

ВИКОРИСТАННЯ СУЧАСНИХ ТЕХНОЛОГІЙ ДЛЯ АВТОМАТИЗАЦІЇ ПЕРЕВІРКИ КОДУ У НАВЧАЛЬНОМУ ПРОЦЕСІ

Вінницький національний технічний університет

Анотація

Розглянуто використання сучасних технологій для автоматизації перевірки програмного коду в навчальному процесі. Проаналізовано важливість автоматизації для зниження навантаження на викладачів та підвищення ефективності оцінювання. Досліджено різні методи автоматизації. Розглянуто переваги та недоліки впровадження таких технологій у навчальний процес.

Ключові слова: автоматизація перевірки коду, статичний аналіз, динамічне тестування, плагіат, штучний інтелект.

Abstract

The use of modern technologies for automating code verification in the educational process is discussed. The importance of automation in reducing the workload for instructors and improving the effectiveness of assessments is analyzed. Various methods of automation are explored. The advantages and disadvantages of implementing such technologies in the educational process are examined.

Keywords: code verification automation, static analysis, dynamic testing, plagiarism, artificial intelligence.

Вступ

Навчання студентів можна умовно поділити на кілька етапів: лекційні заняття, практичні роботи, контрольні завдання та екзамени. Лекції є основою теоретичних знань, практичні роботи дозволяють студентам застосовувати ці знання на практиці, а контрольні та екзаменаційні роботи служать для оцінки засвоєних знань. Важливою складовою всіх цих етапів є не лише надання інформації, а й ефективна перевірка виконаних завдань.

У дисциплінах, де основним завданням є написання програмного коду, перевірка виконаних завдань є хоч і трудомісткою, але й важливою частиною навчального процесу. Цей етап часто займає значно більше часу, ніж власне викладання теоретичного матеріалу, особливо в групах з великою кількістю студентів. Оскільки перевірка програмного коду потребує детального аналізу кожного завдання, це може створювати додаткове навантаження на викладачів, знижуючи ефективність навчального процесу.

Використання сучасних технологій для автоматизації процесів перевірки програмного коду дозволяє значно знизити навантаження на викладачів, раціональніше використовувати обмежені людські ресурси та одночасно підтримувати або навіть підвищувати ефективність перевірки.

Особливу увагу слід приділяти перевірці коду на відповідність академічним вимогам і стандартам оформлення. Це включає виявлення випадків нечесного виконання завдань, таких як плагіат, використання штучного інтелекту для генерації коду без належного розуміння завдання, порушення кодстайлу або подання неробочих рішень.

Звісно, результатом автоматизації не є виставлення оцінки студенту відразу, а лише надання викладачу корисних маркерів для подальшого оцінювання з мінімальними часовими затратами.

Таким чином, впровадження сучасних технологій для автоматизації перевірки коду є необхідним кроком для підвищення якості освіти та оптимізації навчального процесу.

Важливість автоматизації перевірки коду у навчальному процесі

Зі збільшенням обсягу студентських робіт викладачі стикаються з проблемою значних часових витрат на аналіз програмного коду. Окрім цього, оцінювання може бути суб'єктивним через людський фактор, що впливає на справедливість та послідовність результатів. Ручна перевірка також обмежує

швидкість зворотного зв'язку, що може уповільнювати процес навчання та коригування помилок студентами.

У свою чергу, впровадження автоматизованих інструментів перевірки коду дозволяє:

- оптимізувати навантаження на викладачів шляхом автоматичного аналізу рутинних аспектів коду, таких як стиль, синтаксис та відповідність стандартам;
- забезпечити об'єктивність оцінювання, зменшивши вплив людського фактору;
- забезпечити швидке отримання зворотного зв'язку для студентів;
- підвищити якість освіти, інтегруючи сучасні методи аналізу коду, такі як статичний аналіз та автоматичне тестування [1, 2].

Підходи аналізу коду в навчальному процесі

Автоматизація перевірки коду у навчальному процесі базується на кількох підходах, кожен з яких вирішує специфічні аспекти аналізу студентських робіт, зокрема:

1. Аналіз стилю коду та якості реалізації за допомогою AI або інших відповідних інструментів – системи на основі штучного інтелекту можуть автоматично оцінювати відповідність стилю коду стандартам та надавати рекомендації щодо його покращення.
2. Перевірка коду на плагіат у межах навчального закладу – автоматичне порівняння нових студентських робіт з раніше зданими завданнями для виявлення можливого плагіату.
3. Перевірка працездатності коду на основі вхідних та вихідних даних – автоматизовані тестові середовища можуть запускати код студентів із заданими параметрами та перевіряти коректність вихідних даних.

