DOI: 10.34185/1991-7848.itmm.2026.01.032

ТРАНСФОРМАЦІЯ НАВЧАЛЬНОГО СЕРЕДОВИЩА З НАЛАГОДЖЕННЯ В ЕПОХУ GITHUB COPILOT: ЯК ЗБЕРЕГТИ ЕФЕКТИВНІСТЬ НАВЧАННЯ?

Завгородній А.Д.[1] [ORCID], Іванов О.П.[2] [ORCID]

[1]Український державний університет науки і технологій, аспірант, Україна

[2]Український державний університет науки і технологій,к.т.н., доцент, Україна

Анотація. *У роботі досліджено проблему трансформації методики навчання налагодженню програмного забезпечення в умовах активного впровадження ШІ-асистентів. Розглянуто феномен «когнітивної пасивності» студентів та обмеження сучасних LLM при роботі зі складними програмними системами. Запропоновано концепцію формування ШІ-резистентних навчальних завдань, що базуються на архітектурній складності та великих обсягах контексту, що є критичним для обмеження можливостей автоматизованого розв'язання. Обґрунтовано впровадження системи збору телеметрії взаємодії студента із середовищем розробки (IDE) як інструменту верифікації самостійності виконання робіт. Визначено ключові метрики відмінності між діями людини та ШІ-агентів, серед іншого, часові параметри опрацювання контексту та патерни використання інструментів налагодження (breakpoints, stack trace, stepping). Описано прототип інструментарію, що реалізує двокомпонентну модель оцінювання, яка враховує як коректність рішення, так і методологічну цілісність процесу налагодження. Результати дослідження спрямовані на створення адаптивних освітніх систем із можливістю формування індивідуальних рекомендацій для студентів.*

Ключові слова: *великі мовні моделі, інформаційні технології, налагодження, програмне забезпечення, телеметрія розробки, управління процесами.*

У сучасних умовах, коли технології штучного інтелекту (ШІ) невпинно прогресують, спостерігається формування стійкої залежності майбутніх розробників від інтерактивних помічників [0, 0, 0, 0, 0, 0]. Це явище часто проявляється як «когнітивна пасивність» при виконанні навчальних завдань. Попри значні успіхи, застосування ШІ в процесах налагодження (debugging) залишається недосконалим [0, 0]. У той час як типові помилки та прості алгоритмічні задачі легко вирішуються сучасними моделями, реальні виклики, з якими стикаються розробники у професійній діяльності, часто виходять за межі поточних можливостей ШІ.

Практична складова опанування навичок налагодження пов’язана з виконанням лабораторних та самостійних робіт. Проте широка доступність ШІ радикально знижує ймовірність того, що студент пройде весь шлях пошуку помилки власноруч [0, 0]. Особливо гостро ця проблема постає в умовах дистанційного навчання [0], де верифікація самостійності виконання роботи перетворюється на надскладне завдання для викладача [0].

Традиційна перевірка зазвичай обмежується аналізом фінального результату (вихідних даних програми) або, у кращому випадку, дискусією, під час якої студент має пояснити свої дії [0]. Останній метод є надзвичайно часозатратним і складно масштабованим у межах жорстко регламентованих освітніх програм. Це створює запит на створення систем автоматизованого моніторингу процесу вирішення задач.

Для вирішення окреслених проблем потрібен комплексний підхід. Першим кроком є розробка «ШІ-резистентних» задач – таких, що слугують якісним матеріалом для навчання стратегіям налагодження, але при цьому не можуть бути миттєво вирішені простим запитом до LLM [0].

Аналіз показує, що ефективність роботи ШІ значно зростає за умови використання детальних назв змінних та розлогих коментарів, що описують призначення блоків коду [0]. Однак проста обфускація (навмисне заплутування) є помилковим шляхом, оскільки вона створює надмірне когнітивне навантаження на студента, змушуючи його витрачати ресурси на дешифрування тексту, а не на аналіз логічних помилок. Необхідний пошук «золотої середини» між заплутаністю та зрозумілістю, де складність полягає в архітектурних зв’язках, а не в синтаксичному шумі.

Іншим важливим аспектом є протидія передачі готових рішень між студентами через генерацію індивідуальних завдань. Сучасні дослідження часто фокусуються на малих фрагментах коду в межах однієї функції, що є легкою здобиччю для ШІ [0]. Натомість перспективним є підхід, що базується на вразливостях LLM при роботі з великими репозиторіями та розпорошеним контекстом [0, 0, 0]. Коли велика кількість взаємопов’язаних файлів заважає простому «one-shot» вирішенню [0], це змушує систему (або студента)

використовувати складні агентні модифікації та багаторівневу взаємодію, що значно легше піддається аналізу та верифікації.

