УДК 621.391.7.001

Д. А. Маєвский, д-р техн. наук,
Ю. П. Чербаджи

СИСТЕМА МЕТРИК ДЛЯ АТРИБУЦІЇ ПОЧАТКОВОГО КОДУ ПРОГРАМНИХ ПРОДУКТІВ

Анотація. *Розроблено систему метрик ідентифікації авторства програмних продуктів та інформаційну систему на її основі. Виконано дослідження точності атрибуції на підставі даних про наявні проекти. Завдяки інформаційній системі можливо формування бази даних профілів, яка допомагає контролювати можливі правопорушення недобросовісних авторів.*

Ключові слова: *атрибуція, початковий код, встановлення авторства, інформаційні системи, комп'ютерна безпека*

D. A. Maevsky, ScD.,
J. P. Cherbadzhy

SYSTEM OF METRICS FOR ATTRIBUTION OF SOURCE CODE OF SOFTWARE PRODUCTS

Abstract. *The system of metrics of authorship identification of software and information system based on it were created. The research established the accuracy of the algorithm based on the available projects. Thanks information system database of profiles can be released, which helps to control possible violations unscrupulous authors.*

Keywords: *attribution, source code, establishment of authorship, information systems, computer security*

Д. А. Маевский, д-р техн. наук,
Ю. П. Чербаджи

СИСТЕМА МЕТРИК ДЛЯ АТРИБУЦИИ ИСХОДНОГО КОДА ПРОГРАММНЫХ ПРОДУКТОВ

Аннотация. *Разработана система метрик идентификации авторства программных продуктов и информационную систему на ее основе. Выполнены исследования точности атрибуции на основе данных об имеющихся проектах. Благодаря информационной системе возможно формирование базы данных профилей, которая помогает контролировать возможные правонарушения недобросовестных авторов.*

Ключевые слова: *атрибуция, исходный код, установление авторства, информационные системы, компьютерная безопасность*

Вступ

За останні роки значно зросли загрози несанкціонованого шкідливого впливу на інформаційні та керуючі комп'ютерні системи. Втрата інформації через дію комп'ютерного вірусу – це сьогодні, мабуть, найменше зло. Але комп'ютерні віруси сьогодні не тільки видаляють інформацію. Вони інформацію крадуть, можуть втручатися в роботу інформаційних систем критичного застосування й налаштовувати її на шкідливі дії. Існує багато методів пасивного захисту від вірусів та шкідливих програм – антивірусні програми, файрволи та інше. Але активна протидія зловмисникам, яка полягає в їх виявленні та нейтралізації сьогодні практично не використовується. На перешкоді цього стають труднощі в ідентифікації автора шкідливого програмного коду – атрибуції.

Теоретично, атрибуція програмних продуктів має відбуватись або за її двійковим кодом або за початковим кодом (якщо він є відомим).

Перша задача – атрибуція за двійковим кодом на сьогодні не розв'язана. Більш того, навіть не вироблено шляхів її розв'язання.

Друга задача – атрибуція за початковим кодом значно легша і тому в цьому напрямі вже досягнуто певних результатів. Однак, її розв'язання є ще далеким від досконалості.

Всі методи атрибуції за початковим кодом можна умовно поділити на такі категорії:

– методи, що засновані на кластеризації (метод XPlag [1]; метод, заснований на локальному вирівнюванні [2]; метод SCAP за допомогою N-грамм на байтовому рівні [3, 4]);

– метричні методи, що використовують спеціальні метрики (метрики Холстеда [5], метод Plague [6]; метод який враховує особ-

ливості мови програмування С [7]; метод IDENTIFIED [8] та отримання відбитків програми Java [9]).

Підходи більшості методів, заснованих на кластерізації, перейшли з природних мов. Інші методи були створені та використовуються для виявлення авторства лише у штучних мовах, якими є мови програмування.

