DITAA Functional Requirements Specification

Daniel Longendelpher | Christopher Menart | Brad Schneider

# Requirements Listing

The following table outlines functional requirements for the [DIagrams Through Ascii Art (DITAA) software](http://ditaa.sourceforge.net/). This represents the existing features of the DITAA software. A separate listing of requirements will be provided for the proposed extended functionality.

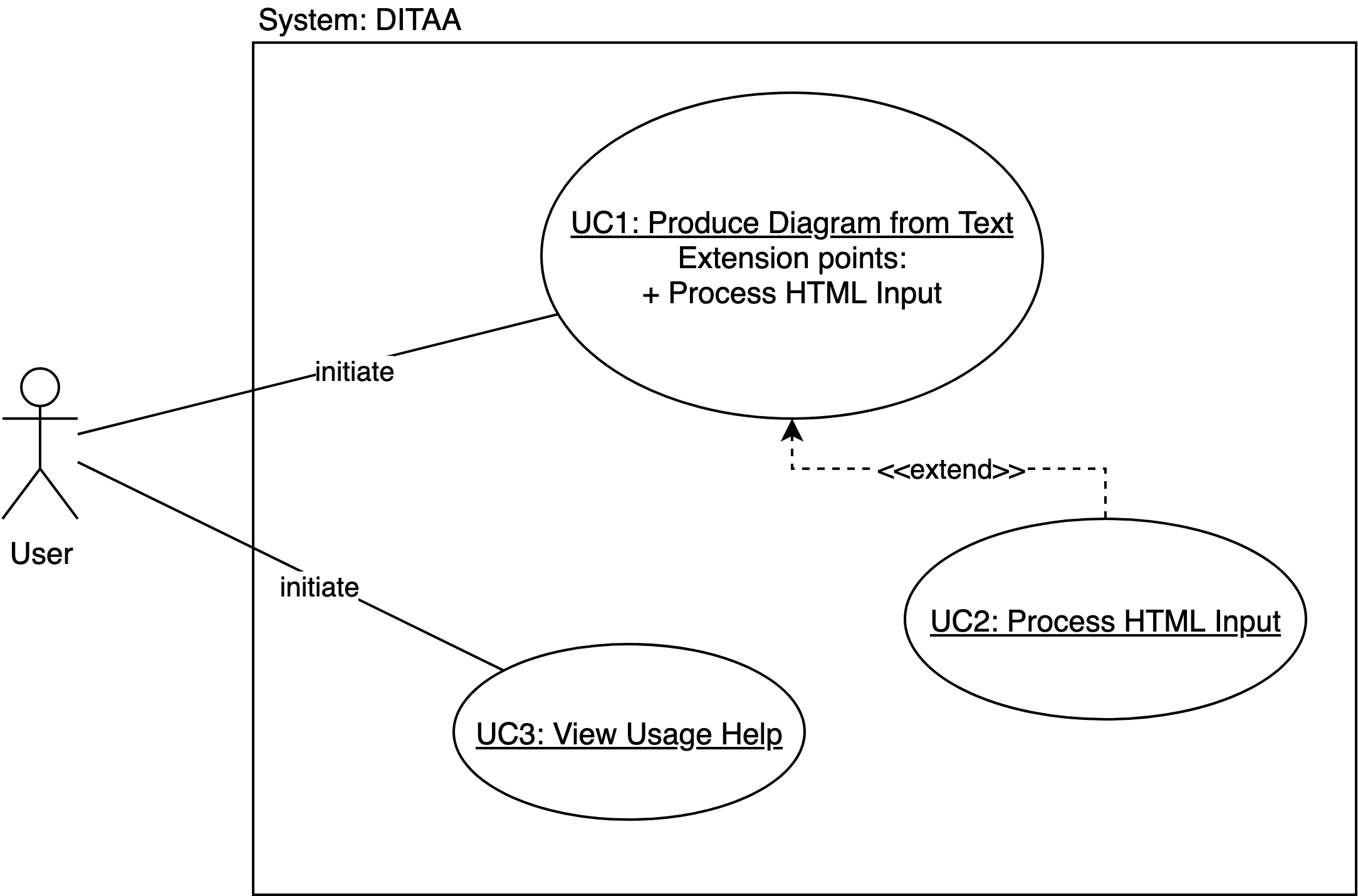
|  |  |
| --- | --- |
| Identifier | Requirement |
| REQ1 | The system shall convert an input plain text file into a bitmap graphics output file. |
| REQ1.1 | The system shall accept plaintext input using ASCII or UNICODE encodings. |
| REQ2 | The system shall provide a command-line interface for operating the tool. |
| REQ3 | The system shall display usage help with details for running the program. |
| REQ4 | The system shall render open and closed polygons with solid lines in the output based on edges defined by characters in the input text file. |
| REQ4.1 | The system shall render closed polygons defined by connected ‘-’, ‘|’, and ‘+’ characters (i.e. ‘-’ for horizontal edges, ‘|’ for vertical edges, and ‘+’ for corners). |
| REQ4.2 | The system shall render open polygons defined by connected ‘-’, ‘|’, ‘+’, ‘<’, ‘>’, ‘^’, and ‘v’ characters (i.e. ‘-’ for horizontal edges, ‘|’ for vertical edges, ‘+’ for corners, and ‘<’, ‘>’, ‘^’, ‘v’ for left-, right-, upward-, and downward-facing arrow heads). |
| REQ4.3 | The system shall render rounded corners when the input contains ‘/’ and ‘\’ characters in place of the ‘+’ (square corners) character on a rectangular shape. |
| REQ4.4 | The system shall render polygons using dashed lines where any edge in an input polygon contains a ‘:’ (for vertical edges) or ‘=’ (for horizontal edges) character. |
| REQ4.5 | The system shall separate or join shared edges of two polygons according to indicated user preferences. |
| REQ5 | The system shall render textual labels where non-polygon-defining characters are encountered in the input. |