Другим критичним напрямком є створення інструментарію для збору телеметрії взаємодії студента із середовищем розробки (IDE) [0]. У 2026 році стає очевидним, що роботу автономних агентів, які діють на рівні термінальних команд, складно перехопити стандартними розширеннями, проте можна зосередитись на виявленні патернів взаємодії [0, 0, 0].

Головним критерієм може бути аналіз швидкості та природи опрацювання інформації. ШІ-агент отримує та аналізує контекст файлів за мілісекунди, тоді як людині для когнітивного сприйняття та перемикання між модулями потрібно від декількох секунд до хвилин [0, 0]. Аномально висока кількість відкритих та змінених файлів за короткий проміжок часу є прямим індикатором використання автоматизованих скриптів [0].

Пропонований інструментарій має оцінювати роботу комплексно, аналізуючи очікувані процеси налагодження в середовищах на кшталт Visual Studio. До зазначених процесів можна віднести, зокрема, такі:

- встановлення та модифікація точок зупину (breakpoints), зокрема умовних;
- покрокова ітерація (stepping) через контексти методів;
- аналіз стека викликів (stack trace) та значень локальних змінних у вікнах Watch;
- часова тривалість фокусування на конкретних ділянках коду.

Заключним елементом запропонованого підходу є синергія інструментарію та комплексних задач. Зібрана телеметрія має стати таким самим валідним підтвердженням успішного виконання завдання, як і правильний фінальний код. Це дозволяє сформувати оцінку з двох компонентів: результативної та процесуальної.

Висновки

Подальша робота має бути спрямована на розширення можливостей генерації ШІ-резистентних задач та формування «маніфесту очікуваних процесів» для кожної вправи. Розробка гнучкого інструментарію збору телеметрії не лише допоможе у виявленні академічної недоброчесності, а й

дозволить збирати цінні дані про якість засвоєння матеріалу. Це відкриває шлях до створення систем індивідуальних рекомендацій, які підкажуть студенту, на які аспекти налагодження йому слід звернути увагу, що є критично актуальним у сучасному освітньому просторі, де роль викладача все більше зміщується в бік посередника складних процесів розробки.

ЛІТЕРАТУРА / REFERENCE

1. Stasiuk O. L., Khomik O. M., Karpiuk D. R. Analiz pravovykh ryzykiv tsyfrovoho otsiniuvannia v umovakh zmishanoho navchannia. 2025. URL: https://doi.org/10.5281/zenodo.16869553 (date of access: 31.03.2026). [in Ukrainian].
2. Amoozadeh M. et al. Student-AI Interaction: A Case Study of CS1 students. 2024. URL: https://doi.org/10.48550/arXiv.2407.00305 (date of access: 31.03.2026).
3. Basha M. et al. CodeWatcher: IDE Telemetry Data Extraction Tool for Understanding Coding Interactions with LLMs. 2025. URL: https://arxiv.org/abs/2510.11536 (date of access: 31.03.2026).
4. Beleulmi S. Challenges Of Online Assessment During Covid-19 Pandemic: An Experience Of Study Skills Teachers. 2022. مجلة آفاق للعلوم. С. 49. URL: https://doi.org/10.37167/1677-007-002-004 (date of access: 31.03.2026).
5. Cotroneo D. et al. Human-Written vs. AI-Generated Code: A Large-Scale Study of Defects, Vulnerabilities, and Complexity. 2025. URL: https://doi.org/10.48550/arXiv.2508.21634 (date of access: 31.03.2026).
6. Denny P. et al. Computing Education in the Era of Generative AI. Communications of the ACM. 2024. URL: https://doi.org/10.1145/3624720 (date of access: 31.03.2026).
7. Eibl P. et al. Exploring the Challenges and Opportunities of AI-assisted Codebase Generation. 2025. URL: https://arxiv.org/abs/2508.07966 (date of access: 31.03.2026).
8. Gerlich M. AI Tools in Society: Impacts on Cognitive Offloading and the Future of Critical Thinking. Societies. 2025. Т. 15, № 1. С. 6. URL: https://doi.org/10.3390/soc15010006 (date of access: 31.03.2026).
9. Ihantola P. et al. Educational Data Mining and Learning Analytics in Programming. ITICSE '15: Innovation and Technology in Computer Science Education Conference 2015. Vilnius, Lithuania. New York, USA, 2015. URL: https://doi.org/10.1145/2858796.2858798 (date of access: 31.03.2026).
10. Korpimies K. et al. Unrestricted Use of LLMs in a Software Project Course: Student Perceptions on Learning and Impact on Course Performance. 2024. URL: https://doi.org/10.1145/3699538.3699541 (date of access: 31.03.2026).
11. Kundu D. et al. Keystroke Dynamics Against Academic Dishonesty in the Age of LLMs. 2024. URL: https://arxiv.org/abs/2406.15335 (date of access: 31.03.2026).
12. Liang S. et al. The SWE-Bench Illusion: When State-of-the-Art LLMs Remember Instead of Reason. 2025. URL: https://arxiv.org/abs/2506.12286 (date of access: 31.03.2026).
13. McDanel B., Novak E. Designing LLM-Resistant Programming Assignments: Insights and Strategies for CS Educators. SIGCSE TS 2025. Pittsburgh, USA. New York, USA, 2025. S. 756–762. URL: https://doi.org/10.1145/3641554.3701872 (date of access: 31.03.2026).

