DOI: https://doi.org/10.32782/2524-0072/2026-85-85

УДК 004.4:339.138

ТРАНСФОРМАЦІЯ ПРОЦЕСІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПІД ВПЛИВОМ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

TRANSFORMATION OF SOFTWARE DEVELOPMENT PROCESS UNDER THE INFLUENCE OF ARTIFICIAL INTELLIGENCE TECHNOLOGIES

Деордиця Олексій Олександрович
здобувач вищої освіти за третім аспірантським рівнем,
Державний вищий навчальний заклад
«Приазовський державний технічний університет»
ORCID: https://orcid.org/0009-0007-4613-2323

Deordytsia Oleksii
State Higher Education Institution «Pryazovskyi State Technical University»

У статті досліджено трансформацію процесів розробки програмного забезпечення під впливом технологій штучного інтелекту. Проаналізовано зміни у структурі життєвого циклу розробки програмного забезпечення та формування концепцій AI-Enhanced Software Development Life Cycle і AI-Driven Software Engineering. Розглянуто використання інтелектуальних інструментів програмування, систем автоматизованої генерації програмного коду, інтелектуального аналізу та тестування програмного забезпечення. Визначено трансформацію професійних ролей у розробницьких командах та формування нових моделей взаємодії між людиною та штучним інтелектом. Охарактеризовано вплив технологій штучного інтелекту на продуктивність розробки програмного забезпечення та архітектурні підходи до побудови програмних систем.

Ключові слова: штучний інтелект, програмне забезпечення, MLOps, програмні системи, AI-Enhanced SDLC, AI-Driven Software Engineering.

The article is devoted to the study of the transformation of software development processes under the influence of artificial intelligence technologies. The paper analyzes current trends in the integration of intelligent algorithms into the software development life cycle and identifies key directions for the formation of new approaches to organizing engineering processes. The concepts of AI-Enhanced Software Development Life Cycle and AI-Driven Software Engineering are considered, which involve the systematic use of intelligent technologies at various stages of software product development. The application of artificial intelligence in requirements analysis, architectural design, code implementation, testing, and DevOps processes is characterized. Particular attention is paid to the use of intelligent programming tools, automated code generation systems, and software module analysis tools that contribute to increased developer productivity, reduced implementation time, and improved software quality. The transformation of professional roles within development teams and the emergence of new models of interaction between humans and artificial intelligence are also examined, reflected in changes in the nature of engineering activities and the redistribution of functions among team members. It is shown that the spread of intelligent technologies affects the organization of development teams, team structures, and the efficiency of engineering processes. The influence of artificial intelligence on architectural approaches to software system design is analyzed, particularly the development of AI-native applications, data-centric architectures, and the integration of ML pipelines and MLOps practices into modern software ecosystems. It is determined that the further development of AI-oriented software engineering is associated with the formation of new approaches to software development automation, the use of autonomous software solutions, and the integration of artificial intelligence into software lifecycle management processes, as well as improving development efficiency and adaptability of modern software systems.

Keywords: artificial intelligence, software, MLOps, software systems, AI-Enhanced SDLC, AI-Driven Software Engineering.

Постановка проблеми. Стрімкий розвиток технологій штучного інтелекту суттєво трансформує сучасні підходи до розробки програмного забезпечення, змінюючи логіку організації процесів створення, тестування та супроводу програмних систем. Інтеграція інтелектуальних алгоритмів у середовище розробки, використання систем автоматизованої генерації коду та інструментів підтримки прийняття рішень створюють можливості для підвищення продуктивності розробників, оптимізації життєвого циклу програмного забезпечення та покращення якості програмних продуктів. Зокрема, інструменти здатні виконувати частину завдань програмування, включаючи генерацію фрагментів коду, аналіз помилок і автоматичне документування.

Водночас активне впровадження технологій штучного інтелекту супроводжується новими викликами, серед яких зміна структури життєвого циклу програмного забезпечення, трансформація ролей учасників розробницьких команд та підвищені вимоги до контролю якості й безпеки програмного коду, згенерованого інтелектуальними системами. Крім того, використання генеративних моделей може призводити до появи прихованих помилок або вразливостей, що потребує розроблення нових підходів до валідації результатів автоматизованої генерації коду. Попри зростання кількості досліджень у цій сфері, у науковій літературі недостатньо комплексно висвітлено питання трансформації процесів розробки програмного забезпечення під впливом технологій штучного інтелекту, що зумовлює необхідність подальших досліджень у цьому напрямі.

