

УДК 005.95:004.8

DOI <https://doi.org/10.32782/2312-7872.3.2026.18>**Огінок-Копильчак Соломія Василівна**

кандидат економічних наук, доцент,

Національний університет «Львівська політехніка»;

Львівський національний університет імені Івана Франка

ORCID: <https://orcid.org/0000-0001-5462-5362>**Лесюк Андрій Іванович**

аспірант,

Національний університет «Львівська політехніка»

ORCID: <https://orcid.org/0009-0009-8498-6279>

## ПРОДУКТИВНІСТЬ РОЗРОБНИКІВ В ЕПОХУ ШТУЧНОГО ІНТЕЛЕКТУ ТА МІКРОСЕРВІСІВ: ПРИКЛАД ІНТЕГРАЦІЇ «GITHUB COPILOT» У КОМПАНІЇ «DUOLINGO»

**Анотація.** У цій статті досліджується реальний досвід та результати стратегічного впровадження інструменту генеративного штучного інтелекту (GenAI) «GitHub Copilot» у розробницьку екосистему компанії «Duolingo» – провідної у світі освітньої платформи з глобальною аудиторією понад 500 мільйонів користувачів [1]. В умовах стрімкого масштабування бізнесу компанія зіткнулася з критичною потребою оптимізації процесів написання коду. У роботі детально висвітлено комплексний підхід до подолання когнітивного тертя та фрагментації робочих процесів, які неминуче виникли внаслідок еволюційного технологічного переходу архітектури від 3 великих монолітних репозиторіїв до складної розподіленої мережі з більш ніж 400 мікросервісів [1; 2]. Така децентралізація значно ускладнила навігацію по кодовій базі та збільшила час онбордингу інженерів. Для оцінки ефективності інтеграції ШІ-асистента в дослідженні застосовано багатовимірну методологію. На основі глибокого комбінованого аналізу об'єктивних кількісних метрик екосистеми «GitHub» (зокрема швидкості роботи з pull requests) та суб'єктивних якісних показників, виміряних за допомогою фреймворку «SPACE» (задоволеність, продуктивність, активність, комунікація, ефективність) [6], емпірично доведено позитивний вплив інновації. Результати засвідчують, що систематичне використання ШІ-асистентів у тісному поєднанні з сучасними хмарними середовищами розробки (як-от GitHub Codespaces) забезпечує суттєвий приріст інженерної швидкості: на 25% серед розробників під час роботи у нових або малознайомих для них репозиторіях, та на стабільні 10% – у знайомому середовищі [1]. Крім того, стаття концептуально показує, що інструменти GenAI сьогодні виходять за межі звичайного автодоповнення. Вони виступають потужним системним мультиплікатором експертизи команди, знижують когнітивне навантаження та слугують ефективним засобом збереження ментального стану потоку (flow state) під час вирішення складних алгоритмічних задач [3; 7; 8]. Отримані висновки створюють науково-практичне підґрунтя для переосмислення ролі штучного інтелекту як повноцінного партнера у процесі інженерії програмного забезпечення.

**Ключові слова:** інженерія програмного забезпечення, мікросервісна архітектура, «GitHub Copilot», «GitHub Codespaces», продуктивність розробників, стан потоку (flow state), фреймворк SPACE, штучний інтелект як партнер.

**Постановка проблеми.** Сучасна розробка великомасштабних споживчих IT-продуктів вимагає від інженерних організацій надзвичайної гнучкості, високої швидкості постачання функціоналу (velocity) та безперервного масштабування інфраструктури. Проте традиційні методики оцінки ефективності розробників – такі як підрахунок чистої кількості згенерованих рядків коду (Source Lines of Code, SLOC) – остаточно втратили актуальність у контексті складних розподілених систем [6].

Головним бар'єром для продуктивності сучасного інженера стає не фізичне введення символів з клавіатури, а когнітивне навантаження. Перемикання контексту, пошук синтаксису у сторонній документації та навігація між сотнями ізольованих сервісів створюють суттєве «когнітивне тертя» [3].

