ПІДХОДИ ДО ОБРОБЛЕННЯ «СИНТАКСИЧНОГО ЦУКРУ» ПРИ ПОШУКУ ПЛАГІАТУ В ПРОГРАМНОМУ КОДІ

УДК 004.91

ШЕВЧУК Михайло Михайлович
магістрант кафедри ПЗКС ФПМ НТУУ «КПІ ім.І.Сікорського»
e-mail: myte@ukr.net.

ЮСИН Яків Олексійович
магістрант кафедри ПЗКС ФПМ НТУУ «КПІ ім.І.Сікорського»
e-mail: yusin.yakiv@gmail.com.

ЗАБОЛОТНЯ Тетяна Миколаївна
к.т.н., доцент кафедри ПЗКС ФПМ НТУУ «КПІ ім.І.Сікорського»
e-mail: tetiana.zabolotnia@gmail.com.

РИБАЧОК Наталія Антонівна
к.т.н, ст.викладач кафедри ПЗКС ФПМ НТУУ «КПІ ім.І.Сікорського»
e-mail: rybachoknata@gmail.com.

ДИЧКА Андрій Іванович
магістрант кафедри ПЗКС ФПМ НТУУ «КПІ ім.І.Сікорського»
e-mail: andriydychka@gmail.com.

ВСТУП

Задача пошуку плагіату в програмному коді стає все більш актуальною для різних галузей людської діяльності. В першу чергу, це стосується галузі комерційного розроблення програмного забезпечення і освітньої галузі. Наприклад, одним з найбільш відомих прецедентів щодо копіювання програмного коду в комерційній галузі є судовий процес компанії Oracle проти Google [1]. В ході цього процесу Google була визнана винною в копіюванні 9 рядків коду мовою Java (функції rangeCheck, яка надалі отримала поширену назву «найбільш відомі 9 рядків коду» [2]).

Через поширення на ринку ПЗ явища так званих «open source» проектів підтримка пошуку плагіату в програмному коді стала вкрай необхідною для ефективного вирішення питань щодо захисту інтелектуальної власності та ліцензування. В мережі Інтернет явище розроблення через копіювання коду навіть отримало назву «Stack Overflow Driven Development» [3] (від назви сайту stackoverflow.com – найбільш популярної вебсистеми питань та відповідей про програмування).

Існуючі методи пошуку плагіату в програмному коді або не враховують можливість наявності в ньому «синтаксичного цукру», або є досить обчислювально складними і потребують великого обсягу додаткової пам'яті для роботи.

Таким чином, можна зробити висновок, що розроблення нових підходів, методів та алгоритмів пошуку плагіату у вихідному коді програмного забезпечення є актуальною задачею.

ПОСТАНОВКА ЗАДАЧІ

Задача пошуку плагіату в програмному коді має свої особливості, які роблять неможливим використан-

ня для її вирішення класичних методів пошуку плагіату в текстових даних. Це пов'язано з тим, що в коді цілком можливі перестановка блоків операторів, перейменування змінних та функцій, додавання коментарів тощо, і ці дії не впливають на коректність програми та можуть слугувати для маскування факту плагіату.

Зазвичай виділяють такі чотири типи схожості фрагментів коду [4], коли:

- два фрагменти коду повністю ідентичні, а змінюватись можуть лише коментарі, пробільні символи та відступи;
- два фрагменти коду ідентичні за структурою та синтаксисом за винятком імен ідентифікаторів, типів, а також відмінностей, що потрапляють до першого типу;
- один фрагмент коду отримано шляхом копіювання другого та додаванням та/або видаленням з нього операторів мови програмування;
- два фрагменти коду повністю синтаксично відрізняються один від одного, але при цьому виконують однакові дії.

Недоліком даної класифікації є те, що вона не розглядає наявність в коді так званого «синтаксичного цукру», що на сьогодні присутній майже в будь-якій мові програмування високого рівня.

