Northeastern University
CS5500 – Managing Software Development
J. Annunziato and M. Weintraub
January 21, 2018

Project Phase A

High Level Objective

The goal of the course project is to build an interactive tool for detecting plagiarism in programming assignments.

Project Overview

The goal of this project is to design, implement, test, and evaluate an application to help instructors detect situations where two or more students submit similar solutions to an assignment, in which one version derives from another version through one or more behavior-preserving transformations. Examples of such transformations are moving functions or methods to another location in the same file, renaming variables, classes, and methods, extracting sequences of statements into methods, etc. In general, the nature of these transformations depends highly on the programming language in which the assignments are written. Therefore, different techniques will be required in plagiarism detectors for different programming languages.

You will implement the system in three phases. In Phase A, you will be asked to develop a set of use cases that informally describe the behavior of the system from a users perspective and a mock-up of the systems user-interface. In Phase B, you will be asked to develop a set of UML Diagrams that reflect the high-level architecture of your system and a set of Java interfaces reflecting this design. Phase C involves implementing the system in Java.

Scope

Your plagiarism detector must work on programs written in a language of your choice, except you **may not apply your system to Java**. Options include but are not limited to: Python, JavaScript, HTML/CSS, vbasic, Rust, or Racket,. . . The main job is to design an algorithm for detecting similarities between two programs written in this language. The detector should go beyond a "textual diff," and consider more sophisticated transformations such as but limited to:

- the renaming of variables
- extracting code into functions
- moving code around

The detector should be able to handle multi-file projects, where files with similar content have different names.

It is up to you to decide how to approach the problem. Possible approaches may rely on syntactic similarity metrics, machine learning techniques, etc. You are encouraged to consult the research literature on plagiarism detection and related topics such as clone detection. However, proper attribution to any previous work upon which your solution relies must be given. Your plagiarism detection tool is expected to be interactive (e.g., clicking on a similarity report could pop up a window explaining in detail how one version could be derived from the other).

Logistics

This project will be done in a team of four people. The instructors will assign students into teams. Each team will be given access to an NEU CCIS GitHub repository. Students are expected to follow provided guidelines regarding issue tracking, using branches, etc.

The implementation of your systems core logic *must be in Java.* The front-end is entirely up to you. You can use a web approach using HTML/JavaScript/CSS or REACT or Ruby on Rails. Or, you can build a desktop app using JavaFX or Swing.

The use of external libraries for basic infrastructure such as parsing source code files, building abstract syntax trees, etc. is permitted. However, please note that the core logic that analyzes the two projects should be written entirely in Java.

Deliverables

In this phase, you are required to produce the following:

1. A one page statement describing your plan and the programming language that you plan to target.
2. Five to 10 use cases that describe the systems functionality (following the style used in the Pressman & Maxim textbook) covering at least two kinds of users.
3. One or more mock-ups of the systems user-interface as you envision it.

These deliverables must be placed in your team repository in a directory named phaseA by 23:59pm Eastern Time on the due date.