Code Duplication Checker

Jennifer Galley - Abebe Biru - Rithesh Baradi - Rishi Barua - Sachin Sebastian

# Motivation

We wanted to experiment with detecting duplicate code in C# because we were inspired by the amount of duplicate code we encountered in our day-to-day work, and the automation possibilities for detecting such clones. There are many tools available which implement clone detection on C#. However, we discovered a powerful, language-agnostic algorithm in the research called Count Matrix Clone Detection, which was initially implemented in and tested on Java, which we thought would prove interesting to recreate in C#. Furthermore, we wanted to test if the CMCD algorithm truly is language independent by testing it on C# code instead of Java.

# Design Decisions

We implemented the project in C# using Visual Studio due to familiarity with the language and the tools. This allowed us to use a simple console application as the user interface during development and expand to a more powerful Windows Forms UI later in the project. We chose to use Windows Forms for displaying the results of the CMCD algorithm because it was a good way to show the methods we had compared, their similarity scores, and enable clickthrough to the code visualization.

For the visualization of the results, we decided to generate an HTML file and highlight the diffs in the clones with CSS because that would be the easiest way to replicate common visualization techniques (such as in Visual Studio, git diffs) for a visualization most familiar to the average developer. The visualization highlights the diffs and includes the filenames, start and end lines of the code clones so that the developer is empowered to seek the clones out and refactor them into a common method.

# Testing and Debugging

We used unit tests, functional tests, and a console application to test and debug our project. For the visualization part, we used unit tests extensively to verify if the right HTML was generated. However, generating and maintaining the expected HTML results took quite a lot of time. One of the harder challenges was also being able to quickly spot the differences between the expected and actual results in a huge block of HTML. This was not easily visualized because the differences primarily lay in the class applied to a span tag in the HTML itself and there was no CSS involved. Sometimes there were also differences in whitespace that were not immediately obvious.

# Advancing the Research

The original research paper introducing the CMCD algorithm focused on its ability to detect the different types of code clones but did not expound upon the visualization of the results nor the UI for using the algorithm. Our implementation improves upon the original research by adding an intuitive UI for selecting the directory of code to inspect, and visualizing the resulting score from the CMCD algorithm for each pair of methods, as well as the differences between the code clones in a manner that is most relevant to the refactoring developer.