МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

МІЖНАРОДНИЙ УНІВЕРСИТЕТ

МІЖНАРОДНА КОНФЕРЕНЦІЯ

ПЕРЕДОВІ ТЕХНОЛОГІЇ В ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНІЙ ІНЖЕНЕРІЇ

(ПТІКІ'2026)

Одеса, Україна, 16 липня 2026 р.

Матеріали конференції

ОДЕСА

2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

МІЖНАРОДНИЙ УНІВЕРСИТЕТ

МІЖНАРОДНА КОНФЕРЕНЦІЯ

«Передові технології в інформаційно-комунікаційній інженерії»

(ПТІКІ'2026)

Одеса, Україна, 16 липня 2026 р.

Матеріали конференції

Одеса

2026

УДК 004.9
ББК 32

Проведення заходу зареєстровано в Державній науковій установі «Український інститут науково-технічної експертизи та інформації» (Реєстраційний номер: 3326U000765 від 01-07-2026 р.).

Рецензенти:
Надія КАЗАКОВА – д.т.н., професор, завідувач кафедри інформаційних технологій Одеського національного університету ім. І.І. Мечнікова
Віктор СТРЕЛЬБИЦЬКИЙ – к.т.н., доцент, доцент кафедри підйомно-транспортні машини та інжиніринг портового технологічного обладнання Одеського національного морського університету

International Conference "Advanced Technology in Information and Communication Engineering" (ATICE'2026): Conference Proceedings. Odesa: International university, 2026, 145 p**.**
DOI: https://doi.org/10.32837/11300.33747

Міжнародна конференція «Передові технології в інформаційно-комунікаційній інженерії» (ATICE'2026): Матеріали конференції. Одеса: Міжнародний університет, 2026, 145 с.

У збірнику подано результати наукових досліджень, висвітлено обговорення актуальних проблем, викликів і тенденцій з інформаційних систем, технологій захисту інформації, використання сучасних інформаційних технологій в управлінні системами у різних галузях народного господарства. Матеріали охоплюють інноваційні підходи, досвід упровадження сучасних рішень і приклади успішних практик трансформації різних галузях народного господарства .

Видання призначене для науково-педагогічних працівників, здобувачів освіти, науковців і фахівців ІТ галузі.

Відповідальні за випуск:
к.т.н., доцент Пунченко Наталія,
к.т.н., доцент Розенвассер Денис.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова

КІВАЛОВ С.В. – д.т.н., проф., Міжнародний університет, м.Одеса, Україна.

Співголови

ГРОМОВЕНКО К.В. – д.ю.н., проф., Міжнародний університет, м. Одеса, Україна.
ЛЕФТЕРОВ В.О. – д.п.н., проф., Міжнародний університет, Одеса, Україна
СТРЕЛКОВСЬКА І.В. – д.т.н., проф., Міжнародний університет, Одеса, Україна.
УРИВСЬКИЙ Л.О. – д.т.н., проф., НТУ України "КПІ імені Ігоря Сікорського", Київ, Україна.

Члени організаційного комітету

СОЛОВСЬКА І.М. – к.т.н., доц., Міжнародний університет, Одеса, Україна.
ЙОНА Л.Г. – к.т.н., доц., Міжнародний університет, Одеса, Україна.
ГРИГОР'ЄВА Т.І. – к.т.н., доц., Міжнародний університет, Одеса, Україна.
РУСУ О.П. – к.т.н., доц., Міжнародний університет, Одеса, Україна.
РОЗЕНВАССЕР Д.М. – к.т.н., доц., Міжнародний університет, Одеса, Україна.
ВАКАРЧУК А.О. – к.т.н., доц., Міжнародний університет, Одеса, Україна.
ПУНЧЕНКО Н.О. – к.т.н., доц., Міжнародний університет, Одеса, Україна.
ГРІБОВА В.В. – к.ф.-м.н., доц., Міжнародний університет, Одеса, Україна.

РЕДАКЦІЙНИЙ КОМІТЕТ

СОЛОВСЬКА І.М. – к.т.н., доц., Міжнародний університет, Одеса, Україна.
РОЗЕНВАССЕР Д.М. – к.т.н., доц., Міжнародний університет, Одеса, Україна.
ПУНЧЕНКО Н.О. – к.т.н., доц., Міжнародний університет, Одеса, Україна.