Аналіз останніх досліджень і публікацій. У наукових працях вітчизняних і зарубіжних дослідників питання використання технологій штучного інтелекту у розробці програмного забезпечення розглядається з позицій автоматизації інженерних процесів та підвищення ефективності створення програмних систем. Зокрема, у працях T. Abbas, A. Turki, S.A. Rathore, K. Sunawar, O. Alghushairy, A. Daud [2], M. Alenezi і M. Akour [3], а також P. Kokol [8] досліджуються сучасні напрями інтеграції штучного інтелекту у software engineering та визначаються основні виклики і перспективи розвитку AI-орієнтованих підходів до розробки програмного забезпечення. Питання трансформації процесів розробки програмних систем під впливом інтелектуальних технологій розглядаються у роботі M. Hernández Bejarano та L. E. Baquero-Rey [5]. Окрему групу досліджень становлять роботи, присвячені використанню штучного інтелекту у тестуванні, забезпеченні якості та безпеки програмного забезпечення, зокрема праці A. Escalante-Viteri, D. Mauricio [1] та J. R. Laracy, Z. Meng, V.D. Kirova, C.S. Ku, T.J. Marlowe [9]. В українських дослідженнях питання застосування штучного інтелекту в технологіях розробки програмного забезпечення висвітлюються у працях В. Соколова, В. Рябцева, О. Успенського, Д. Копича [4], а також С. Турлакової і Б. Логвіненка [6]. Водночас, попри значну кількість наукових публікацій, питання комплексної трансформації процесів розробки програмного забезпечення під впливом технологій штучного інтелекту потребує подальшого наукового дослідження.

Формулювання цілей статті. Метою статті є дослідження трансформації процесів розробки програмного забезпечення під впливом технологій штучного інтелекту та визначення основних напрямів розвитку інтелектуально орієнтованої інженерії програмного забезпечення. Для досягнення поставленої мети визначено такі **завдання дослідження**:

– проаналізувати вплив технологій штучного інтелекту на життєвий цикл розробки програмного забезпечення;

– охарактеризувати особливості формування концепцій AI-Enhanced SDLC та AI-Driven Software Engineering;

– дослідити трансформацію професійних ролей у командах розробки та моделей взаємодії між людиною і штучним інтелектом;

– визначити вплив інтелектуальних технологій на продуктивність розробки та архітектурні підходи до побудови програмних систем;

– окреслити перспективні напрями розвитку інтелектуально орієнтованої інженерії програмного забезпечення.

Виклад основного матеріалу дослідження. Сучасний розвиток інженерії програмного забезпечення характеризується активною інтеграцією технологій штучного інтелекту в процеси створення та супроводу програмних систем. У цьому контексті традиційна модель життєвого циклу розробки програмного забезпечення (Software Development Life Cycle, SDLC) зазнає суттєвої трансформації, оскільки значна частина інженерних процесів поступово автоматизується за допомогою інтелектуальних алгоритмів. Це сприяє формуванню концепції *AI-Enhanced SDLC*, яка передбачає системне застосування технологій штучного інтелекту на різних етапах

розробки з метою підвищення ефективності інженерних процесів та оптимізації управління програмними проєктами.

На етапі аналізу вимог (Requirements Engineering) технології штучного інтелекту використовуються для інтелектуального опрацювання технічної документації та текстових специфікацій. Методи обробки природної мови (Natural Language Processing, NLP) дозволяють автоматизувати структурування вимог, виявляти семантичні залежності між ними та ідентифікувати потенційні суперечності. Крім того, алгоритми машинного навчання можуть застосовуватися для прогнозування ризиків реалізації функціональних вимог на ранніх етапах проєкту [2, с. 7].

На етапі архітектурного проєктування штучний інтелект використовується як інструмент підтримки прийняття архітектурних рішень. На основі аналізу вимог до системи інтелектуальні алгоритми можуть пропонувати оптимальні архітектурні підходи, взаємодію між компонентами та відповідні шаблони проєктування (design patterns), а також генерувати проєктну документацію, зокрема UML-діаграми.

