

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
Науково-навчальний центр прикладної інформатики

ІНСТИТУТ ІННОВАЦІЙНОЇ ОСВІТИ

ІННОВАЦІЙНИЙ РОЗВИТОК СУСПІЛЬСТВА: НАУКА, ПРАВО ТА ТЕХНОЛОГІЇ У ДОБУ ЦИФРОВОЇ ТРАНСФОРМАЦІЇ

МАТЕРІАЛИ
Міжнародної науково-практичної конференції

*20–21 березня 2026 р.
м. Київ*



Київ–Запоріжжя
Інститут інноваційної освіти
2026

УДК 001(063):378.4 (Укр)
Р67

До збірника увійшли матеріали наукових робіт (тези доповідей, статті), надані згідно з вимогами, що були заявлені на конференцію.

Роботи друкуються в авторській редакції, мовою оригіналу.

Автори несуть всю повноту відповідальності за зміст поданих матеріалів.

Редакція не завжди поділяє думки авторів і не несе відповідальності за недостовірність публікованих даних. Претензії до організаторів не приймаються.

При передруку матеріалів посилання обов'язкове.

ISBN 978-966-488-352-5

I66 **Інноваційний розвиток суспільства: наука, право та технології у добу цифрової трансформації** : Матеріали Міжнародної науково-практичної конференції (м. Київ, 20–21 березня 2026 р.) / ГО «Інститут інноваційної освіти»; Науково-навчальний центр прикладної інформатики НАН України. – 2-е вид., випр. і доп. – Київ–Запоріжжя: АА Тандем, 2026. – 146 с.

Матеріали конференції рекомендуються освітянам, науковцям, викладачам, здобувачам вищої освіти, аспірантам, докторантам, студентам вищих навчальних закладів тощо¹.

Відповідальний редактор: С.К. Бурма
Коректор: П.А. Нємкова

Матеріали видано в авторській редакції.

УДК 001(063):378.4 (Укр)

ISBN 978-966-488-352-5

© Усі права авторів застережені, 2026

© Інститут інноваційної освіти, 2026

© АА Тандем, 2026

¹ Відповідає п. 8 Порядку присудження (позбавлення) наукових ступенів Затвердженого Постановою Кабінету Міністрів України від 17 листопада 2021 р. № 1197; п. 28 Постанови Кабінету Міністрів України від 30 грудня 2015 р. № 1187 «Про затвердження Ліцензійних умов провадження освітньої діяльності»; п. 13 Постанови Кабінету Міністрів України від 12 липня 2004 р. № 882 «Питання стипендіального забезпечення»

Список використаних джерел

1. Вакуленко І. А., Колосок С. І. Типологізація «розумних» екологобезпечних енергетичних рішень адаптованих до особливостей вітчизняних енергомереж. Вісник Сумського державного університету. № 2. С. 21–25. DOI:10.21272/1817-9215.2019.2-3.
2. Chen Z., Amani A. M., Yu X. та ін. Control and Optimisation of Power Grids Using Smart Meter Data: A Review. Sensors. Вип. 23, № 4. С. 2118. DOI:10.3390/s23042118.
3. Strezoski L. Distributed energy resource management systems—DERMS: State of the art and how to move forward. WIREs Energy and Environment. Вип. 12, № 1. С. e460. DOI:10.1002/wene.460.
4. Чорній В. ВПЛИВ ВІЙНИ НА ЕНЕРГЕТИЧНУ СИСТЕМУ УКРАЇНИ. Herald of Khmelnytskyi National University. Economic sciences. Vol. 304, Issue 2(2). P. 196–202. DOI:10.31891/2307-5740-2022-304-2(2)-31.
5. Ali M., Guan Y., Vasquez J. C. et al. Grid code requirements for the integration of renewable energy sources in Indonesia—a review. Renewable Energy Focus. Vol. 56, 01.03.2026. P. 100782. DOI:10.1016/j.ref.2025.100782.

Сенишин Роман Васильович

аспірант кафедри комп'ютерних наук та програмної інженерії,
Приватний вищий навчальний заклад «Європейський університет»
ORCID: <https://orcid.org/0009-0007-7900-8679>

ТЕОРЕТИЧНІ ЗАСАДИ ЕВОЛЮЦІЇ ЖИТТЄВОГО ЦИКЛУ ВЕБ-РОЗРОБКИ В КОНТЕКСТІ ВПРОВАДЖЕННЯ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

У традиційній інтерпретації життєвий цикл розробки програмного забезпечення (SDLC) структуровано як послідовність відносно стабільних фаз, що охоплюють планування, аналіз, проектування, реалізацію, тестування, розгортання та супровід. Така фазова модель забезпечує формалізовану організацію процесу, прозорість створюваних артефактів і можливість системного управління ризиками, проте її ефективність є максимальною за умов відносної визначеності та стабільності вимог. У сфері веб-розробки зазначені передумови поступово втрачають актуальність, оскільки вимоги динамічно трансформуються під впливом ринкових факторів і поведінкових патернів користувачів, інтерфейсні рішення та бізнес-логіка розвиваються синхронно, а модель релізного циклу еволюціонує від дискретного проєктного підходу до безперервної доставки програмного продукту [5].

