Software Requirements Specification

Version 1.0

September 13, 2020

Diagrams Through ASCII Art

David Holdren
Kevin Porter
Adam Shiveley

Table of Contents

List of Figures

1.0. Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of a proposed program that will convert ASCII art in documents to diagrams. The program will be titled "Diagrams Through ASCII Art," or DITAA for short. The use of ASCII art in the field of computer science ranges for early text based games from the 1980s to acting as a "Maker's Mark" for code authors. While the use of ASCII art is generally fun to create, the only known method to easily save the ASCII art is to copy and paste the art into a new file. The proposed program will leverage the use of modern computing to enable users to create diagrams from embedded ASCII. These diagrams will be saved as image format files that can be easily printed or loaded into word editing programs for future use.

1.2. Scope of Project

This software system will be a Java-based program executed from the command line. The program will execute as automated as possible, thus reducing input burden on the user. The program will also be able to execute in batch mode, meaning a user can queue up multiple different files to scan and convert.

1.3. Glossary

The DITAA (DIagrams Through ASCII Art) program is a command line utility which provides functionality to convert ASCII diagrams into images.

The GUI is the Graphical User Interface with which the user will interact with the DITAA program.

Mentions to a '*.jar' file are in reference to Java archive files which can be a standalone executable final product for a Java program. This is the planned product format for the executable to run the DITAA GUI.

1.4. References

This document refers to the DITAA guide on Github as well as help documentation in the repository for functionality descriptions and available options.

1.5. Overview of Document

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

2.0. Overall Description

2.1 System Environment

The DITAA program has minimal user input. The only requirement from the user is to have an ASCII text file ready to be converted. The user also has options to select the encoding, input file type, and some output configurations regarding file output and image style.

2.2 Functional Requirements Specification

This section outlines the use cases for both the GUI and command line version.

2.2.1 Command Line Use Case

Use case: **Execute Via Command line**

Brief Description
The User executes the program and types all of the user required options on the command line.

Initial Step-By-Step Description
Before this step can be executed, the user needs to have identified a file with ASCII art to convert.

1. The User locates the ASCII art file (it can be user created, found online, or art buried in a text file.
2. The user opens a shell.
3. The user runs the program with the selected options, passing the text file as an argument.
4. The program processes the user defined file and creates the object file in the same directory.
5. The user is notified of program completion.

2.2.2 GUI Use Case

Use case: **Execute Via GUI Executable *.jar**

This user case allows for the Graphical User Interface (GUI) component of the program to be executed. The GUI is basically a wrapper around the main program, which facilitates ease of program execution for the user.

Brief Description
The user executes a *.jar file by double clicking on it. The program opens to a screen which allows the user to decide which parameters to use and which file to scan looking for ASCII art.

Initial Step-By-Step Description

1. The user double clicks the *.jar file
2. The GUI opens and displays program options to the user.
3. The user chooses the file with the ASCII art from a button click.
4. The user chooses a save location for the output file.
5. The user selects options for creating the output file.
6. The program is executed via a “Run” button and the image is saved to a chosen location.

2.3 User Characteristics

The user is expected to be able to operate a computer. The user is also expected to be able to operate a common GUI with push buttons and file browser dialogs. The user is expected to understand common file formats and text encodings.

2.4 Non-Functional Requirements

The DITAA GUI program will operate on the user’s computer. It will be a *.jar file and be easily executed by double clicking the file. The data will be saved in the user chosen location through the GUI.

3.0. Requirements Specification

3.1 External Interface Requirements

The DITAA program operates on external files. The program also depends on user input from the command line or a GUI interface.

3.2 Functional Requirements

The Logical Structure of the Data is contained in Section 3.3.1.

3.2.1 Command Line

Use Case Name	Command Line
Trigger	User presses the "Run" button
Precondition	The User has an ASCII Art file to convert
Basic Path	1. The User types the required program options. 2. The User types the file to be converted. 3. The User presses "Enter"
Alternative Paths	The User can choose to quit the program without converting text to an image.
Postcondition	The file is verified to exist and the conversion is complete
Exception Paths	The User may abort the program at any time.
Other	None

3.2.2 Graphical User Interface

Use Case Name	GUI
Trigger	The double clicks the *.jar file
Precondition	The user has an ASCII Art file to convert.
Basic Path	1. The user selects the file to be converted 2. The user choose which options to operate with 3. The user presses "Run" to execute the program.
Alternative Paths	The user may abort the program without converting text files to images. The user may also continue to repeat the process any number of times.
Postcondition	The file is verified to exist and the conversion is complete.
Exception Paths	The user may abort the program at any time.
Other	None

3.3 Detailed Non-Functional Requirements

3.3.1 Logical Flow of DITAA Graphical User Interface

The logical structure of the DITAA program is given below.

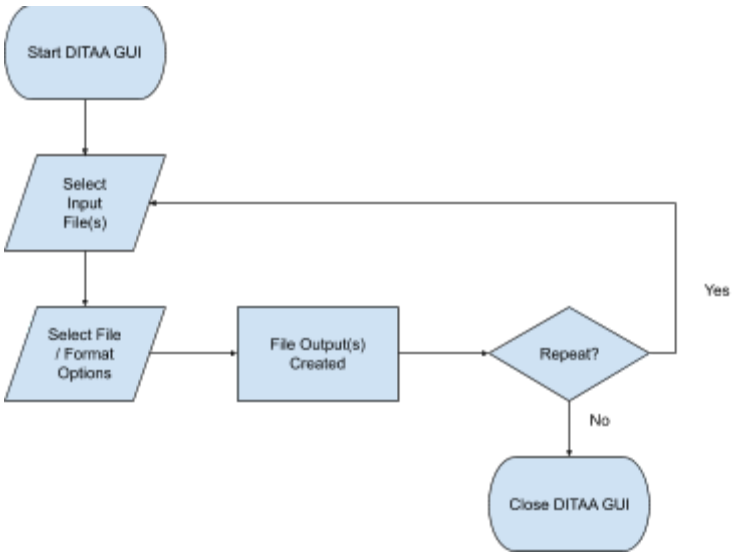

Figure 1 - Logical Flow of DITAA Graphical User Interface

The DITAA program will output a *.png file containing a diagram of the chosen ASCII art.

3.3.2 Security

The program will attempt to be developed with cyber security in mind. All loops will contain bounds, all memory will be properly managed, and error messages to the user will be easy to understand. The GUI interface will be tested for proper input into the requested fields. The program should not crash or hinder access to the user's machine nor access, store, or transmit any unauthorized data.