Методи визначення авторства текстів на природних мовах почали розвиватися з тридцятих років минулого сторіччя і на даний час вважаються добре дослідженими. Частина з них перейшла до встановлення авторства штучних мов. Прикладом є метод N-грам, який використовується у більшості підходів. Саме ці методи можна віднести до методів, заснованих на кластеризації. Одним з таких методів є метод, у якому проводяться експерименти за допомогою підходу SCAP (профіль автора початкового коду), який полягає у використанні N-грам [3; 4]. Однією з переваг такого методу є те, що він не залежить від мови, оскільки заснований на інформації низького рівня. В результаті, він може бути застосований без будь-яких додаткових змін для наборів даних, де програми написано на C++, Java, Perl і т.д. Більш того, він не вимагає численних навчальних прикладів від кожного учасника, оскільки ґрунтується на одному профілі від одного автора. Чим більше програм початкового коду доступно для кожного автора, тим більшу довжину профілю можна вибрати і тим надійніше профіль автора. Недоліком цього методу є те, що він суб'єктивний і потенційно упереджений через вибір розміру N-грам і довжини профілю. Метод SCAP напівавтоматичний і тому відкритий для суб'єктивних маніпуляцій. Знаходження N-грам та їх позиціонування повністю автоматизоване, але вибір розміру N-грам і довжини профілю є відкритими для суб'єктивних маніпуляцій.

Для визначення авторства текстів на штучних мовах найчастіше використовуються метричні методи, які в більшості випадків розробляються та застосовуються тільки для певної мови програмування.

Усі відомі метричні методи різняться тільки кількістю та змістом показників, які використовуються в них для формування метрики. Наприклад, у одного з перших методів атрибуції програмних продуктів, що використовує метрики Холстеда [5], таких показників лише 16. Іншим метричним методом є метод отримання «відбитків» програми. Ці «відбитки» розраховуються тільки для програм, написаних на Java [9]. При створенні цей метод використовував метрику з 56 показників, але пізніше, за результатами випробувань, їх кількість було скорочено до 48. Точність атрибуції методу «відбитків» досить невелика – вона становить лише близько 62 відсотків.

Окрім того, що метричні методи не забезпечують достатньої точності атрибуції, вони є залежними від мови програмування. Налаштувавши одного разу інформаційну систему на певну мову програмування, в подальшому її можна використовувати для розрахунку метрик тільки для цієї мови. Подолання перелічених недоліків та створення незалежної від мови програмування метрики й визначає мету даної роботи.

Розробка мультимовної метрики для атрибуції програмних продуктів

Основним підходом до створення мультимовної метрики є вибір та включення до неї таких показників, які є притаманними усім сучасним мовам програмування. Для аналізу було обрано декілька популярних мов програмування: Java, Java Script, C# та C++. Вони є одними з найбільш використовуваних програмістами мов. Початкові коди програмних продуктів є підставою для атрибуції.

Для створення метрики було обрано такі основні категорії показників: роздільники, знаки операцій, ключові слова, пропуски перед та після знаків операцій, власні імена ідентифікаторів, які вживає програміст.

Розглянемо більш детально кожну категорію.

1. Роздільники. Дана категорія містить 10 метрик, які показують частоту і особливості використання програмістом роздільників. У багатьох мовах програмування такі роздільники, як пробіл, знак табуляції або перехід на новий рядок не є обов'язковими. Однак саме використання роздільників формує «зовнішній вигляд» лістингу програми і відображає усталені і важко змінювані особливо-

сті почерку програміста. Ця група показників показує частоту використання програмістом окремих роздільників.

2. Знаки операцій. Категорія вміщує показники, що включають знаки операцій в найбільш часто використовуваних мовах програмування та їх можливі варіації, наприклад x = x + 1 або еквівалентні варіанти типу x++ та x += 1. Як і у випадку з роздільниками, розраховується частота використання тих чи інших знаків операцій.