| REQ5.1 | The system shall render a bulleted list where a line of input text is of the form ‘ o XXXX’ where X represents any text (a single space is required before and after the ‘o’ character). |
| REQ6 | The system shall render a ‘point marker’ (i.e. a node) when the ‘\*’ character is encountered on an edge. |
| REQ7 | The system shall render non-rectangular shapes in the place of a rectangular closed polygon when tags of the form ‘{XX}’ are encountered inside of a closed polygon. |
| REQ7.1 | The system shall render a UML Document symbol when the tag ‘{d}’ is encountered within a closed polygon. |
| REQ7.2 | The system shall render a UML Database symbol when the tag ‘{s}’ is encountered within a closed polygon. |
| REQ7.3 | The system shall render a UML Data (I/O) symbol when the tag ‘{io}’ is encountered within a closed polygon. |
| REQ8 | The system shall render closed polygons with a colored fill when the input contains a color code of the form ‘cXXX’ where ‘X’ represents a valid hexadecimal digit. The digits shall be interpreted as the red, green, and blue components of the resulting color, respectively. |
| REQ8.1 | The system shall support the following shorthand color codes:   * cRED (cD32) * cBLU (c55B) * cGRE (c9D9) * cPNK (cEAA) * cBLK (c000) * cYEL (cEE3) |
| REQ9 | The system shall accept HTML files as input. For each <pre> tag with the class attribute set to ‘textdiagram’ shall be interpreted as a separate diagram. The value of the ‘id’ attribute shall be used as the resulting diagram filename if provided. Otherwise, the filename shall be generated in the format ‘ditaa\_diagram\_X.png’ where ‘X’ represents a unique number. |
| REQ10 | The system shall output a copy of the input HTML file with <pre> tags replaced with <img> tags. The <img> tags shall have an appropriate ‘src’ attribute set to the location of the bitmap diagram that replaces the <pre> tag from the original input file. If no output filename is provided the produced HTML file shall be named ‘xxxx\_processed.html’ where ‘xxxx’ represents the original input filename. |