«Синтаксичний цукор» - це загальноприйнятий термін для позначення доповнень синтаксису мови програмування, які не додають їй нових функціональних можливостей, а є більш зручними замінами інших конструкцій, вже наявних в мові [5]. «Синтаксичний цукор» може бути використаний для маскування плагіату програмного коду без особливих зусиль (наприклад, популярні інструменти для розроблення програм автоматично пропонують заміни конструкцій мови на «синтаксичний цукор»), проте відповідно до розглянутої класифікації такі фрагменти коду потрапляють тільки до четвертого типу – найменш схожого на плагіат.

Таким чином, **метою** даної роботи стало підвищення ефективності пошуку плагіату в програмному коді за критерієм точності завдяки врахуванню можливої наявності в коді «синтаксичного цукру».

Відповідно до вказаної мети в роботі поставлені і розв'язані такі **задачі**:

- вивчення існуючих методів пошуку плагіату в програмному коді;
- розроблення підходів до оброблення «синтаксичного цукру» при пошуку плагіату в програмному коді;
- формування переліку «синтаксичного цукру» для використання при реалізації запропонованих підходів при пошуку плагіату в програмному коді на мові C#.

«СИНТАКСИЧНИЙ ЦУКОР» ТА ІСНУЮЧІ МЕТОДИ ПОШУКУ ПЛАГІАТУ В ПРОГРАМНОМУ КОДІ

Існуючі методи пошуку плагіату в програмному коді прийнято поділяти на три великі групи [6]:

- текстові методи;
- структурні методи;
- семантичні методи.

Даний перелік не включає в себе методи, які розглядають програмний код в первинному вигляді, без будь-яких його додаткових перетворень. Це пов'язано з тим, що такі методи працюють з програмним кодом як зі звичайним текстом, через це забезпечуючи занадто низьку точність оброблення вхідних даних. У зв'язку з цим автори статті вважають їх розгляд недоцільним.

Текстові методи пошуку плагіату в програмному коді базуються на розгляданні коду як послідовності токенів – символів, що представляють собою оператор або групу операторів мови програмування [7]. При цьому параметри цих операторів повністю ігноруються.

Токенізація (процес перетворення програмного коду на послідовність токенів) методами цієї групи виконується наступним чином:

1.Кожному оператору мови програмування, який не є операндом, приписується цифровий код, що раніше був обраний для відповідного класу операторів. Також коди можна приписувати блоковим операторам (в мові програмування C#, наприклад, це фігурні дужки) та підключенням бібліотек (для C# – просторів імен).

2.Будується рядок з отриманих кодів, зберігаючи порядок їх слідування відповідно до порядку в програмному коді.

На основі результатів аналізу цієї процедури можна зробити висновок, що дані методи пошуку плагіату дозволяють ефективно працювати з фрагментами коду, що належать до перших двох типів розглянутої класифікації. Разом з тим слід зазначити, що текстові методи були першими методами пошуку плагіату в програм-

ному коді і на сьогодні до них належать найбільш сучасні та ефективні методи [7].

Методи, що базуються на дослідженні структури програми (яка часто представляється ними у вигляді графу потоку керування або абстрактного синтаксичного дерева), відповідно отримали назву *структурних методів* [8]. Методи даної групи також є ефективними для оброблення фрагментів коду перших двох типів розглянутої класифікації і непогано працюють з кодом, що відноситься до третього типу. Головним недоліком методів пошуку плагіату в програмному коді цієї групи є їхня обчислювальна складність і потреба в досить великій кількості додаткової пам'яті для зберігання відповідного подання програмного коду (графу потоку керування або абстрактного синтаксичного дерева).

Семантичні методи є подібними до структурних методів, проте в їх основі лежить знання семантики операторів [9]. Наприклад, методами цієї групи може використовуватись подання програмного коду у вигляді особливого семантичного графу, що має вершини двох типів. Вершини одного типу будуються на основі послідовностей операторів, що характеризуються певною семантикою (наприклад, це можуть бути арифметичні операції, умовні оператори, оператори циклу тощо). Вершини іншого типу задають відношення, в якому перебувають сусідні з ними вершини – таким чином з'єднуючи пари вершин першого типу.