3. Ключові слова. Дана категорія включає в себе найбільшу кількість показників за кількістю ключових слів у різних мовах програмування. Розраховується частота використання різних ключових слів програмістом. Для подолання залежності цієї групи показників використано заздалегідь підготовлені словники ключових слів кожної мови. Кількість словників та їх склад можуть змінюватись, що дозволяє робити легкий перехід до іншої мови програмування. За допомогою даної групи можна виявити службові слова, які найчастіше використовує програміст. Так, наприклад, існують взаємозамінні ключові слова і складені з їх допомогою оператори, використання яких не змінює логіку роботи програми (оператори if/else, наприклад, можуть замінюватися на switch/case замість оператору циклу for завжди може стояти while, та інше). Це відкриває можливості відслідковування особливостей використання окремими програмістами ключових слів і операторів.

4. Роздільники до і після знаків операцій. Дана категорія містить 4 показники, які характеризують вживання необов'язкового обрамлення знаків операцій пробілами. Розраховується частота використання роздільників до і після кожного знака операцій. Категорія показує використання розробником так званого «гарного» стилю програмування.

5. Ідентифікатори. У дану категорію входять метрики, що показують середню довжину використовуваних програмістом ідентифікаторів глобальних і локальних змінних, назв класів та методів. Враховуються також деякі стилістичні особливості використання програмістом слів для назв змінних, наприклад, чи вживає програміст російські/українські назви, написані англійськими літерами чи ні, чи використовує скорочення повних назв і так далі. До цієї категорію входять деякі структурні показники, наприклад, середня кількість глобальних змінних у класі.

Для того, щоб позбутися залежності показників від довжини програмного коду запропоновано використовувати відносне значення кожного показника. Це дає можливість зіставити відповідні величини для різних програмних продуктів.

На початку розрахунку метрик визначається абсолютне значення кожного з показників, а потім обчислюється їх відносне значення.

Для того, щоб визначити, до якого автора відноситься невідомий програмний проект, необхідно виявити значення характеристик, що притаманні даному автору. Сукупність цих характеристик утворює так званий «профіль автору».

Можна сказати, що профіль автору відповідає поточному стану його стилю програмування. Як показує досвід авторів, стиль програмування закладається вже в перші роки професійної діяльності програміста і згодом практично не змінюється.

Формування профілю автора складається з трьох етапів.

Етап 1. Збираються всі програмні проекти, про які достовірно відомо, що їх створив даний автор.

Етап 2. Для кожного з відібраних проектів розраховуються абсолютні та відносні значення показників метрики.

Етап 3. Розраховані відносні значення показників метрик всіх програмних проектів усереднюється. Отримані середні значення показників утворюють профіль автора.

Розробка інформаційної системи атрибуції програмних продуктів

На підставі описаної мультимовної метрики створено інформаційну систему для атрибуції програмних продуктів. Вона складається з таких підсистем (рис. 1): підсистеми створення та модифікації профілю авторів та власне підсистеми атрибуції. Вхідними даними для підсистеми створення та модифікації профілю авторів є програмні проекти, авторство яких достеменно відомо.

Рис. 1. Підсистеми інформаційної системи

Після успішної атрибуції профіль автора може бути доповнено даними, отриманими з підсистеми атрибуції На вхід підсистеми атрибуції поступає початковий код досліджуваної підсистеми. програмні проекти невідомих авторів. До підсистеми створення профілю авторів входять п'ять модулів: модуль вибору мови програмування, модуль обчислення абсолютних значень метрик, модуль обчислення відносних значень метрик, візуалізація профілю автора, запис інформації з профілю автора. Друга підсистема включає в себе чотири модулі – це модуль вибору мови програмування, модуль обчислення абсолютних значень метрик, модуль обчислення відносних значень метрик, запис інформації з програмного проекту.

Як видно, майже всі модулі у обох частинах схожі, адже атрибуція виконується на підставі порівняння профілю початкового коду досліджуваної програми з профілем програм відомих авторів. Тому усі ці модулі є спільними для обох підсистем.

В модулі прийняття рішень, який входить до складу підсистеми атрибуції, обчислюються значення відхилень профілю проекту від профілів авторів по кожному з показників окремо. На підставі статистичного аналізу цих відхилень обирається найбільш ймовірний автор. Основні етапи роботи інформаційної системи атрибуції представлено на рис. 2.

