DOI: https://doi.org/10.32782/2524-0072/2025-80-113

УДК 005.94:004.8

МОВНІ МОДЕЛІ ДЛЯ АВТОМАТИЗАЦІЇ ДОКУМЕНТУВАННЯ ТА ПІДТРИМКИ УПРАВЛІННЯ ЗНАННЯМИ У ВІДДАЛЕНИХ КОМАНДАХ IT-КОМПАНІЙ

LANGUAGE MODELS FOR AUTOMATING DOCUMENTATION AND SUPPORTING KNOWLEDGE MANAGEMENT IN REMOTE TEAMS IN IT-COMPANIES

Румик Ігор Іванович
доктор економічних наук, професор,
Вищий навчальний заклад «Університет економіки та права "КРОК»
ORCID: https://orcid.org/0000-0003-3943-639X

Rumyk Ihor
«KROK» University

У статті досліджено застосування великих мовних моделей (LLM) для автоматизації створення технічної документації та підтримки управління знаннями у віддалених IT-командах. Дослідження проведено за допомогою методу систематичного аналізу наукових публікацій для огляду сучасних підходів інтеграції ШІ у процеси документування, методу типології для класифікації підходів інтеграції, а також методу моделювання для розробки моделі впровадження RAG-архітектури. Розглянуто ключові напрями інтеграції ШІ у процеси документування, проаналізовано сучасні наукові підходи та наведено практичні рекомендації щодо впровадження технологій штучного інтелекту у життєвий цикл розробки програмного забезпечення. Особливу увагу приділено архітектурі Retrieval-Augmented Generation (RAG) як ефективному способу побудови внутрішніх систем підтримки знань. Показано, що використання LLM сприяє підвищенню ефективності командної роботи, стандартизації документації та оптимізації процесів комунікації в умовах віддаленої роботи.

Ключові слова: мовні моделі, управління IT-компанією, документування, управління знаннями, віддалена робота, штучний інтелект, цифровізація.

The article examines the potential of Large Language Models (LLMs) for automating technical documentation and supporting organizational knowledge management processes in IT projects, particularly those implemented within remote and distributed work environments. The growing complexity of modern software systems and the inherent challenges of collaboration across different time zones necessitate new approaches to standardize documentation practices and enhance internal communication efficiency. The relevance of this study lies in addressing the critical need for reliable mechanisms to capture, update, and retrieve expertise within decentralized teams, thus mitigating knowledge silos and reducing cognitive load on developers. The research employed a structured methodology, combining the method of systematic literature analysis to comprehensively review contemporary AI integration approaches in documentation workflows, the typological method to categorize and define distinct models of AI-assisted software engineering, and the modeling method to propose an implementable framework based on the Retrieval-Augmented Generation (RAG) architecture. The core findings highlight how AI technologies optimize the Software Development Life Cycle (SDLC), specifically by automating initial drafts, ensuring consistency between code and documentation, and assisting in the coordination among remote team members. Special attention is given to the RAG architecture as an efficient way to build internal, context-aware knowledge systems, allowing LLMs to access and synthesize information from an organization's proprietary codebase and internal documentation. The proposed implementation model strategically positions LLMs not as a definitive replacement for human expertise but as a cognitive augmentation tool that handles repetitive, data-intensive tasks. This approach enables human technical writers and engineers to focus on strategic validation, contextual accuracy, and quality assurance of the generated content. Ultimately, the use of AI-driven language models is demonstrated to accelerate knowledge transfer, improve documentation quality and consistency, and enhance overall project transparency and collaborative productivity in the demanding context of remote IT management.

Keywords: language models, IT-companies management, documentation, knowledge management, remote work, artificial intelligence, digitalization.

