ІНЖЕНЕРНІ НАУКИ

УДК 004.415.2:004.41:005.32 DOI https://doi.org/10.35546/kntu2078-4481.2025.3.1.1

Р. В. АБАКУМОВ
інженер-програміст, DispatchHealth Management, LLC, Денвер, США
ORCID: 0009-0004-2743-3815

КОРЕЛЯЦІЯ МІЖ КУЛЬТУРОЮ CODE REVIEW ТА РІВНЕМ НАДІЙНОСТІ У ВЕЛИКИХ МІЖНАРОДНИХ ІНЖЕНЕРНИХ КОМАНДАХ

У статті представлено ґрунтовний аналіз кореляції між культурою рецензування коду та рівнем надійності у великих міжнародних інженерних командах. Застосовуючи міждисциплінарний підхід, що проводить широкі паралелі між програмною інженерією та науковим методом, аргументовано, що такі ключові інженерні практики, як code review, є прямими аналогами академічного рецензування (peer review), які забезпечують колективну верифікацію та підвищення достовірності програмного коду. Цей підхід дає змогу переосмислити процес розробки як постійний цикл висування гіпотез, експериментальної перевірки та колективного аналізу, що веде до системного поліпшення якості. Проведене дослідження показало, що code review має не лише очевидний технічний вплив, пов'язаний зі зниженням кількості дефектів, виявленням архітектурної ерозії та підвищенням читабельності коду, але й глибоке культурне значення, що сприяє формуванню довіри, покращенню комунікації та створенню колективної відповідальності за якість програмного продукту, що є особливо важливим в умовах роботи розподілених команд. На основі окреслених складних взаємозв'язків розроблено модель, яка візуалізує причинно-наслідкові зв'язки, демонструючи, як зріла культура рецензування безпосередньо діє на ключові метрики надійності, зокрема на частоту відкатів та кількість критичних інцидентів, що виникають після релізу. Результати дослідження мають вагоме практичне значення для менеджерів та інженерних команд, оскільки підкреслюють, що інвестування в організаційну культуру, яка підтримує відкрите та конструктивне рецензування, є переломним фактором для гарантування довгострокової стабільності та якості програмного продукту, перетворюючи його з технічного процесу на фундаментальний атрибут успішної інженерної практики. Отже, зазначена модель втілює універсальний інструмент для оцінювання та покращення ефективності роботи команд, орієнтуючись на культуру як на основний драйвер надійності.

Ключові слова: *рецензування коду, якість програмного забезпечення, культура розробки, командна взаємодія, метрики надійності, розподілені команди, міждисциплінарний підхід.*

R. V. ABAKUMOV
Software Engineer, DispatchHealth Management, LLC, Denver, USA
ORCID: 0009-0004-2743-3815

CORRELATION BETWEEN CODE REVIEW CULTURE AND RELIABILITY IN LARGE INTERNATIONAL ENGINEERING TEAMS

This article presents an in-depth analysis of the correlation between code review culture and reliability in large international engineering teams. Using an interdisciplinary approach that draws broad parallels between software engineering and the scientific method, it argues that key engineering practices such as code review are direct analogues of academic peer review, providing collective verification and enhancing the credibility of software code. This perspective makes it possible to reconceptualize the development process as a continuous cycle of hypothesis generation, experimental testing, and collective analysis that leads to systematic quality improvement.

The study shows that code review exerts not only an obvious technical impact, including the reduction of defects, the detection of architectural erosion, and the improvement of code readability, but also a deep cultural influence. It fosters trust, strengthens communication, and establishes collective responsibility for software quality, which is especially critical in distributed teams. Based on these complex interconnections, a model was developed to visualize causal relationships, demonstrating how a mature review culture directly affects key reliability metrics such as rollback frequency and the number of critical incidents that occur after release.

The results of the research have substantial practical value for managers and engineering teams, as they highlight that investing in an organizational culture that supports open and constructive review is a decisive factor in ensuring long-term stability and product quality. This transforms code review from a purely technical procedure into a fundamental

© Абакумов Р. В., 2025
Стаття поширюється на умовах ліцензії CC BY 4.0

attribute of successful engineering practice. Accordingly, the proposed model represents a universal tool for assessing and improving team performance by positioning culture as the primary driver of reliability.

