Розвиток продуктивних сил і регіональна економіка

УДК 004.4’242:004.415.2

DOI https://doi.org/10.5281/zenodo.17140774

Роль практик рецензування в забезпеченні надійності Java-сервісів на етапі архітектурного проєктування

Абакумов Роман Валерійович,

інженер-програміст, DispatchHealth Management, LLC,

3827 North Lafayette St., Denver, CO 80205, USA,

https://orcid.org/0009-0004-2743-3815

Прийнято: 04.09.2025 | Опубліковано: 17.09.2025

Анотація. *Актуальність дослідження зумовлена зростаючими вимогами до надійності, масштабованості та безпеки програмних систем, що особливо проявляється в сфері розробки Java-сервісів, які активно застосовуються в бізнес-критичних доменах. Саме на етапі архітектурного проєктування визначаються ключові параметри стабільності й довговічності системи, а недооцінювання цього етапу призводить до суттєвих ризиків і фінансових утрат у майбутньому. У цих умовах практики peer review розглядають як ефективний інструмент попередження архітектурних вразливостей та підвищення якості програмного продукту.*

Мета статті полягає в науковому обґрунтуванні значення процедур рецензування в забезпеченні надійності Java-сервісів під час архітектурного проєктування та у встановленні напрямів поліпшення результативності їх застосування в ході розроблення.

Методологія дослідження базується на порівняльному аналізі наукових джерел, вивченні міжнародних стандартів ISO/IEC та IEEE та критичному

розгляді практик рецензування в контексті сучасних підходів до архітектурного проєктування. Використано методи систематизації для окреслення ключових характеристик надійності, узагальнення – для створенння рекомендацій та аналітичний підхід до оцінювання взаємодії peer review з автоматизованими інструментами контролю якості.

Результати роботи засвідчили, що впровадження рецензування на архітектурному рівні дає змогу своєчасно виявляти ризики масштабованості, відмовостійкості та безпеки, підвищувати прозорість ухвалення рішень і знижувати вартість усунення дефектів. Доведено, що поєднання експертного аналізу та інструментальної підтримки (статичний аналіз коду, CI/CD-платформи, засоби візуалізації архітектури) вибудовує інтегрований контур контролю якості. Виявлено, що ефективність peer review зростає за умов стандартизації процедур і формалізації результатів.

Висновки підтвердили, що проблематика застосування peer review охоплює організаційні, методичні та культурні бар'єри, подолання яких є необхідним для поліпшення якості архітектурного проєктування. Запропоновано рекомендації щодо стандартизації підходів, інтеграції рецензування в середовище DevOps та CI/CD, а також формування практики менторських оглядів.

Перспективи подальших досліджень пов'язані з розробленням уніфікованих методик поєднання peer review з інструментами автоматизованої верифікації, оцінюванням впливу організаційних культур на результативність рецензування та створенням інтегрованих моделей забезпечення надійності, орієнтованих на довгостроковий розвиток Java-сервісів.

Ключові слова: *програмна інженерія, контроль якості, колективна експертиза, DevOps-середовище, стандартизація процесів.*

The role of peer review practices in ensuring the reliability of Java services at the stage of architectural design

Roman Abakumov,
Software Engineer, DispatchHealth Management, LLC,
3827 North Lafayette St., Denver, CO 80205, USA,
https://orcid.org/0009-0004-2743-3815

Abstract. *The relevance of the research is determined by the growing requirements for reliability, scalability, and security of software systems, which are particularly evident in the development of Java services widely used in business-critical domains. It has been established that the stage of architectural design defines the key parameters of system stability and durability, while its underestimation leads to significant risks and financial losses. Under these conditions, peer review practices are considered an effective tool for preventing architectural vulnerabilities and improving the overall quality of software products.*

The purpose of the article is to provide a scientific justification of the significance of peer review procedures in ensuring the reliability of Java services during architectural design and to identify directions for improving their effectiveness in the development process.

Research methodology is based on a combination of comparative analysis of scientific literature, examination of international standards such as ISO/IEC and IEEE, and critical analysis of peer review practices within the context of modern approaches to architectural design. Methods of systematization were applied to define the key reliability characteristics, generalization was used to formulate recommendations, and analytical approaches were employed to evaluate the interaction between peer review and automated quality control tools.

