УДК 004.02

ОБЗОР АЛЬТЕРНАТИВНЫХ МЕТРИК ПРОГРАММНОГО КОДА ДЛЯ ПОИСКА ПЛАГИАТА

М.В. МАТЮШ
(Полоцкий государственный университет)

Представлен обзор альтернативных метрик программного кода. Анализ программного кода используется для решения актуальных задач определения качества, поиска плагиата и определения авторства исходного кода. Для каждой конкретной задачи могут подходить только определённые методы и алгоритмы анализа исходного текста программ. В свою очередь, для каждого метода и алгоритма требуется собственный, уникальный набор характеристик программного кода. На текущий момент существует множество методов и алгоритмов для анализа и поиска уникальных характеристик программного кода, часть из которых будет кратко представлена и обобщена в работе.

Актуальность анализа исходного кода. При создании программного обеспечения все затраты приходятся на создание первого образца, а дальнейшее его тиражирование стоит несравненно меньше, т.е. себестоимость программного обеспечения составляет стоимость исходного кода этого программного обеспечения. При модификациях программного обеспечения все изменения ведутся непосредственно над программным кодом. Анализ исходных кодов программ может использоваться для решения актуальных задач: определения качества, поиска плагиата, определения авторства.

Качество кода в свою очередь включает определение метрик, «читаемости» кода, документирования кода, надлежащего форматирования, количества ошибок проектирования. Качество кода в большинстве случаев определяет стоимость исходного кода. Качественный код легче поддается дальнейшим модификациям, содержит в себе меньше ошибок, может поддерживаться и модифицироваться программистами, не относящимися к числу авторов первоначального варианта.

Хищение программного продукта – преступление, связанное с использованием копии программного обеспечения без соблюдения лицензионного соглашения, по которому оно распространяется. Но хищение исходных кодов программного продукта – более тяжкое преступление. Поэтому поиск плагиата именно в исходных текстах программ является актуальной задачей. Качество поиска плагиата определяет возврат затрат на создание первой копии программного продукта.

В случае исходного кода вредоносного программного обеспечения задача правоохранительных органов состоит в определении авторства для дальнейшего привлечения к ответственности. Также определение первоначального авторства может быть актуальным при определении авторства исходного кода при двух и более лиц. Задачи поиска плагиата и определения авторства различные по реализации, хотя могут преследовать одну и ту же цель. Так, задача поиска плагиата подразумевает сравнение двух исходных текстов программ и поиск одинаковых блоков, а задача определения авторства – поиск общих черт у различных по назначению программ одного автора.

Цель данного исследования – обобщить известные и предложить теоретическое описание собственных, альтернативных метрик программного кода, которые используют совокупность стандартных характеристик и смогут охватить большой спектр решения задач анализа программного кода.

Определение качества программного кода. К стандартным характеристикам программного кода можно отнести качество. Качественные характеристики носят субъективный характер и для различных ситуаций определяются различными способами. Характеристики программного кода определяются на основе метрик – численных показателей, определяемых на основе программного кода. Однако зачастую разработчики программного обеспечения не в состоянии адекватно оценить значение той или иной метрики, поскольку не до конца понимают её природу [1].

Из основных видов метрик программного кода можно выделить [2]:

1) *количественные метрики*. Самой элементарной количественной метрикой является количество строк кода. Различают физические и логические строки, при этом к последним относят количество команд в программе. Также к количественным метрикам относят: количество пустых строк; количество комментариев; процент комментариев; среднее число строк для функций (классов, файлов); среднее число строк, содержащих исходный код для функций (классов, файлов); среднее число строк для модулей и многие другие характеристики, основанные на подсчете некоторых единиц в коде программы;

2) *метрики сложности потока управления программы*. Суть данного типа метрик заключается в строительстве ориентированного графа, содержащего лишь один вход и один выход. При этом вершины графа соотносят с теми участками кода программы, в которых имеются лишь последовательные вычисления и отсутствуют операторы ветвления и цикла, а дуги соотносят с переходами от блока к блоку и ветвями выполнения программы. Условие при построении данного графа: каждая вершина достижима из начальной и конечная – из любой другой вершины. Данная метрика определяет сложность программного кода;