Key words: *code review, software quality, development culture, team collaboration, reliability metrics, distributed teams, interdisciplinary approach.*

Постановка проблеми

Із розширенням міжнародних інженерних команд зростає потреба в гарантуванні високої надійності програмного забезпечення. Розподілене робоче середовище, що характеризується різними культурними нормами, комунікаційними стилями та інженерними практиками, створює суттєві перешкоди для підтримки уніфікованої якості коду. Хоча практика code review визнана ключовим механізмом контролю якості, її ефективність у мультикультурних командах часто знижується. Проблема полягає в тому, що code review розглядають переважно як технічний інструмент, ігноруючи його важливу роль як культурного феномену, що базується на колективній відповідальності та взаємній довірі.

На сьогодні недостатньо вивчена кількісна та якісна кореляція між культурою code review та метриками надійності програмного продукту. Необхідно заповнити цю прогалину, застосовуючи аналогії з науковим методом, зокрема з концепціями peer review та відтворюваності, щоб довести, що зріла культура code review є не просто рекомендованою практикою, а критично вагомим чинником, що передбачає надійність, яку можна порівняти з безпекою в інших інженерних дисциплінах. Це дослідження має на меті переосмислити code review як інструмент наукової валідації в рамках програмної інженерії, що дасть змогу розробити ефективніші стратегії для підвищення якості в глобальних командах.

Аналіз останніх досліджень і публікацій

У теперішню епоху створенням програмного забезпечення все частіше займаються великі міжнародні команди, що ставить на перше місце питання надійності. Для гарантування високої якості та безпеки програмна інженерія має запозичувати підходи з класичних інженерних дисциплін, де надійність є ключовою. У цьому контексті культура code review виходить за межі суто технічного процесу, стаючи фундаментальним механізмом, що передбачає стабільність. Ця практика багато в чому подібна до наукового рецензування, що підтверджує достовірність наукових знань.

Учені активно продовжують пошуки оптимальних підходів до code review. Так, E. Witter dos Santos та I. Nunes [1] досліджували ефективність code review у розподілених командах, наголошуючи, що, попри технічний характер, ця практика є важливим інструментом для обміну знаннями. Схожу методологію застосували D. I. De Silva, W. A. C. Pabasara, S. V. Sangkavi, L. G. A. T. D. Wijerathne, W. M. K. H. Wijesundara та Reezan [2] під час вивчення впливу рецензування коду на якість програмного забезпечення, з'ясувавши, що це дієвий метод для виявлення багів та покращення читабельності коду.

На окрему увагу заслуговують праці, присвячені викликам у міжнародних командах. M. Hoffmann, D. Mendez, F. Fagerholm та A. Luckhardt [3] зосередилися на людському аспекті командної роботи, а D. Welsch, D. Mötefindt, L. Burk та M. Neumann [4] визначили, як культурні відмінності можуть створювати бар'єри. B. Zaghloul, D. Riehle та M. Zhou [5] окреслили особливості комунікації в глобальних командах із Китаєм, а C. Miller, P. Rodeghero, M-A. Storey, D. Ford та T. Zimmermann [6] аналізували виклики, які виникли під час віддаленої роботи в умовах пандемії. Важливу цінність має дослідження P. Ofem, B. Isong та F. Lugayizi [7] щодо прозорості в інженерії програмного забезпечення, що є ключовим для формування довіри.

Для оцінювання надійності програмного забезпечення активно використовують чіткі метрики. Науковці P. Haindl та R. Plösch [10] наголошують на важливості таких показників, як час відновлення, частота відкатів та кількість критичних інцидентів. Ці метрики дають змогу командам проактивно вимірювати ефективність своїх практик. M. Jureczko, Ł. Kajda та P. Górecki [11] підтвердили, що якість процесу рецензування безпосередньо впливає на кількість дефектів, які потрапляють у реліз.

Наукову цінність має публікація R. Li, M. Soliman, P. Liang та P. Avgeriou [12], присвячена виявленню симптомів архітектурної ерозії на ранній стадії за допомогою code review. Варта уваги робота F. Touré, M. Badri та L. Lamontagne [13] щодо модульного тестування, яке по суті є мікроекспериментом. R. B. Silva та C. Bezerra [14] досліджували вплив шкідливих практик безперервної інтеграції на якість програмного забезпечення.

Цікаві паралелі з науковою відтворюваністю провели M. H. Touati та A. R. Moanassar [15], яківивчали, як обчислювальні контейнери та безперервна інтеграція можуть покращити відтворюваність результатів наукових досліджень.