Research results demonstrated that the introduction of peer review at the architectural level enables the prompt identification of risks related to scalability, fault tolerance, and security, increases transparency in decision-making, and reduces the cost of defect elimination. It has been proven that combining expert analysis with tool support (static code analysis, CI/CD platforms, architecture visualization tools) creates an integrated quality assurance framework. It has been revealed that the efficiency of peer review increases significantly under conditions of standardized procedures and formalized results.

Keywords: *software engineering, quality assurance, collective expertise, DevOps environment, process standardization.*

Постановка проблеми. На сучасному етапі розроблення складних інформаційних систем особливого значення набуває питання забезпечення надійності програмних сервісів на стадії архітектурного проєктування. Вибір архітектурних рішень визначає подальшу якість програмного продукту, його масштабованість, безпечність та здатність витримувати навантаження в реальних умовах експлуатації. Недостатньо обґрунтовані або неперевірені рішення на цьому етапі призводять до критичних похибок, які складно виправити на наступних фазах життєвого циклу системи. Одним із ефективних інструментів мінімізації таких ризиків є практика рецензування, що передбачає систематичний аналіз і критичне оцінювання архітектурних рішень фахівцями, незалежними від їх безпосереднього автора.

Наукова значущість проблеми зумовлена тим, що рецензування є не лише методом контролю, але й засобом формування нових знань у команді розробників, сприяючи поширенню кращих практик архітектурного проєктування та інтеграції міждисциплінарних підходів. З практичного погляду, впровадження якісних процедур peer review допомагає виявити архітектурні вразливості ще до їх матеріалізації у вигляді програмних

помилок, забезпечує відповідність сервісів вимогам надійності та безпеки, а також знижує витрати на подальше доопрацювання системи. Отже, проблема набуває подвійної актуальності: у науковому вимірі вона постає як вивчення механізмів колективної експертизи в програмній інженерії, а в практичному аспекті – як завдання підвищення ефективності розробки Java-сервісів, що активно застосовуються в бізнес-критичних сферах.

Аналіз останніх досліджень і публікацій. Аналіз наукових праць дає змогу виокремити чотири основні напрями досліджень, що формують комплексне уявлення про роль практик рецензування у гарантуванні надійності Java-сервісів на етапі архітектурного проєктування. Перший напрям охоплює публікації, пов'язані з основами програмної організації та їхнім впливом на архітектурні рішення. P. Gowda (П. Говда) та A. N. Gowda (А. Н. Говда) показали, що належне застосування потоків у Java потребує систематичного рецензування для зниження ризиків неузгодженості [1]. B. Vyas (Б. Вйас) проаналізував виклики інформаційної безпеки в Java-додатках, довівши, що практики рецензування допомагають виявляти критичні вразливості на ранніх етапах [2]. L. Pascarella (Л. Пашкарелла) та співавтори класифікували коментарі в Java-системах, що відображають рівень зрілості командного рецензування й впливають на архітектурні рішення [3]. T. Durieux (Т. Дюрьє) та співавтори встановили, що ефективність інструментів виправлення Java-програм значно зростає за умови інтеграції з рецензуванням [4]. Цей джерельний блок засвідчує, що рецензування відіграє фундаментальну роль у запобіганні помилкам, які прямо позначаються на архітектурній надійності. Подальші дослідження варто зосередити на кількісному вимірюванні впливу рецензування на якість архітектури.

Другий напрям зосереджений на інтеграції рецензування з автоматизованими інструментами забезпечення якості. N. Cassee (Н. Кассі) та співавтори аргументували, що безперервна інтеграція діє як «мовчазний

помічник» у процесі рецензування, мінімізуючи дефекти [5]. R. Tufano (Р. Туфано) та співавтори зауважили, що попередньо натреновані моделі дають змогу автоматизувати рецензування, пришвидшуючи ухвалення архітектурних рішень [6]. S. AlAraj (С. Аларадж) та M. AlAraj (М. Аларадж) обґрунтували, що Java може слугувати основою для інтелектуальних систем рецензування з елементами штучного інтелекту [7]. A. Khatami (A. Хатамі) та A. Zaidman (А. Зайдман) узагальнили практики гарантування якості у відкритих Java-проєктах, підкресливши, що рецензування є ключовим чинником їхньої стійкості [8]. Отже, автоматизація підсилює рецензування, а рецензування надає автоматизованим інструментам архітектурної глибини. Подальші дослідження доцільно спрямувати на синтез штучного інтелекту та колективної експертизи в автоматизованому рецензуванні.