Компанія «Duolingo», обслуговуючи понад 500 мільйонів активних користувачів, зіткнулася з класичним «податком на масштабування»: інженерна екосистема розрослася з кількох монолітів до понад 400 мікросервісів [1]. Щоб зберегти високий темп інновацій, керівництво «Duolingo» зробило ставку на концепцію інженера як «мультиплікатора експертизи», інтегрувавши «GitHub Copilot» безпосередньо у повсякденні розробницькі пайплайни [1; 2].

**Аналіз останніх досліджень і публікацій.** Проблема вимірювання продуктивності розробників протягом тривалого часу була предметом дискусій у науковій спільноті. Фундаментальний зсув відбувся із запровадженням фреймворку «SPACE» (розробленого Н. Форсгрена, М.-А. Сторі, С. Мадділа, Т. Ціммерманном та ін.), який пропонує оцінювати продуктивність через багатовимірні показники, включаючи задоволеність, активність, співпрацю та стан потоку [6].

З появою генеративних нейромережових моделей постало питання їх практичного впливу на кодинг. Закордонні вчені, зокрема А. Ціглер, Е. Калліамваку, Х. Лі, А. Райс, у своїх працях підтвердили, що нейромережовий автокомпліт безпосередньо підвищує задоволеність інженерів за рахунок підтримки сфокусованості [7]. У вітчизняному науковому дискурсі питання ефективності автоматизованого генерування програмного коду засобами штучного інтелекту, оцінювання продуктивності та впливу LLM на інженерні процеси детально дослідили А. І. Купін та Д. Ю. Духов [8].

Згідно з індустріальними прогнозами аналітиків (зокрема звітами Panto AI) [5], застосування AI-асистентів глобально скорочує час виконання завдань та знижує ментальну втомлюваність.

**Мета статті.** Системно проаналізувати практичний досвід та оцінити реальну ефективність впровадження інструментів генеративного штучного інтелекту (зокрема «GitHub Copilot») у розробницькі процеси компанії «Duolingo» в умовах складної, фрагментованої мікросервісної архітектури.

**Виклад основного матеріалу дослідження.** На початкових етапах інженери «Duolingo» фокусувалися на мобільних додатках та базовій інфраструктурі, використовуючи лише 3 основні репозиторії. Згодом, прагнення пришвидшити автономні релізи та розмежувати зони відповідальності зумовило міграцію «Duolingo» на мікросервісну архітектуру, що призвело до розширення екосистеми до понад 400 окремих репозиторіїв [1; 2]. Звільнивши архітекторів від блокуючих залежностей, цей крок створив два нові системні виклики:

- культурна ізоляція: кожен репозиторій сформував власну унікальну культуру розробки та специфічні процеси перевірки пулл-реквестів (запитів на внесення змін до спільного репозиторію коду);
- податок на мобільність: перехід розробника з однієї команди в іншу або спроба внести правки у суміжний сервіс вимагали значного часу на онбординг (структурований процес інтеграції нового співробітника в команду), вивчення чужих конвенцій та копання в локальних бібліотеках.

Замість того, щоб розширювати штат інженерів лінійно відповідно до зростання користувацької бази, «Duolingo» зробила ставку на автоматизацію. Близько 300 розробників компанії тісно співпрацюють із науковцями в галузі лінгвістики, експертами з машинного навчання та дизайнерами. Впровадження «GitHub Copilot» розглядалося як інструмент, здатний прибрати когнітивне тертя, дозволяючи інженерам фокусуватися на високорівневих завданнях проектування, а не на написанні шаблонного коду та boilerplate-конструкцій.

