CS 7140 - Advanced Software Engineering

Fall 2020

Project Requirements Documentation

Team Members:

Lancius (Lance) Matthieu

Abhishek Pandya

Daniel Ketterer

Griffin Mosley

Table of Contents

1. Introduction

1.1 Purpose of Document

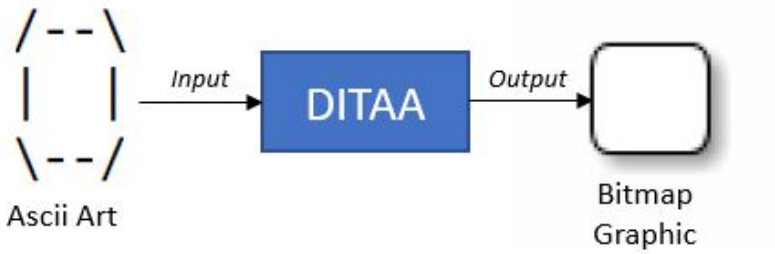
This Requirements document is for a new conversion software that converts ascii art drawings into proper bitmap graphics. This conversion software will be titled Diagrams Through Ascii Art (DITAA). This document describes the scope, objectives, and goal of DITAA. Functional and non-functional objectives of DITAA will be covered and functional objectives will be modeled with use cases and a context diagram. The intention of this requirements document is to guide the design and implementation of DITAA in an object-oriented language.

1.2 Project Summary

Project Name:	Diagrams Through Ascii Art (DITAA)
Project Team:	Lancius (Lance) Matthieu
	Abhishek Pandya
	Daniel Ketterer
	Griffin Mosley
Responsible Users:	Dr. Prabhaker Mateti

1.3 Problem Statement

Our team is tasked with developing software that will enable the conversion of new and potentially legacy ascii art drawings into proper bitmap graphics. Ascii art are drawings that are constructed with characters that resemble lines, such as |, /, -, and).



Motivation:

There are legacy frequently asked questions (FAQs) that utilize ascii drawings and/or diagrams that could be aesthetically improved with the use of rendered graphics. Currently, these ascii art drawings would need to be recreated from scratch with a separate graphic design software (e.g. GIMP, Photoshop, Inkscape, etc.). DITAA aims to reduce the burden to update and maintain ascii art drawings into usable graphics, by directly converting the ascii art into bitmap graphics.

Vision:

In addition to the standard version of the software, the following enhancement(s) will be made to the baseline DITAA software:

- Reduce baseline codebase by a minimum of 5%
- Update baseline codebase to utilize Java 8 functionality
- Provide wider range of pre-set background colors
- Provide ability to render wider range of shapes
- Provide ability to specify texture of lines drawn (e.g. solid, dashed, dotted)

DITAA will be offered as free open source software under the GPL license with no intention of monetizing the software.

1.4 Project Scope

The scope of this project is a command line interface executable that takes ascii art drawings in a specified format (see section 5.3) and converts the drawings into proper bitmap graphics. Note: some graphics will not be rendered, such as “Display” and “Off-Page Reference”.

Examples of graphics not rendered: (Left) Display (Right) Off-Page Reference

A design for a graphical user interface (GUI) look and feel will also be developed.

Integration of DITAA into other text-based formats (e.g. HTML, DocBook, LaTeX) is not part of this project.

1.5 System Purpose

1.5.1 Users

Users who will directly benefit from the newly developed software are identified below:

End User:

Once software is complete, end users will have an offline, standalone command line interface tool able to convert legacy and/or new ascii art drawings into usable bitmap graphics.

1.5.2 Location

The software (including source code) will be available for anyone with access to GitHub. The software is intended to be utilized as a standalone command line interface tool on the end user's local machine.

1.5.3 Responsibilities

The primary responsibilities of the software are:

- Provide end user ability to convert ascii art diagrams to bitmap graphics
- Provide end user with documentation of expected ascii art input expectations
- Provide end users with an intuitive, easy to use command line interface
- Provide end users a common -help option that specifies usage and optional flags
- Allow the end user to customize output graphics with the following:
    - anti-aliasing
    - graphic background color
    - edge separation (default on)
    - rounded corners
    - shadows
    - transparency
    - scale
    - fixed width or slope for parallelograms
    - dashed lines
    - point markers
    - text insertion – to include bullets
- Allow end users to overwrite file if it already exists
- Allow end users to specify encoding of input ascii art drawing
- Allow end users to specify the number of spaces "tabs" are interpreted as (default 8)

Other desired features of the new software:

- Allow end users access to a wider range of pre-set background colors
- Allow end users to specify line texture (solid, dashed, dot, etc.)