Формулювання мети дослідження

Метою статті є систематизація та верифікація емпіричних даних щодо кореляційного взаємозв'язку між інженерною культурою code review та рівнем надійності програмного забезпечення в контексті функціонування великих міжнародних команд.

Викладення основного матеріалу дослідження

У сучасну епоху розробкою програмного забезпечення все частіше займаються великі міжнародні команди, що ставить на перше місце питання надійності. Для гарантування високої якості, безпеки програмного забезпечення та процесів його створеннякомп'ютерна інженерія має запозичувати підходи з класичних інженерних дисциплін, де надійність є ключовою. У цьому контексті культура code review виходить за межі суто технічного процесу, стаючи фундаментальним механізмом, що гарантує стабільність. Ця практика багато в чому подібна до наукового рецензування, що підтверджує достовірність наукових знань.

Modern code review еволюціонував від більш формалізованих практик минулого, як-от інспекція коду, що потребувала офіційних зустрічей, до менш номінальних та гнучкіших процесів. Сьогоднішній підхід часто підтримується спеціалізованими інструментами. Попри технічну природу, він є значно вагомішим, ніж простою перевіркою синтаксису чи виявленням багів. Це визнаний спосіб сприяти обміну знаннями, що приносить користь як авторам, так і рецензентам [1, с. 28]. По суті, code review покращує командну взаємодію, оскільки створює колективну відповідальність за код, перетворюючи його з результату індивідуальної роботи на спільне надбання команди. Зазначене формує спільноту, у якій взаємоповага, довіра та спільні знання стають основою для підвищення загальної якості програмного продукту.

Для повнішого розуміння глибини code review як процесу, його варто розглядати крізь призму академічного рецензування. Подібно до того, як наукова рецензія спрямована на оцінювання методів, які використовує дослідник, code review дає змогу команді охарактеризувати архітектурні рішення та загальну стратегію розробки, перевіряючи її відповідність стандартам. Якщо метою наукової рецензії є розкриття помилок у дослідженні, то code review ефективно виявляє баги та інші потенційні проблеми в програмному коді ще до його інтеграції в основну кодову базу [2, с. 8].

Принцип роботи code review як проактивного бар'єру проти архітектурної ерозії візуалізовано на рисунку 1.

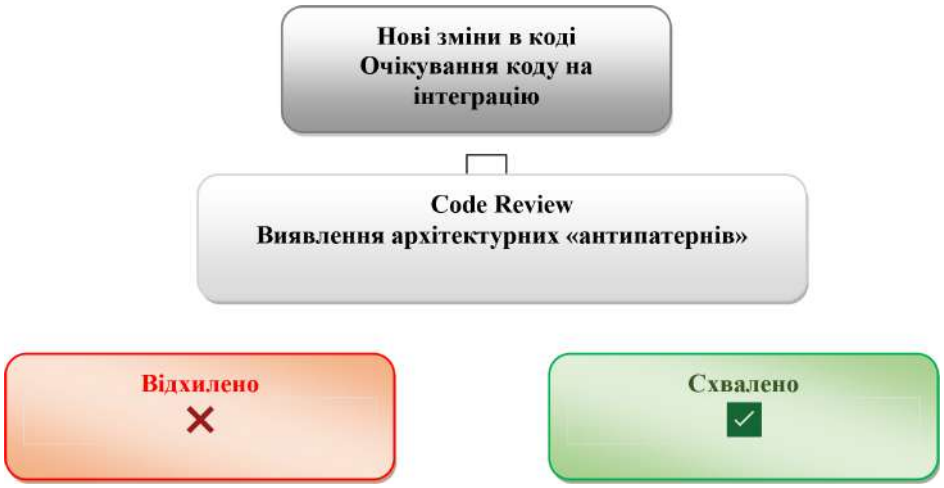

Рис. 1. Бар'єр від архітектурної ерозії

Джерело: сформовано автором

Окреслена схема наочно ілюструє, як code review виступає в ролі критично важливого контрольного пункту в процесі розробки. Вона показує, як цей механізм фільтрує небажані архітектурні рішення та потенційні проблеми, не дозволяючи їм інтегруватися в основну кодову базу. Таким чином, рецензування коду діє як проактивний захист від накопичення технічного боргу та архітектурної ерозії, забезпечуючи довготривалу стабільність і надійність програмного продукту.