Третій напрям стосується комунікативного та організаційного вимірів рецензування. D. H. Choi (Д. Х. Чой) та T. S. Seo (Т. С. Со) розробили модель відкритого рецензування з використанням блокчейну та алгоритмів рекомендації, яка підвищує прозорість і знижує суб'єктивність оцінювання [9]. M. Паіхао (M. Paixao) та співавтори хоча й вивчали соціально-економічний контекст, акцентували на значенні комплексного підходу до оцінювання надійності систем, що може бути перенесено й на практики рецензування [10]. A. Uchôa (А. Учоа) та співавтори продемонстрували, що рецензування має прогностичну цінність у виявленні архітектурно значущих змін [11]. D. Guamán (Д. Гуаман) та співавтори запропонували процес реконструкції архітектури, де рецензування є ключовим елементом узгодження рішень із фактичною реалізацією [12]. Це свідчить, що рецензування є не лише інструментом перевірки коду, але й засобом підтримки цілісності архітектури. Подальші дослідження мають зосередитися на розробленні методик прогнозування архітектурних змін на основі даних рецензування.

Четвертий напрям розширює бачення рецензування, пов'язуючи його з новітніми технологіями безпеки та продуктивності. Y. Horbenko (Ю. Горбенко) наголошує на важливості застосування WebAssembly та блокчейну для підвищення безпеки й продуктивності фронтенд-систем, підкреслюючи, що аналогічні принципи можуть поширюватися й на архітектуру Java-сервісів у процесі рецензування [13]. У наступній праці того ж автора показано, що конфіденційні обчислення із використанням безпечних анклавів і гомоморфного шифрування створюють нові стандарти довіри, які здатні бути інтегрованими в системи рецензування для зміцнення архітектурної надійності [14]. Отже, рецензування є не лише інструментом контролю, але й частиною комплексних технологій захисту. Подальші дослідження мають бути спрямовані на інтеграцію рецензування з підходами конфіденційних обчислень і високопродуктивних технологій.

Виділення не вирішених раніше частин загальної проблеми. Попри значні досягнення у вивченні архітектурного проєктування та практик peer review, залишаються невирішеними кілька важливих аспектів. Недостатньо розкрито роль архітектурного проєктування як системоутворюючого етапу у формуванні надійності програмних систем, бракує глибокого аналізу механізмів peer review у контексті архітектурних рішень та їхнього впливу на якість Java-сервісів. Обмеженими є дослідження взаємодії рецензування з інструментами автоматизованого контролю якості, що ускладнює створення цілісного підходу до оцінювання архітектури. Водночас не подолані організаційні та методичні бар'єри, пов'язані з відсутністю уніфікації процесів та ризиками суб'єктивності оцінювання, що знижує ефективність рецензування в практиці.

Запропоноване дослідження покликане заповнити ці прогалини через комплексне обґрунтування значення архітектурного проєктування в забезпеченні надійності систем, детальний аналіз механізмів peer review та

їхню інтеграцію з автоматизованими інструментами. Виявлення ключових проблем упровадження рецензування в реальних умовах дає змогу сформувати практичні рекомендації щодо його стандартизації та вдосконалення.

Формулювання цілей статті (постановка завдання). Мета статті полягає в науковому обґрунтуванні ролі практик рецензування в гарантуванні надійності Java-сервісів на етапі архітектурного проєктування та визначенні шляхів підвищення ефективності їх застосування в процесі розроблення.

Завдання статті:

1. Дослідити значення архітектурного проєктування та механізмів peer review у забезпеченні якості й надійності Java-сервісів.

2. Проаналізувати взаємодію рецензування з інструментами автоматизованого контролю та виявити проблеми його впровадження.