- Allow end users to render more shapes (see 1.4)

1.5.4 Need

This new software is needed as current, legacy ascii art diagrams would need to be re-created from scratch via a separate graphics design software and presents maintainability concerns when updates are required. DITAA would allow a user to take advantage of newer versions by simply re-rendering their previous ascii art diagram.

1.6 Overview of Document

The rest of this document gives the detailed specifications for the new sales system. It is organized as follows:

- ***Section 2: Functional Objectives***
  Each functional objective details a desired behavior for the software These objectives are organized by priority. All high priority objectives must be met for the software to be considered successful.
- ***Section 3: Non-Functional Objectives***
  Non-functional objectives are organized by category. Each objective specifies a technical requirement or constraint on the overall characteristics of the software.
- ***Section 4: Context Model***
  This section provides a description of the overall goal of the software and a context diagram depicting the scope of the software.External entities that interact with the software are also described.
- ***Section 5: Use Case Model***
  Behavioral requirements of the software are detailed in a use case. The use case shows the interaction between software and the end user. A use case diagram is also presented showcasing all interactions with the software.
- ***Section 6:*** An appendix containing a glossary of terms specific to this project as well as team member project journals

2. Functional Objectives

2.1 High Priority

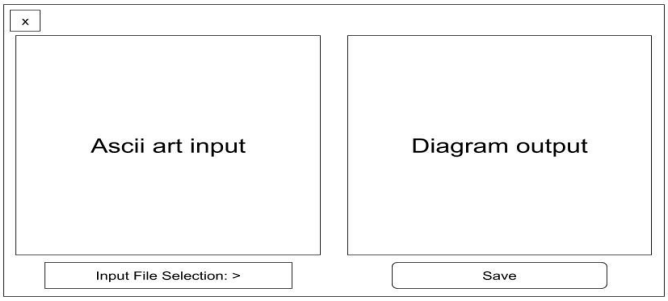
Identifier	Requirement
REQ1	The program shall provide an easy way to convert ascii art(characters that resemble lines like | / - + *) to diagrams.
REQ2	The program shall allow the user to draw any design using ascii art and in return the program should give diagrams as an output from that ascii art.
REQ3	The program shall convert each ascii value to a particular design.
REQ4	The program shall provide information on which ascii value converts into which diagram.

2.2 Medium Priority

Identifier	Requirement
REQ5	The program shall provide the user a choice for filling the color in particular shape.
REQ6	The program shall provide the user a list of tags from which the user can render a particular diagram.

2.3 Low Priority

Identifier	Requirement
REQ7	The program shall provide GUI to the user, so that, user can easily convert ascii art to diagram.
REQ8	The program shall provide enhancements to improve the quality of the program. Examples are command line flags, more line conversions (double lined), and shapes.



This is an example of what the GUI might look and feel like..

3. Non-Functional Objectives

3.1 Reliability

- The program will run 90-95% times. There will be no error in the program

3.2 Usability

- Any one in the world who wants to use this program can use it.

3.3 Performance

- The executable time of the program will not exceed more than 10 seconds.

3.4 Supportability

- This program will run on all devices who have java in their system.
- This program will run in all environments whether it's windows or mac or Linux.

3.5 Online user Documentation and Help

- The program shall provide a README file that explains how to render each diagram from characters(ascii art).
- This README file will be available for everyone who will download the whole program.

3.6 Interfaces

- The program must interface with:
    1. Current version of JDK(Java Development Kit)
    2. Current version of JRE(Java Runtime Environment)

3.7 Maintainability

- A user who wants to add some new feature can make a request after giving the correct code.

