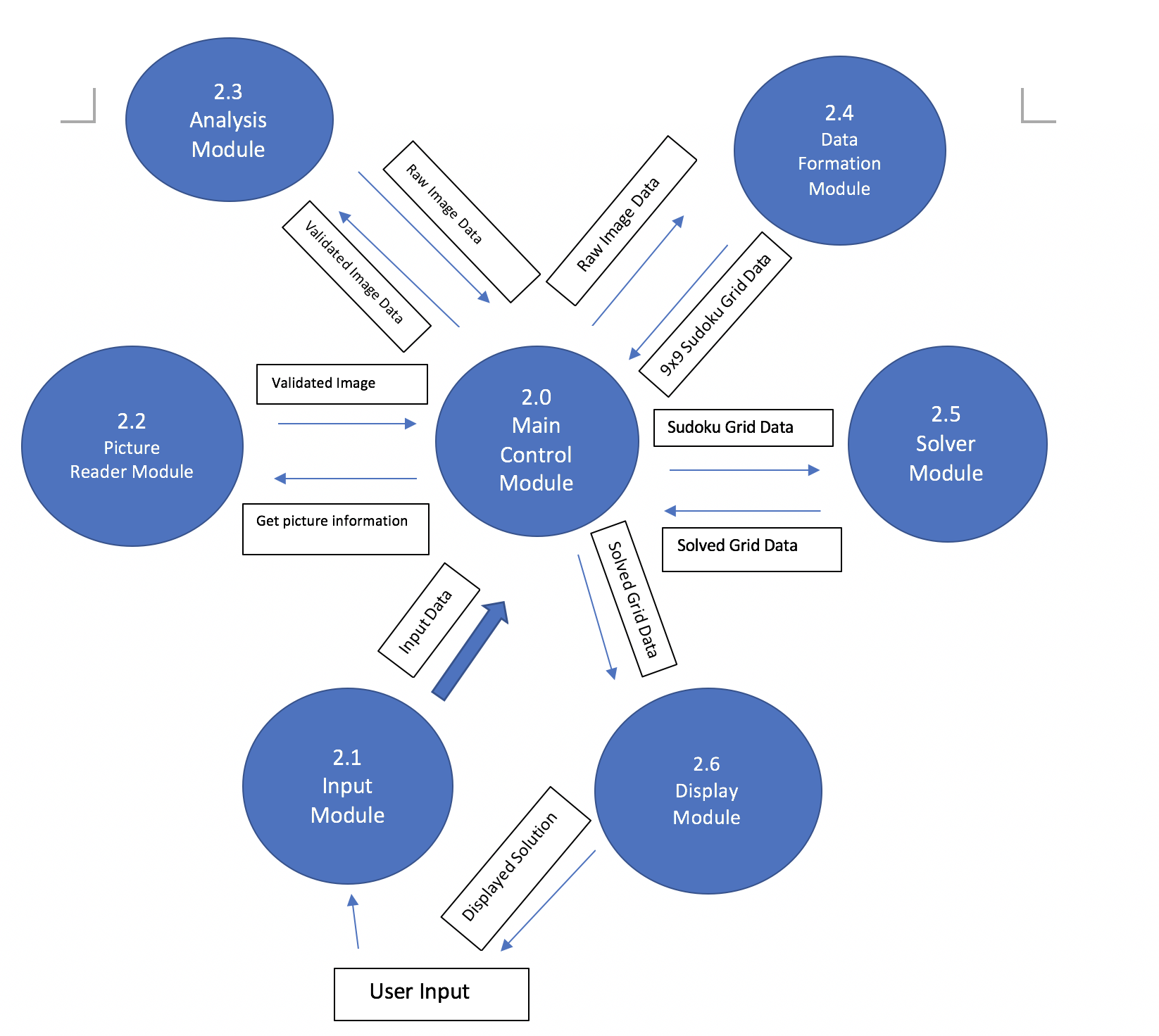
**2.0 DETAILED FUNCTIONAL DESCRIPTION OF SOLVEDOKU**

**2.1 Detailed SOLVEDOKU Functional Description.**

The major tool used to design SOLVEDOKU is the Data Flow Diagram, DFD. The rationale for the selection of DFDs as the preferred design tool is its simplicity and versatility. In the future additional tools may be used if a stronger correlation from Design to Requirement to Implementation and Testing is required.

**2.1.1 Level 1 DFD**

SolveDoku’s major functional subunits are shown in the DFD Level 1 shown below:

****

**Figure 2.1: Level 1 DFD**

**2.1.2 Detailed Functional Description of SOLVEDOKU’s Major Units.**

The description of SOLVEDOKU’s major functional units shown in Figure 2.1 follows. Most of SOLVEDOKU’s operations depend on the image provided by the user. Image provided can only be a 9x9 grid.

Input - Module 2.1

The Input Module (IM) allows the user to insert the image file for processing.

Picture Reader - Module 2.2

The Picture Reader Module (PRM) validates the image that is entered. If an invalid picture is entered into the program, it rejects the picture. Also, the grid that is entered must contain 17 digits already pre-entered.

Analysis - Module 2.3

The Analysis Module (AM) analyzes the image’s 17 pre-filled digits. The Analysis Module will take the 17 or more pre-filled digits will be extracted from the image and stored.

Data Formulation - Module 2.4

The Data Formulation Module (DFM) processes the extracted image data and stores the data into a data structure.