КОМІТЕТ МІЖНАРОДНИХ ЗВ'ЯЗКІВ

СТРЕЛКОВСЬКА І.В. – д.т.н., проф., Міжнародний університет, м.Одеса, Україна.
ГЛОБА Л. С. – д.т.н., проф., Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м.Київ, Україна.

ВЧЕНИЙ СЕКРЕТАР

ПУНЧЕНКО Н.О. – к.т.н., доц., Міжнародний університет, Одеса, Україна.

ТЕХНІЧНО-ПРОГРАМНИЙ КОМІТЕТ

СТРЕЛКОВСЬКА І.В. – д.т.н., проф., Міжнародний університет, м.Одеса, Україна.
УРИВСЬКИЙ Л.О. – д.т.н., проф., Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м.Київ, Україна.
ГЛОБА Л.С. – д.т.н., проф., Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м.Київ, Україна.

ЗМІСТ

СЕКЦІЯ 3.
КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

УДК 004.8:004.432

DOI: https://doi.org/10.32837/11300.33756

ПОРІВНЯЛЬНЕ ДОСЛІДЖЕННЯ СТРАТЕГІЙ RAG-КОНТЕНТУ ДЛЯ ПІДВИЩЕННЯ ПОВНОТИ АВТОМАТИЗОВАНОГО CODE REVSEW

Проценко М.М.
здобувач вищої освіти третього (аспірантського рівня)
спеціальності F7 – Комп'ютерна інженерія
Міжнародний університет
місто Одеса, Україна
protsenko.mike@gmail.com
Науковий керівник: Мірошник М.А., д.техн.н., професор

Анотація. *Проведено порівняльне дослідження того, як RAG та агентний підхід підвищують повноту автоматизованого code review. Встановлено, що текстовий контекст дає більший приріст, ніж розширений код, і в промисловій системі піднімає якість review на 61.98%, репозиторний контекст дає 285% приросту покриття дефектів, а інструментальний забезпечує реакцію розробника на 38.70% зауважень за скорочення часу review на 30.8%. Показано, що агентні системи замінюють одноразовий пошук ітеративним дослідженням коду з перевіркою достатності контексту. Запропоновано двоетапний підхід: попередню гіпотезу про клас дефекту за самим diff і цілеспрямований добір контексту під цей клас із перевіркою достатності доказів.*

Автоматизоване code review полягає у перевірці змін у коді засобами мовної моделі з метою виявити дефект до його потрапляння в основну гілку. Якщо моделі подано лише змінені рядки, вона виявляє локальні дефекти, проте не розпізнає причину, що міститься в іншому файлі, у пов'язаній задачі чи в конфігурації. Тому здатність системи знаходити реальні дефекти визначається передусім тим, який контекст дібрано на перевірку. Способи такого добору, що відрізняються тим, з чого саме береться контекст (з тексту задачі, зі структури коду чи з дій зовнішніх інструментів [9]), об'єднують під поняттям retrieval-augmented generation (RAG), а їх розвитком є агентні системи, що самостійно досліджують репозиторій. Мета роботи полягає у з'ясуванні того, як саме пошук контексту та агентна поведінка підвищують повноту виявлення дефектів. Щоб результат був вимірюваним, одиницею оцінювання прийнято

підтверджений дефект: автори бенчмарку c-CRAB перетворюють зауваження рецензентів на виконувані тести, і дефект вважається знайденим, якщо після виправлення тест проходить [2].

Найпростіша форма RAG додає до зміни текст, що пояснює її намір: опис задачі, пов'язаний issue, обговорення в pull request. Перевірка тоді стосується не синтаксичної коректності, а відповідності реалізації задуму. Значущість цього контексту підтверджує бенчмарк ContextCRBench, що містить 67 910 записів із прив'язкою issue до pull request у понад 90 репозиторіях та 9 мовах. Його автори показують, що саме текстовий контекст, а не додатковий код, дає більший приріст якості [1]. У промисловому застосуванні цей ефект виражений кількісно: коли ContextCRBench використали як сигнал для самонавчальної системи review у ByteDance, якість зросла на 61.98% [1]. Подібний ефект дають приклади вже перевірених змін: коли моделі показують найбільш схожий випадок review з минулого, вона краще відтворює усталені зауваження команди, підвищуючи відтворення рідкісних, але змістовних формулювань на 24.01% [6], хоча, як буде видно далі, цей приріст стійкий лише за невеликої кількості таких прикладів. Отже, навіть простий текстовий RAG істотно підвищує здатність моделі співвідносити зміну з її призначенням.

