Code Visualization

Project Plan

Project Title

Code Visualization

Team Members

- Curtice Gough cgough2019@my.fit.edu
- Joshua Hartzfeld jhartzfeld2020@my.fit.edu
- Catherine DiResta cdiresta2019@my.fit.edu

Faculty Advisor

- Ryan Stansifer ryan@fit.edu

Client

- Ryan Stansifer Florida Institute of Technology

Client Meeting Dates

- 19 January 2024

Goal/Motivation

The primary objective of this project is to achieve code visualization at a medium/high level (as opposed to algorithm visualization). Code tracing is often a complex task. Visualizing structures and the movement of data will aid users with debugging tasks while not spending too much time tracing execution by hand.

Approach

Key features:

- Interactive GUI

This program will feature an interactive GUI. The GUI includes a display pane used for showing diagrams to represent various data structures. In addition to showing the data structures themselves, there will also be animations for demonstrating the movement of data.

- Dynamic code analysis

Users will be able to step through source code during program execution line-by-line in a fashion reminiscent of popular debuggers such as GDB and x64dbg. This allows them to keep track of where they are in the program flow.

- User intervention

The data structure visualization engine may occasionally make incorrect assumptions about which data structure is being represented. To solve this problem, we take inspiration from popular decompilers such as Binary Ninja and Ghidra to allow the user to rename/retype visual elements during analysis.

Novel features:

- Our cutting-edge code analysis system offers users a remarkable level of flexibility by enabling users to specify their data structure. This innovative feature empowers users to enhance the accuracy and effectiveness of the system according to their specific needs and preferences.

Algorithms and Tools

- PyQt

  The GUI will be written using the PyQt framework. PyQt is a simple widget-based GUI framework that allows for easy setup, detailed adjustments, and cross-platform compatibility. Custom widgets will be created to represent each data structure.

- Traceprinter

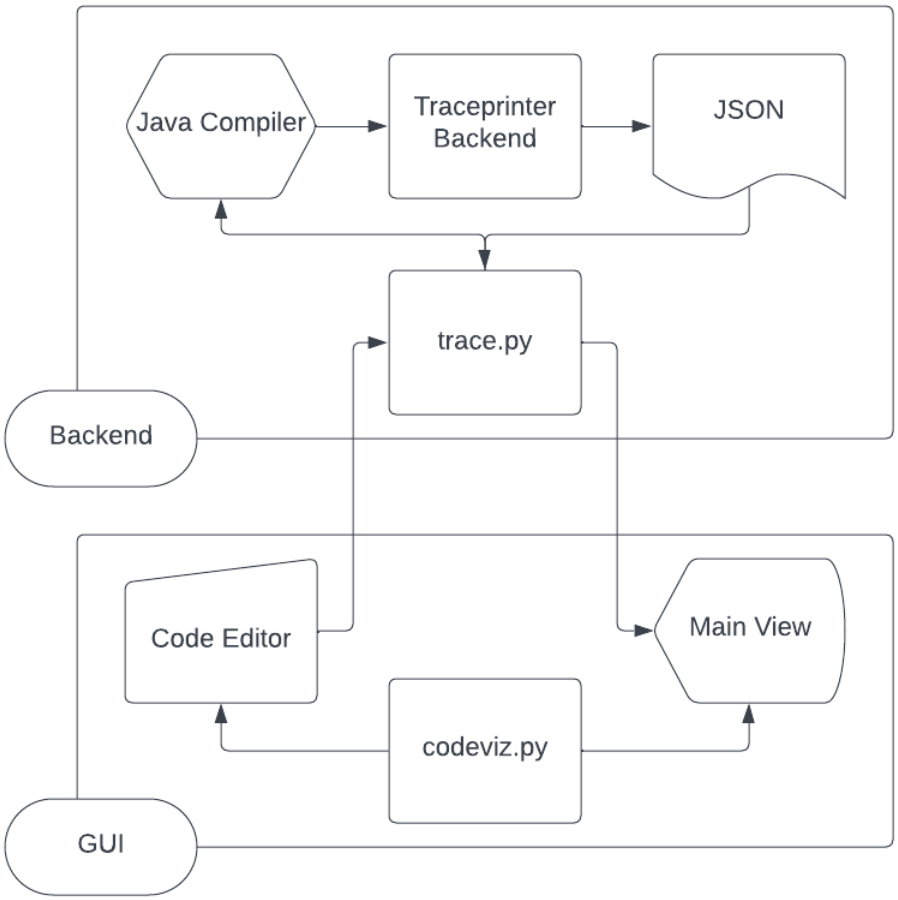
  Much of the backend functionality is borrowed from the University of Waterloo's *java_jail*. This backend was originally created for use in the University of Waterloo's *Java Visualizer* and is easy to adapt to our needs.

Technical Challenges

- Backend and frontend development are done separately. Integrating all of the pieces may prove difficult.
- Some of us are not very familiar with GUI development. Those who are more experienced will need to help the others along the way.

- Our faculty advisor would like Traceprinter to support modern versions of Java. Certain libraries used by Traceprinter are deprecated in modern version of Java, so the feasibility of this task is still in question.

System Architecture Diagram



Evaluation

- Speed

  How long does it take to fully generate the visual elements after submitting code?

- Reliability

  How often does the system correctly identify data structure types?

Progress Summary

Module/feature	Completion %	To do

Traceprinter backend	90%	Integrate with frontend
GUI	0%	Everything
Custom data structures	10%	Write the rest of the data structures listed in the requirements document

Milestone 4 Tasks

- Set up main window in PyQt
- Implement code editor
- Write custom List and Map implementations
- Modify Traceprinter to add multiple files to classpath

Milestone 5 Tasks

- Implement data structure diagrams
- Conduct evaluation and analyze results
- Create poster and ebook page for Senior Design Showcase

Milestone 6

- Implement data structure diagrams
- Test/demo of the entire system
- Conduct evaluation and analyze results
- Create user/developer manual
- Create demo video

Task matrix for Milestone 4

Task	Curtice	Josh	Catherine
1. PyQt main window	50%	50%	0%
2. Implement code editor	50%	50%	0%
3. Custom List/Map implementations	0%	0%	100%

4. Modify Traceprinter compile-time options	100%	0%	0%

1. Task 1 is to create a functional primary window and separate it into three panes. This task exists mostly as incentive to set up the boilerplate for PyQt.
2. Task 2 is to implement the Code Editor pane in the main window. At a minimum, there should be functionality for typing code and pressing a “start” button which sends the code to the traceprinter backend.
3. Task 3 is to write custom implementations of the List and Map data structures for use in the visualizer.
4. Task 4 is to modify the source code of Traceprinter to iterate over and compile every Java file in a directory rather than just the StdLib source files. This task is very simple and should require minimal effort.

Approval from Faculty Advisor

I have discussed with the team and approve this project plan. I will evaluate the progress and assign a grade for each of the three milestones.

Signature: _____ Date: _____