Постановка проблеми. Ефективна співпраця в команді є критичною для успіху будь-якого проєкту життєвого циклу розробки програмного забезпечення (SDLC), проте вона часто стикається з низкою викликів. Непорозуміння між розробниками, особливо в умовах віддаленої та розподіленої роботи команд, можуть призводити до помилкового трактування вимог або стану проєкту. Значну частину часу витрачають на створення та підтримку документації, яка часто є повторюваною та одноманітною, що знижує ефективність та мотивацію команди. Це підкреслює необхідність автоматизації процесу документування та надання інструментів, що полегшують роботу з документацією, сприяючи своєчасному та якісному виконанню проєктів.

Принципи організації віддаленої роботи в українських IT-компаніях сформувалися як системна практика, що пов'язана як із пандемією COVID-19, так і з військовими викликами на території України. Внаслідок цього віддалена співпраця перестала бути тимчасовим заходом і трансформувалася у сталу організаційну модель, що потребує перегляду підходів до управління командами та обігу знань. Такий стан справ потребує детального розгляду та наукового обґрунтування з позицій підвищення ефективності управління компаніями IT-сфери.

Аналіз останніх досліджень і публікацій. Під час повномасштабної війни IT-сектор демонстрував високу здатність до адаптації, зокрема завдяки впровадженню гнучких моделей роботи [1]. Завдяки таким стратегіям IT-сектор зміг зберегти продуктивність і продовжував надавати критично важливі послуги як для внутрішнього, так і для міжнародного ринку [2]. Тому питання управління знаннями всередині IT-команд, що працюють у віддаленому режимі, досліджувалося багатьма українськими та зарубіжними вченими.

Зокрема, Триус Ю.В. та Ткаченко Є.Ю. [3] підкреслювали важливість використання сучасних інформаційних технологій і спеціалізованого програмного забезпечення для ефективного управління, координації та комунікації всередині команди IT-проєкту в умовах віддаленої роботи. Це безпосередньо корелює з ідеєю використання мовних моделей як наступного кроку в автоматизації та інтеграції інформації, щоб мінімізувати необхідність у ручній роботі з документацією.

Ціла низка зарубіжних учених досліджували можливості використання великих мовних моделей (LLM) для автоматизації створення технічної документації до програмного забезпечення (Diggs та ін.) [4]. Автори розробили підхід до генерації коментарів та описів коду за допомогою LLM і запропонували критерії оцінювання якості автоматично створеної документації, серед яких повнота, читабельність, корисність та відсутність галюцинацій.

Інші зарубіжні дослідники провели порівняльний аналіз великих мовних моделей (LLM), зокрема GPT-3.5, GPT-4, Bard, Llama2 та StarChat, щодо їх здатності генерувати документацію до коду (Dvivedi та ін.) [5]. Результати показали, що більшість моделей здатні створювати таку документацію, яка за чіткістю та повнотою переважає оригінальні варіанти, написані людьми, зокрема на рівні коментарів до функцій.

Інтеграція мовних моделей у процеси командної роботи може покращити комунікацію, зменшити час на розв'язання технічних питань і підвищити загальну ефективність команди (Dhanuka) [6]. Зокрема, LLM можуть автоматизувати рутинні завдання, такі як генерація документації та управління проєктами, що дозволяє командам зосередитися на більш складних аспектах розробки. Wang та ін. [7] у своїх працях продемонстрували, як технології ШІ здатні генерувати технічні макети та специфікації, а також допомагати у визначенні та повторному використанні архітектурних рішень.

Виділення невирішених раніше частин загальної проблеми. Попри активний розвиток мовних моделей залишається недостатньо дослідженим питання їхньої інтеграції у внутрішні системи управління знаннями IT-компаній, особливо в умовах віддаленої роботи. Також потребує подальшого вивчення вплив використання LLM на ефективність командної взаємодії та збереження корпоративної експертизи.

Метою статті є розробка шляхів упровадження мовних моделей для вдосконалення процесів документування під час реалізації проєктів із залученням віддалених команд IT-компаній.