Наступний рівень добору враховує структуру коду, а не лише текст. Тут важливо не лише вибрати потрібний фрагмент, а й зберегти його цілісність: метод cAST показує, що поділ коду за рядками розриває функції, тоді як поділ за синтаксичним деревом дає самодостатні фрагменти й підвищує точність вибору релевантного коду (Recall@5) на 4.3 пункти [10]. Саме від цього вибору залежить, чи побачить модель потрібний фрагмент під час перевірки. На рівні цілого репозиторію виграш ще помітніший. Бенчмарк AACR-Bench, який надає моделі граф залежностей між файлами та експертно розмічені приховані дефекти, фіксує 285% приросту покриття дефектів порівняно зі стандартними наборами [4], а SWR-Bench із 1000 pull request показує, що агрегування кількох незалежних проходів review підвищує F1 виявлення проблем до 43.67% [3]. Структурний і репозиторний контекст відкриває моделі ті дефекти, що виникають на межі модулів і недосяжні з самого diff.

Окрему групу становить інструментальний RAG, де моделі подають не текст, а результати засобів, що самостійно перевіряють код: статичного аналізатора, перевірки типів, тестів, покриття. У цьому випадку контекст є не підказкою, а доказом, і це найбільше наближає review до практичної користі. Розгортання RovoDev Code Reviewer у компанії Atlassian, де система працювала понад рік на більш ніж 1900 репозиторіях і згенерувала понад 54 000 зауважень, показало, що близько 38.70% її зауважень призвели до фактичної зміни коду, час проходження pull request

скоротився на 30.8%, а обсяг написаних людьми зауважень зменшився на 35.6% [7]. Ці показники відображають реальну реакцію розробників, а не подібність до зразка, і свідчать, що поєднання моделі з перевірювальними інструментами дає вимірюваний приріст продуктивності.

Спільним для всіх перелічених режимів є те, що контекст добирається один раз і подається моделі цілком. Агентний підхід знімає це обмеження: замість одноразового пошуку модель сама керує тим, що шукати далі, спираючись на вже знайдене. Показовим є AutoCodeRover: агент починає з ключових слів опису задачі, викликає пошукові операції над структурою коду, а далі переходить по викликах методів та означеннях класів, на кожному кроці уточнюючи, яких саме відомостей ще бракує, і поступово нарощуючи контекст. Перед формуванням висновку він окремо перевіряє, чи зібраного контексту достатньо [11]. Така цілеспрямована навігація дає змогу розв'язати близько 22% задач на наборі SWE-bench-lite, причому з помітно меншою витратою токенів, ніж у агентів без структурного пошуку, оскільки система швидше виходить на місце помилки [11]. Дослідники репозиторних бенчмарків незалежно фіксують той самий перехід: від подання моделі готових фрагментів контексту до активного дослідження коду, у якому система сама перевіряє власні припущення через кілька кроків взаємодії [4].

Ця самостійність помітна й у методології оцінювання AACR-Bench: для методів на основі подібності кількість фрагментів контексту задають наперед і фіксовано, тоді як агентному методу дозволено самому визначати, скільки контексту потрібно для конкретного pull request. За висновками авторів, саме вибір такої агентної парадигми поряд із рівнем деталізації контексту істотно впливає на результат, і цей вплив по-різному виявляється залежно від моделі та мови програмування [4]. Природним розвитком цієї ідеї є ітеративний пошук із перевіркою достатності доказів, коли система нарощує обсяг пошуку лише доти, доки наявних свідчень бракує для обґрунтування дефекту, і тим самим уникає зайвого контексту [12]. Таким чином агентна поведінка не лише розширює доступний контекст, а й керує його обсягом залежно від складності конкретного випадку.