У відповідь на обмеження лінійних моделей SDLC сформувалася Agile-парадигма, яка концептуалізує процес розробки як адаптивне створення цінності в умовах високої невизначеності. Зокрема, у межах Scrum розробка

тракується як емпіричний процес, що базується на принципах прозорості, інспекції та адаптації, тоді як Kanban орієнтований на оптимізацію потоку створення цінності через візуалізацію робочих процесів, управління незавершеною роботою та безперервне вдосконалення. Для веб-розробки, яка характеризується одночасною інтеграцією змін користувацького досвіду, API-взаємодій, оновлення залежностей, вимог безпеки та операційних аспектів, зазначені підходи демонструють вищу методологічну релевантність порівняно з жорстко детермінованими фазовими моделями [1].

Сучасний етап розвитку програмної інженерії характеризується не відмовою від SDLC як концептуальної основи, а його трансформацією шляхом інтеграції гнучких і гібридних підходів. Емпіричні дані засвідчують, що переважна частка організацій застосовує Agile у межах SDLC, при цьому значна кількість практик реалізується у форматі комбінованих моделей, що поєднують елементи Agile, DevOps та інших підходів. Зокрема, за результатами 17th State of Agile Report, 71% респондентів інтегрують Agile у життєвий цикл розробки, 42% використовують гібридні моделі, а Scrum залишається домінантною практикою на рівні команд. Така тенденція відображає зміщення від жорсткої методологічної уніфікації до прагматичної адаптації інструментів і підходів відповідно до специфіки організаційного контексту [4]. Отже, результати досліджень підкреслюють зростання ролі стабільності пріоритетів, орієнтації на користувача та здатності інтегрувати інноваційні технології без порушення цілісності процесів розробки.

Цифрові двійники виступають важливим інструментом цифрової трансформації, що забезпечує безперервний зв'язок між фізичними та цифровими об'єктами. Використання таких технологій у поєднанні з генеративним штучним інтелектом дозволяє автоматизувати процеси аналізу вимог, генерації програмного коду, тестування та підтримки вебзастосунків, що сприяє еволюції сучасних моделей розробки програмного забезпечення [8].

У цьому контексті генеративний штучний інтелект доцільно розглядати не як чинник, що заміщує класичний життєвий цикл розробки, а як надбудову, яка формує додатковий рівень інтелектуалізованої автоматизації процесів. Систематичний огляд застосування великих мовних моделей у програмній інженерії, що охоплює 947 досліджень і 112 прикладних задач, демонструє їхню інтеграцію в усі ключові фази інженерного workflow від формування вимог і створення документації до тестування, рев'ю, рефакторингу та супроводу програмних продуктів. Відповідно, функціональне поле LLM виходить за межі генерації коду і трансформується у засіб підтримки повного життєвого циклу розробки. Аналогічний підхід репрезентовано в технічній документації IBM, де концепція AI in the SDLC

інтерпретується як системна інтеграція інструментів штучного інтелекту у всі етапи циклу з метою підвищення швидкості розробки, якості програмного продукту та обґрунтованості управлінських рішень. Така інтерпретація узгоджується із сучасними теоретичними підходами програмної інженерії, у яких ключовим аналітичним об'єктом виступає не окрема фаза, а цілісний ланцюг трансформації артефактів і переходів між ними [6].

Практичний вимір зазначених трансформацій простежується на прикладі сучасних інструментальних рішень. Зокрема, офіційна документація GitHub Copilot фіксує розширення функціональності від базових механізмів автодоповнення коду до підтримки процедур code review, реалізації агентних режимів у середовищах розробки, а також до сценаріїв створення повноцінних програмних продуктів на основі натураломовних інструкцій. Важливим є також те, що інструмент здійснює багатовимірний аналіз pull request-ів із урахуванням контексту репозиторію, ідентифікує потенційні проблеми та пропонує варіанти їх усунення. У ширшому контексті розвитку інтелектуальних інструментів програмної інженерії Codex позиціонується як хмарно-орієнтований агент розробки, здатний виконувати комплексні завдання — від реалізації функціональності до тестування та статичного аналізу коду, тоді як інтерактивні можливості ChatGPT забезпечують підтримку циклу розробки через редагування, інтерпретацію та візуалізацію результатів. Отже, генеративний штучний інтелект поступово інтегрується у суміжні фази життєвого циклу, змінюючи саму логіку їх взаємодії [3].