Виклад основного матеріалу дослідження. Поява потужних великих мовних моделей (LLM), таких як GPT-3/4/5 від OpenAI, Gemini від Google, LLaMA від Meta та ін., спричинила численні дослідження щодо їх застосування у сфері розробки програмного забезпечення. Ці моделі відкрили нові можливості для автоматизації та оптимізації процесів програмування, зокрема під час аналізу

вимог, проєктування архітектури, написання коду, тестування, документування та супроводу систем. Завдяки здатності обробляти природну мову, LLM забезпечують перехід від традиційного програмування до інтерактивної розробки за допомогою штучного інтелекту, що дозволяє значно підвищити продуктивність інженерів програмного забезпечення, скоротити кількість помилок і стандартизувати підхід до розробки [8-9].

Зазвичай життєвий цикл програмного забезпечення охоплює такі основні етапи: аналіз і формування вимог, проєктування та розроблення архітектури, реалізацію (програмування), тестування й налагодження, впровадження та експлуатацію, а також подальший супровід і технічну підтримку [10]. На кожному етапі формується специфічний набір документів, що забезпечує цілісність, прозорість і відтворюваність процесу розробки.

На етапі аналізу та формування вимог створюються бізнес- та користувацькі вимоги, функціональні й нефункціональні специфікації, а також сценарії використання (use cases). Під час проєктування архітектури розробляються технічні описи системи, архітектурні схеми, UML-діаграми, специфікації інтерфейсів та опис структур даних. Такі документи дозволяють узгодити бачення системи між аналітиками, архітекторами та розробниками.

На етапі реалізації створюються коментарі в коді, технічні описи модулів, API-документація, файли README та історія змін, які забезпечують зрозумілість і підтримуваність коду в довгостроковій перспективі. У процесі тестування й налагодження формується документація, що підтверджує якість продукту: тест-плани, тест-кейси, звіти про дефекти та підсумкові звіти про тестування. Під час впровадження та експлуатації створюються інструкції з інсталяції, керівництва користувача, адміністратора, а також навчальні матеріали.

На завершальному етапі – супроводу та технічної підтримки – ведуться журнали обслуговування, бази знань, звіти про інциденти та ретроспективи, що забезпечують накопичення досвіду й безперервне вдосконалення продукту.

Науковці активно вивчають, як великі мовні моделі можуть підтримувати або доповнювати різні етапи процесу розробки. Зокрема, LLM продемонстрували високу ефективність у генерації коду та створенні його документації [5-7].

У Таблиці 1 виділено шість основних сфер застосування штучного інтелекту (мовних моделей) для автоматизації документування в IT-проєктах.

Дослідження показують, що практичне впровадження мовних моделей для автоматизації процесів документування в IT-проєктах забезпечується за рахунок інтеграції таких моделей в існуючі робочі процеси розробки та управління знаннями. Ключовий підхід полягає у використанні LLM не як повного замінника технічного письменника чи інженера програмного забезпечення, а як потужного інструменту автоматизації рутини та інтелектуального пошуку, який підвищує продуктивність, мінімізує час на рутинні завдання та дозволяє фахівцям зосередитися на стратегічній валідації та забезпеченні високої якості кінцевого продукту.

Наразі доволі активно досліджуються питання інтеграції таких моделей у середовища розробки (IDE), управління знаннями в IT-компаніях, генерації технічної документації та підтримки прийняття рішень на основі даних, що свідчить про формування нового етапу еволюції інтелектуальних систем розробки програмного забезпечення.

Згадані напрямки сприяють підтримці прийняття рішень на основі даних, оскільки LLM можуть швидко аналізувати великі обсяги технічного боргу, вимог та метрик, пропонуючи зважені висновки. Це все свідчить про формування нового еволюційного етапу переходу документування до парадигми програмної інженерії з інтелектуальною підтримкою великими мовними моделями.

Крім того, LLM поступово інтегруються у системи контролю версій, тестування та CI/CD-процеси, де вони можуть автоматично генерувати опис змін, створювати звіти про тести або допомагати виявляти невідповідності між кодом і документацією. Це значно зменшує ризик помилок, пов’язаних із застарілою або неповною технічною інформацією.