3. Сформулювати практичні рекомендації щодо підвищення ефективності процедур peer review в архітектурному проєктуванні.

Виклад основного матеріалу дослідження. Архітектурне проєктування є фундаментальним етапом створення програмних систем, адже саме на цьому рівні визначаються базові принципи організації компонентів, способи їх взаємодії та загальна логіка функціонування сервісів. Якість ухвалених архітектурних рішень безпосередньо впливає на надійність програмного забезпечення, його стійкість до збоїв, можливість масштабування та інтеграції з іншими системами. Помилки або недоопрацювання на цьому етапі здатні зумовити значні витрати в майбутньому, оскільки їх виправлення на пізніших фазах життєвого циклу є значно складнішим і дорожчим. Саме тому архітектурне проєктування розглядають не лише як технічну діяльність, а і як стратегічний процес, який окреслює довгострокову ефективність програмних систем (табл. 1).

Таблиця 1

Ключові аспекти архітектурного проєктування в забезпеченні надійності програмних систем

Аспект	Зміст	Вплив на надійність
Структурна організація	Визначення компонентів і модулів та їх взаємозв'язків	Забезпечує узгодженість і цілісність системи
Масштабованість	Передбачення можливостей для розширення та оптимізації	Дає змогу адаптувати систему до зростання навантаження
Відмовостійкість	Використання механізмів резервування та відновлення	Мінімізує ризики збоїв і втрат даних
Безпека	Інтеграція принципів захисту на рівні архітектури	Гарантує стійкість до зовнішніх і внутрішніх загроз
Гнучкість	Можливість адаптації до змін вимог	Підвищує довговічність і життєздатність системи

Джерело: сформовано автором на основі [1, p. 1564–1566; 2, p. 269–272; 3, p. 1503–1505; 10, p. 1044–1048; 12, p. 594–596; 13, p. 2280–2283; 15; 16; 17]

У сучасних умовах архітектурне проєктування програмних систем тісно пов'язане з необхідністю забезпечення балансу між технічною складністю та практичною надійністю. Структурна організація відіграє роль базового каркаса, який визначає логіку взаємодії компонентів і мінімізує ризик появи «вузьких місць». Наприклад, у Java-сервісах застосування патернів проєктування на зразок layered architecture чи hexagonal architecture гарантує чітке розмежування рівнів, що сприяє локалізації помилок і зручності тестування. Масштабованість у практиці реалізується через мікросервісні архітектури: такі компанії як Netflix чи Amazon використовують розподілені сервіси з автоматичним балансуванням навантаження, що дає можливість безперервно розширювати обчислювальні ресурси відповідно до динаміки користувацького трафіку [10, p. 1045–1046]. Відмовостійкість проєктується через введення механізмів резервування та реплікації. У реальних рішеннях це може бути впровадження кластеризації Java-додатків у Kubernetes чи застосування механізмів circuitbreaker у SpringCloud, які допомагають

запобігати каскадним відмовам. Безпека інтегрується на архітектурному рівні через багаторівневі підходи: шифрування даних, контроль доступу на основі ролей та використання API-gateway для централізованого управління потоками інформації. Ці принципи успішно застосовуються у фінтех-системах, де Java-сервіси повинні гарантувати захист транзакцій і відповідність міжнародним стандартам, таким як PCI DSS. Гнучкість архітектури проявляється в адаптивності до трансформацій зовнішнього середовища та бізнес-вимог. На практиці це означає можливість поступового впровадження нових функцій без порушення роботи наявних модулів, наприклад, через методології DevOps і CI/CD [5, p. 429]. В українських IT-компаніях такий підхід дає змогу швидко інтегрувати зміни, пов’язані з оновленням державних регламентів чи міжнародних стандартів, не зупиняючи критично важливі сервіси.

