Обнаружение плагиата исходного кода программного обеспечения при помощи математических метрик сложности

А.Н. Мироненко
mironim84@mail.ru

Омский государственный университет им. Ф.М. Достоевского, Омск, Россия

Аннотация

В работе предлагается метод, основанный на двух алгоритмах: алгоритм предобработки исходного кода программы и алгоритм проверки является ли код плагиатом по его количественным характеристикам, вычисляемым на основе метрик сложности. Были выбраны метрики сложности Холстеда и Джилба. Предложенный метод был реализован в виде программы и протестирован. Тестирование проводилось на множестве программ написанных студентами в ходы выполнения лабораторных работ.

Введение

Использование метрик программного обеспечения (ПО) является достаточно новым направлением в сфере разработки. Применение метрик возможно практически на всех этапах жизненного цикла ПО. Традиционное их использование — это оценка и обеспечение качества ПО. Достаточно большое количество работ посвящено обзору основных подходов к измерению ПО. Например, в работах [1, 2]. Авторы [3] описывают различные аспекты применения метрик ПО в структурном тестировании, которое позволяет произвести оценку усилий, необходимых для его осуществления. Стоимость поиска и исправления ошибок или дефектов ПО является одной из самых крупным статей расходов. Исправление ошибок начинается на этапе описания требований и продолжается на протяжении всего процесса разработки. Как утверждает автор [4], за 25 – летнюю продолжительность жизни крупной программной системы размером 10 000 функциональных точек, почти 50 центов с каждого доллара тратится на поиск и исправление ошибок. Для сокращения этих расходов так же возможно использование метрик сложности. Однако метрик сложности программного кода могут использоваться не только для оценки сложность разработки, с целью повысить качество разработки, но и может помочь в решение задачи установления авторства исходного кода. В работе [5], исследуется проблема плагиата в исходном коде работ студентов. «С этой же проблемой столкнулись и массовые открытые онлайн курсы. Так, например, 73 из 239 студентов Массачусетского Технологического института, проходивших курс «CE 1.00: Introduction to Computers and Problem Solving», подверглись дисциплинарному наказанию за использование в своих работах чужих фрагментов кода» [5]. Работа [6], так же посвящена проблеме плагиата в программном коде. Данная статья посвящена определению плагиата в программном коде на основе его математических метрик сложности.

In: Sergey V. Belim, Nadezda F. Bogachenko (eds.): Proceedings of the Workshop on Data, Modeling and Security 2017 (DMS-2017), Omsk, Russia, October 2017, published at http://ceur-ws.org

1 Постановка задачи и ее решение

Под плагиатом в работе мы понимаем полное или частичное копирование программного кода без изменение либо с незначительной его модификацией. В работе [6] прилагается следующая классификация копирования исходного кода:

- Заимствования 1 рода -– код заимствуется без каких-либо изменений.
- Заимствования 2 рода -– код заимствуется без каких-либо серьезных изменений, но меняются имена переменных, их тип.
- Заимствования 3 рода -– код заимствуется, но изменяется посредствам добавления и/или удаления фрагментов кода, возможно, даже ни как не влияющих на работы программы. Так же могут быть заимствования 2 рода.
- Заимствования 4 рода – данный род заимствования характеризуется копированием логики работы и функциональности. Синтаксически код оригинала может абсолютно отличаться от заимствованного кода.

Четвертый род заимствования можно не учитывать, в данном случае сложно говорить о плагиате в явном виде т.к. человек заимствующих код копирует только «идею» (логику, функционал), но при этом создает собственную программу. Наиболее простым методом выявления плагиата является простое посимвольное сравнение кодов программ. Данный метод не зависит от языка, на котором реализована программа, но достаточно легко обходится элементарным переименованием переменных, функций, добавлением/удалением комментариев и т.п. Подходит для выявления заимствований только 1 рода. Существует различные методы выявления плагиата на основе токенизации исходного кода, например, алгоритм Хескела, выравнивание строк, жадное строковое замощение (Greedy String Tiling) и т.п. Процедура токенизации происходит следующим образом [7]:

- определяются классы операторов и им назначаются коды. Всем операторам языка (не являющимся операндами) присваиваются соответствующие коды классов
- строится строка полученных кодов (токенов), при этом последовательность кодов соответствует порядку появления операторов в коде.