3.8 Expectations

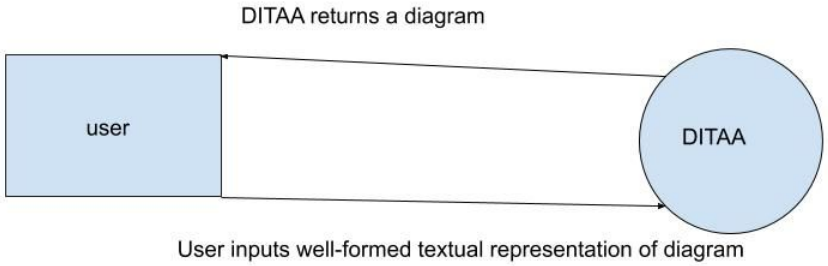
- After test, it is expected that the final software product be free of errors, while also decreasing the overall project code size by a minimum of 5% and increasing maintainability of the codebase

4. Context Model

4.1 Goal Statement

The goal of the system is to allow users to convert ascii textual representations into diagrams rendered in bitmap graphics

4.2 Context Diagram

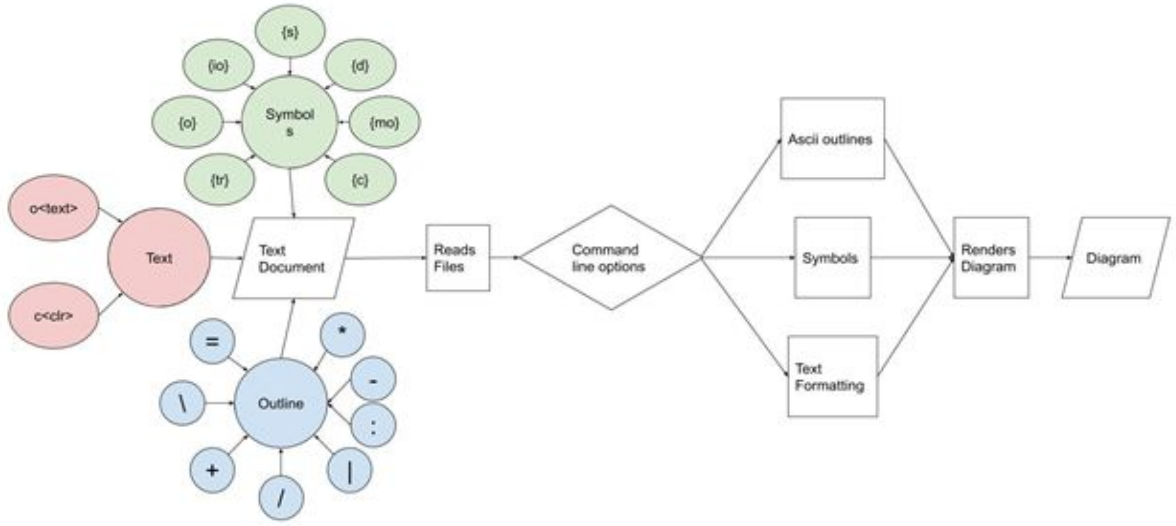


4.3 System Externals

User : A user is anyone using the software.

5. The Use Case Model

5.1 System Use Case Diagram



5.2 System Use Case Description

Default

Use Case:	Default
Summary:	Default functionality of DITAA. Takes in a file of ascii characters and converts it to a diagram formed from the characters.
Basic Flow:	1. User specifies which document as input. 2. User specifies an output file. 3. Reads in the input file. 4. Checks and applies options from the command line. 5. Creates outlines based on specified characters. 6. Converts outlines surrounding symbols to corresponding formats. 7. Applies text formatting options. 8. Renders the diagram to the output file.
Alternative Flow:	Step 2: If the user does not specify an output file, an output file with the given input name is created rendered to a .png file. Input.txt would be rendered to Input.png. Step 4: No command options need to be specified.
Preconditions	Input file must not be empty. Input file must be in a readable encoded file, examples: .txt or .html.
Postconditions	Users can access the ascii art converted to a picture..