У сфері управління знаннями мовні моделі стають основою для створення внутрішніх корпоративних асистентів, здатних швидко знаходити потрібну інформацію у великих базах даних та підтримувати процес навчання нових співробітників. Застосування LLM також сприяє стандартизації форматів документації, уніфікації термінології та підвищенню узгодженості між технічними і бізнес-вимогами, що в результаті покращує комунікацію між усіма учасниками IT-проєктів.

Таблиця 1

Сфери застосування штучного інтелекту для автоматизації документування в IT-компаніях

Сфера вдосконалення	Характеристики LLM	Переваги для IT-команди
1. Генерація контенту (чернетки)	*Автоматична генерація:* створює перші чернетки документації (наприклад, описи API, інструкції, технічні звіти) на основі коду, коментарів або структурованих даних	*Економія часу:* значно зменшує час, який розробники витрачають на написання документації з нуля
2. Актуальність та синхронізація	*Автоматичне оновлення:* аналізує зміни в системі контролю версій (VCS) (наприклад, Git-комміти) та автоматично генерує пропозиції щодо оновлення відповідних розділів документації.	*Підтримка актуальності:* документація завжди відповідає останній версії коду, усуваючи проблему застарілих інструкцій.
3. Послідовність та якість	*Стандартизація стилю:* забезпечує єдиний тон, стиль та термінологію в усіх документах, використовуючи корпоративні глосарії.	*Зменшення помилок:* підвищує якість і надійність документації завдяки уніфікації та перевірці на суперечності.
4. Зрозумілість для користувачів	*Спрощення мови:* перетворює складний технічний жаргон (наприклад, архітектурні діаграми, логи) на прості та зрозумілі пояснення для кінцевих користувачів або нетехнічних стейкхолдерів.	*Покращення доступу:* документація стає корисною для ширшої аудиторії (продажі, маркетинг, підтримка клієнтів).
5. Інтелектуальний пошук (RAG)	*Чат-бот на основі знань:* використовує моделі в архітектурі ***Retrieval-Augmented Generation***, щоб відповідати на запитання користувачів природною мовою, використовуючи внутрішню базу знань.	*Швидке вирішення проблем:* співробітники та клієнти отримують миттєві та точні відповіді, не переглядаючи сотні сторінок.
6. Узагальнення та резюмування	*Конденсація інформації:* стисло підсумовує довгі документи, протоколи нарад або звіти про інциденти, виділяючи ключові рішення та завдання.	*Підвищення ефективності:* команди можуть швидко засвоювати основну інформацію та зосереджуватися на дії.

Джерело: сформовано автором

На основі проведеного аналізу можна виділити наступні практичні кроки та моделі впровадження документування та обробки даних: застосування RAG-систем для управління внутрішньою базою знань та їх інтеграція у робочі процеси розробки.

1. Впровадження RAG-систем для внутрішньої бази знань. Найпоширенішим і водночас найбезпечнішим підходом до інтеграції великих мовних моделей (LLM) для документування є використання архітектури Retrieval-Augmented Generation (RAG) [11].

Процес починається з накопичення даних: вся внутрішня документація організації (Wiki, Confluence, Markdown-файли, технічні звіти, записи в Jira тощо) завантажується у систему. Документи розбиваються на невеликі фрагменти та перетворюються на числові вектори за допомогою спеціалізованої моделі. Отримані вектори зберігаються у векторній базі даних. При обробці користувацьких запитів RAG-система визначає найбільш релевантні документи з бази, після чого ці фрагменти разом із запитом передаються як контекст до LLM, яка формує відповідь, ґрунтуючись на представленій інформації.

2. Інтеграція в робочий процес розробки. LLM інтегруються безпосередньо в середовище розробки (IDE) та системи контролю версій для автоматичного створення і оновлення технічних описів (Таблиця 2).