Недостатком методов, в основе которых лежит токенизация является зависимость от конкретного языка программирования, однако эту проблему можно решить путем создания различных наборов, не пересекающихся кодов для различных классов языков, что усложняет процесс автоматизированной проверки, т.к. добавляется необходимость определения языка, при том, что существуют операторы, которые присущи разным языкам. Еще одним видом методов выявления плагиата, являются методы с использованием метрик. Они используют совокупность количества циклов, переменных, условий или их количества по отдельности. В данных методах нет привязки к языку программирования. Однако недостатком является высокий уровень ложных срабатываний при проверки небольших программ. Проанализировав различные подходы к выявлению плагиата предлагается метод на основе метрик, а именно, на основе метрик сложности Холстеда, Джилба. Метрики Холстеда строятся на большом количестве показателей таких как: ко-личество уникальных операторов программы, количество уникальных операндов программы, общее количество операторов, общее количество операндов, теоретическое количество уникальных операторов и теоретическое количество уникальных операн-дов. На основе этих показателей вычисляются: уровень качества программирования, сложность понимания программы, трудоемкость кодирования программы, уровень языка выражения, информационное содержание программы и оценка интеллектуальных усилий при разработке программы [8]. Метрика Джилба определяет сложность ПО по насыщенности кода программы условными операторами или операторами цикла. CL – абсолютная сложность, как количество операторов условия, cl – относительная сложность программы, как насыщенность кода операторами условия, т. е. cl=CL/n, где n – общее количество операторов [8]. Процедуру выявления плагиата можно разделить на два этапа: подготовка кода программы для дальнейшей работы с ним и определение является ли программа плагиатом. Алгоритм подготовки данных [9]: Программа представляется в виде точки на n-мерном пространстве, каждая i-я координата которой — это некоторая количественная характеристика. Для вычисления координат было предложено использовать метрики оценки качества ПО и количество используемых в программе операторов цикла. Алгоритм выявления плагиата [9]:

1. Представляем программу, которая подозревается в плагиате в виде точки A(x1, x2, . . . , xn), где xi — значение количественной метрики;

2. Представляем программу, с которой мы хотим сравнить подозреваемую программу B(x1, x2, . . . , xn), где xi — значение количественной метрики;

3. Размещаем полученные точки на n-мерном пространстве

4. Определяем расстояние между точками A(x1, x2, . . . , xn) и B(x1, x2, . . . , xn) в пространстве, если оно мало, то можно говорить, что подозреваемая в плагиате программа проверку не прошла.

2 Апробация

В качестве тестового множества были взяты программы, написаны студентами Омского государственного университета им.Ф.М. Достоевского в ходе выполнения лабораторных работ по курсу «Криптографические методы защиты информации». Каждая программа представляет собой реализацию, какого-либо криптографического алгоритма, начиная от простейшего шифра Цезаря (100 – 150 строк программного кода), заканчивая более сложным блочными шифрами. В общей сложности было обработано более 70 программ. Было сформировано семейство из семи множеств, каждое из которых представляло собой набор программ выполненных одним студентом, вида: X01.cs; X02.cs; X03.cs; X04.cs; X05.cs; X06.cs; X07.cs, X01.cs — это исходный код программы на языке C#. Элементы множеств сравнивались попарно «каждый с каждым», кроме того, проводилось сравнение «каждый с каждым» и внутри самих множеств, с целью проверить насколько будут «похожи» между собой программы, написанные одним и тем же человеком. Для проведения апробации предлагаемого метода выявления плагиата было реализовано два приложения: первое – для подробного анализа кода, без проверки на плагиат; второе -– для вычисления основных количественных характеристик, на основе которых происходит проверка на плагиат. На рис. 1 показан результат подробного анализа исходного кода программы, который может анализироваться человеком для выявления плагиата в ручном режиме или для проверки корректности работы программы выявляющей плагиат в автоматическом режиме. На рис. 3 представлен результат обработки тестового множества, показаны основные характеристики исходных кодов программы из него. F01.cs и K01.cs, F02.cs и K02.cs и т.п. – это исходные коды программы решающие одинаковые задачи и написанные разными студентами. После выполнения обработки тестового множества выбираются программы, которые необходимо проверить на плагиат, сравнение будет проводиться по принципу «каждый с каждым».