Можливі способи автоматизації перевірки коду

Сучасні технології пропонують кілька підходів до автоматизованої перевірки коду:

1. Статичний аналіз коду (DeepSource, SpotBugs, Checkstyle, PMD, SonarQube, JArchitect) – передбачає аналіз вихідного коду без його виконання для виявлення потенційних помилок, недотримання стандартів кодування та інших проблем [3].
2. Динамічне тестування (JUnit для Java, PyUnit та PyTest для Python, Jest, Mocha, Jasmine та Nightwatch для JavaScript, NUnit, MSTest та xUnit.Net для C#, Capybara та RSpec для Ruby) – виконання коду з метою перевірки його працездатності за допомогою заданих тест-кейсів. Також можливе використання власних реалізацій, які працюють за принципом вхідних та вихідних даних, що забезпечує більшу гнучкість щодо вибору студентами мов програмування [4].
3. Системи перевірки на плагіат (Moss, JPlag, Codequiry) – аналіз схожості кодових рішень між студентами [5, 6, 7].
4. Штучний інтелект та машинне навчання – автоматизоване оцінювання якості коду та надання рекомендацій щодо його покращення.

Висновок

Переваги автоматизації перевірки коду:

1. Швидкість перевірки коду в порівнянні з ручним аналізом. Автоматизовані системи здатні швидко обробляти великі обсяги коду, що значно зменшує час, який викладач витрачає на перевірку робіт. Це дозволяє прискорити весь процес навчання та оцінювання.
2. Мінімізація суб'єктивності під час оцінювання, що підвищує справедливість оцінок.
3. Надання швидшого зворотного зв'язку студентам завдяки зменшеним часовим затратам на перевірку коду.
4. Можливість інтеграції алгоритмів автоматизації у навчальні платформи.

Недоліки автоматизації перевірки коду:

1. Обмеженість в оцінюванні складних алгоритмічних рішень, які потребують глибокого контекстного розуміння. Автоматичні системи не завжди здатні адекватно оцінити складність чи оригінальність рішення, що вимагає глибокого розуміння контексту завдання. Тому в деяких випадках важко врахувати інноваційність підходу або ефективність алгоритму, якщо вони не повністю відповідають стандартним критеріям тестів.
2. Ризик помилкових висновків при використанні AI-алгоритмів, особливо у випадках некоректно складених тест-кейсів. Неправильне або неповне формулювання тестів може призвести до помилкових

результатів. Автоматична система може помилково оцінити код як неефективний чи некоректний, якщо не враховані всі можливі варіанти вхідних даних або сценаріїв.

3. Потреба в оновленні та підтримці автоматизованих систем, що може вимагати додаткових ресурсів.

Таким чином, впровадження автоматизації перевірки коду в навчальний процес є необхідним елементом, який дозволить викладачам подолати складнощі, які притаманні процесу вивчення аспектів практичного програмування. Однак, слід також звертати увагу на подолання недоліків, які можуть супроводжувати впровадження автоматизації перевірки коду в навчальний процес.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ручне тестування vs Автоматизація [Електронний ресурс] – режим доступу до ресурсу <https://training.qatetestlab.com/blog/technical-articles/manual-testing-vs-automation-testing/> (дата звернення: 22.03.2025). — Назва з екрана.

2. Повний посібник з автоматизації тестування програмного забезпечення [Електронний ресурс] – режим доступу до ресурсу <https://www.zaptest.com/uk/повний-посібник-з-автоматизації-тест> (дата звернення: 22.03.2025). — Назва з екрана.

3. Java Code Quality Tools Recommended by Developers [Електронний ресурс] – режим доступу до ресурсу <https://dev.to/saifsadiq1995/java-code-quality-tools-recommended-by-developers-3j3e> (дата звернення: 22.03.2025). — Назва з екрана.

4. Огляд мов програмування для автоматизованого тестування [Електронний ресурс] – режим доступу до ресурсу <https://training.qatetestlab.com/blog/technical-articles/overview-of-programming-languages-for-automated-testing/> (дата звернення: 22.03.2025). — Назва з екрана.

5. Moss. A System for Detecting Software Similarity [Електронний ресурс] – режим доступу до ресурсу <https://theory.stanford.edu/~aiken/moss/> (дата звернення: 22.03.2025). — Назва з екрана.

6. JPlag - Detecting Software Plagiarism [Електронний ресурс] – режим доступу до ресурсу <https://codesandbox.io/p/github/1lker/JPlag/main> (дата звернення: 22.03.2025). — Назва з екрана.

7. Codequiry. A Code Plagiarism Checker [Електронний ресурс] – режим доступу до ресурсу <https://codequiry.com/code-plagiarism-checker> (дата звернення: 22.03.2025). — Назва з екрана.

Якубенко Олександр Владиславович – студент групи ІПІ-24м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: olexandr2000xd@gmail.com

Катєльніков Денис Іванович - к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, Вінниця, fuzzy2dik@gmail.com.

Yakubenko Oleksandr Vladyslavovich – student of group ІПІ-24m , Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: olexandr2000xd@gmail.com

Katienikov Denys Ivanovich - Ph.D., Associate Professor of the Department of Software Engineering, Vinnytsia National Technical University, Vinnytsia, fuzzy2dik@gmail.com.