Математическое и компьютерное моделирование:
материалы III Международной научной конференции
(Омск, 12 ноября 2015 г.). Омск, 2015. С. 195–196. УДК 004.05

А.Н. Мироненко

*Омский государственный университет им. Ф.М. Достоевского,
г. Омск*

ИСПОЛЬЗОВАНИЕ МЕТРИК ОЦЕНКИ КАЧЕСТВА ПРОГРАММНОГО КОДА ДЛЯ ОБНАРУЖЕНИЯ ЗАИМСТВОВАНИЙ

Существует много решений позволяющих выявить плагиат в рефератах, курсовых, дипломах и т.п., но они абсолютно не применимы для проверки на плагиат программного кода. Можно говорить, что программа является плагиатом, если она содержит в себе значительную часть (на уровне языка программирования) другой программы. При этом субъект, который пытается выдать чужую программу за свою вносит незначительные изменения в код, например, добавляет пустые строки, переименовывает переменные или функции, переносит отдельные функции в другой участок кода, что позволяет успешно проходить проверку на плагиат, если используются системы выявляющие заимствования в рефератах, курсовых работах и т.п., но по сути, все это не делает «новую» программу оригинальной, она по-прежнему является плагиатом.

Прежде чем приступить выявлению плагиата в программном коде необходимо каким-то образом его подготовить для дальнейшей работы. Существуют различные подходы к решению данной задачи [1], мною был выбран подход, в котором программа представляется в виде точки на n-мерном пространстве [2], каждая i-я координата которой – это некоторая количественная метрика, т.е. программа – это точка. В качестве координат могут выступать, например, средняя длина строки, средняя длина имени переменной или функции, количество используемых стандартных команд или операторов цикла/ветвления и т.п.

Для вычисления i-й координаты (количественной метрики) предлагается использовать метрики оценки качества программного продукта (Метрика Холстеда [3] и метрика Джилба [3]) и количество используемых в программе операторов цикла.

Описанные действия можно назвать подготовительным этапом.

Алгоритм выявления плагиата состоит из следующих шагов:

1. Представляем программу, которая подозревается в плагиате в виде точки $A(x_1, x_2, ..., x_n)$, где xi – значение количественной метрики.

2. Представляем программу, с которой мы хотим сравнить подозреваемую программу $B(x_1, x_2, ..., x_n)$, где x_i – значение количественной метрики.

3. Размещаем полученные точки на n-мерном пространстве.

4. Определяем расстояние между точками $A(x_1, x_2, ..., x_n)$ и $B(x_1, x_2, ..., x_n)$ в пространстве, если оно мало, то можно говорить, что подозреваемая в плагиате программа проверку не прошла.

Очевидно, что по предложенному алгоритму можно осуществлять проверку не одной программы, а сразу нескольких и сравнивать можно с программами из уже существующей базы уникальных программ, причем хранить коды в базе не обязательно, достаточно сохранить только метрики.

Литература.

1. Обзор автоматических детекторов плагиата в программах. URL: `http://detector.spb.su/bin/view/Sandbox/ProjectOutput`.

2. Обзор алгоритмов обнаружения плагиата в исходных кодах программ. URL: `http://rain.ifmo.ru/cat/data/theory/unsorted/plagiarism-2006/article.pdf`.

3. Метрики кода и их практическая реализация в Subversion и ClearCase. URL: `http://cmcons.com/articles/CC_CQ/dev_metrics/mertics_part_1`.