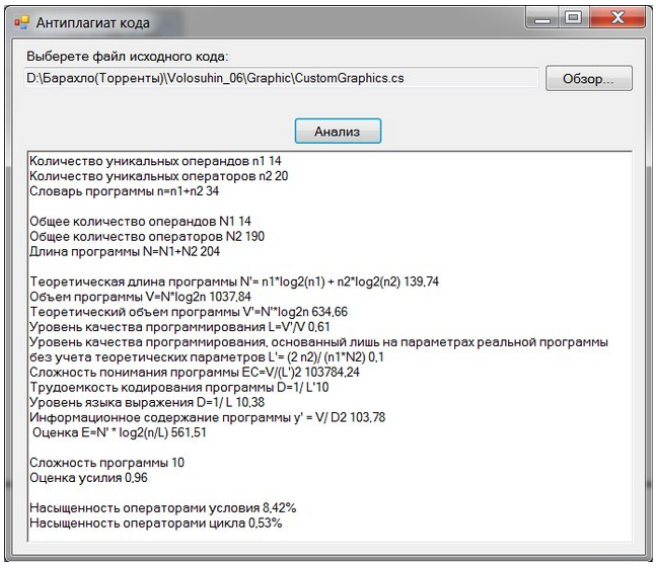

Рис. 1: Подробный анализ исходного кода

На рис. ?? представленным результаты работы алгоритма выявления плагиата. В результатах показывается, какие программы сравнивались и их схожестью в процентах. Проанализируем результаты проверки. Возьмем пару F01.cs и K01.cs, это программы решающие одну и ту же задачу, но написанные

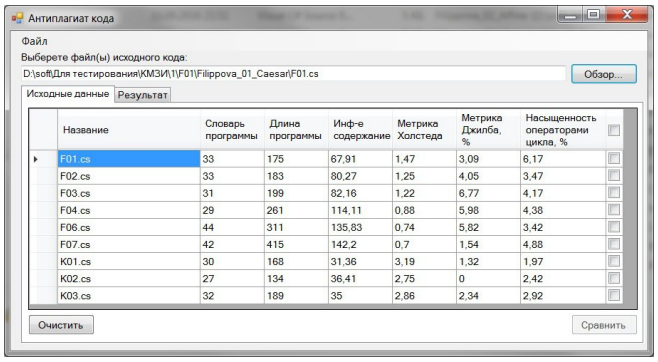

Рис. 2: Результат обработки тестового множества

разными студентами имеют схожесть менее 50% (48,15%). При этом известно, что программы реализуют один из простейших криптографических алгоритмов, шифр Цезаря, поэтому их размер невелик, F01.cs -– 175 строк кода, K01.cs – 168 строк кода. Как было отмечено ранее, недостатком методов выявления плагиата на основе метрик является высокий уровень ложных срабатываний, при проверки небольших программ исходя из этого и можно говорит, что при схожести менее 50% программы будут оригинальными. Так же анализируем другие пары.