Дослідження базується на змішаному методі оцінки інженерних метрик та якісних відгуків лідерів розробки «Duolingo». Щоб ШІ-асистент працював ефективно, компанія не просто роздала ліцензії, а розгорнула комплексну екосистему, яка включає [1]:

- «GitHub Copilot»: контекстне автодоповнення коду та підказка синтаксису безпосередньо в IDE (інтегроване середовище розробки, від англ. *Integrated Development Environment*);
- «GitHub Codespaces»: хмарні середовища розробки, які дозволяють підняти повністю налаштоване робоче місце для будь-якого з 400+ мікросервісів за лічені секунди;
- кастомні API-інтеграції: інструменти автоматичного контролю за дотриманням єдиних стандартів та конвенцій написання коду по всій компанії.

Кількісні вимірювання інженерної продуктивності після повноцінної інтеграції «GitHub Copilot» продемонстрували чіткі відмінності залежно від досвіду роботи інженера з конкретною кодовою базою (Таблиця 1) [1].

У реальних індустріальних випадках, таких як досвід компанії «Duolingo», розрахунок показників продуктивності від впровадження ШІ (зокрема 25% та 10%) базується на поєднанні експертної оцінки менеджменту (внутрішнього трекінгу лід-тайму) та об'єктивних метрик екосистеми GitHub.

Оскільки чистий підрахунок рядків коду (LOC) визнано неефективним, інженерна команда «Duolingo» використовувала комплексний підхід до вимірювання швидкості (velocity). Пропонуємо вашій увазі розглянути цей підхід більш детально.

Показники приросту швидкості (25% для розробників у нових репозиторіях і 10% для роботи у знайомому середовищі) – це внутрішня інженерна оцінка, яку сформував старший інженерний менеджер «Duolingo» Джонатан Буркет.

Вона розраховувалася на основі порівняння часу, який розробники витрачали на виконання завдань (Lead Time to Task Resolution) у двох сценаріях:

- у нових репозиторіях (+25%): вимірювався час, необхідний інженеру для внесення перших корисних змін (PR) у мікросервіс суміжної команди. Раніше цей процес вимагав тривалого вивчення документації та локальних конвенцій. Асистент «Copilot» скоротив це «когнітивне тертя», підказуючи потрібний синтаксис та локальні паттерни безпосередньо в контексті кодової бази компанії.
- у знайомих репозиторіях (+10%): оцінювалося прискорення рутинних операцій розробниками, які вже володіють контекстом свого сервісу. Тут ШІ економить час суто на написанні шаблонного коду (boilerplate), повторюваних структур та автотестів.

**Таблиця 1 – Диференціація приросту інженерної швидкості (Velocity) після впровадження «GitHub Copilot» залежно від рівня контексту розробника**

| Категорія інженерів                                                       | Показник приросту швидкості розробки | Основне джерело ефективності                                                                                     |
|---------------------------------------------------------------------------|--------------------------------------|------------------------------------------------------------------------------------------------------------------|
| Новачки в репозиторії (розробники, що адаптуються до чужого мікросервісу) | +25%                                 | Швидка підказка локальних патернів, функцій та внутрішнього синтаксису без читання сотень сторінок документації. |
| Досвідчені розробники (працюють у своєму звичному сервісі)                | +10%                                 | Автоматичне дописування рутинних конструкцій, автотестів та зменшення потреби в ручному копіюванні шаблонів.     |

Джерело: сформовано авторами на основі [1] та внутрішньої аналітики компанії «Duolingo»

**Таблиця 2 – Метрики приросту ефективності в інженерній екосистемі «Duolingo»**

| Вимір                                    | Показник                 | Технічний та організаційний механізм                                                    |
|------------------------------------------|--------------------------|-----------------------------------------------------------------------------------------|
| Час підготовки середовища («Codespaces») | 1 хвилина (замість днів) | Автоматичний запуск хмарного контейнера з готовими залежностями та мережею «Tailscale». |
| Обсяг розробки (Pull Request Volume)     | +70%                     | Зростання пропускної здатності (throughput) завдяки зменшенню ручної роботи.            |
| Затримка ревію (Code Review Turnaround)  | -67% (з 9.6 до 2.4 днів) | Покращення читабельності коду завдяки стандартним патернам, які генерує ШІ.             |

Джерело: сформовано авторами на основі [1; 3] та аналітики екосистеми GitHub

Щоб перевірити, чи суб'єктивні оцінки менеджменту відповідають реальності, «Duolingo» проаналізувала «тверді» метрики життєвого циклу розробки через інструменти аналітики GitHub (Таблиця 2) [1; 3].