У процесі архітектурного проєктування складних програмних систем одним із ключових інструментів підвищення їхньої надійності є peer review. Його сутність полягає в колективному аналізі й обговоренні архітектурних рішень, що допомагає виявляти недоліки, суперечності або приховані ризики ще до їх реалізації в коді. На відміну від автоматизованих інструментів перевірки, peer review ґрунтується на експертних знаннях та досвіді членів команди, що передбачає багатогранний погляд на архітектуру системи. Для Java-сервісів, які зазвичай функціонують у середовищі з високими вимогами до масштабованості й безпеки, значення цього підходу зростає, оскільки якісно організоване рецензування дає змогу досягти стабільності та стійкості архітектури в довгостроковій перспективі (табл. 2).

Таблиця 2

Механізми peer review архітектурних рішень та їх вплив на якість Java-сервісів

Механізм	Характеристика	Вплив на якість Java-сервісів
Формальні архітектурні огляди	Регулярні зустрічі та документування результатів	Підвищують прозорість рішень і відповідність стандартам
Інтерактивні обговорення (design review meetings)	Дискусії в режимі реального часу з використанням моделей і діаграм	Забезпечують оперативне виявлення слабких місць
Застосування контрольних списків	Упровадження стандартизованих критеріїв перевірки	Гарантують системність і зменшують ризик пропуску критичних деталей
Інструментальна підтримка (Jira, Confluence, GitHub)	Фіксація, відстеження та аналіз рішень у цифрових платформах	Створюють базу знань і підтримують повторюваність процесу
Менторські рецензії	Залучення досвідчених архітекторів до оцінювання рішень молодших розробників	Поліпшують якість рішень і сприяють професійному розвитку команди

Джерело: сформовано автором на основі [5, p. 428–432; 6, p. 2295–2298; 7, p. 102–104; 8, p. 1415–1418; 9, p. 106–109; 11, p. 474–476; 18]

Практичне застосування peer review у сучасних умовах демонструє значну ефективність у запобіганні архітектурним помилкам і підвищенні якості Java-сервісів. Формальні архітектурні огляди активно застосовують у великих міжнародних компаніях, таких як Google та Microsoft, де документування результатів створює основу для подальшої оптимізації й повторного використання успішних рішень. Інтерактивні обговорення у форматі design review meetings, які все частіше проводяться за допомогою візуальних інструментів моделювання на кшталт UML чи ArchiMate, дають змогу командам швидко узгоджувати альтернативні варіанти архітектури та оперативно реагувати на зміну вимог. Упровадження контрольних списків набуває особливої актуальності в аутсорсингових компаніях України, де системність перевірки гарантує відповідність міжнародним стандартам ISO/IEC [15–19]. Інструментальна підтримка через Jira чи GitHub створює

єдине інформаційне середовище для відстеження рішень і спрощує взаємодію між віддаленими командами. Менторські рецензії в практиці компаній, що працюють з Java у фінансовому секторі, не лише підвищують надійність архітектури, але й забезпечують передання досвіду, що є важливим у формуванні культури якості всередині команди. У результаті peer review постає як багатокомпонентний механізм, що поєднує технічний контроль, навчання та стратегічне управління якістю сервісів.

У сучасному архітектурному проєктуванні Java-сервісів зростає значення комплексного підходу до гарантування якості, який ураховує людську експертизу та автоматизовані інструменти контролю. Peer review дає змогу виявляти логічні й концептуальні недоліки, що виходять за межі можливостей алгоритмів, тоді як автоматизовані засоби аналізу коду й архітектури забезпечують системність і масштабованість перевірок. Їхня взаємодія створює подвійний рівень захисту від архітектурних помилок: поєднання глибокого експертного оцінювання й швидкої перевірки великого обсягу технічних деталей. Це допомагає командам розробників не лише підвищувати якість ухвалених рішень, але й зменшувати витрати часу на рутинні перевірки (табл. 3).