Solver Module - Module 2.5

The Solver Module (SM) uses backtracking to fill the blank spaces as random digits first. As it fills the puzzle with random digits, collisions shall occur. Through collisions, the Solver Module shall find the digits that do not belong in certain spaces and reject the digit.

Display - Module 2.6

The Display Module (DM) will take the solution of the solved puzzle. Through the solved puzzle, the Display Module will create a visual representation for the user.

**3.0 SOLVEDOKU REQUIREMENTS**

**3.1 SOLVEDOKU Functional Requirements**

This Section lists SOLVEDOKU’s functional requirements. This section includes the complete set of functional requirements, along with explanations for cases in which the statement of the requirement was deemed insufficient or requires additional clarification. All requirements relate to the design modules described in Section 2. An effort has been made to standardize the correlation between the design modules and the requirements to make access and organization more consistent. For example, requirement number “n” affecting module 2.1 will be labeled 3.1.n

|  |  |
| --- | --- |
| **Requirements Related to Design Module 2.2: Input Module** | |
| Requirement No. | Requirement Description |
|  | **Input Module (IM)** |
| 3.1.1 | IM shall process inserted data |

|  |  |
| --- | --- |
| **Requirements Related to Design Module 2.2: Picture Reader Module** | |
| Requirement No. | Requirement Description |
|  | **Picture Reader Module (PRM)** |
| 3.2.1 | PRM shall validate the image file |
| 3.2.2 | PRM shall reject non 9x9 image grids |

|  |  |
| --- | --- |
| **Requirements Related to Design Module 2.3: Analysis Module** | |
| Requirement No. | Requirement Description |
|  | **Analysis Module (AM)** |
| 3.3.1 | AM shall collect Sudoku data from the validated image |
| 3.3.2 | AM shall recognize blank spaces as zeros |

|  |  |
| --- | --- |
| **Requirements Related to Design Module 2.4: Data Formation Module** | |
| Requirement No. | Requirement Description |
|  | **Data Formation Module (DFM)** |
| 3.4.1 | DFM shall store Sudoku grid data in a data structure |
|  |  |

|  |  |
| --- | --- |
| **Requirements Related to Design Module 2.5: Solver Module** | |
| Requirement No. | Requirement Description |
|  | **Solver Module (SM)** |
| 3.5.1 | SM shall use backtracking |
| 3.5.2 | SM shall populate grid using random digits from 1-9 |
| 3.5.3 | SM shall test the random digits against the rules of a Sudoku puzzle |
| 3.5.4 | SM shall reject incorrect digits entered into the puzzle |

|  |  |
| --- | --- |
| **Requirements Related to Design Module 2.6: Display Module** | |
| Requirement No. | Requirement Description |
|  | **Display Module** |
| 3.6.1 | SOM shall be able to process the populated data for displaying it |
| 3.6.2 | SOM shall be able to create a visual representation for the user’s interpretation |

**3.2 SOLVEDOKU Non-Functional Requirements**

This Section collects all the SOLVEDOKU’s Non-Functional Requirements. All non-functional requirements are numbered “NF – n” where “n” indicates the nth requirement.

NF - 1 OLYMPIA requires sufficient data storage to store a library of rehearsal recordings as well as machine-read librettos and scores.

NF - 2 OLYMPIA must execute cue events as rapidly as a professional stage crew.

NF - 3 OLYMPIA must be operable by creative professionals who have little technical training.

**3.3 SOLVEDOKU Hardware Requirements**

This Section collects all SOLVEDOKU’s electronic hardware requirements. All hardware requirements are numbered “H – n” where “n” indicates the nth requirement.

H - 1 SOLVEDOKU will run on standard microcomputer hardware

**5.0 SOLVEDOKU’sELEMENTS OF IMPLEMENTATION**

**5.1 Introduction**

This section descries the implementation of the requirements described above. We include brief descriptions of each of the coded classes. Java code files are included separately.

**5.2 Descriptions of Java Classes**

**Tess4JDigit GUI:**

*Tess4JDigit* uses optical character recognition to read the individual images of digits and outputs the integer value of the inserted digit.

**DetectSudoku**

*DetectSudoku* grabs the puzzle and cuts out unnecessary border. This class also crops the puzzle into individual 81 digit squares.

**FeatureDetecter**

*FeatureDetecter* detects the digit inside the individual squares. This class then determines if the square contains a digit or not. If not then this class enters 0 as a substitute for a blank digit.

**Board**

*Board* orients the oupt as a 9 by 9 grid during testing.

**DancingLinks**

**Special Effects:**

*SpecialEffectsMain* is the top level controlling class for the Special Effects Module.

*SFXMachine* is an interface implemented by the effects machines and specifying some key methods and attributes used to standardize the machine controls.

*AirMachine, Fog Machine, MistMachine,* and *PyroMachine* simulates special effects hardware.

*AirSFX, FogSFX, MistSFX,* and *PyroSFX* are the controllers for the machinery listed above.

**Supertitles:**

*SupertitlesMain* is the main controller for the Supertitles module

*DisplayScreen* provides a GUI for patron display screens

*Libretto* represents the text of an opera. It contains data storage and functionality to construct and iterate through operas.

*LibrettoLine* represents one line of a libretto, in several languages.

*PatronDisplay* is the controller for a single patron screen.