УДК 004.43+004.9

О.О. Киричек[1], Г.Г. Киричек[2],
[1] Компанія «Інфопульс Україна», м. Київ
[2]Запоріжський національний технічний університет, м. Запоріжжя

ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ З ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ВИЯВЛЕННЯ ПЛАГІАТУ ПРОГРАМНОГО КОДУ C#

Анотація

Киричек О.О., Киричек Г.Г. Експериментальні дослідження з підвищення ефективності виявлення плагіату програмного коду C#. *Проведено експериментальні дослідження. Реалізація даного алгоритму дозволяє отримати високу точність оцінки наявності плагіату в програмному коді за рахунок застосування до проектів алгоритму фільтрації шаблонних ділянок коду та формально перетворених ділянок.*

Ключевые слова: *токенізація, формальне перетворення, вкладений токен, фільтрація шаблонного коду.*

Постановка задачи. Інтенсивний розвиток інформаційних технологій привів до того, що над розробкою та супроводом автоматизованих систем зараз працюють мільйони програмістів, які пишуть величезну кількість програмного коду. У зв'язку з цим важною проблемою стає пошук плагіату у програмному коді. Розробка застосувань на платформі Framework.Net виконується переважно мовою *C#*, яка найбільш оптимізована під платформу. Одною з переваг розробки з використанням сучасних засобів розробки є автоматична генерація базового каркасу застосувань, форм та спеціалізованих класів. Це знімає з програміста роботу по виконанню рутинних задач із формування однотипного програмного коду, який не відрізняється від аналогічного коду у інших проектах. Але з точки зору аналізу програмного коду на плагіат це вносить суттєву помилку в загальну оцінку.

В статті пропонується методом експерименту дослідити запропонований спосіб підвищення ефективності виявлення плагіату програмного коду *C#* за рахунок виключення автогенерованого програмного коду. Даний підхід розвиває алгоритм виявлення плагіату [1], що інтегровано до розробленої системи виявлення плагіату [2].

Пошук плагіату у програмних застосувань краще виконувати на основі програмного коду. Переваги в тому, що він містить більше специфічних властивостей та характеристик притаманних автору. Вміст *.exe, .dll* та інших бінарних кодів теж можна аналізувати, але вони містять вже набагато менше специфіки і при виконанні різних компіляторів бінарних кодів на основі вихідних кодів або різних налаштувань оптимізації на виході будуть різні за

вмістом бінарні файли. Слід зазначити, що при експертних юридичних оцінках теж використовується програмний код.

Алгоритми виявлення плагіату у програмних кодах прийнято поділяти на атрибутно-статистичний, структурний, комбінований [3, 4]. Атрибутні методи основані на чисельному виразі деяких при знаків програми та порівняння отриманих чисел. Програми з близькими чисельними характеристиками потенціально схожі [5]. Недоліком такого підходу є те, що різні програми можуть отримувати близькі характеристики [4]. Структурні методи враховують контекст програм. Такі підходи використовують токенізацію програмного коду для більш компактного його представлення [4,6].

Видалення з файлів шаблонних ділянок коду, досягається, виключивши токенізоване подання фрагментів коду, що відповідають шаблонам з використанням яких створюються проекти [3,7].

Задача токенізованого представлення коду – збереження суттєвих та ігнорування легко модифікованих деталей. Токенізація виконується наступним чином:

– кожному оператору мови приписується код. Коди токенів сформовані для мови *C#*;

– з отриманих кодів будується рядок, зберігаючи послідовність токенів у відповідності до програмного коду. Один токен – один оператор;

– тимчасово при токенізації змінними призначаємо токени з діапазону.

Видалення з файлів автосгенерованих ділянок коду, досягається, виключивши токенізоване подання фрагментів коду, що відповідають шаблонам з використанням яких створюються проекти [1,8].

Для поліпшення пошуку схожих токенів можна використовувати базу предтокенізованих уявлень схожих алгоритмів. Таке рішення дозволить скоротити час на пошук міток і оцінки плагіату. Використання відкритих суспільних систем для наповнення подібної бази дозволить за короткий проміжок часу створити еталонні токенізовані подання відомих алгоритмів рішення тривіальних завдань. Використовуючи настільки глибокий аналіз коду викладачеві стане простіше оцінити творчу складову рішення студентом завдання, або оцінити тривіальність запропонованого рішення.