Результати верифікації

Для перевірки коректності роботи розробленої системи було обрано 154 програмних проектів написаних мовою *Java* та 51 проект на C++ тринадцяти відомих авторів. В контрольну групу для формування профілю авторів було відібрано від 35 до 50 відсотків проектів кожного автора. Решта проектів склала групу для виявлення авторства згідно з описаною вище методикою. Результати співставлення показали в середньому 85% точність атрибуції незалежно від мови програмування. Цей результат майже на 20 відсотків перевищує результати відомих методів атрибуції.

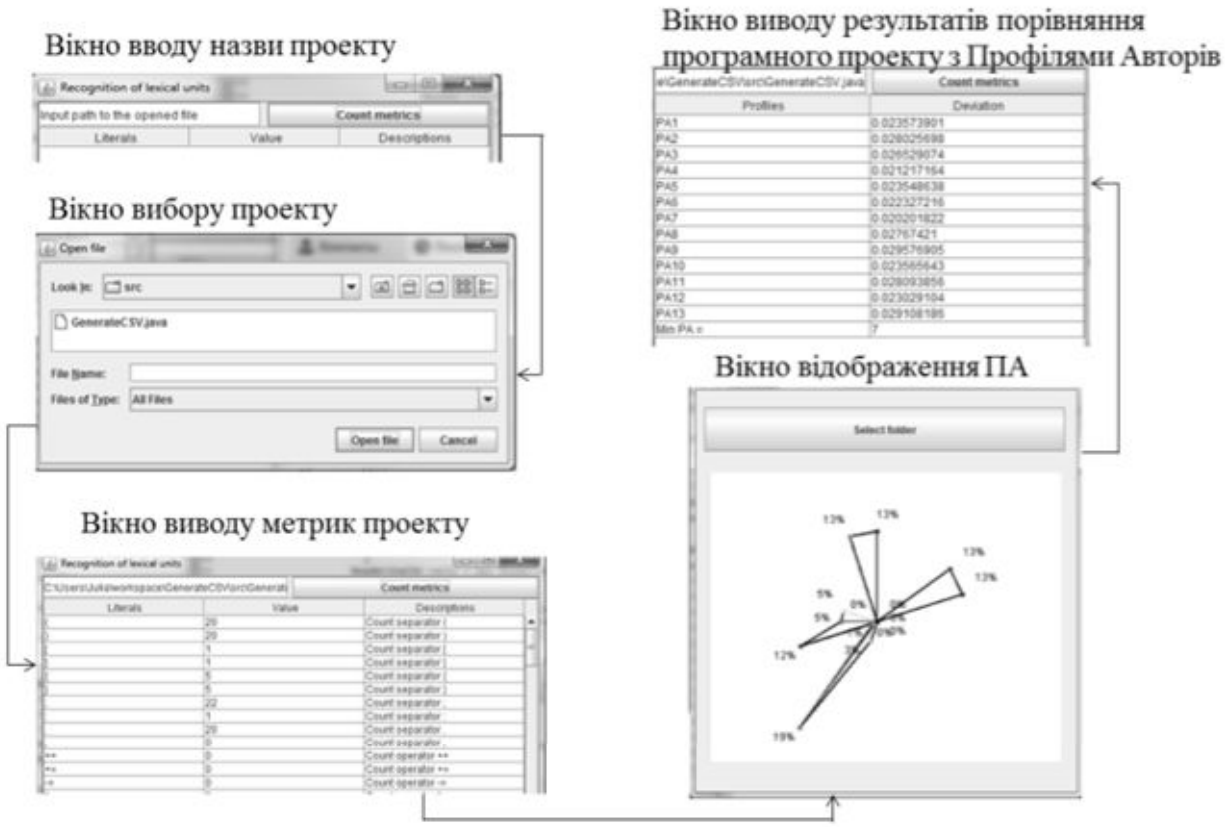

Рис. 2**.** Вікна інформаційої системи

Висновки

Запропонована в роботі мультимовна метрика дозволяє виконувати атрибуцію програмних продуктів на підставі їх початкового коду. За рахунок оптимального вибору набору значущих показників метрики точність атрибуції підвищено на 15 % у порівнянні з відомими результатами