3) *метрики сложности потока управления данными*. Необходимы для расчёта информационной прочности использования различных типов данных, к примеру, анализа характера использования переменных из списка ввода-вывода;

4) *метрики сложности потока управления и данных программы*. Данный класс метрик устанавливает сложность структуры программы как на основе количественных подсчетов, так и на основе анализа управляющих структур;

5) *объектно-ориентированные метрики.* Эти метрики основаны на анализе методов класса, дерева наследования и т.д. В этом классе можно выделить следующие метрики: центростремительное сцепление, т.е. количество классов вне этой категории, зависящих от классов внутри этой категории; центробежное сцепление, т.е. количество классов внутри этой категории, зависящих от классов вне этой категории; нестабильность; абстрактность; суммарная сложность всех методов класса; глубина дерева наследования; количество потомков; сцепление между классами, показывающее количество классов, с которыми связан исходный класс; множество методов, которые могут быть потенциально вызваны методом класса в ответ на данные; недостаток сцепления методов;

6) *метрики надежности.* Метрики, близкие к количественным, но основанные на количестве ошибок и дефектов в программе: количество структурных изменений, произведенных с момента прошлой проверки; количество ошибок, выявленных в ходе просмотра кода; количество ошибок, выявленных при тестировании программы, и количество необходимых структурных изменений, необходимых для корректной работы программы;

7) *гибридные метрики*. Метрики данного класса основываются на более простых метриках и представляют их взвешенную сумму.

Как правило, все перечисленные метрики являются показателями качества программного кода, но плохо подходят для поиска плагиата и определения авторства. В то же самое время совокупность метрик может определять другие нестандартные характеристики программного кода.

Стандартная задача поиска плагиата. Если рассматривать задачу плагиата исходных текстов программ, то здесь нужно четко разделять *плагиат дизайна программного обеспечения* и *плагиат исходных кодов*. Сходства программного продукта – основная причина анализа исходных кодов. Однако это не дает 100 % гарантии использования тех же самых исходных кодов. В данном случае похищен может быть только дизайн приложения. Следующей причиной анализа на плагиат может послужить общая или схожая функциональность двух приложений. Но схожая функциональность – это больше чем причина анализа, это неотъемлемое свойство плагиата, причем тут нужно понимать и видеть разницу между общим и модульным функционалом, так как могут быть различные назначения двух программных продуктов, т.е. различная общая функциональность, но при детальном анализе и делении на составные компоненты, каждый из которых имеет собственную функциональность.

Эти отдельные компоненты или группы могут совпадать по функциональности с другим программным обеспечением, и здесь может иметь место плагиат. Компоненты также могут делиться на более мелкие части-модули, и так до эмпирической константы, которая в случае исходного кода может быть функцией, процедурой или методом.

Более детальное рассмотрение до уровня переменных не имеет смысла, так как особенность исходных кодов в том, что используется конечный алгоритм, поэтому при анализе плагиата важен не столько лексический набор инструкций, сколько синтаксический.

Когда мы определили эмпирический минимум в виде функции, мы имеем в виду реализацию этой функции. Функции являются идентичными, если они идентичны по семантической составляющей. В качестве подобного минимума допустимо использовать отдельную строку кода в исходном файле. Это объясняется, с одной стороны, тем, что все основные операции во время кодирования, включая копирование и вставку, выполняются над одной или несколькими строками, а с другой – позволяет упростить предварительную обработку кода. Однако использование строк негативно сказывается на качестве обнаруживаемых клонов, а также существенно затрудняет определение синтаксических характеристик кода [2]. Наиболее предпочтительным и целесообразным является использование токенов в качестве единицы. Такой подход уменьшает количество ложных срабатываний, но приводит к существенному увеличению количества единиц кода в сравнении со строками кода и потребует реализации лексического анализатора для каждого поддерживаемого языка программирования. Введение единиц кода позволяет абстрагироваться от конкретных особенностей работы с исходным кодом. Наиболее предпочтительным и целесообразным является использование токенов в качестве единицы. Такой подход уменьшает количество ложных срабатываний, но приводит к существенному увеличению количества единиц кода в сравнении со строками кода и потребует реализации лексического анализатора для каждого поддерживаемого языка программирования. Введение единиц кода позволяет абстрагироваться от конкретных особенностей работы с исходным кодом. Но методы и алгоритмы, применяемые в поиске плагиата, не позволяют определить качество программного кода, а проверять качественный и не качественный программный код на плагиат нецелесообразно, так как присутствие плагиата в разных программах подразумевает, что эти программы схожи по своему качеству.