Таблиця 3

Взаємодія peer review та інструментів автоматизованого контролю якості

Компонент	Роль у процесі	Результат взаємодії
Peer review	Експертний аналіз архітектурних моделей, розкриття логічних і концептуальних недоліків	Забезпечення стратегічної обґрунтованості рішень
Статичний аналіз коду (SonarQube, PMD, Checkstyle)	Автоматичне виявлення синтаксичних і структурних дефектів	Зниження ризику дрібних помилок і технічного боргу
CI/CD-платформи (Jenkins, GitHubActions)	Автоматизований запуск тестів і перевірок під час інтеграції	Підтримка стабільності коду під час швидких релізів
Інструменти моделювання (Archi, PlantUML)	Візуалізація архітектури для колективного аналізу	Оптимізація дискусій та швидше узгодження змін

ISSN: 2786-9431

Компонент	Роль у процесі	Результат взаємодії
Інтегровані панелі контролю (Jira, Confluence)	Фіксація результатів рецензування й автоматизованих перевірок	Формування єдиної бази знань і прозорості процесу

Джерело: сформовано автором на основі [4, p. 310–312; 5, p. 431–433; 6, p. 2299–2301; 8, p. 1420–1422; 11, p. 478–480; 19]

У сучасній практиці розроблення Java-сервісів взаємодію peer review та автоматизованих засобів контролю якості розглядають як доповнювальний процес. Peer review допомагає встановлювати логічні неузгодженості й концептуальні ризики, які часто залишаються поза увагою автоматизованих інструментів, тоді як статичний аналіз коду чи CI/CD гарантують оперативність і системність перевірок. Наприклад, у фінансових сервісах колективний аналіз архітектури дає змогу оцінити відповідність моделі вимогам безпеки, тоді як SonarQube автоматично виявляє уразливості в коді, пов’язані з неправильним обробленням виключень. У сфері електронної комерції використання PlantUML для візуалізації архітектури спрощує колективні обговорення, а GitHubActions передбачає стабільність релізів через автоматичні тести. Завдяки цьому досягається подвійний результат: глибина експертного аналізу поєднується з ефективністю машинних перевірок, що забезпечує підвищення надійності та скорочення часу виходу Java-сервісів на ринок.

Попри очевидні переваги процедур рецензування, їх упровадження в процес архітектурного проєктування супроводжується низкою проблем, які суттєво знижують дієвість цього інструменту. Організаційні бар’єри пов’язані насамперед із браком часу у високонавантажених проєктах: команди часто віддають перевагу швидкій реалізації функціоналу, ігноруючи системні огляди архітектурних рішень [5, p. 428-430]. До того ж немає чітко визначених ролей і відповідальності під час проведення рецензій, коли обговорення втрачає структурованість, а результати залишаються неформалізованими.

Суттєвою проблемою є й відсутність методичної уніфікації. У різних командах використовуються несумісні підходи до документування та характеристики архітектурних рішень, що ускладнює порівняння результатів і накопичення колективного досвіду [7, p. 102–104; 12, p. 594–596; 16]. Це призводить до фрагментарності знань та унеможливлює формування єдиної бази практик. Окремо варто відзначити ризики суб'єктивності: оцінювання архітектурних моделей значною мірою залежить від досвіду та пріоритетів конкретних експертів. Як наслідок, одне й те саме рішення може отримати як схвалення, так і критику, що створює невизначеність у процесі ухвалення остаточних рішень.

На практиці спостерігаються й культурні бар'єри. У багатьох колективах рецензування сприймають як форму контролю, а не співпраці, що породжує опір і знижує відвертість обговорень [2, p. 273–274]. Це особливо помітно в міжнародних проєктах, де стикаються різні стилі комунікації й організаційні культури. Додатковою проблемою є недостатня інтеграція peer review з інструментами автоматизації: огляди проводять ізольовано від систем CI/CD чи статичного аналізу коду, що унеможливлює комплексне оцінювання архітектурних рішень [4, p. 310–311]. Також варто згадати проблему «поверхневого рецензування», коли формально здійснені огляди зводяться до поверхових коментарів без глибокого аналізу, що лише створює ілюзію контролю.

Удосконалення застосування peer review у процесі архітектурного проєктування Java-сервісів потребує системного підходу, який поєднує організаційні, методичні та технологічні рішення. Передусім доцільним є формування чіткої процедури рецензування, яка має охоплювати стандартизовані етапи підготовки, обговорення та фіксацію результатів. Така структурованість зменшує ризики суб'єктивності та забезпечує накопичення досвіду, що може використовуватися під час наступних ітерацій проєктування.