Замість того, щоб міряти суху кількість натискань на клавіші, «Duolingo» сперлася на філософію фреймворку «SPACE», де швидкість розглядається через задоволеність та збереження фокусу [6].

Менеджмент фіксував зменшення кількості випадків, коли розробник змушений залишати IDE, відкривати браузер і перемикає контекст на пошук синтаксису чи документації сторонніх бібліотек. Збереження фокусу автоматично конвертувалося у вищу швидкість розробки (velocity) [3; 7; 8].

Якісний аналіз підтвердив, що ключова цінність інструменту полягає не лише у швидкості, а й у психологічному комфорті розробників. Керівники інженерних напрямків відзначають суттєву зміну ментальних звичок команд:

«З «GitHub Copilot» наші розробники залишаються в стані потоку та зберігають динаміку, замість того щоб постійно прочісувати бібліотеки коду чи зовнішню документацію», – Джонатан Буркет, старший інженерний менеджер «Duolingo» [1].

Зменшення потреби перемикатися на браузер для пошуку банальних синтаксичних конструкцій у сторонніх сервісах кардинально знизило рівень щоденної втоми та підвищило задоволеність роботою.

Одним із найцікавіших висновків кейсу «Duolingo» є те, що «GitHub Copilot» фактично став інструментом м'якої стандартизації архітектури. В умовах 400 окремих репозиторіїв з різною культурою ШІ-асистент, підлаштовуючись під контекст компанії, пропонує рішення, які відповідають загальноприйнятим інженерним практикам організації. Це нівелює колишні розбіжності в pull-реквестах різних команд.

Стратегія «Duolingo» щодо застосування ШІ не обмежується інженерією. Аналогічний підхід застосовано для створення навчального контенту. Було розроблено платформу «DuoFactory» на базі «Apache Airflow», яка оркеструє пайплайни великих мовних моделей (Таблиця 3) [4].

Цей паралельний кейс підтверджує: максимальний ефект від ШІ досягається лише тоді, коли він інтегрований як глибокий елемент загальної інфраструктури, а не як ізольований інструмент [4].

Таблиця 3 – Еволюція моделей генерації контенту: від ручної праці до оркестрації ШІ

| Етап еволюції  | Підхід до контенту                     | Часова ефективність              | Роль людини                                | Економічний ефект                                  |
|----------------|----------------------------------------|----------------------------------|--------------------------------------------|----------------------------------------------------|
| Патріархальний | Традиційний (повністю вручну)          | 1 місяць (на 1 епізод)           | Творець (кише, редагує, кодує)             | Базова вартість (дорого, не масштабується)         |
| Перехідний     | Гібридний (із залученням LLM)          | 6 годин (на 1 епізод)            | Редактор (керівляє та коригує ШІ)          | Суттєве прискорення процесів                       |
| Проривний      | Оркестрований (платформа «DuoFactory») | Лічені хвилини (Потоковий режим) | Супервізор (контроль фінальних пайплайнів) | Скорочення витрат на 99% (експоненційне зростання) |

Джерело: сформовано авторами на основі [4]

**Висновки.** Приклад компанії «Duolingo» демонструє успішний сценарій подолання мікросервісного бар'єру за допомогою генеративного ШІ. Основні висновки дослідження:

Генеративні ШІ-асистенти дають найбільший приріст продуктивності (+25%) під час онбордингу та крос-репозиторної взаємодії інженерів у розподілених мікросервісних екосистемах;

Реальна інженерна продуктивність визначається часом перебування розробника у стані безперервного когнітивного потоку (flow state) без вимушеного перемикавання контексту;

Максимальна віддача від «GitHub Copilot» можлива лише за умови його комбінування з хмарними інфраструктурами («Codespaces») та автоматизованими пайплайнами.

