Enabling Repeatable Graph Based Experimentation and Learning

Team:
Austin Gregory, Blake Mulnix, Kyle Ferguson, Matthew Schaffer, and Peter Marasco

Senior Design
May 2020
Team 54

Client: Knowledge Centric Software Lab

Advisors: Suresh Kothari and Payas Awadhutkar

Problem Statement

Professors want to enable education via creating Control Flow Graphs of snippets of code. The Knowledge Centric Software lab (KCSL) had developed an open source prototype that enabled some of desired graphing functionalities. The prototype could produce visualizations of software graphs within a Jupyter Notebook. However, the prototype was buggy, did not work on all systems, and did not provide the features that our client desired.

Solution

Our team has improved upon the prototype to provide a product that provides the features our client desired. We achieved the following:

- Eliminated bugs within the existing prototype
- Got the application working on all systems and browsers
- Refactored and the existing code to make the project more maintainable and expandable for teams in the future
- Added the desired visualization features

Functional Requirements

- Ability to **visualize** Control Flow Graphs with appropriate styling and layout
- Ability to **interact** with graph
- **Integration** of visualizations with the Jupyter Notebook

Non-functional Requirements

- **Simplicity:** Installation and setup should be simple
- **Ease of Use:** Newcomers should be able to use the application easily
- **Performance**: The application's speed should not impede education

Operating Environment

Jupyter Notebook running in Chrome, Firefox, or Safari on Windows 10, Mac OS, or Linux.

System Design

Atlas

Code Snippet

```java
public class ExampleClass {

    public void compareIntegers(int a, int b) {
        if (a > b) {
            Logger.log("A is larger than B");
            return;
        } else if (b > a) {
            Logger.log("B is larger than A");
            return;
        } else {
            Logger.log("A and B are equal");
            return;
        }
    }

}
```

CHPG

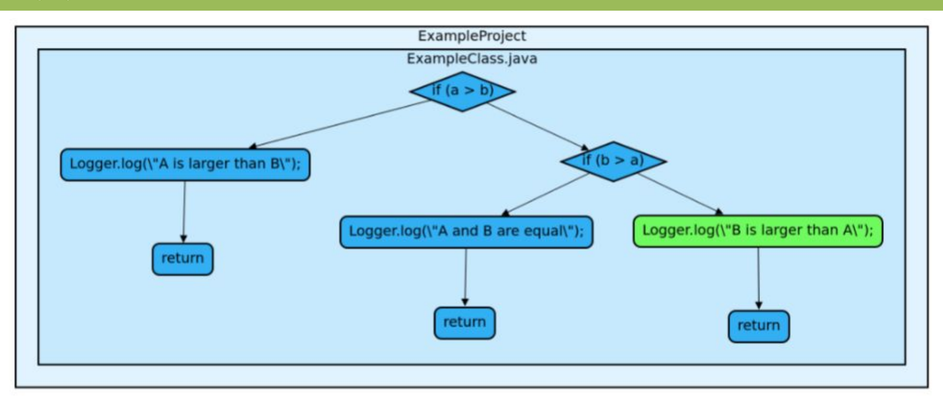
Visualization Server

Graph Visualization Algorithm

Graph Conversion Algorithm

Web Browser

Jupyter Notebook



Technical Details

Our application is contained within a JAR file that can be easily distributed to students and imported into a Jupyter Notebook. That notebook can then be opened within a browser and used to visualize CHPG graph files.

Testing

Our advisors agreed that manual testing to ensure that our application matches Atlas's output would be sufficient for our project.

Engineering Standards and Design Practices

Google's Java Style Guide:
We wrote and documented our code using these standards for Java development.

Agile SCRUM:
We used this development lifecycle to execute our project in an iterative manner and get rapid feedback.

Development Process

Our team elected to use an Agile SCRUM development lifecycle to execute our project. We elected to use flexible length sprints of approximately four weeks in duration. Our team held weekly discussions with our client to discuss near and long-term goals, priorities, and our progress. Our team then took this feedback to create and assign tasks that each team member would complete during the current sprint.