Таблиця 2

Впровадження LLM в середовище розробки (IDE) в IT-компаніях

Завдання	Практичне впровадження	Приклади інструментів
Документація API/Коду	*Автоматична генерація коментарів та описів:* LLM підключається до системи контролю версій (наприклад, Git) і генерує чернетки технічних описів або оновлює документацію API щоразу, коли відбувається злиття.	GitHub Copilot, Amazon CodeWhisperer, спеціалізовані конвеєри, які генерують специфікації з коду.
Створення початкових чернеток	*Генерація структури та змісту:* технічний письменник надає LLM запит з основними пунктами чи функціоналом, а модель генерує повний шаблон чи чорновий текст.	Використання GPT-4 або Claude для створення чернеток.
Аналіз змін	*Автоматичне оновлення інструкцій:* LLM аналізує зміни коду і пропонує, які розділи внутрішньої документації потрібно змінити, створюючи «пропозицію оновлення» для рецензування.	Автоматизація інтегрована в Jira або GitLab/GitHub

Джерело: сформовано автором

Варто зазначити, що при реалізації даних підходів стандартною практикою має стати впровадження механізмів безпеки, щоб LLM не використовувала конфіденційні дані і не генерувала некоректну або невідповідну інформацію.

Висновки. Сучасні дослідження свідчать про високий потенціал великих мовних моделей та ШІ в автоматизації документації та підтримці командної взаємодії. Водночас існують обмеження, зокрема точність моделей залежить від складності коду та контексту, а автоматичні метрики поки що не можуть повністю замінити людську оцінку. Це підкреслює необхідність комплексного підходу при впровадженні LLM у роботу віддалених IT-команд, де якісна документація та ефективна комунікація є ключовими факторами успіху IT-компанії.

Результати дослідження свідчать, що практична імплементація великих мовних моделей у процеси IT-документування повинна розглядатися не як повна автоматизація, що замінює людську працю, а як створення гібридного, напівавтоматизованого робочого процесу. У цій парадигмі LLM ефективно виконують функції рутинної генерації та пошуку інформації, тоді як людські експерти зберігають ключову роль у перевірці, формуванні стратегії та забезпеченні кінцевої контекстуальної якості документації.

Перспективними напрямами подальших досліджень є розроблення моделей оцінювання якості автоматично згенерованої документації, а також створення гібридних систем, що поєднують когнітивні можливості ШІ з експертними знаннями фахівців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Куничак О. Українська IT-індустрія: робота в умовах війни та подальші перспективи. *European Business Association*. 2022. URL: https://eba.com.ua/ukrayinska-it-industriya-robota-v-umovah-vijny-ta-podalshi-perspektyvy/ (дата звернення 25.10.2025)

2. Makovoz O. & Buriak M. Agile Change Management of Ukraine's IT Sector's and Value Creation Amidst Conflict. *London Journal of Social Sciences*. 2024. Vol. 7, pp. 77-89. DOI: https://doi.org/10.31039/ljss.2024.7.189 (дата звернення 25.10.2025)

3. Триус Ю., Ткаченко Є. Особливості управління командою IT-проєкту в умовах віддаленої роботи. *Управління розвитком складних систем*. 2024. №60. С. 105-112. DOI: https://doi.org/10.32347/2412-9933.2024.60.105-112 (дата звернення 22.10.2025)

4. Diggs C. et al. Leveraging LLMs for Legacy Code Modernization: Challenges and Opportunities for LLM-Generated Documentation. 2024. DOI: 10.48550/arXiv.2411.14971 (дата звернення 21.10.2025)

5. Dvivedi S.S., Vijay V., Pujari S.L.R., Lodh S., & Kumar D. A Comparative Analysis of Large Language Models for Code Documentation Generation: *46th International Conference on Software Engineering*; April 2024; Lisbon, Portugal. URL: https://arxiv.org/html/2312.10349v1 (дата звернення 23.10.2025)

6. Dhanuka D. Impact of LLMs on Team Collaboration in Software Development. 2025. DOI: 10.48550/arXiv.2510.08612 (дата звернення 21.10.2025)