Як вже було зазначено в попередньому розділі, під «синтаксичним цукром» розуміється будь-який елемент синтаксису мови програмування, що дублює існуючий аналог, але є більш зручним у використанні, або більш коротким при записуванні, або виглядає більш природнім чи більш зрозумілим при читанні програмного коду. Таким чином, «синтаксичний цукор» призначений лише для того, щоб зробити мову програмування більш зручною для програміста.

За своїм визначенням найбільш стійкими до заміни операторів на «синтаксичний цукор» для маскування плагіату є структурні та семантичні методи. Проте ці методи, як зазначено, є обчислювально складними і часто потребують залучення великої кількості додаткової пам'яті. Текстові методи є більш привабливими з точки зору обчислювальної складності та необхідної пам'яті, проте є зовсім нестійкими до маскування плагіату за допомогою «синтаксичного цукру».

На думку авторів, вирішенням проблеми неможливості виявлення плагіату в програмному коді, що містить «синтаксичний цукор», текстовими методами може стати розроблення та практичне впровадження нових підходів до попереднього аналізу програмного коду на наявність в ньому «синтаксичного цукру» та подальшої нейтралізації впливу останнього на результати процедури пошуку плагіату.

ПІДХОДИ ДО ОБРОБЛЕННЯ «СИНТАКСИЧНОГО ЦУКРУ» ПРИ ПОШУКУ ПЛАГІАТУ В ПРОГРАМНОМУ КОДІ

Принциповими ознаками «синтаксичного цукру» є те, що [5]:

1) все, що може бути написаним за допомогою використання «синтаксичного цукру», може бути написаним без нього цією ж мовою програмування;

2) заміна частини програмного коду на «синтаксичний цукор» не змінює хід виконання програми.

Також слід відмітити, що конструкції «синтаксичного цукру» є попередньо відомими і вони чітко визначаються у коді.

На цих ознаках базуються два наступні запропоновані підходи до оброблення «синтаксичного цукру». Обидва підходи передбачають певні дії на етапі попереднього оброблення програмного коду перед виконанням будь-якого методу пошуку плагіату у програмному коді і розглядають останній як звичайний текст.

Reduce. Основна ідея цього підходу полягає в заміні базових конструкцій мови програмування, що мають еквіваленти у вигляді «синтаксичного цукру», на «синтаксичний цукор».

Map. Основна ідея цього підходу є протилежною до *Reduce*-підходу, і полягає в заміні операторів «синтаксичного цукру» на їх еквіваленти, що використовують базові конструкції мови програмування.

Свої назви дані підходи отримали відповідно до результату їх застосування:

- у зв'язку з тим, що в більшості випадків код з використанням «синтаксичного цукру» є більш коротким, підхід, що передбачає заміну базових конструкцій мови програмування на «синтаксичний цукор», начебто «стискає» програмний код – звідси походить назва ***Reduce***;

- протилежний підхід в більшості випадків збільшує обсяг програмного коду, «відображаючи» «синтаксичний цукор» на базові синтаксичні конструкції мови програмування – звідси походить назва ***Map***.

Заміна базових конструкцій мови програмування на «синтаксичний цукор» та навпаки може бути реалізована по-різному. Наприклад, можливе використання регулярних виразів для пошуку та заміни конструкцій. Також очевидним є те, що дані підходи потребують попередньо створеного списку «синтаксичного цукру» конкретної мови програмування та його еквівалентів, записаних базовими конструкціями цієї ж мови.

Результатами реалізації даних підходів є те, що всі оператори програми приводяться до одного виду – або з використанням «синтаксичного цукру», або без. Отже, у випадку, якщо плагіат програмного коду було замасковано за допомогою «синтаксичного цукру», після оброблення програмного коду буде отримано його початкову форму і плагіат буде знайдено при подальшому виконанні будь-якого методу пошуку плагіату.

Загальний алгоритм використання запропонованих підходів оброблення «синтаксичного цукру» при пошуку плагіату програмного коду буде мати такий вигляд:

1.Читання сирцевого коду, який потрібно перевірити на наявність в ньому плагіату.

2.Читання коду, який є потенційним джерелом плагіату.