Зрештою, це призводить до значного поліпшення якості коду та його надійності. Цей процес не тільки допомагає знаходити дефекти, але й робить код більш зрозумілим та легким для подальшої модифікації, що безпосередньо покращує його загальну якість. Два виміри цього впливу, технічний та культурний, які сукупно визначають надійність програмного забезпечення, візуалізовано на рисунку 2.

Як видно з рис. 2, культура рецензування коду має подвійний вплив на показники надійності. По-перше, вона здійснює прямий технічний вплив завдяки виявленню дефектів та їх усуненню ще до виходу продукту в реліз, що безпосередньо сприяє підвищенню якості коду. По-друге, культура рецензування виконує важливу соціально-культурну функцію, формуючи здорову командну взаємодію, яка ґрунтується на довірі, відкритій комунікації та колективній відповідальності.

Сукупність цих двох аспектів визначає ключові показники надійності, зокрема кількість багів після релізу, частоту відкатів та рівень критичних інцидентів. Отже, модель демонструє, що інвестиції у становлення зрілої

Рис. 2. Подвійний вплив Code Review

Джерело: сформовано автором

культури code review є визначальним чинником для забезпечення довгострокової стабільності та високої якості програмного продукту.

У міжнародних командах, де члени мають різну національність, виникають значні виклики, які позначаються на ефективності комунікації та взаємодії [3, с. 3]. Культурні відмінності впливають на те, як ми думаємо та діємо, і можуть створювати бар'єри для якості роботи команди та кінцевого продукту [4].

Одне з головних ускладнень – це різне розуміння критики та зворотного зв'язку, що може впливати на процес code review [3, с. 12]. Наприклад, в одних культурах прямий і чесний фідбек сприймається як конструктивний, тоді як в інших його розцінюють як особисту образу, що призводить до проблеми «збереження обличчя» (face-saving) та уникнення відкритого обговорення [5, с. 136]. Крім того, у глобальному розподіленому середовищі відсутність неформальних розмов, як-от «розмови біля кулера», може спричинити відчуття роз'єднаності та зниження соціальних зв'язків [6]. Це підкреслює необхідність глибокого осмислення культурних відмінностей у мультикультурних командах [4].

Для вирішення зазначених проблем критично важливим є впровадження спільних метрик та прозорих практик. Прозорість, яку розглядають як розкриття інформації про програмні продукти та процеси, є ключовою для створення довіри та впевненості зацікавлених сторін під час розробки програмного забезпечення [7, с. 33]. Це допомагає не лише підвищити якість, але й сприяє подоланню культурних розбіжностей, щоб кожен член команди мав спільне розуміння цілей та операцій.

Варто акцентувати, що якщо не враховувати культурні виклики в міжнародних командах, вони можуть значно знизити ефективність code review, у такий спосіб підриваючи якість коду. Отже, усвідомлене формування прозорої та відкритої культури рецензування є значно вагомим для подолання окресленого бар'єру. Це переводить нас до наступного визначального поняття – надійності, яка є безпосереднім наслідком зрілих інженерних практик. У цьому сенсі програмна інженерія має запозичувати підходи з класичних дисциплін, як-от авіація чи будівництво, де надійність є синонімом безпеки. Несправність програмного забезпечення, подібно до відмови механізму літака чи руйнування мосту, може мати катастрофічні наслідки, що свідчить про те, що в сучасному світі надійність програмного забезпечення є не просто технічною характеристикою, а фундаментальним атрибутом безпеки. Щоб оцінити надійність програмного забезпечення, що є ключовим атрибутом якості та безпеки, слід використовувати чіткі метрики. Їх відстеження дає змогу ухвалювати обґрунтовані рішення щодо необхідних оперативних коригувань, виправлення помилок та вдосконалення коду в подальших фазах розробки. Найбільш важливі метрики надійності представлено нижче в таблиці 1.