14. Messer M. et al. How Consistent Are Humans When Grading Programming Assignments? 2025. T. 25, № 4. URL: https://doi.org/10.1145/3759256 (date of access: 31.03.2026).
15. Pădurean V. et al. BugSpotter: Automated Generation of Code Debugging Exercises. 2024. URL: https://arxiv.org/abs/2411.14303 (date of access: 31.03.2026).
16. Pitts G. et al. Students' Reliance on AI in Higher Education: Identifying Contributing Factors. Communications in Computer and Information Science. Cham, 2026. С. 86–97. URL: https://doi.org/10.1007/978-3-032-12773-0_9 (date of access: 31.03.2026).
17. Shen J. H., Tamkin A. How AI Impacts Skill Formation. 2026. URL: https://arxiv.org/abs/2601.20245 (date of access: 31.03.2026).
18. Shihab M. et al. The Effects of GitHub Copilot on Computing Students' Programming Effectiveness, Efficiency, and Processes in Brownfield Coding Tasks. 2025. URL: http://dx.doi.org/10.1145/3702652.3744219 (date of access: 31.03.2026).
19. Wang Z. et al. How Does Naming Affect Language Models on Code Analysis Tasks? 2024. URL: http://dx.doi.org/10.4236/jsea.2024.1711044 (date of access: 31.03.2026).
20. Xu Z. et al. CodeVision: Detecting LLM-Generated Code Using 2D Token Probability Maps and Vision Models. 2025. URL: https://arxiv.org/abs/2501.03288 (date of access: 31.03.2026).
21. Yuan E. et al. Debug-gym: an environment for AI coding tools to learn how to debug code like programmers. 2024. URL: https://www.microsoft.com/en-us/research/blog/debug-gym-an-environment-for-ai-coding-tools-to-learn-how-to-debug-code-like-programmers/ (date of access: 31.03.2026).

TRANSFORMATION OF THE DEBUGGING LEARNING ENVIRONMENT IN THE ERA OF GITHUB COPILOT: HOW TO MAINTAIN LEARNING EFFECTIVENESS?

Andrii Zavhorodnii, Oleksandr Ivanov

Abstract. *This paper investigates the transformation of software debugging instructional methodologies amidst the active integration of AI assistants. It explores the phenomenon of student «cognitive passivity» and the limitations of current Large Language Models (LLMs) when dealing with complex software systems. A framework for designing «AI-resistant» learning tasks is proposed, leveraging architectural complexity and extensive context volumes to mitigate the potential for automated problem-solving. The study justifies the implementation of a telemetry collection system to monitor student interactions within the Integrated Development Environment (IDE) as a mechanism for verifying academic integrity and task autonomy. Key metrics for distinguishing between human and AI-agent activities are identified, including temporal context-processing parameters and debugging tool usage patterns (breakpoints, stack trace, stepping). Furthermore, the paper describes a prototype toolkit that implements a dual-component assessment model, considering both the correctness of the final code and the methodological integrity of the debugging process. The findings are aimed at developing adaptive educational systems capable of generating personalized recommendations for students.*

Keywords: *debugging, information technology, large language models, process management, software.*