Особенности анализа программного кода. Исходные коды программ отличаются от обычного текста, им присущи специфические характеристики: структурированность, зависимость от входных данных, ограниченный набор лексем, часто повторяющиеся синтаксические конструкции и т.д. Применение стандартных алгоритмов поиска плагиата будет давать неверные результаты, так как в текстах программ всегда будут встречаться одинаковые конструкции, операторы, переменные, литералы и т.д. [3; 4]. Также стандартные алгоритмы поиска плагиата не учитывают паттерны проектирования, это такая повторимая архитектурная особенность, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста [5]. Очевидно, что использование одних и тех же правил проектирования исходного кода программ приводит к большой их схожести, затрудняя анализ исходных текстов на наличие плагиата. Далее предложим собственные характеристики с учетом ранее рассмотренной специфики программного кода.

Почерк программиста. В исходном тексте программ, помимо стандартных характеристик, сохраняется больше характеристик, свойственных конкретному автору (наименование переменных, стиль написания текста, лингвистические особенности комментариев, количество разных типов данных, стабильный уровень определенных метрик и др.) [3; 6 – 8], т. е. уместно будет употреблять понятие *почерк программиста*.

Задача определения почерка программиста заключается в поиске необязательных характеристик кода, которые присущи конкретному программисту, но это не является тривиальной задачей, как в случае с рукописным почерком. Почерк программиста может проявиться в архитектурном решении, например, в злоупотреблении каким-либо шаблоном проектирования, или при кодировании в правилах именования переменных, методов и других структурных единиц языка программирования, таких как:

- именование классов, методов и переменных, локальных переменных метода, параметров метода;
- место инициализации переменных класса; создание неиспользуемых методов класса (т.е. избыточность, пришедшая из проектирования, но не используемая в текущей реализации);
- количество комментариев, вид комментариев, место комментариев;
- длина имени методов; использование типов;
- использование перечислений;
- объявление интерфейсов, но не использование полиморфизма;
- употребление определённого шаблона во всех программах автора, определённого эталонного набора метрик программного кода, которые имеют одинаковое значение во всех программах конкретного программиста;
- и множество других.

Почерк программиста в первую очередь должен однозначно определять автора исходного кода, а также в задачах поиска плагиата находить почерк автора. Так как под почерком программиста мы также подразумеваем определённый поддерживаемый уровень метрик исходного кода, то качество почерка имеет эквивалентную зависимость с качеством программного кода. Таким образом, данная характеристика охватывает одновременно все три вышеперечисленные задачи: расчет качества кода, поиск плагиата и определение авторства.

Акцент программиста. Как и в случае с людьми, владеющими различными языками, у программистов также есть такое понятие, как «родной язык программирования». В отличие от языка общения, «родным» языком программирования является не тот, что изучался первым, а тот, на котором программист пишет постоянно, тот, на котором создавать программный продукт ему комфортно. И если случается, что программисту следует написать программу на не «родном» для него языке программирования, в исходном тексте этой программы могут появиться особенности, присущие только его «родному» языку программирования, т.е. проявится *акцент программиста*. Акцент программиста полностью не отделен от почерка программиста и является составной частью этой характеристики. Но в то же самое время выделение акцента программиста позволяет помимо большей точности в задачах поиска плагиата и определения авторства еще и проводить статистические исследования по миграции программистов с того или иного языка программирования. А также акцент позволяет определить отрицательный коэффициент соответствия стандартов программирования для исследуемого программного кода, что в свою очередь является качеством кода.

Эвристические методы для выделения акцента программиста должны выделить априорное множество обязательных и необязательных характеристик, присущих определённому языку программирования, сведенное в общий класс особенностей, и с последующим поиском этих особенностей в исходном коде с отличным языком программирования данного класса. Данная характеристика также может предполагать неопределенность в том случае, когда исходный код не соответствует ни одному классу акцентов из проверяемого множества, причем даже классу соответствующего языка программирования исходного кода, над которым будет осуществляться анализ. В этом случае считается, что это не акцент, а простая безграмотность программиста.