**Перспективи подальших досліджень у цьому напрямі.** Перспективними напрямками дослідження є довгостроковий аналіз накопичення технічного боргу та уразливостей у коді, згенерованому ШІ (з урахуванням останніх даних 2025–2026 років про якість AI-комітів), аналіз ефектів для Junior- та Senior-розробників окремо, а також вивчення етичних аспектів використання коду, згенерованого на основі публічних репозиторіїв.

#### СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. GitHub Customer Stories. Duolingo empowers its engineers to be force multipliers for expertise with GitHub Copilot. URL: <https://github.com/customer-stories/duolingo>
2. ZenML LLMops Database. Duolingo: GitHub Copilot Integration for Enhanced Developer Productivity. URL: <https://www.zenml.io/llmops-database/github-copilot-integration-for-enhanced-developer-productivity>
3. GitHub Blog. The economic impact of the AI-powered developer lifecycle and lessons from GitHub Copilot. URL: <https://github.blog/news-insights/research/the-economic-impact-of-the-ai-powered-developer-lifecycle-and-lessons-from-github-copilot> (дата звернення: 07.09.2026).
4. Astronomer Blog. Orchestrating AI At Duolingo To Unlock 99% Lower Costs, 10x User Growth (Belle Romea, Software Engineer at Duolingo). URL: <https://www.astronomer.io/blog/airflow-in-action-duolingo/>
5. Panto AI. GitHub Copilot Statistics 2026 – Users, Revenue & Adoption. URL: <https://www.getpanto.ai/blog/github-copilot-statistics> (дата звернення: 07.09.2026).
6. Forsgren N., Storey M.-A., Maddila C., Zimmermann T., Houck B., Butler J. The SPACE of Developer Productivity. *ACM Queue*. 2021. Vol. 19, Is. 1. P. 20–48. DOI: <https://doi.org/10.1145/3454122.3454124>
7. Ziegler A., Kalliamvakou E., Li X., Rice A., Rifkin D., Simister S., Aftandilian E. Productivity Assessment of Neural Code Completion. *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming (MAPS 2022)*. 2022. P. 21–29. DOI: <https://doi.org/10.1145/3520312.3534864>
8. Купін А. І., Духов Д. Ю. Дослідження ефективності автоматизованого генерування програмного коду засобами штучного інтелекту. *Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки*. 2026. Т. 37 (76), № 2, ч. 1. С. 157–163. DOI: <https://doi.org/10.32782/2663-5941/2026.2.1/23>

#### REFERENCES

1. GitHub. (n.d.). *Duolingo empowers its engineers to be force multipliers for expertise with GitHub Copilot*. GitHub Customer Stories. Retrieved September 7, 2026, from <https://github.com/customer-stories/duolingo>
2. ZenML. (n.d.). *Duolingo: GitHub Copilot Integration for Enhanced Developer Productivity*. ZenML LLMops Database. Retrieved September 7, 2026, from <https://www.zenml.io/llmops-database/github-copilot-integration-for-enhanced-developer-productivity>
3. GitHub. (n.d.). *The economic impact of the AI-powered developer lifecycle and lessons from GitHub Copilot*. GitHub Blog. Retrieved September 7, 2026, from <https://github.blog/news-insights/research/the-economic-impact-of-the-ai-powered-developer-lifecycle-and-lessons-from-github-copilot/>
4. Romea, B. (n.d.). *Orchestrating AI At Duolingo To Unlock 99% Lower Costs, 10x User Growth*. Astronomer Blog. Retrieved September 7, 2026, from <https://www.astronomer.io/blog/airflow-in-action-duolingo/>
5. Panto AI. (2026). *GitHub Copilot Statistics 2026 – Users, Revenue & Adoption*. Retrieved September 7, 2026, from <https://www.getpanto.ai/blog/github-copilot-statistics>
6. Forsgren, N., Storey, M.-A., Maddila, C., Zimmermann, T., Houck, B., & Butler, J. (2021). The SPACE of Developer Productivity. *ACM Queue*, 19(1), 20–48. <https://doi.org/10.1145/3454122.3454124>