Водночас більший контекст не є монотонно корисним, і це обмеження стосується як пасивного, так і агентного RAG. Та сама AACR-Bench виявляє, що агенти, ефективні за широкого контексту, подеколи пропускають очевидні локальні помилки, а для значної частини мов надлишковий контекст діє як шум і знижує результат. До того ж для сильної моделі простий пошук на основі BM25 чи векторної подібності нерідко додає більше шуму, ніж користі, причому різні моделі віддають перевагу різним способам пошуку [4]. Дослідження RARe незалежно встановлює, що додавання прикладів понад один найрелевантніший погіршує результат [5], а CR-Bench показує, що зі зростанням кількості

знахідок зростає й кількість хибних зауважень, які знижують довіру розробника до інструмента [8]. Отже, приріст забезпечує не більший обсяг контексту, а точність його добору під конкретну модель і клас дефекту.

Ідея визначати стратегію до отримання контексту не нова: у загальній RAG-літературі так званий adaptive retrieval класифікує складність запиту ще до звернення до джерел, а в самому code review İçöz і Biricik нещодавно показали, що класифікація pull request перед генерацією дає 13.2% приросту якості [13]. Втім, у їхній системі класифікація обирає модель для генерації, а вид контексту, історичні приклади через векторний пошук, лишається незмінним для всіх категорій. У цій роботі класифікується не тип коментаря, а ймовірна причина дефекту, і вона визначає не модель, а сам вид контексту для перевірки. Модель за самим diff, без додаткового контексту, висуває таку гіпотезу, спираючись на те, що навіть режим без RAG впевнено вказує на локальну аномалію, а вже контекст, який найбільше пасує ймовірному класу, залучається лише після цього.

Клас гіпотези визначає, який контекст залучається для перевірки: для невідповідності наміру це опис задачі чи issue, для міжфайлової помилки граф залежностей між модулями, а для помилки, яку виявляють інструменти розробки, результат статичного аналізу, перевірки типів чи тестів. Здобутий таким чином контекст перевіряють так само, як це роблять агентні системи. Якщо доказів для конкретного дефекту бракує, пошук продовжують у межах цього самого виду контексту або гіпотезу відхиляють. Картку з розташуванням у коді, типом помилки, джерелом контексту та способом перевірки формують лише за наявності підтвердження. Зауваження без такої картки розробникові не показують.

Перевірити такий підхід можна порівнянням із фіксованими стратегіями: режимом, що завжди залучає лише один вид контексту, і режимом, що залучає одразу всі види незалежно від класу гіпотези. Критерієм слугує не подібність згенерованого тексту до зразка, а кількість підтверджених дефектів і частка хибних зауважень серед усіх поданих. Перевірку доцільно провести на одному наборі pull request за фіксованої моделі, де еталоном є зауваження, прийняті людьми, та тести, що проходять після виправлення. Це дасть змогу з'ясувати, чи справді добір контексту за передбаченим класом дефекту дає кращий баланс повноти й точності, ніж стратегія з фіксованим або надлишковим контекстом.

Висновки. Огляд показує, що RAG і агентний підхід підвищують повноту автоматизованого code review, відкриваючи моделі ті дефекти, що недосяжні з самого diff. Текстовий контекст відновлює намір зміни, структурний і репозиторний розкривають міжфайлові зв'язки, а інструментальний надає перевірені докази. Агентна поведінка йде далі, замінюючи одноразовий пошук ітеративним дослідженням коду з

перевіркою достатності контексту. Водночас більший контекст не гарантує кращого результату: для сильних моделей надлишковий контекст подеколи лише знижує якість і додає шум. Тому найбільший приріст дає не максимізація обсягу контексту, а його добір під передбачуваний клас помилки та конкретну модель, поєднаний з агентною перевіркою обґрунтованості знахідки. Подальшим напрямом є експериментальний стенд, на якому режими порівнюються за кількістю підтверджених дефектів, а не за подібністю до еталонних зауважень.