Грамотность программиста. Очевидным является то, что при поиске акцента программиста этот процесс становится частью процесса определения уникального почерка программиста. Поэтому обязательные стилистические особенности программирования не панацея к действию, так как изначально человек может придерживаться правил, и это из задачи поиска акцента перейдет в поиск почерка программиста. Если сравнивать с языком как словесным выражением мысли, то такие программисты являются неграмотными. Последнее предположение выделяет еще одну метрику – *грамотность программиста.*

Грамотность программиста – это соответствие исходного кода программиста стандартам, предложенным и повсеместно применяемым для конкретного языка. Эта задача актуальна, к примеру, при приеме на работу и при отслеживании качества программного кода. Решение задачи сводится к определению критериев грамотности, правил оформления и далее – последовательная проверка кода программиста на соответствие. К правилам может относиться именование переменных, функций или методов, далее принятая синтаксическая последовательность лексем языка. Также следует сразу отметить, что правила оформления для одного языка не соответствуют правилам оформления для другого. Из-за этого и появляется акцент при переходе от одного языка программирования к другому.

Грамотность программиста, с одной стороны, является противоположной характеристикой к акценту программирования, с другой – неотъемлемой частью почерка программиста, определяющего его качество. Но помимо этого грамотность определяет корректность инструкций и правил, которые характерны для большой части языков программирования.

Уникальный стиль программиста. Почерк программиста охватывает все остальные характеристики, но однозначно не определяет ту уникальную особенность, которая характерна именно для этого программиста. Эту особенность назовем *уникальным стилем программиста*, который будет рассчитываться путем поиска уникальных показателей ранее рассматриваемых характеристик и иметь как положительную, так и отрицательную тенденцию к изменению качества исходного кода. Типичными метриками, отражающими уникальный стиль программиста, могут быть нестандартное комментирование исходного кода, добавление собственных приставок и суффиксов к именам классов, методов и переменных.

Отличительной особенностью данной характеристики является то, что ее может намеренно изменять сам программист. Данная характеристика может быть специально введена программистом для защиты своего исходного кода. Также можно разделить стиль программирования на плохой и хороший по отношению к правилам оформления программного кода, а соответственно, и к качеству программного кода. Причем в этом контексте плохой может означать только, что автор исходного текста специально не следует общим правилам языка программирования, придерживается собственных по причине безграмотности или с целью оставить код уникальным. Хорошим стилем считается стиль, который не противоречит правилам оформления, но имеет избыточные комментарии или инструкции, являющиеся отличительными чертами автора.

Чистота программного кода. Положительная динамика данной характеристики является пропорционально зависимой от низких показателей акцента программиста, нулевых показателей уникальности стиля программиста, высоких показателей грамотности программиста и остальных хороших показателей почерка программиста. Предлагается считать программный код «чистым», если он идеален для конкретно решенной задачи. Основное затруднение в поиске данного критерия – это заранее определённые идеальные решения, что является утопичным. Поэтому предлагается применять данную характеристику только к определённым инструкциям и соответствию шаблонам проектирования.

Особенности командного программирования. Современные высоконагруженные проекты – результат работы не одного, а группы программистов. Разрабатываться командой могут как отдельные несвязанные модули, так и один класс (как синтаксическая единица языка). Эта особенность может потребовать применять вышеперечисленные метрики не к одному программисту, а к команде. В этом случае это будет общая усредненная характеристика всех участников команды. Также в случае командного программирования, если будут известны работы каждого программиста и при предварительном наполнении эталонных характеристик для каждого программиста, станет возможным проводить анализ на процент участия в написании исходного кода каждого программиста в команде.

Еще одной особенностью командного программирования является разный уровень навыков у каждого программиста, что выражается в различных характеристиках, а именно показателях почерка, грамотности, акцента и т.д. Данная особенность позволяет определить максимум и минимум, а также измерить динамику изменения характеристик в программном коде. Данный показатель позволит более качественно выделить общую динамику изменения показателей всей команды при изменении ее участников.