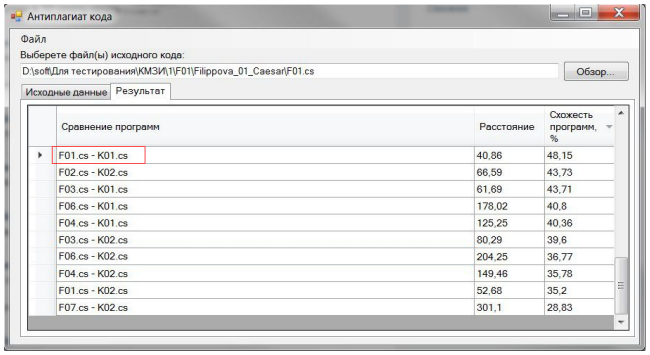

Рис. 3: Результаты работы алгоритма выявления плагиата

Анализ результатов и выводы

В работе был предложен метод выявления плагиата в исходном коде программ на основе математических метрик сложности. С целью апробации предлагаемого метода было реализовано два приложения: первое – для подробного анализа кода, без проверки на плагиат; вторая -– для вычисления основных количественных характеристик, на основе которых происходит проверка на плагиат. Оба реализованных приложения справились с поставленными задачами.

Список литературы

[1] Software measurement URL: `http://www.uio.no/studier/emner/matnat/ifi/INF5181/h11/undervisningsmateriale/reading-materials/Lecture-06/morasca-handbook.pdf` (дата обращения: 04.04.17).

[2] Measuring the difficulty of code comprehension tasks using software metrics URL: `http://crpit.com/confpapers/CRPITV136Kasto.pdf` (дата обращения: 04.04.17).

[3] Mrinal Kanti Debbarma, Swapan Debbarma, Nikhil Debbarma, Kunal Chakma, and Anupam Jamatia A Review and Analysis of Software Complexity Metrics in Structural Testing *International Journal of Computer and Communication Engineering. - 2013. - №2.*

[4] Software Quality Metrics: Three Harmful Metrics and Two Helpful Metrics URL: http://www.ppi-int.com/systems-engineering/free/\%20resources/Software\%20Quality\%20Metrics\%20Capers\%20Jones\%20120607.pdf (дата обращения: 20.04.17).

[5] Определение плагиата в исходном коде работ студентов *Молодежный научный форум: Технические и математические науки: электр. сб. ст. по материалам VII студ. междунар. заочной науч.-практ. конф. — М.: «МЦНО». — 2013 —№ 7(7)* URL: https://nauchforum.ru/archive/MNF_social/7(7).pdf.

[6] Ступенчатый метод проверки исходного кода программы на плагиат URL: http://nauchkor.ru/pubs/stupenchatyy-metod-proverki-ishodnogo-koda-programmy-na-plagiat-587d36555f1be77c40d58cf6 (дата обращения: 20.04.17).

[7] Обзор алгоритмов обнаружения плагиата в исходных кодах программ URL: http://rain.ifmo.ru/cat/data/theory/unsorted/plagiarism-2006/article.pdf (дата обращения: 19.04.17).

[8] Метрики сложности кода. Технический отчет 2012-2 URL: http://www.ispras.ru/preprints/docs/prep_25_2013.pdf (дата обращения: 19.04.17).

[9] А. Н. Мироненко Метод определения заимствований в программном коде с использованием его метрик сложности *Информационная безопасность и защита персональных данных. Проблемы и пути их решения: Материалы VIII Всероссийской научно-практической конференции.* Брянск: БГТУ, 2016.

Detection of Plagiarism of the Software Source Code Using the Mathematical Metrics of Complexity

Anton N. Mironenko

In this pepper offers a method based on two algorithms: the algorithm for preprocessing the program's source code and the validation algorithm is plagiarized by its quantitative characteristics, calculated on the basis of complexity metrics. The Holstead and Dzhilba complexity metrics were selected. The proposed method was implemented as a program and tested. Testing has been done on many of the programs written by the students in the lab progress.