Найбільш помітний вплив штучного інтелекту спостерігається на етапі реалізації програмного коду. Сучасні інструменти підтримки програмування, зокрема GitHub Copilot, ChatGPT та Amazon CodeWhisperer, використовують великомасштабні мовні моделі для генерації фрагментів коду, автоматичного завершення програмних конструкцій та пропонування альтернативних алгоритмічних рішень. Це сприяє підвищенню продуктивності розробників та скороченню часу створення програмних модулів.

Технології штучного інтелекту активно застосовуються і на етапі тестування програмного забезпечення, зокрема для автоматизованої генерації тестових сценаріїв, аналізу програмного коду та прогнозування дефектів. Такі підходи дозволяють оптимізувати процес тестування та підвищити ефективність виявлення помилок у складних програмних системах.

Важливим напрямом є також інтеграція штучного інтелекту у практики DevOps, зокрема у процеси безперервної інтеграції (Continuous Integration, CI) та безперервного розгортання (Continuous Deployment, CD). Алгоритми машинного навчання можуть використовуватися для аналізу функціонування систем, прогнозування збоїв та автоматичного виявлення аномалій у продакшн-середовищі.

Таким чином, інтеграція технологій штучного інтелекту у різні етапи життєвого циклу розробки програмного забезпечення формує нову парадигму інженерії програмного забезпечення, у межах якої інтелектуальні системи виступають інструментом підтримки інженерних рішень та підвищення ефективності створення сучасних програмних систем [9, с. 30].

Логічним продовженням аналізу трансформації життєвого циклу розробки програмного забезпечення під впливом технологій штучного інтелекту є формування нового напряму в межах інженерії програмного забезпечення, який у сучасній науковій літературі визначається як AI-Driven Software Engineering. На відміну від традиційних підходів, де інтелектуальні інструменти виконували допоміжні функції, сучасні системи штучного інтелекту дедалі активніше інтегруються у ключові процеси розробки, виступаючи повноцінним елементом інженерної екосистеми створення програмних систем.

У межах цієї концепції штучний інтелект використовується не лише для автоматизації окремих технічних операцій, а й для інтелектуальної підтримки широкого спектра інженерних завдань. Одним із найбільш поширених напрямів є інтелектуальна підтримка програмування (AI-assisted coding). Сучасні інструменти на основі великих мовних моделей здатні аналізувати контекст програмного коду, генерувати програмні конструкції та пропонувати варіанти реалізації алгоритмів, що сприяє підвищенню продуктивності розробників і зменшенню кількості рутинних операцій.

Важливим напрямом розвитку AI-Driven Software Engineering є також автоматизоване налагодження програмного забезпечення (automated debugging). Алгоритми машинного навчання дозволяють аналізувати структуру програмного коду, виявляти потенційні помилки та пропонувати варіанти їх усунення, що сприяє підвищенню якості програмного забезпечення. Подібні підходи застосовуються і в системах рекомендацій щодо рефакторингу коду (code refactoring recommendation), які допомагають оптимізувати структуру програмних модулів, підвищити підтримуваність системи та зменшити технічний борг [2, с. 13].

Крім того, технології штучного інтелекту використовуються для генерації програмної документації (intelligent documentation generation) та інтелектуального аналізу програмного коду (AI-based code review). Такі системи здатні автоматично формувати опис

функціональності програмних модулів, перевіряти код на відповідність стандартам програмування та виявляти потенційні вразливості безпеки.

Загалом розвиток AI-Driven Software Engineering свідчить про трансформацію традиційної моделі створення програмного забезпечення. Якщо раніше процес розробки був переважно орієнтований на діяльність програміста, то нині формується нова парадигма - перехід від developer-centric development до AI-augmented development, у межах якої штучний інтелект виступає активним учасником інженерного процесу та інструментом підтримки прийняття технічних рішень [1, с. 717].