7. Wang L. et al. Large Action Models: From Inception to Implementation. 2025. DOI: 10.48550/arXiv.2412.10047 (дата звернення 23.10.2025)

8. Rasheed Z. et al. Large Language Models for Code Generation: The Practitioners Perspective. 2025. DOI: 10.48550/arXiv.2501.16998 (дата звернення 21.10.2025)

9. Fan A. et al. Large Language Models for Software Engineering: Survey and Open Problems. 2023. DOI: 10.48550/arXiv.2310.03533 (дата звернення 23.10.2025)

10. Сеспедес Гарсія Н., Сеспедес Гарсія П. Моделі життєвого циклу розробки програмного забезпечення. *Молодий вчений*. 2023. № 2 (114). С. 17-20. DOI: https://doi.org/10.32839/2304-5809/2023-2-114-4 (дата звернення 25.10.2025)

11. Lewis P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. 2020. DOI: 10.48550/arXiv.2005.11401 (дата звернення 23.10.2025)

REFERENCES:

1. Kunychak O. (2022) Ukrainska IT-industriia: robota v umovakh viiny ta podalshi perspektyvy [Ukrainian IT industry: work in war conditions and further prospects]. *European Business Association*. Available at: https://eba.com.ua/ukrayinska-it-industriya-robota-v-umovah-vijny-ta-podalshi-perspektyvy/ (accessed October 25, 2025)

2. Makovoz O. & Buriak M. (2024) Agile Change Management of Ukraine's IT Sector's and Value Creation Amidst Conflict. *London Journal of Social Sciences,* vol. 7, pp. 77-89. Available at: https://doi.org/10.31039/ljss.2024.7.189 (accessed October 25, 2025)

3. Tryus Yu., Tkachenko Ye. (2024) Osoblyvosti upravlinnia komandoiu IT-proiektu v umovakh viddalenoi roboty [Features of team management in conditions of remote work]. *Upravlinnia rozvytkom skladnykh system – Management of Development of Complex Systems*, vol. 60, pp. 105-112. Available at: https://doi.org/10.32347/2412-9933.2024.60.105-112 (accessed October 22, 2025)

4. Diggs C. et al. (2024) Leveraging LLMs for Legacy Code Modernization: Challenges and Opportunities for LLM-Generated Documentation. Available at: 10.48550/arXiv.2411.14971 (accessed October 21, 2025)

5. Dvivedi S.S., Vijay V., Pujari S.L.R., Lodh S. & Kumar D. (2024) A Comparative Analysis of Large Language Models for Code Documentation Generation: *46th International Conference on Software Engineering*; April 2024; Lisbon, Portugal. Available at: https://arxiv.org/html/2312.10349v1 (accessed October 23, 2025)

6. Dhanuka D. (2025) Impact of LLMs on Team Collaboration in Software Development. Available at: 10.48550/arXiv.2510.08612 (accessed October 21, 2025)

7. Wang L. et al. (2025) Large Action Models: From Inception to Implementation. Available at: 10.48550/arXiv.2412.10047 (accessed October 23, 2025)

8. Rasheed Z. et al. (2025) Large Language Models for Code Generation: The Practitioners Perspective. Available at: 10.48550/arXiv.2501.16998 (accessed October 21, 2025)

9. Fan A. et al. (2023) Large Language Models for Software Engineering: Survey and Open Problems. Available at: 10.48550/arXiv.2310.03533 (accessed October 23, 2025)

10. Sespedes Harsiia N., Sespedes Harsiia P. (2023) Modeli zhyttievoho tsyklu rozrobky prohramnoho zabezpechennia [Life cycle models of software development]. *Molodyi vchenyi – Young Scientist*, vol. 2(114), pp. 17-20. Available at: https://doi.org/10.32839/2304-5809/2023-2-114-4 (accessed October 25, 2025)

11. Lewis P. et al. (2020) Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Available at: 10.48550/arXiv.2005.11401 (accessed October 23, 2025)