Важливо також інтегрувати peer review у середовище сучасних інструментів автоматизації. Поєднання експертного аналізу з результатами статичного аналізу коду, автоматизованого тестування й CI/CD-процесів дає змогу створити єдиний контур контролю якості, у якому людська оцінка доповнюється машинною об'єктивністю. Це особливо актуально для Java-сервісів, що працюють у високонавантажених середовищах, де будь-яка помилка може призвести до значних фінансових утрат. До рекомендацій належить і розвиток культури співпраці всередині команд. Рецензування має сприйматися не як механізм контролю, а як інструмент колективного навчання й підвищення рівня компетентності. Запровадження менторських практик, коли досвідчені архітектори беруть участь у рецензуванні рішень молодших спеціалістів, створює додану цінність – професійне зростання команди. Водночас необхідно забезпечити прозорість процесу за допомогою цифрових платформ, які зберігають історію рецензій, аргументацію й ухвалені рішення. Окремої уваги заслуговує уніфікація методик оцінювання. Використання контрольних списків, заснованих на міжнародних стандартах ISO/IEC та IEEE, надає процесу рецензування об'єктивності й гарантує відповідність архітектурних рішень загальноприйнятим нормам. Це дає змогу не лише поліпшити якість Java-сервісів, але й формує підґрунтя для їхньої сертифікації та інтеграції в глобальні цифрові екосистеми. Отже, практичні кроки з удосконалення peer review в архітектурному проєктуванні повинні базуватися на стандартизації процедур, інтеграції з інструментами автоматизації та розвитку командної культури. Реалізація цих рекомендацій забезпечить підвищення надійності Java-сервісів і водночас оптимізує процеси їхньої розробки, роблячи їх більш передбачуваними та економічно ефективними.

Висновки. У дослідженні встановлено, що архітектурне проєктування є головним чинником гарантування надійності Java-сервісів, адже саме на цьому етапі формуються умови для масштабованості, відмовостійкості, безпеки та

гнучкості систем. Доведено, що застосування практик peer review під час ухвалення архітектурних рішень передбачає багаторівневий контроль, сприяє колективному виробленню оптимальних підходів і знижує ризик стратегічних помилок, які складно виправити на подальших етапах життєвого циклу. Виявлено ключові проблеми впровадження рецензування: організаційні обмеження, пов'язані з браком часу та ресурсів; відсутність єдиної методичної основи, що перешкоджає накопиченню досвіду; ризики суб'єктивності оцінювання та культурні бар'єри, які знижують результативність командної взаємодії. Додатковим викликом є недостатня інтеграція peer review з інструментами автоматизованого контролю якості, що ускладнює комплексну перевірку архітектурних моделей. Сформульовані рекомендації окреслюють стандартизацію процедур рецензування з опорою на міжнародні норми, синтез експертної оцінки з автоматизованими засобами аналізу та розвиток культури співпраці всередині команд. Такий підхід забезпечує підвищення надійності Java-сервісів, скорочення витрат на виправлення помилок і формування прозорого процесу ухвалення рішень. Перспективи подальших досліджень полягають у створенні інтегрованих методик поєднання peer review з автоматизованими системами верифікації, а також у вивченні впливу організаційних та культурних чинників на ефективність процедур рецензування.

Список використаних джерел

1. Gowda P., Gowda A. N. Power of Java streams and its best practice. *International Journal of Science and Research (IJSR)*. 2021. Vol. 10, № 11. P. 1563–1567. DOI: https://dx.doi.org/10.21275/SR24801081541.

2. Vyas B. Security challenges and solutions in java application development. *Eduzone: International Peer Reviewed/Refereed Multidisciplinary Journal*. 2023. Vol. 12, № 2. P. 268–275. URL:

https://www.eduzonejournal.com/index.php/eiprmj/article/view/467 (дата звернення: 08.07.2025).

3. Pascarella L., Bruntink M., Bacchelli A. Classifying code comments in Java software systems. *Empirical Software Engineering*. 2019. Vol. 24, № 6. P. 1499–1537. DOI: https://doi.org/10.1007/s10664-019-09694-w.

4. Durieux T., Madeiral F., Martinez M., Abreu R. Empirical review of Java program repair tools: A large-scale experiment on 2,141 bugs and 23,551 repair attempts. *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2019. P. 302–313. DOI: https://doi.org/10.1145/3338906.3338911.