Подальший розвиток технологій штучного інтелекту у сфері інженерії програмного забезпечення зумовлює не лише трансформацію технологічних процесів розробки, але й формування нових моделей взаємодії між людиною та інтелектуальними системами. У цьому контексті дедалі більшого значення набуває концепція співпраці людини та штучного інтелекту (Human-AI collaboration), яка передбачає інтеграцію інтелектуальних систем у процеси прийняття інженерних рішень. На відміну від традиційних підходів, у межах яких програмні інструменти виконували переважно допоміжні функції, сучасні системи штучного інтелекту поступово стають активними елементами середовища розробки програмного забезпечення.

Інтеграція інтелектуальних технологій у процеси розробки програмного забезпечення спричиняє поступову трансформацію професійних ролей учасників розробницьких команд. Автоматизація значної частини рутинних інженерних операцій, зокрема генерації програмного коду, аналізу програмних помилок та створення тестових сценаріїв, призводить до зміни функціонального навантаження фахівців та появи нових моделей розподілу відповідальності між людиною та інтелектуальними системами.

У науковій літературі такі трансформаційні процеси дедалі частіше описуються як перехід до нових професійних ролей, орієнтованих на координацію взаємодії з інтелектуальними системами, аналіз результатів їхньої роботи та забезпечення якості програмних рішень. Основні зміни у професійних ролях розробників програмного забезпечення в умовах інтеграції технологій штучного інтелекту наведено у таблиці 1.

Як видно з таблиці 1, інтеграція технологій штучного інтелекту у процеси розробки програмного забезпечення поступово змінює характер професійної діяльності фахівців у сфері програмної інженерії. У нових умовах значна частина їхньої роботи пов’язана не лише зі створенням програмного коду, але й з управлінням взаємодією з інтелектуальними системами, аналізом результатів їх функціонування та забезпеченням надійності програмних рішень.

Таблиця 1

Трансформація професійних ролей у процесах розробки програмного забезпечення під впливом технологій штучного інтелекту

Традиційна роль у розробці програмного забезпечення	Трансформована роль у середовищі AI-орієнтованої розробки	Основні функції в умовах використання технологій штучного інтелекту
Програміст	AI-orchestrator	Формування запитів до інтелектуальних систем, координація процесу генерації програмного коду, інтеграція згенерованих фрагментів у програмну систему, контроль коректності отриманих результатів
Тестувальник	AI-validation specialist	Перевірка результатів автоматизованого тестування, валідація тестів, згенерованих інтелектуальними системами, аналіз потенційних дефектів та забезпечення якості програмного забезпечення
Архітектор програмного забезпечення	AI-supported architect	Використання інтелектуальних систем для аналізу альтернативних архітектурних рішень, оцінювання архітектурних сценаріїв та підтримка прийняття складних архітектурних рішень

Джерело: сформовано автором на основі [5]

Крім трансформації професійних ролей, використання технологій штучного інтелекту може впливати на організаційну структуру команд розробки програмного забезпечення. Зокрема, інтеграція інтелектуальних інструментів програмування створює передумови для перегляду традиційних моделей розподілу завдань між учасниками команди, змінює співвідношення між досвідченими фахівцями та розробниками початкового рівня, а також впливає на загальну продуктивність розробницьких колективів.

У зв'язку з цим особливої актуальності набуває дослідження низки питань, пов'язаних із впливом технологій штучного інтелекту на організацію інженерної діяльності у сфері програмної інженерії, зокрема:

– яким чином інтеграція інтелектуальних інструментів змінює структуру команд розробки програмного забезпечення;

– чи призводить використання систем штучного інтелекту до зменшення потреби у junior-розробниках;

– яким є вплив інтелектуальних інструментів на продуктивність розробницьких команд.

Таким чином, формування нових моделей взаємодії між людиною та штучним інтелектом є важливим напрямом сучасних досліджень у галузі інженерії програмного забезпечення, оскільки дозволяє глибше зрозуміти трансформаційні процеси, що відбуваються у середовищі розробки програмних систем під впливом технологій штучного інтелекту.

Одним із ключових напрямів сучасних досліджень у галузі інженерії програмного забезпечення є оцінювання впливу технологій штучного інтелекту на продуктивність процесів розробки програмних систем. Активне впровадження інтелектуальних інструментів у середовище розробки програмного забезпечення створює передумови для суттєвого підвищення ефективності інженерної діяльності, зокрема за рахунок автоматизації рутинних операцій, інтелектуальної підтримки програмування та оптимізації процесів тестування і супроводу програмних продуктів.