У цьому зв'язку принципового значення набуває питання ідентифікації джерел втрат ресурсів і часу в сучасному процесі розробки. Емпіричні дослідження засвідчують, що такі втрати локалізуються переважно не на етапі кодування як такому, а в інтерфейсних зонах взаємодії між фазами життєвого циклу. Зокрема, результати аналізу діяльності Agile-команд вказують на перевантаження беклогів, часті перемикання контексту, фрагментацію бачення продукту та додаткові часові витрати, зумовлені жорсткими процедурами спринт-планування. Додатково, міждисциплінарні дослідження продуктивності розробників демонструють, що інструментальні фрикції, процесні неузгодженості та особливості командної взаємодії суттєво впливають на ефективність праці та суб'єктивну задоволеність діяльністю. Інтеграція генеративного ШІ формує новий тип витрат, пов'язаний із необхідністю верифікації результатів роботи моделей: зокрема, близько 22,4% часу сесії витрачається на перевірку запропонованих рішень, а стани взаємодії з інструментами штучного інтелекту охоплюють понад половину загальної тривалості робочого циклу. У контексті веб-розробки це дозволяє зробити висновок, що ключовий резерв підвищення ефективності полягає у мінімізації втрат на переходах

між етапами «вимоги – проектування – реалізація – тестування – розгортання», а не виключно в оптимізації процесу написання коду [2].

Отже, на основі проведеного аналізу, еволюція життєвого циклу веб-розробки в умовах інтеграції генеративного штучного інтелекту не становить розриву з класичною логікою SDLC, а репрезентує її закономірну трансформацію. Зберігаючи фазову структурованість як методологічний каркас, сучасний SDLC зазнає суттєвих змін у внутрішній організації, оскільки тривалість окремих фаз скорочується, інтенсивність взаємозв'язків між ними зростає, а значна частина рутинних операцій делегується автоматизованим інструментам. У результаті відбувається перехід від дискретної моделі виконання до безперервного, інтегрованого процесу створення програмного продукту.

На думку Ю. А. Перегуди, інтеграція технологій штучного інтелекту в різні сфери діяльності зумовлює структурні трансформації економічних систем та формує нові підходи до організації цифрових процесів, що створює передумови для перегляду традиційних моделей розроблення програмного забезпечення та вебсистем [7]. Теоретично обґрунтованою для сучасної веб-розробки є синтетична конструкція, що поєднує принципи SDLC, Agile, DevOps і генеративного ШІ. У такій моделі генеративні інструменти виконують функцію інтенсифікації потоку створення цінності, водночас не елімінуючи ролі суб'єкта розробки, а навпаки – актуалізуючи значення експертного контролю, архітектурного мислення та забезпечення безпеки як інваріантних компонентів інженерної діяльності.

Практична імплементація зазначених положень передбачає цілеспрямовану інтеграцію генеративного штучного інтелекту насамперед у ті сегменти життєвого циклу, які характеризуються повторюваністю та частковою формалізованістю, зокрема аналіз вимог, підготовку проектної документації, генерацію шаблонних фрагментів коду, тестових сценаріїв і супровідних матеріалів релізу. Водночас гнучкі підходи, зокрема Agile і Kanban, доцільно інтерпретувати як інструменти оптимізації потоків робіт у межах SDLC, а не як альтернативні моделі організації розробки.

Оцінювання ефективності впровадження генеративного ШІ має ґрунтуватися на системних метриках, що відображають скорочення часу проходження завдань між фазами, зниження рівня дефектності та мінімізацію втрат, пов'язаних із перемиканням контексту. Ключовою умовою забезпечення якості та надійності результатів залишається формалізація процедур людського контролю в критичних точках життєвого циклу, зокрема на етапах формування вимог, архітектурного проектування, визначення публічних інтерфейсів, забезпечення безпеки, виконання релізів і пострелізного моніторингу.

Список використаних джерел

1. Швабер К., Сазерленд Д. Посібник зі Скраму: Повний посібник зі Scrum: Правила гри. 2020. URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Ukrainian.pdf> (дата звернення: 28.01.2026).
2. Chaban C. B., Korol C. B. Tasks Distribution in Agile Teams: Challenges and Solutions. Problems of Modern Transformations. Series: Economics and Management. 2025. № 17. URL: <https://doi.org/10.54929/2786-5738-2025-17-04-09>
3. GitHub Copilot features. GitHub Docs. URL: <https://docs.github.com/en/copilot/get-started/features> (дата звернення: 30.04.2026).
4. 17th State of Agile Report. 2024. 26 с. URL: <https://2288549.fs1.hubspotusercontent-na1.net/hubfs/2288549/RE-SA-17th-Annual-State-Of-Agile-Report.pdf> (дата звернення: 28.01.2026).
5. What is the Software Development Lifecycle (SDLC)? IBM. URL: <https://www.ibm.com/think/topics/sdlc> (дата звернення: 25.01.2026).
6. Zhang Q., Fang C., Xie Y., Zhang Y., Yang Y., Sun W., Yu S., Chen Z. A survey on large language models for software engineering. arXiv. 2023. URL: <https://arxiv.org/abs/2312.15223> (дата звернення: 25.01.2026).
7. Перегуда Ю. А., Чалюк Ю. О. Штучний інтелект як фактор структурних змін у глобальній економіці в умовах ентропії: виклики та можливості. Актуальні проблеми сталого розвитку. 2026. Т. 3, № 3. С. 105–113. DOI: 10.60022/3(3)-13S.
8. Перегуда Ю. А. Цифрові двійники як інструмент підвищення економічної ефективності управління міською інфраструктурою. Управління змінами та інновації. 2025. № 15. С. 48–55. DOI: 10.32782/CMI/2025-15-7.