Также стоит отметить, что именно особенность командного программирования в совокупности с миграцией разработчиков между компаниями-производителями программного обеспечения и порождает

прецеденты плагиата. Подразумевается, что предлагаемые выше альтернативные метрики программного кода после их реализации смогут выявлять и такие нарушения авторского права.

В заключение проведенного исследования можно сделать следующие **выводы:**

- вопросы анализа программных кодов в виде решения трех задач: расчет качества, поиск плагиата и определение автора исходного кода – являются актуальными;

- приведены стандартные метрики исходного кода. Показана стандартная задача поиска плагиата в исходных текстах программ. Рассмотрены особенности анализа программного кода в сравнении с аналогичным литературным текстом;

- предложены собственные альтернативные метрики: почерк программиста, акцент программиста, грамотность программиста, уникальный стиль программиста и чистота исходного кода. Также показана ситуация анализа при командном программировании;

- представлен обобщённый обзор предлагаемых альтернативных метрик для дальнейшего продолжения исследования в данной области анализа исходного текста программных кодов, а именно планируется подготовка алгоритмического обеспечения по выделению каждой из предложенных характеристик с целью решения задач расчета качества, поиска плагиата и определения авторства.

ЛИТЕРАТУРА

1. Проблемы автоматизированного исследования качества программного кода // Журнал научных публикаций [Электронный ресурс]. – 2010. – Режим доступа: http://jurnal.org/articles/2010/inf26.html. – Дата доступа: 26.07.2013.
2. Метрики кода программного обеспечения // PVS-Studio [Электронный ресурс]. – 2008. – Режим доступа: http://www.viva64.com/ru/a/0045/. – Дата доступа: 01.08.2013.
3. Анализ алгоритмов выявления плагиата в кодах программ, написанных на языках высокого уровня // Современная техника и технологии: XVIII междунар. науч.-практ. конф. [Электронный ресурс]. – 2012. – Режим доступа: http://www.lib.tpu.ru/fulltext/v/Conferences/2012/C2/V2/v2_212.pdf. – Дата доступа: 10.07.2013.
4. Использование статистических характеристик программного кода в задачах дедупликации и оценки качества // Науч.-попул. интернет-журнал [Электронный ресурс]. – 2012. – Режим доступа: http://novainfo.ru/ /archive/9/ispolzovanie-statisticheskih-harakteristik-programmnogo-koda-v-zadachah-deduplikacii-i-ocenki-kachestva. – Дата доступа: 25.07.2013.
5. Шаблон проектирования // Википедия: свободная энцикл. [Электронный ресурс]. – 2002. – Режим доступа: http://ru.wikipedia.org/wiki/Шаблон_проектирования. – Дата доступа: 20.09.2013.
6. Идентификация дублирования и плагиата в исходном тексте прикладных программ // Институт проблем управления РАН [Электронный ресурс]. – 2005. – Режим доступа: http://lab18.ipu.ru/projects/conf2006/1/15.htm. – Дата доступа: 22.08.2013.
7. Поиск плагиата в исходных текстах программ // Белорус. гос. ун-т [Электронный ресурс]. – 2011. – Режим доступа: http://www.fpmi.bsu.by/ImgFpmi/Cache/36173.pdf. – Дата доступа: 03.09.2013.
8. Метрики сложности программного обеспечения как критерии оценки плагиата в исходных кодах программ // Материалы междунар. науч.-практ. конф. [Электронный ресурс]. – 2012. – Режим доступа: http://www.apriori-nauka.ru/uploads/files/RIBANOVK6.pdf. – Дата доступа: 21.09.2013.

Поступила 23.09.2013

REVIEW OF ALTERNATIVE METRICS PROGRAM CODE FOR SEARCH OF PLAGIARISM

M. MATSIUSH

Analysis of the alternative metrics of program code is presented, which is used to solve the actual problems of determining the quality, searching plagiarism and authorship of source code. For each specific target can be approached only certain methods and algorithms of analyzing the source code of programs. In turn, each method and algorithm requires its own unique set of characteristics of program code. At the moment there are a lot of methods and algorithms for analyzing and searching for unique characteristics of the program code, part of which will be briefly presented and summarized in the article. The purpose of this paper is to summarize the well-known and provide a theoretical description of own, irregular specifications of the software code, which use a set of standard features and will be able to cover a wide range of solutions the problems of analyzing the program code.