5.3 Text and Command Line Input Details

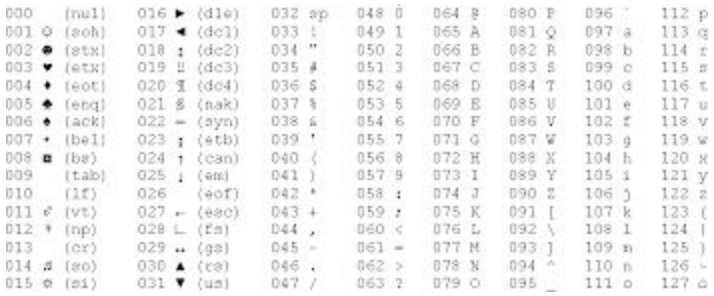
Outline Characters:	· |: Creates left and right side of the outline. · -: Creates top and bottom of the outline. · / and \: Creates rounded corners of the outline. · +: Creates squared corners of the outline. · =: Creates dotted lines on the top and bottom part of the outline. · :: Creates dotted lines on the left and right side of the outline. · *: Creates a point on the outline.
Symbols	· {d}: Creates shape representing a document for diagrams. · {s}: Creates shape representing a storage for diagrams. · {io}: Creates shape representing I/O for diagrams. · {mo}: Creates a shape representing manual operation for diagrams. · {c}: Creates a shape representing a choice for diagrams. · {o}: Creates an ellipse. · {tr}: Creates a trapezoid.
Text	· o<text>: Creates a bullet point followed by the text. · c<XXX>: Creates a background color for the outlined shape. XXX is the hex number corresponding to a color.
Options	· -A: Turns off anti-aliasing. · -b: Takes in color and sets background color of the image. · -d: Renders the debug grid. · -E: Prevents the default separation of shapes. · -e: Takes in encoding and specifies the encoding of the input file. · -h: Specifies the input is an HTML and produces HTML file with tags. · -o: Overwrites a file if the file is already created. · -r: Makes all corners round corners. · -S: Turns off drop-shadow effect. · -s: Takes in scale and uses it to determine the size of the rendered image. · -T: Makes the background transparent. · -t: Takes in length of tabs and uses it to change the spacing for tab characters · -v: Makes ditaa more verbose. · -w: Makes sides of shapes with fixed slopes instead of widths.

6. Appendix

6.1 Glossary of Terms

ASCII

ASCII (American Standard Code for Information Interchange) is a character encoding that uses numeric codes to represent characters in 7 bits. The numeric codes are the decimal numbers 0-127, inclusive on both ends. There are some ascii representations that are in 8 bits, however the original ASCII format is 7 bits.



EPS

EPS (Encapsulated PostScript) is a graphics file format. Using Postscript code, a language that describes the format of a file, EPS is able to switch between different file formats. Ditaa uses EPS to be able to display the output using a variety of different image formats, a few being: png, jpeg, and html.

Bitmap

An array of bits correlating to which pixels turn on on a screen. Using the array of bits, the pixels will either turn on or turn off. The pixels turned on will form to display an image.

6.2 Sources

http://web.cse.ohio-state.edu/~bair.41/616/Project/Example_Document/Req_Doc_Example.html

https://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf

http://www.plcdev.com/ascii_chart

https://techterms.com/definition/eps

6.3 Team Member Journals

6.3.1 Team Journal

30 August 2020:

- Discord created, created by Lance
- Invite links sent out

8 September 2020:

- Griffin added to group
- Team members finalized
- Email sent for team members to professor

14 September 2020:

- Discussion of first meeting, set up by Lance
    - Set to 15 September 2020 2-3 pm
- Goal set to have ditaa downloaded and looked over prior to first meeting
- Github repository created, created by Daniel
- Google drive created, created by Griffin

15 September 2020:

- First meeting (All group members attended)
    - Discussed how ditaa is ran
    - How the requirements document will be organized
    - Who will be completing each task
        - Lance M - 1) Introduction
        - Abhishek - 2) Functional 3) non-functional Objectives
        - Daniel - 4) Context Model
        - Griffin - 5) Use Case Model
    - Links shared on formatting and what to include
- Discussion of second meeting, set up by Lance
    - Set to 16 September 2020 5-6 pm
    - Goals
        - Review each portion
        - Make edits
        - Input on additional items

16 September 2020:

- Second meeting (All group members attended)
    - All sections completed
    - Reviewed each section
    - Filled in portions where things still needed to be discussed
    - Discussed additional details to be added

        - Appendix, to be completed by Griffin
            - Glossary of terms
            - Sources
            - Journals
        - Additions to introduction, to be completed by Lance
    - Made small edits in formatting
- Discussion of third meeting, set up by Lance
    - Set to 17 September 2020
    - Goals
        - Final changes
        - Review edits and things added on
        - Determine who will submit document