3.Попереднє перетворення обох кодів відповідно до одного з запропонованих підходів (Map або Reduce) до оброблення «синтаксичного цукру» в коді програмного забезпечення.

4.Виконання основного класичного методу пошуку плагіату в програмному коді.

ПЕРЕЛІК КОНСТРУКЦІЙ «СИНТАКСИЧНОГО ЦУКРУ» МОВИ C#

Поняття «синтаксичного цукру» в деякій мірі є умовним – різні конструкції можуть бути як віднесені до «синтаксичного цукру» мови програмування, так і не бути віднесеними до нього. Наприклад, більшість сучасних мов програмування як високого, так і низького рівня, мають декілька конструкцій для запису циклів (з умовою, без умови, з лічильником, з післяумовою) або декілька умовних конструкцій. Віднесення однієї з них до базових конструкцій, а інших варіантів до «синтаксичного цукру» є помилковим і часто не доцільним, особливо при пошуку плагіату. До таких же конструкцій можна віднести різноманіття арифметичних операторів: наприклад, оператор += та подібні до нього оператори формально підпадають під визначення «синтаксичного цукру», проте вони є досить низькорівневими. В той час, як програмний код стає все більш складним та високорівневим, доцільним є віднесення до «синтаксичного цукру» саме конструкцій високого рівня абстракції для пошуку плагіату. Розгляд «синтаксичного цукру» лише високого рівня також дозволяє зменшити кількість хибно-позитивних помилок, таким чином підвищуючи якість отримуваних результатів за різними оцінками (наприклад, за такими, як точність та F-міра).

До «синтаксичного цукру» мови програмування C# версії 7.0 та вище [10], за допомогою якого можливе маскування плагіату програмного коду, автори пропонують відносити:

1) оператор null-об'єднання;
2) неявну типізацію локальних змінних;
3) ініціалізатори колекцій;
4) ініціалізатори об'єктів;
5) лямбда-вирази;
6) методи розширення;
7) автоматичні властивості;
8) іменовані параметри;
9) null-умовні оператори;
10) лямбда-визначення функцій;
11) інтерполяція рядків;
12) out змінні;
13) опціональні параметри.

Перелік конструкцій «синтаксичного цукру» мови C# та відповідні йому базові конструкції наведено в табл. 1.

До окремого переліку слід включити «цукор», що відноситься до LINQ (Language Integrated Query). Цей механізм платформи .NET додає до мов програмування, що її підтримують (в тому числі і до мови програмування C#), синтаксис мови запитів (подібну до SQL), яка дозволяє оперувати даними з різних джерел – бази даних, пам'яті, XML і т.д.

LINQ в мові програмування C# можна використовувати двома шляхами:

- за допомогою операторів мови;
- за допомогою методів розширення.

Таблиця 1.

«Синтаксичний цукор» мови C#

№	«Синтаксичний цукор»	Еквівалент
1	return obj1 **??** obj2;	return obj1 != null ? obj1 : obj2;
2	**var** obj = 1	int obj = 1
3	string[] arr = **{"A", "B"}**	string[] arr = new string[2]; arr[0] = "A"; arr[1] = "B";
4	Customer c = new Customer **{** **Name = "James",** **Age = 30** **};**	Customer c = new Customer(); c.Name = "James"; c.Age=30;
5	obj.Act(**x => x**)	obj.Act(delegate(obj x) { return x; })
6	x.**Do()**	StaticClass.Do(x)

7	public string A **{ get; set; }**	private string a;
		public string A
		{
		get { return a; }
		set { a = value; }
		}

8	Method(**a:** 10, **b:** 25)	Method(10, 25)
9	var obj1 = a**?.**Do(); var obj2 = arr**?[**0];	var obj1 = a != null ? a.Do() : null; var obj2 = arr != null ? arr[0] : null;

10	public object Do() **=> new object();**	public object Do()
		{
		return new object();
		}

11	var s = **$**"A**{obj.ToString()}**"	var s = string.Format("A{0}", obj)
12	int.TryParse("1", **out int** value);	int value; int.TryParse("1", out value);
13	public void Method(int a **= 5**); Method();	public void Method(int a); Method(5);

Методи розширення виступають в ролі «синтаксичного цукру» для операторів LINQ, таким чином вони також відносяться до області застосування запропонованих підходів до пошуку плагіату у вихідному програмному коді. Причиною їх виділення в окремий перелік (див. табл. 2) авторами цієї роботи є те, що перетворення методів розширення LINQ на оператори та навпаки потребує розроблення та застосування більш складних та комплексних правил для того, щоб зберегти правильний порядок оброблення даних та викликів.