7. Ziegler, A., Kalliamvakou, E., Li, X., Rice, A., Rifkin, D., Simister, S., ... & Aftandilian, E. (2022). Productivity Assessment of Neural Code Completion. *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming (MAPS 2022)*, 21–29. <https://doi.org/10.1145/3520312.3534864>

8. Kupin, A. I., & Dukhov, D. Yu. (2026). Doslidzhennia efektyvnosti avtomatyzovanoho heneruvannia prohramnoho kodu zasobamy shtuchnoho intelektu [Research on the effectiveness of automated generation of program code by means of artificial intelligence]. *Vcheni zapysky Tavriiskoho natsionalnoho universytetu imeni V. I. Vernadskoho. Seriya: Tekhnichni nauky*, 37(2, Pt. 1), 157–163. <https://doi.org/10.32782/2663-5941/2026.2.1/23>

**Solomiya Ohinok-Kopylchak**

*Lviv Polytechnic National University;  
Ivan Franko National University of Lviv*

**Andrii Lesiuk**

*Lviv Polytechnic National University*

### **DEVELOPER PRODUCTIVITY IN THE ERA OF ARTIFICIAL INTELLIGENCE AND MICROSERVICES: A CASE STUDY OF GITHUB COPILOT INTEGRATION AT DUOLINGO**

**Abstract.** *This article examines the real-world experience and results of the strategic implementation of the generative artificial intelligence (GenAI) tool "GitHub Copilot" into the development ecosystem of Duolingo – the world's leading educational platform with a global audience of over 500 million users [1]. Amid rapid business scaling, the company faced a critical need to optimize code writing processes. The paper details a comprehensive approach to overcoming cognitive friction and workflow fragmentation that inevitably arose from the evolutionary technological transition of the architecture from 3 large monolithic repositories to a complex distributed network of over 400 microservices [1; 2]. Such decentralization significantly complicated codebase navigation and increased engineer onboarding time. To evaluate the effectiveness of the AI assistant's integration, a multidimensional methodology was applied in the study. Based on an in-depth combined analysis of objective quantitative GitHub ecosystem metrics (specifically pull request processing velocity) and subjective qualitative indicators measured using the "SPACE" framework (satisfaction, performance, activity, communication, efficiency) [6], the positive impact of the innovation is empirically proven. The results demonstrate that the systematic use of AI assistants in close combination with modern cloud development environments (such as GitHub Codespaces) provides a substantial increase in engineering velocity: 25% among developers working in new or unfamiliar repositories, and a consistent 10% in a familiar environment [1]. Furthermore, the article conceptually shows that GenAI tools today go beyond simple autocompletion. They act as a powerful systemic multiplier of team expertise, reduce cognitive load, and serve as an effective means of maintaining mental flow state while solving complex algorithmic tasks [3; 7; 8]. The findings create a scientific and practical foundation for rethinking the role of artificial intelligence as a full-fledged partner in the software engineering process.*

**Keywords:** *Software engineering, microservices architecture, GitHub Copilot, GitHub Codespaces, developer productivity, flow state, SPACE framework, artificial intelligence as a partner.*

*Дата надходження статті: 07.09.2026*

*Дата прийняття статті до публікації: 20.10.2026*

*Дата публікації статті: 23.10.2026*