Застосування систем штучного інтелекту у процесах розробки програмного забезпечення потенційно сприяє скороченню часу реалізації програмних рішень, оскільки значна частина операцій, пов'язаних із написанням програмного коду, аналізом документації або створенням тестів, може виконуватися автоматизовано. Інтелектуальні інструменти програмування дозволяють розробникам швидше формувати програмні конструкції, генерувати шаблонні фрагменти коду та отримувати рекомендації щодо реалізації алгоритмічних рішень, що в цілому скорочує тривалість виконання окремих етапів життєвого циклу програмного забезпечення.

Крім того, використання технологій штучного інтелекту може сприяти зменшенню кількості програмних помилок за рахунок автоматизованого аналізу програмного коду та раннього виявлення потенційних дефектів. Інтелектуальні системи здатні ідентифікувати типові помилки програмування, виявляти невідповідності стандартам програмування та сигналізувати про можливі вразливості безпеки ще на етапі розробки програмного коду. Це дозволяє підвищити загальну надійність програмних систем та зменшити витрати на подальше виправлення дефектів.

Окремим аспектом дослідження є вплив інтелектуальних асистентів програмування на якість програмного коду (code quality). Сучасні системи підтримки розробки програмного забезпечення здатні аналізувати структуру програмного коду, пропонувати більш оптимальні реалізації алгоритмів, а також рекомендувати застосування відповідних архітектурних підходів або шаблонів проєктування. У результаті використання таких інструментів може сприяти підвищенню читабельності програмного коду, покращенню його структурної організації та зменшенню технічного боргу у програмних проєктах [7, с. 134-135].

Для об'єктивного оцінювання впливу технологій штучного інтелекту на ефективність розробки програмного забезпечення у наукових дослідженнях широко застосовуються різноманітні метрики продуктивності програмної інженерії. Серед найбільш поширених показників можна виділити такі:

– lead time – час, необхідний для реалізації функціональної зміни від моменту формулювання вимоги до її впровадження у програмну систему;

– defect density – показник щільності дефектів програмного коду, що відображає кількість помилок на одиницю обсягу програмного коду;

– development velocity – швидкість виконання розробницьких завдань, яка характеризує продуктивність команди у межах ітерацій розробки;

– cognitive load developer – рівень когнітивного навантаження на розробника під час виконання інженерних завдань.

Застосування зазначених метрик дозволяє комплексно оцінити вплив інтелектуальних технологій на продуктивність розробницьких команд, якість програмного коду та ефективність реалізації програмних проєктів. У цьому контексті дослідження впливу технологій штучного інтелекту на продуктивність розробки програмного забезпечення набуває особливої актуальності, оскільки дозволяє сформувати науково обґрунтоване уявлення про переваги та обмеження інтеграції інтелектуальних систем у сучасні інженерні практики [3, с. 1344].

Окрім впливу на продуктивність інженерної діяльності, технології штучного інтелекту спричиняють також суттєві зміни в архітектурних підходах до побудови програмних систем. На відміну від традиційних програмних архітектур, у яких інтелектуальні модулі виступали допоміжними компонентами, сучасні програмні системи дедалі частіше проєктуються як AI-native applications, де алгоритми машинного навчання інтегруються у базову функціональну логіку системи. Такий підхід передбачає переорієнтацію архітектурного проєктування на data-centric architectures, у межах яких ключову роль відіграють механізми збору, обробки та управління даними, необхідними для навчання і функціонування інтелектуальних моделей.

Важливим елементом сучасних AI-орієнтованих архітектур є інтеграція ML pipelines, що автоматизують підготовку даних, навчання, тестування та розгортання моделей машинного навчання. У цьому контексті ключову роль відіграє концепція MLOps, яка поєднує принципи інженерії програмного забезпечення з управлінням життєвим циклом моделей та забезпечує їх безперервне оновлення і моніторинг.