# Use Case Description

The DITAA software is a very simple command line tool with only a few modes of operation. While there are several features exposed through its ASCII syntax, there are only the following limited use cases identified:

1. Produce Diagram from Text
2. Process HTML Input
3. View Usage Help

The following diagram illustrates the relationship between the use cases and actors.

[](https://app.diagrams.net/?page-id=T5S9OR-i3GHdE-8xFNX2&scale=auto#G1by_JEGlmTloQEYnR8agsPQzdxeN2Z4uH)

Non-Functional Requirements

These non-functional requirements are intended to be met by the end of the Testing Phase.

|  |  |
| --- | --- |
| Identifier | Requirement |
| REQ11 | The system shall contain 5% fewer SLOC than the pre-existing implementation |
| REQ12 | The system shall pass tests for at least 1 software bug known to exist in the pre-existing implementation. |

Added Functional Requirements

These functional requirements are proposed in addition to those in the pre-existing implementation.

|  |  |
| --- | --- |
| Identifier | Requirement |
| REQ13 | The system shall render diagonal polygon edges in place of horizontal/vertical edges when two or more diagonally-consecutive ‘/’ or ‘\’ characters are used. ‘/’ or ‘\’ characters must have row offsets of 1, and column offsets of +1 and -1 from the character below them, respectively, to indicate a diagonal line. Diagonal edges may not be rendered as dashed lines. |
| REQ13.1 | The system shall render ‘r’ characters at the juncture of two diagonal edges as a rounded corner. |
| REQ14 | The system shall render the consecutive character sequences ‘:)’, ‘:(‘, and ‘:D’ as smiley face icons, in line with any surrounding free text. |

Maintainability Improvements

Improvements to the clarity and maintainability of the pre-existing implementation are also intended:

1. Adding Design-by-Contract specifications to each class, including class invariants, pre-conditions, post-conditions, and loop invariants.
2. Reviewing the class structure: Consolidating classes, and potentially creating new classes, leading to a more logical organization with only valid uses of inheritance.

# Team Journals

## Brad Schneider

**2020-09-10**

* 17:00-18:00
  + Created journal for tracking project work
  + Set up some infrastructure for project/homework development
    - Create new instance of Linux VM
    - Checked out DITAA code
    - Downloaded Intellij IDEA
* 18:00-18:30
  + Read through DITAA usage docs
  + Began skimming DITAA source code

**2020-09-13**

* 13:00-14:30
  + Held team meeting on Discord - see meeting minutes

**2020-09-14**

* 16:30-17:00
  + Read deeper into the Requirements Specification example in the appendices of the Marsic book

**2020-09-16**

* 19:00-21:00
  + Produced initial draft requirements table based on the DITAA documentation of features
  + Took a second pass over the initial draft to consolidate/re-number based on related requirements (e.g. refactor REQX, REQY, REQZ into REQX, REQX.1, REQX.2)
  + Created draft Use Case diagram
    - Identified very few use cases for DITAA; it is simple software
  + Started draft document for Requirements deliverable
    - Included both artifacts mentioned above

Christopher Menart

**2020-08-30**

* Installed Intellij IDEA
  + Initial impressions are positive; this IDE has a similar feel to PyCharm does not obstruct editing
    - Later realize that PyCharm and IDEA are in fact closely related

**2020-09-12**

* Downloaded DITAA source code
* Installed and ran Doxygen on DITAA source code

**2020-09-13**

* Compiled DITAA source code
  + Need to link com.sun.javadoc in order to compile successfully
* Held team meeting on Discord--see meeting minutes

**2020-09-15**

* Initial read of DITAA source code
  + Code is already quite clean by real-world standards. Code is sufficiently commented, classes are generally clear.
  + No obvious cases of egregious design such as lots of global variables
  + Possibly one demi-god class? ‘Diagram’ is rather long for one class, mainly constructor.
    - Possible target for refactoring

**2020-09-17**

* Reviewed initial functional requirements from Brad, added 1 new Req
* Added non-functional requirements (as dictated by project givens)
* Add initial discussion of possible maintenance improvements

**2020-9-21**

* Added initial proposed new functional requirements

Daniel Longendelpher

**2020-09-02**

* 12:00-12:30
  + Created Team Discord Channel

**2020-09-13**

* 13:00-14:30
  + Team meeting on Discord
  + Took action items to ask requirements questions on discussion forums and to look over DITAA code

**2020-09-14**

* 10:00-13:00
  + Installed Intellij IDEA
  + Downloaded and compiled DITAA source code

**2020-09-17**

* 19:00-20:00
  + Reviewing requirements document
* 21:30
  + Submitted initial requirements doc draft