Експериментальна частина. В експерименті використовуємо 18 проектів на мові C# платформи .NET Framework компанії Microsoft. Усі проекти однотипні, створені з використанням патерну MVVM.

Об’єми проектів залежать від виду роботи та курсу, на якому вона виконується. Приблизна оцінка об’ємів на основі виконаних робіт з програмування, які виконуються на кафедрі АУТС: 100 – 800 рядків коду – об’єм лабораторної роботи в залежності від курсу; близько 1600 – розрахункова робота; 3000 – об’єм курсової роботи; 4000 – 5000 робота освітньо-кваліфікаційного рівня бакалавр. Визначимо залежність кількості

шаблонного коду від об’єму програми. Для отримання результатів проаналізуємо по три проекти на кожний об’єм програми.

Результати дослідження залежності кількості шаблонного та автогенерованого коду від об'єму програми наведено на рис. 1.

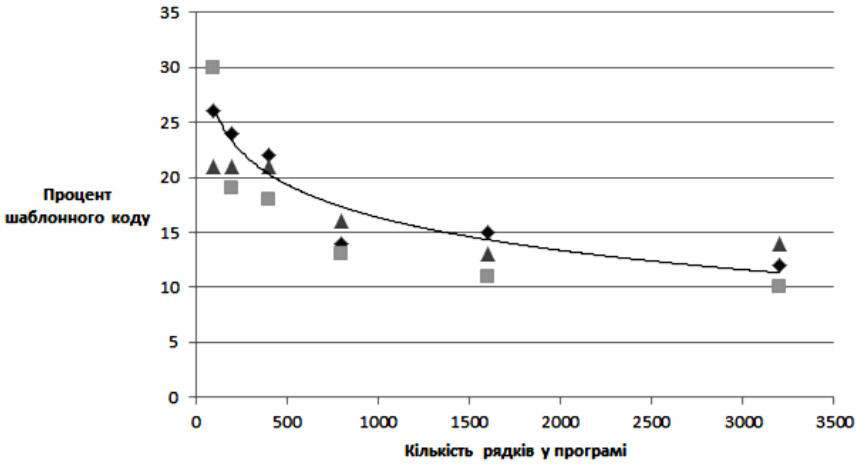

Рисунок 1 - Залежність проценту шаблонного коду від об’єму програми

Внесемо зміни в програмний код для отримання унікальності ~80% для шести різних по розміру програм, інші 20% будуть автогенерованими та шаблонними ділянками коду. У табл. 1 представлено результат порівняння проектів різними алгоритмами пошуку плагіату.

Таблиця 1 - Ефективність відсіювання шаблонних ділянок коду

	100	200	400	800	1600	3200
Без використання відсіювання	25,5	22,55	20,8	20,31	20,1	20,05
З відсіюванням	12	8,35	6,3	5,61	5,15	4,95
Різниця між двома алгоритмами	13,5	14,2	14,5	14,7	14,95	15,1

Результати застосування алгоритму без відсіювання наведено на рис. 2 та з відсіюванням шаблонного коду наведено на рис. 3.