Архітектурна еволюція програмних систем також пов’язана з використанням microservices та serverless-обчислень, що забезпечують гнучке масштабування інтелектуальних компонентів. У таких умовах формується модель AI pipelines, яка інтегрує обробку даних, моделі машинного навчання та прикладну логіку програмної системи [4, с. 226-227].

Подальший розвиток технологій штучного інтелекту створює передумови для формування нових парадигм інженерії програмного забезпечення, у межах яких інтелектуальні системи відіграватимуть дедалі більш активну роль у процесах створення, оптимізації та супроводу програмних систем. Одним із перспективних напрямів є концепція autonomous software development, яка передбачає високий рівень автоматизації процесів розробки програмного забезпечення за допомогою інтелектуальних систем, здатних самостійно генерувати програмні рішення на основі аналізу вимог, даних та контексту використання програмної системи.

Значну увагу в сучасних дослідженнях приділено розвитку self-healing systems - програмних систем, здатних автоматично виявляти та усувати збої у процесі експлуатації. Використання алгоритмів машинного навчання для аналізу поведінки програмних систем дозволяє прогнозувати можливі відмови, автоматично коригувати конфігурацію системи та забезпечувати її стабільне функціонування без безпосереднього втручання людини [6, с. 32].

Ще одним перспективним напрямом є формування підходів до AI-generated architectures, у межах яких інтелектуальні системи можуть аналізувати функціональні вимоги до програмного продукту та генерувати оптимальні архітектурні рішення. Такі системи потенційно здатні оцінювати різні архітектурні сценарії, визначати найбільш ефективні структурні конфігурації програмних систем та пропонувати оптимальні архітектурні патерни для реалізації складних програмних продуктів/

Окремим напрямом еволюції інженерії програмного забезпечення є розвиток концепції AI-driven DevOps, яка передбачає використання інтелектуальних алгоритмів для оптимізації процесів безперервної інтеграції, тестування, розгортання та моніторингу програмних систем. У межах такого підходу штучний інтелект може використовуватися для прогнозування збоїв у програмному середовищі, автоматичного масштабування ресурсів, оптимізації процесів розгортання та підвищення загальної ефективності управління програмною інфраструктурою [8, с. 354].

Таким чином, подальша еволюція AI-орієнтованої інженерії програмного забезпечення пов’язана з поступовим переходом до інтелектуально керованих систем розробки та експлуатації програмних продуктів, у межах яких штучний інтелект виступає важливим елементом підтримки прийняття інженерних рішень, автоматизації складних процесів та підвищення адаптивності програмних систем.

Висновки. Отже, технології штучного інтелекту суттєво трансформують сучасну інженерію програмного забезпечення, впливаючи

на організацію життєвого циклу програмних систем, підходи до розробки, тестування та супроводу програмного забезпечення. Їх інтеграція у процеси розробки сприяє формуванню концепції AI-Enhanced Software Development Life Cycle, у межах якої інтелектуальні алгоритми застосовуються на різних етапах створення програмних продуктів.

Поширення інтелектуальних інструментів програмування формує напрям AI-Driven Software Engineering, що характеризується використанням систем генерації програмного коду, інтелектуального аналізу та автоматизованого тестування. Це супроводжується змінами у моделях взаємодії розробницьких команд, підвищенням продуктивності розробки та трансформацією архітектурних підходів до побудови програмних систем, зокрема розвитком AI-native applications, data-centric architectures та практик MLOps.

Подальші дослідження можуть бути спрямовані на аналіз ефективності використання інтелектуальних інструментів у розробці програмного забезпечення, вивчення моделей взаємодії між людиною та штучним інтелектом у розробницьких командах, а також розвиток підходів до створення автономних програмних систем, self-healing software та інтелектуально орієнтованих архітектур програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Escalante-Viteri A., Mauricio D. Artificial Intelligence in Software Testing: A Systematic Review of a Decade of Evolution and Taxonomy. *Algorithms.* 2025. № 18 (11). P. 717. DOI: https://doi.org/10.3390/a18110717

2. Abbas T., Rathore S. A., Turki A., Sunawar K., Alghushairy O., Daud A. Enhancing Software Engineering With AI: Innovations, Challenges and Future Directions. *IET Software.* 2025. P. 25. DOI: https://doi.org/10.1049/sfw2/5691460

