ВИКОРИСТАННЯ ПІДХОДІВ РЕФАКТОРИНГУ ПРОГРАМНИХ ПРОДУКТІВ ДЛЯ ВИЗНАЧЕННЯ ПОДІБНОСТІ ПРОГРАМНОГО КОДУ В АКАДЕМІЧНИХ РОБОТАХ

PhD, consultant at GlobalLogic Ukraine, Д.М. Главчев, м. Харків; PhD, директор НТБ, Ю.М. Главчева, Національний технічний університет "Харківський політехнічний інститут", м. Харьков

Обґрунтовано необхідність розробки та удосконалення підходів аналізу програмних кодів з метою визначення подібностей та підтвердження унікальності. Такі підходи можуть буди використані при вирішенні наступних завдань: оптимізації програмного коду для забезпечення якісних характеристик роботи програмних продуктів; комп'ютерної криміналістичної експертизи програмних продуктів; визначенні авторського стилю для ідентифікації авторів вихідного коду шкідливого програмного забезпечення; тощо.

Розглянуто актуальні проблеми, пов’язані з аналізом програмних кодів (ПК) в освітньо-науковому середовищі та корпоративному сегменті.

Значна увага університетів приділяється вживанню заходів щодо забезпечення академічної доброчесності та запобіганню академічному плагіату (АП). Відповідно до пояснень Міністерства освіти і науки України: «академічний плагіат не зводиться до текстових збігів, а може стосуватися також некоректного запозичення фактів, гіпотез, числових даних, методик, ілюстрацій, формул, моделей, програмних кодів тощо» [1]. Антиплагіатні системи, які використовуються на практиці в академічних установах орієнтовані на визначення ознак академічного плагіату у текстових даних. Викладачі з ІТ дисциплін доступними засобами та технологіями аналізують програми, написані студентами на наявність ознак АП у ПК. Зазвичай, існує категорія студентів, які намагаються здійснити маніпуляції та приховати запозичений фрагмент програмного коду.

Для виявлення ознак академічного плагіату в програмних кодах використовують спеціалізовані програми, в яких реалізовано спеціальні підходи до аналізу кодів. Підходи, які застосовуються для визначення подібностей у текстах не можуть бути ефективними при застосуванні для програмних кодів з причини кардинально різних маніпуляцій, які можуть бути проведені автором з текстовими даними та з програмним кодом. Таким чином, розвиток існуючих та створення нових підходів до виявлення ознак академічного плагіату в програмних кодах є актуальним завданням

Щодо корпоративного сегменту, розглянуто програмне забезпечення (ПЗ), що створюється для корпоративних замовників та відрізняється складністю та масштабністю. До нього висуваються вимоги з стабільності, безпеки, швидкодії, та можливостей розширення функціоналу програмного продукту [2]. Саме корпоративний сегмент потерпає від проявів

недоброчесного використання програмних кодів, витоків інформації, тощо [3]. Для уникнення фінансових втрат та падіння репутації, вживаються певні заходи для перевірки та підтвердження унікальності програмного коду, що дозволяє зберегти права власника на відповідний компонент його програмного продукту та оптимізувати код задля підвищення якості роботи продукту [4].

Проаналізовано підхід, який використовується для забезпечення підвищення стабільності програмних продуктів. Рефакторинг – це переробка та вдосконалення коду програми без зміни її зовнішнього функціоналу з метою покращення читаності коду та полегшення процесу внесення подальших змін в програмний код [5]. До теми рефакторингу можна віднести підхід до розробки ПЗ, DRY (Don’t repeat yourself), основний зміст якого полягає в тому, що нераціонально повторювати однакові фрагменти коду в проекті. За цим підходом кожна частина системи повинна мати унікальне представлення та не повторюватися [6, 7]. Більшість середовищ розробки мають відповідні інструменти для виконання автоматизованого рефакторингу. Інструменти використовують прийоми, що дозволяють розбити код на дрібніші частини, дозволяють забезпечити додаткову абстракцію, тощо. Зокрема, прийом "Відокремлення методу" (Extract Method) виконує переміщення відповідного фрагменту коду в окремий метод, та виконання пошуку подібних до нього фрагментів коду. Надається пропозиція замінити повторювані фрагменти на виклик того самого окремого методу. На основі виявлення подібних фрагментів проводяться зміни у програмному коді, що забезпечує оптимізацію та зменшення об’ємів коду, робить його відповідним до принципів DRY [7, 8].