Таблиця 2.

Оператори та методи розширення LINQ

№	Метод розширення	Відповідний оператор
1	GroupBy	group … by
2	Join	join … in … on … equals …
3	OrderBy	orderby …
4	OrderByDescending	orderby … descending
5	Select	select
6	SelectMany	декілька операторів from
7	ThenBy	orderby …, …
8	ThenByDescending	orderby …, … descending
9	Where	where

Перспективним напрямом подальшої роботи над цією задачею автори вважають актуалізацію переліку конструкцій «синтаксичного цукру» у відповідності до нових версій мови C# та змін, що вони несуть, а також розгляд можливості застосування запропонованих підходів до оброблення «синтаксичного цукру» при пошуку плагіату в програмному коді, написаному іншими мовами програмування.

ВИСНОВКИ

Таким чином, в рамках даного дослідження показана необхідність оброблення «синтаксичного цукру» при пошуку плагіату в програмному коді і запропоновано два підходи до його визначення на етапі попереднього оброблення коду. Запропоновані підходи орієнтовані на текстові методи пошуку плагіату та не потребують структурного подання програми або її скомпільованого вигляду в машинних кодах. Реалізація цих підходів при пошуку плагіату у вихідному програмному коді дозволить підвищити точність автоматизованого визначення наявності запозичень в коді.

Також наведено перелік «синтаксичного цукру» мови програмування C# та його еквіваленти в базових конструкціях, що є необхідним для практичного застосування запропонованих підходів.

REFERENCES

1. Google vs. Oracle: sud dlinoj v sem' let [Electronic Resource]. – Access mode: https://ain.ua/special/google-vs-oracle/. – Title from the screen. – (application date: 23.03.2018).
2. Google's 9 lines [Electronic Resource]. – Access mode: https://majadhondt.wordpress.com/2012/05/16/googles-9-lines/. – Title from the screen. – (application date: 23.03.2018).
3. Stack Overflow Driven Development [Electronic Resource]. – Access mode: https://dzone.com/articles/stack-overflow-driven-development-sodd-its-really. – Title from the screen. – (application date: 23.03.2018).

4. Roy C. K. and Cordy J. R.. A survey on software clone detection research, Tech.Rep. 2007-541, School of Computing, Queen's University, Kingston, Ontario,Canada, 2007., pages 43-59.
5. Syntaksychnyi tsukor [Electronic Resource]. – Access mode: https:// goo.gl/uaP469/. – Title from the screen. – (application date: 23.03.2018).
6. Prechelt L., Malpohl G., Philippsen M. JPlag: Finding plagiarisms among a set of programs. // Technical Report No. 1/00, Universityof Karlsruhe, Department of Informatics. March 2000.
7. Huang X., Hardison R.C., Miller W.A space-efficient algorithm for local similarities. // Computer Applications in the Biosciences 6. 1990. P. 373–381.
8. Baxter I., Yahin A., Moura L., Anna M.S., Bier L. Clone Detection Using Abstract Syntax Trees. // Proceedings of ICSM. IEEE. 1998.
9. Moussiades L.M., Vakali A.P Detect: A Clustering Approach for Detecting Plagiarism in Source Code Datasets. // The Computer Journal Advance Access. June 24, 2005.
10. C# Reference [Electronic Resource]. – Access mode: https://docs.microsoft.com/en-gb/dotnet/csharp/language-reference/. – Title from the screen. – (application date: 23.03.2018).

Рецензент: д.т.н, проф. І.А. Дичка
НТУУ «КПІ ім. І.Сікорського»