3. Alenezi M., Akour M. AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions. *Applied Sciences.* 2025. № 15 (3). P. 1344. DOI: https://doi.org/10.3390/app15031344

4. Соколов В., Рябцев В., Успенський О., Копич Д. Напрями застосування штучного інтелекту в технологіях розробки програмного забезпечення. *Collection «Information Technology and Security»*. 2024. № 12 (2). С. 219-235. DOI: https://doi.org/10.20535/2411-1031.2024.12.2.315741

5. Hernández Bejarano M., Baquero-Rey L. E. AI-Driven Software Development: A Paradigm Shift in Software Engineering. *Industrial Engineering and Management. IntechOpen.* 2025. DOI: https://doi.org/10.5772/intechopen.1007786

6. Turlakova S., Lohvinenko B. Artificial intelligence tools for managing the behavior of economic agents at micro level. *Neuro-Fuzzy Modeling Techniques in Economics.* 2023. № 12. P. 3-39. DOI: http://doi.org/10.33111/nfmte.2023.003

7. Borodiyenko O. Opportunities and Risks of Using AI-Based Applications in Research: The Case of Ukrainian Universities. *Information Technologies and Learning Tools.* 2025. № 105 (1). P. 125-143. DOI: https://doi.org/10.33407/itlt.v105i1.5794

8. Kokol P. The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature. *Information.* 2024. № 15 (6). P. 354. DOI: https://doi.org/10.3390/info15060354

9. Laracy J. R., Meng Z., Kirova V. D., Ku C. S., Marlowe T. J. Software Quality Assurance and AI: A Systems-Theoretic Approach to Reliability, Safety, and Security. *Software.* 2025. № 4 (4). P. 30. DOI: https://doi.org/10.3390/software4040030

REFERENCES:

1. Escalante-Viteri A., Mauricio D. (2025). Artificial Intelligence in Software Testing: A Systematic Review of a Decade of Evolution and Taxonomy. *Algorithms***,** 18(11), 717. DOI: https://doi.org /10.3390/a18110717

2. Abbas T., Rathore S. A., Turki A., Sunawar K., Alghushairy O., Daud A. (2025). Enhancing Software Engineering With AI: Innovations, Challenges, and Future Directions. *IET Software*, 25. DOI: https://doi.org/10.1049/sfw2/5691460

3. Alenezi M., Akour M. (2025). AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions. *Applied Sciences*, 15(3), 1344. DOI: https://doi.org/10.3390/app15031344

4. Sokolov V., Riabtsev V., Uspenskyi O., Kopych D. (2024). Napriamy zastosuvannia shtuchnoho intelektu v tekhnolohiiakh rozrobky prohramnoho zabezpechennia [Directions of application of artificial intelligence in software development technologies]. *Information Technology and Security*, 12(2), 219-235. DOI: https://doi.org/10.20535/2411-1031.2024.12.2.315741

5. Hernández Bejarano M., Baquero-Rey L. E. (2025). AI-Driven Software Development: A Paradigm Shift in Software Engineering. *Industrial Engineering and Management*. DOI: https://doi.org/10.5772/intechopen.1007786

6. Turlakova S., Lohvinenko B. (2023). Artificial intelligence tools for managing the behavior of economic agents at micro level. *Neuro-Fuzzy Modeling Techniques in Economics*, 12, 3-39. DOI: https://doi.org/10.33111/nfmte.2023.003

7. Borodiyenko O. (2025). Opportunities and Risks of Using AI-Based Applications in Research: The Case of Ukrainian Universities. *Information Technologies and Learning Tools*, 105(1), 125-143. DOI: https://doi.org/10.33407/itlt.v105i1.5794

8. Kokol P. (2024). The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature. *Information*, 15(6), 354. DOI: https://doi.org/10.3390/info15060354

9. Laracy J. R., Meng Z., Kirova V. D., Ku C. S., Marlowe T. J. (2025). Software Quality Assurance and AI: A Systems-Theoretic Approach to Reliability, Safety and Security. *Software*, 4(4), 30. DOI: https://doi.org/10.3390/software4040030

Дата надходження статті: 03.04.2026
Дата прийняття статті: 24.04.2026
Дата публікації статті: 30.04.2026