На практиці в освітньому середовищі вже реалізуються певні підходи до виявлення подібностей у програмних кодах, але дослідження з пошуку більш ефективних технологій у цьому напряму продовжуються [9, 10].

Автори [11] представили детальний огляд області виявлення плагіату вихідного коду в освітньо-науковому середовищі: огляд визначень плагіату, інструментів виявлення плагіату ПК, показників порівняння, методів обфускації, та типів алгоритмів. У статті [11] також виконано категоризацію доступних інструментів виявлення АП у ПК та представлено аналіз їх ефективності.

Висновок. Використання існуючих підходів, що застосовуються для рефакторингу програмного коду у корпоративному сегменті з метою аналізу коду та пошуку подібностей в ньому, може бути застосовано до визначення ознак академічного плагіату програмних кодів в академічному середовищі. Зокрема, увага приділяється реалізації принципу "DRY" (Don't Repeat Yourself), виконанні прийому "Відокремлення методу" (Extract Method), тощо. Експерту має надаватися звіт, у якому проведено детальне маркування подібних фрагментів. Таким чином, планується проведення

емпіричного дослідження щодо використання рефакторингу для пошуку академічного плагіату програмних кодів в академічних роботах.

Список литературы: 1. До питання уникнення проблем і помилок у практиках забезпечення академічної доброчесності : Лист Міністерства освіти і науки України від 20.05.2020 р. № 1/9-263. **2.** *Robert C. Martin* Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert C. Martin – Prentice Hall, 2018. **3.** 533 million Facebook users' phone numbers and personal data have been leaked online [Електронні данні]. – Режим доступу: https://www.businessinsider.com/stolen-data-of-533-million-facebook-users-leaked-online-2021-4. **4.** Zuckerberg Loses $6 Billion in Hours as Facebook Plunges [Електронні данні]. – Режим доступу: https://www.bloomberg.com/news/articles/2021-10-04/zuckerberg-loses-7-billion-in-hours-as-facebook-plunges. **5.** *Мартин Фаулер* Рефакторинг: улучшение проекта существующего кода / *Мартин Фаулер, Кент Бек, Джон Брант, Уильям Опдайк, Дон Робертс*. – Диалектика, 2019. – 448 с. **6.** *Robert C. Martin* Clean Code: A Handbook of Agile Software Craftsmanship / Robert C. Martin. – Prentice Hall, 2019. – 464 p. **7.** The Don't Repeat Yourself Principle and the Wormhole Anti-Pattern [Електронні данні]. – Режим доступу: http://codebetter.com/jeremymiller/2007/03/22/the-dont-repeat-yourself-principle-and-the-wormhole-anti-pattern/. **8.** *Hunt Andrew* The Pragmatic Programmer: Your Journey To Mastery: 20th Anniversary Edition / *Hunt Andrew, Thomas David*. – Addison-Wesley Professional, 1999. – 352 с. **9.** *Cheers H.* Academic Source Code Plagiarism Detection by Measuring Program Behavioral Similarity / *H. Cheers, Y. Lin and S. P. Smith.* // IEEE Access. 2021. – Vol. 9. – P. 50391-50412. doi: 10.1109/ACCESS.2021.3069367. **10.** *Pandit A. A.* Review of plagiarism detection technique in source code / *Pandit Anala A., Gaurav Toksha* // International Conference on Intelligent Computing and Smart Communication. – Springer, Singapore, 2020. – P. 393-405. **11.** *Matija Novak* Source-code Similarity Detection and Detection Tools Used in Academia: A Systematic Review / *Matija Novak, Mike Joy, Dragutin Kermek* // ACM Transactions on Computing Education. – 2019. – Vol. 19, № 3. – Article 27. – DOI: https://doi.org/10.1145/3313290.