17 September 2020

- Third meeting (Daniel was not there (sick))
    - Reviewed final additions
    - Finalized document
    - Added journals
    - Abhishek volunteered to submit document

6.3.2 Lancius (Lance) Matthieu Journal

26 August 2020:

- Posted on Pilot forum introducing myself and requesting teammates for group project

30 August 2020:

- Received responses from fellow students and sent a team email organizing ongoing communications
- Team decided on utilizing discord, so I set up a group project channel and provided invitation link to team members

08 September 2020:

- Reached out to Griffin Mosely to join our current team, which he accepted as he was looking to join a team

14 September 2020:

- Joined group GitHub repository for project created by Daniel Ketterer
- Obtained access to Google Docs requirements document, set up by Griffin
- Reviewed Marsic requirements documentation example (Appendix G)
- Researched other examples of requirements documentation

    - Found example from Ohio State University that team will likely use for our requirements document
- Organized team meeting to discuss requirements document on 15 September at 1400

15 September 2020:

- Cloned ditaa repository utilizing IntelliJ IDEA IDE
- After obtaining ditaa dependencies, was able to build and run ditaa
- Team meeting: Discussed outline of Requirements documentation and handed out responsibilities
    - Lance: Introduction
    - Abhishek: Functional & Non-Functional Objectives
    - Daniel: Context Model and GUI Look & Feel
    - Griffin: Use Case Model
- Organized follow-up team meeting for 16 September 2020 at 1700
- Started Introduction of requirements document

16 September 2020:

- Completed first draft of introduction section and included in shared google drive document
    - Submitted before team meeting to allow for review beforehand
- Attended team meeting to discuss progress on requirements documentation
    - Decided GUI look & feel would be completed by Griffin (done before end of meeting)
- Reviewed team members portions of the requirements document and provided feedback
- Scheduled final team meeting on requirements documentation for 17 September 2020 at 1700
- Finished incorporating feedback to introduction portion of requirements documentation
- Added cover page and table of contents

17 September 2020:

- Attended final team meeting before turn-in of requirements documentation
    - Provided feedback on non-functional requirements section, glossary of terms, and sources
- Finalized requirements document and Abhishek took point to submit via email
- Plan is to reconvene on 28 September 2020 to start discussion on specifications document

6.3.3 Griffin Mosley Journal

14 September 2020:

- Installed ditaa.jar from link given in discussion.
- Pull ditaa from github.
- Ran jar file with tests.
- Created google drive folder for group
- Joined group github repository, created by Daniel

15 September 2020:

- Group meeting 2-3pm, organized by Lance
    - Discussed different parts for the requirements document.
    - Discussed who will complete each part.
        - Lance M - 1) Introduction
        - Abhishek - 2) Functional 3) non-functional Objectives
        - Daniel - 4) Context Model
        - Griffin - 5) Use Case Model
    - Completed Use Case Model

16 September 2020:

- Group meeting 5-6 pm, organized by Lance
    - Discussed any changes for specific parts
    - Discussed a GUI for the project.
    - Discussed the additional features to be included
    - Reformatted sections to be consistent

17 September 2020:

- Group meeting 5-6 pm, organized by Lancee
    - Discussed final changes
    - Discussed who will submit document
- Completed glossary of terms
- Added a sources section

6.3.4 Abhishek Pandya Journal

28th August 2020:

- Responded to Lancius (Lance) Matthieu post to join the team in pilot discussion.

30th August 2020:

- Joined discord channel and meet everyone in the team

10th September 2020:

- Downloaded the DITAA Project and tried to build and run the program.

14th September 2020:

- Joined group github repository, created by Danie

15th September 2020:

- Group meeting: 2-3 PM
- Discuss the requirement document and decide which part will be done by who.
    1. Lance: 1) Introduction
    2. Abhishek: 2) Functional & 3) Non-Functional Objectives

3. Daniel: 4) Context Model
4. Griffin: 5) Use Case Model

- Wrote Functional and Non-Functional Objective for the requirement document

16th September 2020:

- Group meeting: 5-6 PM
- Discussed about the changes in the requirement document
- Discussed about what additional feature team can add

17th September 2020:

- Group meeting: 5-6 PM
- Finalized the requirement document
- Sent final requirement document to professor