Напрямками подальшої роботи повинні стати:

– наповнення бази даних системи профілями авторів на підставі розроблених ними програмних продуктів;

– розробка нових, більш точних методик статистичного аналізу розбіжностей при порівнянні профілів. Цей напрямок повинен ще більше підвищити точність атрибуції.

Список використаної літератури
(References)

1. Arwin C., and Tahaghoghi S.M.M., (2006), Plagiarism Detection across Programming Languages [Electronic Resource], *ACSC '06 The 29th Australasian Computer Science Conference*, Vol. 48, pp.277 – 286, url: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.1199 &rep=rep1&type=pdf.

2. Burrows S., Tahaghoghi S.M.M., and Zobel J., (2007), Efficient plagiarism detection for large code repositories [Electronic Resource], *SOFTWARE—PRACTICE AND EXPERIENCE*, Vol. 37, pp.151 – 175, url: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.7103&rep=rep1&type=pdf.

3. Frantzeskou G., Frantzeskou G., Gritzalis S., and Macdonell S.G., (2004), Source Code Authorship Analysis for supporting the Cybercrime Investigation Process, *International Conference on Telecommunications Networks,* Setúbal, pp. 85 – 92.

4. Frantzeskou G., MacDonell S.G, Stamatatos E., Georgiou S., and Gritzalis S., (2011), The Significance of User-defined Identifiers in Java Source code Authorship Identification, *International Journal of Computer Systems Science and Engineering,* Samos, Greece, pp.139 – 148.

5. Berghel H.L., and Sallach D.L., (1984), Measurements of Program Similarity in Identical

Task Environments [Electronic Resource], *ACM SIGPLAN Notices*, Vol. 19, No. 8, pp. 65 – 76, url:http://ulgtcsp.berghel.com/publications/measprogsim/measprogsim.pdf.

6. Whale G., (1990), Software Metrics and Plagiarism Detection [Electronic Resource], *Journal of Systems and Software*, Vol. 13, pp.131 – 138, url : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.5047&rep=rep1&type =pdf.

7. Krsul I., and Spafford E.H., (1995), Authorship Analysis: Identifying The Author of a Program [Electronic Resource], *Proceedings of the 18th National Information Systems Security Conference*, pp. 514 – 525, url: http://www.krsul. org/ivan/ articles/krsul-authorship_analysis_nissc.pdf.

8. Gray A., Sallis P., and MacDonell S.G., (1998), IDENTIFIED (Integrated Dictionary-based Extraction of Non-language-dependent Token Information for Forensic Identification, Examination, and Discrimination): A Dictionary-based System for Extracting Source Code Metrics for Software Forencis [Electronic Resource], *The 18th National Information Systems Security Conference*, pp. 252 – 259, url: http://www.academia.edu/2669584/ IDENTIFIED_Integrated_Dictio-nary.

9. Ding H., and Samadzadeh M.H., (2004), Extraction of Java Program Fingerprints for Software Authorship Identification [Electronic Resource], *The Journal of Systems and Software*, Vol. 72, No. 1, pp. 49 – 57, url: http://www.sciencedirect.com/science/article/pii/S0164121203000499.

Отримано 27.03.2015

Маєвський
Дмитро Андрійович,
д-р техн. наук, проф.,
зав. каф. Теоретичних основ та загальної електротехніки Одеського політехн. у-ту,
пр-т Шевченко, 1,
тел. (048) 734-84-54.
E-mail:
dmiry.a.maevsky@gmail.com

Чербаджи
Юлія Павлівна, магістр з інформаційних управляючих систем та технологій Одеського політехн. у-ту,
тел. (093) 048-35-60.
E-mail:
jcherbadzhy@gmail.com