5. Cassee N., Vasilescu B., Serebrenik A. The silent helper: the impact of continuous integration on code reviews. *Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. London, ON, Canada. 2020. P. 423–434. DOI: 10.1109/SANER48275.2020.9054818.

6. Tufano R., Masiero S., Mastropaolo A., Pascarella L., Poshyvanyk D., Bavota G. Using pre-trained models to boost code review automation. *Proceedings of the 44th International Conference on Software Engineering*. 2022. P. 2291–2302. DOI: https://doi.org/10.1145/3510003.3510621.

7. AlAraj S., AlAraj M. Java-powered AI: using code to implement intelligent systems–systematic review. *International Journal of Theory of Organization and Practice (IJTOP)*. 2024. Vol. 4, № 1. P. 100–106. URL: https://journals.gaftim.com/index.php/ijtop/article/view/344 (дата звернення: 08.07.2025).

8. Khatami A., Zaidman A. State-of-the-practice in quality assurance in Java-based open source software development. *Software: Practice and Experience*. 2024. Vol. 54, № 8. P. 1408–1446. DOI: https://doi.org/10.1002/spe.3321.

9. Choi D. H., Seo T. S. Development of an open peer review system using blockchain and reviewer recommendation technologies. *Science Editing*. 2021. Vol. 8, № 1. P. 104–111. DOI: https://doi.org/10.6087/kcse.237.

10. Paixao M., Krinke J., Han D., Ragkhitwetsagul C., Harman M. The impact of code review on architectural changes. *IEEE Transactions on Software Engineering*. 2021. Vol. 47, № 5. P. 1041–1059. DOI: 10.1109/TSE.2019.2912113.

11. Uchôa A., Barbosa C., Coutinho D., Oizumi W., Assunçao W. K., Vergilio S. R., Garcia A. Predicting design impactful changes in modern code review: a large-scale empirical study. *Proceedings of the 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. Madrid, Spain. 2021. P. 471–482. DOI: 10.1109/MSR52588.2021.00059.

12. Guamán D., Pérez J., Diaz J., Cuesta C. E. Towards a reference process for software architecture reconstruction. *IET Software*. 2020. Vol. 14, № 6. P. 592–606. DOI: https://doi.org/10.1049/iet-sen.2019.0246.

13. Horbenko Y. Web assembly and blockchain for high-performance secure front-end systems. *International Journal of Current Science Research and Review*. 2025. Vol. 8, № 5. P. 2279–2285. DOI: https://doi.org/10.47191/ijcsrr/V8-i5-36.

14. Horbenko Y. Confidential computing in front-end: enhancing data security with secure enclaves and homomorphic encryption. *International Journal of Advanced Multidisciplinary Research and Studies*. 2025. Vol. 5, № 3. P. 308–321. URL: https://www.multiresearchjournal.com/admin/uploads/archives/archive-1747130538.pdf (дата звернення: 08.07.2025).

15. ISO/IEC 25019:2023. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality-in-use model. *International Organization for Standardization*: вебсайт. 2023. URL: https://www.iso.org/standard/78177.html (дата звернення: 04.07.2025).

16. ISO/IEC/IEEE 42010:2022. Software, systems and enterprise – Architecture description. *International Organization for Standardization*: вебсайт. 2022. URL: https://www.iso.org/standard/74393.html (дата звернення: 04.07.2025).

17. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems — Requirements. *International Organization for Standardization*: вебсайт. 2022. URL: https://www.iso.org/standard/82875.html (дата звернення: 04.07.2025).

18. IEEE 1471-2000. IEEE recommended practice for architectural description of software-intensive systems. *Institute of Electrical and Electronics Engineers*: вебсайт. 2000. URL: https://standards.ieee.org/ieee/1471/2293/ (дата звернення: 04.07.2025).

19. OWASP Application Security Verification Standard (ASVS) v4.0.2. *Open Worldwide Application Security Project*: вебсайт. 2020. URL: https://owasp.org/www-project-application-security-verification-standard/ (дата звернення: 04.07.2025).

ISSN: 2786-9431