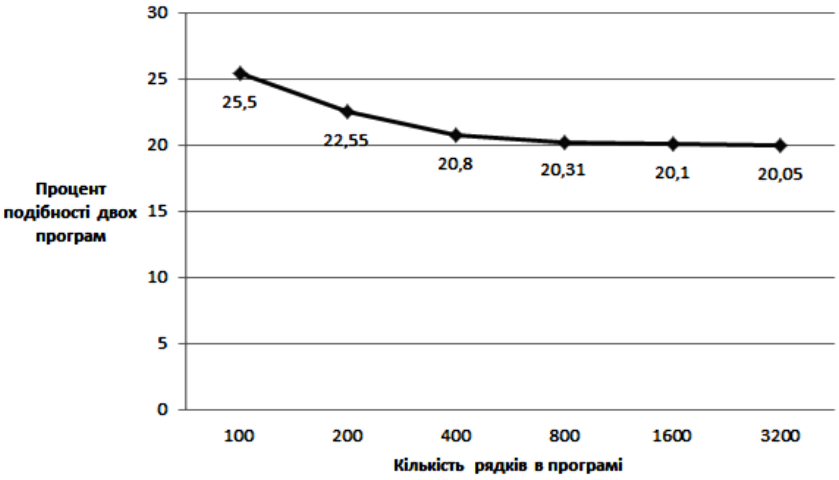

Рисунок 2 - Процент подібності програмного коду без відсіювання

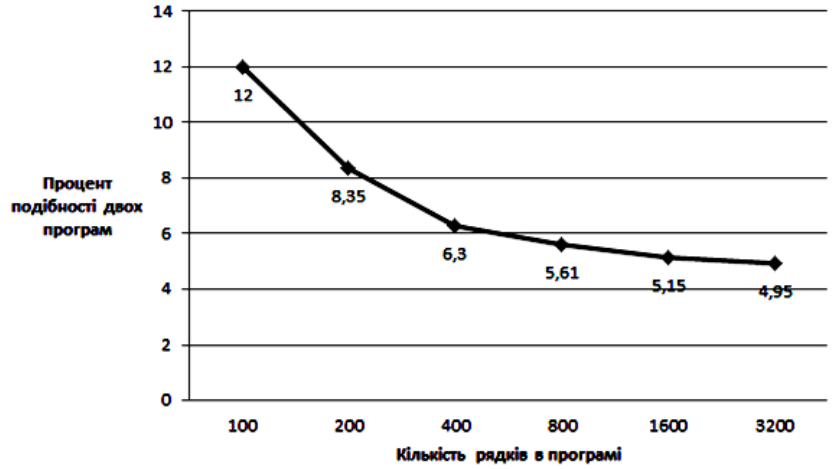

Рисунок 3 - Процент подібності програмного коду з використанням відсіювання

Різниця між значеннями проценту подібності програмного коду для двох алгоритмів (рис.4) залежить від похибки виконаня алгоритму, і приблизно дорівнює одному проценту.

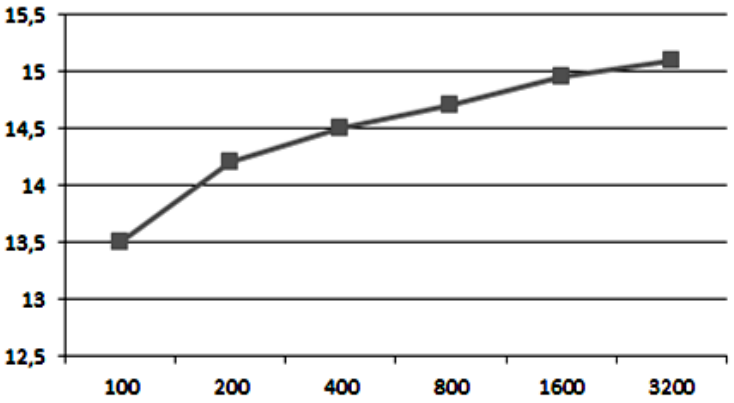

Рисунок 4 - Різниця між показами подібності програмного коду для двох алгоритмів

Проведемо експеримент швидкості виконання аналізу для тих самих програм. Один з проектів залишаємо без змін, в другому моделюємо деякі спроби приховати плагіат:

– заміна назв змінних, властивостей, методів, класів;

– зміна послідовності чергування методів (функцій), оголошення змінних та властивостей;

– зміна типів оголошення змінних.

Швидкість виконання алгоритмів наведено в табл. 2.

Таблиця 2 - Швидкість роботи алгоритмів, мс

	100	200	400	800	1600	3200
Без використання відсіювання	40	71	120	228	452	948
З відсіюванням	86	149	267	429	710	1434
Час на фільтрацію	35	71	152	298	446	737

