SOFTWARE REQUIREMENTS SPECIFICATION

for

Code Visualization

Version 1.0

Prepared by : **Curtice Gough**
Joshua Hartzfeld
Catherine DiResta

Submitted to : **Dr. Philip Chan**
Instructor

September 26, 2023

Contents

1 Introduction

1.1 Purpose

The purpose of this Software Requirement Specification (SRS) is to clearly define the features, attributes, function, and general use cases of the software associated with the project titled "Code Visualization".

The intended audience of this SRS includes, but is not limited to, any developers, clients, and advisors associated with the project.

1.2 Scope

The scope of this document is restricted to the software developed for the project "Code Visualization".

This software will allow users to perform dynamic analysis on programs written in the TBD programming language through the use of visual diagrams and animations representing data structures present in said program, as well as the movement and behavior of data among those structures. It is important to note that this software will not perform algorithm visualization, but code visualization. The primary difference being the detection and delineation of data structures and storage mechanisms (e.g. stack, queue, list) as opposed to algorithms (e.g. search, sort, insert).

To allow for the correction of potential errors in the detection of data structures, the user will be able to redefine each diagram via the GUI menu options.

1.3 Overview

The following chapters and sections of this document contain information on the technical, functional, and logical requirements of the software associated with the project, as well as explanations of planned features/functionality.

2 Overall description

2.1 Product perspective

2.1.1 System interfaces

This software is designed to run under most Linux-based operating systems. A graphical window manager is necessary for the operation of this software.

2.1.2 User interfaces

Users will interact with the software via a graphical user interface (GUI).

2.1.3 Hardware interfaces

In addition to standard PC components, this software will require the use of a monitor, keyboard and mouse.

2.1.4 Software interfaces

This software is designed to send user-provided Java code to a tracing program called ”traceprinter” on the backend. Traceprinter will compile and run the code, then return JSON-formatted data representing everything the program did during the execution (program states, operations, data values, class types, etc.).

The software will also feature an interactive GUI built upon the PyQt5 framework.

2.1.5 Operations

Users will have the following actions available to them:

- Input source code
- Analyze code
- Step to next line
- Retype data structure/variable
- Hide/show data structure/variable

2.1.6 Site adaptation requirements

The software is only capable of operating on source code written in the Java programming language. The system on which the software will run must have a functioning X server/window manager.

2.2 User characteristics

The intended users of this product are beginners learning to write code in the Java programming language. Ideally, this product will be completely intuitive, requiring very little prerequisite knowledge to understand.

2.3 Constraints

At present, the code tracing program implemented on the backend only supports Java. Thus, any and all source code analyzed by the software must be written in the Java programming language.

Analyzing and detecting complex/custom data structures in arbitrary code is beyond the scope of this project. Data structures to be visualized are limited to exclusively those defined in section 3.2. Any and all others will be represented by a generic default diagram showing the different fields/values present in the object in question.

2.4 Assumptions and dependencies

2.4.1 Assumptions

- Users will provide valid Java source code
- Windows, Linux, and MacOS environments handle PyQt5 universally
- Java programs compiled at versions more recent than Java 8 can be traced by Traceprinter in the same way as programs compiled at Java 8

2.4.2 Dependencies

- Python 3
- PyQt5
- Java JDK 8 or newer

3 Specific requirements

3.1 External interfaces

3.1.1 Graphical user interface (GUI)

The GUI will be the primary and solitary interface with which the user will interact with the software. The primary window will consist of three sections:

- Data Structures View
  This main section in the center of the primary window contains visual representations of any data structures currently selected. Each diagram will be labeled, and appropriate animations will occur each time data is moved or modified. The user has the option to right-click a diagram to re-type or re-name the structure. Diagrams may be moved in this space using click-and-drag.

- Source View
  On the left side of the primary window is a section dedicated to input and viewing of user-provided source code. The user can paste Java code here, then click a button to begin execution. During execution, the line of code corresponding to the current program state will be highlighted.

- Structures List
  On the right side of the primary window is a list of all data structures present in the program up until the current point in execution. Similarly to the functionality in the Data Structures View, users may right-click on a list entry to rename or retype the associated data structure.

3.1.2 Java Traceprinter interface

Code pasted in the Source View will be sent as input to Traceprinter for analysis. Traceprinter will compile and execute the program, then return JSON-formatted data as output. This data will be used to determine data types/values at each given state in the target program.

3.2 Supported Data Structures

The will exclusively support and visualize the following data structures. All others will be represented using a generic standard diagram that simply lists fields/values. Classes listed below beginning with "codeviz.structures.*" are custom classes written for the purpose of providing a standard implementation, and are automatically imported before compilation. This list is subject to expansion as the project is developed.

- Array
- java.util.ArrayList
- java.util.LinkedList
- java.util.Queue
- java.util.Stack
- codeviz.structures.Tree
- codeviz.structures.BinaryTree
- codeviz.structures.Graph

3.3 Functions

- The system shall allow users to paste/input Java code for analysis
- The system shall spawn an instance of Traceprinter as a child process
- The system shall correctly parse JSON output from Traceprinter
- The system shall visualize pre-defined detected data structures in the GUI
- The system shall visually animate the movement of data between data structures
- The system shall allow users to re-type and re-name data structure diagrams
- The system shall represent code execution line-by-line at the user's discretion