Література

1. Hu R., Wang X., Wen X.-C., Zhang Z., Jiang B., Gao P., Peng C., Gao C. Benchmarking LLMs for Fine-Grained Code Review with Enriched Context in Practice. arXiv Computer Science, 2025. URL: https://arxiv.org/abs/2511.07017 (дата звернення: 29.06.2026).
2. Zhang Y., Pan Z., Yusuf I.N.B., Ruan H., Shariffdeen R., Roychoudhury A. Code Review Agent Benchmark. arXiv Computer Science, 2026. URL: https://arxiv.org/abs/2603.23448 (дата звернення: 29.06.2026).
3. Zeng Z., Shi R., Han K., Li Y., Sun K., Wang Y., Yu Z., Xie R., Ye W., Zhang S. SWR-Bench: Assessing LLM Performance in Real-World Code Review Comment Generation. arXiv Computer Science, 2025. URL: https://arxiv.org/abs/2509.01494 (дата звернення: 29.06.2026).
4. Zhang L., Yu Y., Yu M., Guo X., Zhuang Z., Rong G., Shao D., Shen H., Kuang H., Li Z., Wang B., Zhang G., Xiang B., Xu X. AACR-Bench: Evaluating Automatic Code Review with Holistic Repository-Level Context. arXiv Computer Science, 2026. URL: https://arxiv.org/abs/2601.19494 (дата звернення: 29.06.2026).
5. Meng Q., Zhang X., Ren Z., Visser J. When More Retrieval Hurts: Retrieval-Augmented Code Review Generation. arXiv Computer Science, 2026. URL: https://arxiv.org/abs/2511.05302 (дата звернення: 29.06.2026).
6. Hong H., Baik J. Retrieval-Augmented Code Review Comment Generation. arXiv Computer Science, 2025. URL: https://arxiv.org/abs/2506.11591 (дата звернення: 29.06.2026).
7. Tantithamthavorn K., Zou Y., Wong A., Gupta M., Wang Z., Buller M., Jiang R., Watson M., Jeong M., Chen K., Wu M. RovoDev Code Reviewer: A Large-Scale Online Evaluation of LLM-based Code Review Automation at Atlassian. arXiv Computer Science, 2026. URL: https://arxiv.org/abs/2601.01129 (дата звернення: 29.06.2026).
8. Pereira K., Sinha N., Ghosh R., Dutta D. CR-Bench: Evaluating the Real-World Utility of AI Code Review Agents. arXiv Computer Science, 2026. URL: https://arxiv.org/abs/2603.11078 (дата звернення: 29.06.2026).
9. Tao Y. et al. Retrieval-Augmented Code Generation: A Survey with Focus on Repository-Level Approaches. arXiv Computer Science, 2025. URL:

https://arxiv.org/abs/2510.04905 (дата звернення: 29.06.2026).
10. Zhang Y., Zhao X., Wang Z.Z., Yang C., Wei J., Wu T. cAST: Enhancing Code Retrieval-Augmented Generation with Structural Chunking via Abstract Syntax Tree. Findings of EMNLP, 2025. URL: https://arxiv.org/abs/2506.15655 (дата звернення: 29.06.2026).
11. Zhang Y., Ruan H., Fan Z., Roychoudhury A. AutoCodeRover: Autonomous Program Improvement. arXiv Computer Science, 2024. URL: https://arxiv.org/abs/2404.05427 (дата звернення: 29.06.2026).
12. Song S. Knowledge-Aware Iterative Retrieval for Multi-Agent Systems. arXiv Computer Science, 2025. URL: https://arxiv.org/abs/2503.13275 (дата звернення: 29.06.2026).
13. İçöz B., Biricik G. Context-Aware Code Review Automation: A Retrieval-Augmented Approach. Applied Sciences, 2026, 16(4), 1875. URL: https://doi.org/10.3390/app16041875 (дата звернення: 02.07.2026).

УДК 629.58:007.52

DOI

UNDERWATER ROBOTICS AND THE DEVELOPMENT OF ENGINEERING SOLUTIONS FOR REAL-WORLD PROBLEMS THROUGH THE MATE ROV COMPETITION

Volodymyr Yarovenko
Undergraduate Student, Engineering One
Memorial University of Newfoundland
yarovenkovova2008@gmail.com

Анотація: *Underwater robotics is focused on the development of robotic systems for ocean exploration and investigation. The MATE ROV Competition is encouraging students from around the world to learn about underwater robotics by challenging them to design, build, and operate the ROVs and vertical-profiling floats.*

Underwater robotics is a field of engineering that focuses on the development and deployment of robotic systems in underwater environments where direct human access may be difficult or impossible. The application of underwater robotics includes the exploration and recording of environmental information, investigation of ocean floor, documenting shipwrecks, and more. The two main types of robotic systems used for these applications are AUV (Autonomous Underwater Vehicle) and ROV (Remotely Operated Vehicle) [1], [2].