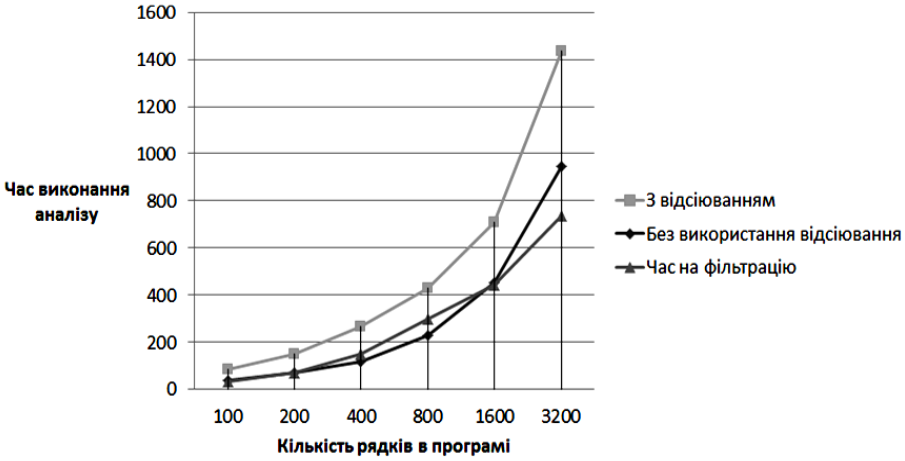

Рисунок 5 - Час виконання аналізу

Аналізуючи результати дослідження (рис.5) можна сказати, що алгоритм з високою якістю проводить відсіювання шаблонних ділянок коду і цим самим суттєво підвищує точність визначення подібності программи.

Натомість майже в два рази збільшився час аналізу програми. Проте наведене порівняння є аналізом один до одного. При великій кількості проектів швидкість обробки збільшиться, бо зменшиться хешоване представлення програми до якої застосовується аналіз. Час на фільтрацію затрачується одноразово.

Висновки.

В результаті проведеного експерименту підтверджено ефективність способу виявлення плагіату програмного коду *C#* на основі шаблонів. Отриманий алгоритм забезпечує виключення з проекту незмінних у різних проектах фрагментів коду, які є автозгенерованими, в результаті чого збільшена точність визначення наявності плагіату в програмному коді.

Подальшим розвитком є формалізація приведень структур циклів, що містять анонімні делегати, а також пошук інших шаблонів програмного коду, який може бути представлений у різному вигляді. Зараз виконуються роботи з пошуку методів збільшення швидкості порівняння проекту з базою проектів.

Література

1. Киричек А.А. Алгоритм фильтрации для системы определения плагиата в программном коде [Текст] / А.А. Киричек, А.А. Амонс, Г.Г. Киричек // Вестник Национального технического университета «ХПИ». Серия: Новые решения в современных технологиях. – Харьков: ХНТУ «ХПИ», 2013. – С.78-82.

2. Амонс О.А. Виявлення плагіату в програмному коді C# [Текст] / О.А.Амонс, С.Ю.Зайцев, О.О.Киричек // Вісник НТУУ «КПІ», 2011 р. – С.170-179.

3. Irving,R.W. Plagiarism and collusion detection using the Smith Waterman algorithm // DCS Technical Report, Dept of Computing Science, University of Glasgow 2004. – pp 1-24.

4. Stepanova E.B. Krivtsov V.E. Process modeling in Educational IT-Projects / E.B. Stepanova, V.E Krivtsov // CSIT'2008: Proceedings of the 10-rd International Workshop on Computer Science and Information Technologies (Antalya, Turkey, September 15-17, 2008). – Ufa: Ufa State Aviation Technical University, 2008. – v. 1. – pp. 227-230.

5. Обзор автоматических детекторов плагиата в программах [Электронный ресурс].- Режим доступа: http://logic.pdmi.ras.ru/~yura/

6. Baker B.S. On Finding Duplication and Near-Duplication in Large Software Systems / B.S. Baker // In Proceedings of the second IEEE Working Conference on Reverse Engineering (WCRE), July 1995. – P. 86–95.

7. Prechelt L. JPlag: Finding plagiarisms among a set of programs / L.Prechelt, G. Malpohl, M. Philippsen // Technical Report No. 1/00, University of Karlsruhe, Department of Informatics, March 2000.

8. Киричек А.А. Фильтрация шаблонов программного кода в студенческих проектах [Текст] / А.А. Киричек, А.А. Амонс, Г.Г. Киричек // Інформаційні управляючі системи та комп'ютерний моніторинг (ІУС КМ - 2013) : IV Всеукраїнська науково-технічна конференція студентів, аспірантів та молодих вчених, 24-25 квітня 2013 р., м.Донецьк : зб. доп. / Донец. національ. техн. ун-т; редкол. В.А. Світлична. – Донецьк: ДонНТУ, 2013. – В 2 т. - Т.1. – С.36-42.