Отже, представлені метрики надають якісні показники для оцінювання надійності програмного забезпечення. Їхнє використання дає змогу командам не просто реагувати на збої, а й проактивно вимірювати ефективність своїх інженерних практик, зокрема code review, таким чином гарантуючи стабільність та високу якість продукту. Адже саме якість процесу рецензування безпосередньо впливає на кількість дефектів, що потрапляють у реліз, а отже, і на частоту відкатів та критичних інцидентів. Дієвий code review, що знаходить функціональні помилки

Таблиця 1

Основні метрики надійності програмного забезпечення

Метрика	Сутність	Значення
Кількість багів після релізу	Кількість дефектів, виявлених кінцевими користувачами або в робочому середовищі після розгортання продукту	Прямий індикатор якості, що відображає ефективність виявлення помилок на ранніх етапах розробки
Mean Time to Recovery (MTTR)	Середній час, необхідний для повного відновлення роботи системи після збою або інциденту	Чим менше значення, тим вища стійкість і надійність системи
Частота відкатів (Rollbacks)	Кількість випадків, коли новий реліз довелося відкотити назад через критичні помилки	Висока частота свідчить про низьку якість тестування та перевірки перед розгортанням
Кількість критичних інцидентів	Загальна кількість серйозних збоїв системи, що мали значний вплив на її роботу за певний період	Фундаментальний показник надійності та стабільності продукту

Джерело: сформовано на основі аналізу [8–10].

ще до інтеграції коду, допомагає мінімізувати подальші ризики, забезпечуючи стабільність та високу якість продукту [11, с. 804].

Ефективність code review не обмежується лише пошуком явних дефектів. Дослідження показують, що він також є цінним інструментом для виявлення ранніх симптомів архітектурної ерозії – поступового погіршення структури програмного забезпечення [12]. Це може значно ускладнити подальшу підтримку та еволюцію системи, спричинивши довготривалі проблеми з надійністю та стабільністю. Тож рецензування коду дає змогу розкрити такі архітектурні «антипатерни» на ранній стадії, перш ніж вони призведуть до серйозних наслідків. Завдяки цьому code review перетворюється на проактивний механізм, який не тільки знаходить помилки, а й підтримує загальну здорову «архітектуру» проєкту, що є запорукою його довгострокового успіху.

Зазначений проактивний підхід переносить нас до ширших емпіричних паралелей між науковим методом та програмною інженерією. Одним із найяскравіших прикладів є модульне тестування (unit testing), що по суті є мікроекспериментом, де кожен тест перевіряє найменшу гіпотезу про поведінку окремого фрагмента коду [13, с. 70]. Це допомагає розробникам системно та науково підтверджувати правильність роботи найменших компонентів, закладаючи міцний фундамент надійності, що є ключовим для подальшого масштабування та розвитку проєкту.

Якщо модульне тестування можна порівняти з мікроекспериментом, то інтеграційне тестування має свою аналогію в системній біології. Системна біологія вивчає, як взаємодіють різні елементи всередині складної біологічної системи, а інтеграційне тестування фокусується на дослідженні взаємодії між окремими модулями програмного забезпечення. Воно дає змогу виявляти помилки, що виникають на стиках компонентів, і гарантувати, що їхня спільна робота відповідає загальним вимогам, забезпечуючи у такий спосіб цілісність та стабільність усієї системи [14].

Також варто наголосити, що code review є не лише механізмом перевірки коду, а й має глибокі паралелі з академічним рецензуванням (peer review), що є основою для верифікації наукових знань. Як і наукова рецензія, що оцінює методи дослідження та виявляє помилки, code review допомагає команді колективно встановлювати недоліки, підвищувати читабельність коду та забезпечувати його відповідність стандартам [2, с. 4]. Така практика сприяє розкриттю та запобіганню дефектів, що є критично важливим для поліпшення якості програмного забезпечення. По суті, code review є колективною перевіркою валідності та надійності коду, перетворюючи його з індивідуальної роботи на спільне надбання, за яке несе відповідальність уся команда.

Продовжуючи емпіричні паралелі, варто зауважити, що безперервну інтеграцію (Continuous integration) можна порівняти з науковою відтворюваністю. Подібно до того, як відтворюваність у науці гарантує, що експеримент, проведений у різних умовах, дасть ідентичні результати, безперервна інтеграція забезпечує стабільність та відтворюваність програмного продукту [15, с. 32]. Завдяки використанню таких інструментів, як обчислювальні контейнери, CI автоматизує процеси компіляції, збірки та виконання тестів, гарантуючи, що код поводитиметься однаково в будь-якому середовищі розробки, тестування чи розгортання. Це є дуже важливим для високої якості продукту та мінімізації помилок на всіх етапах розробки.

Загалом у контексті сучасної програмної інженерії ключові практики розробки можуть бути осмислені через призму фундаментальних принципів наукового методу. Окреслені паралелі візуалізує наступна схема (рис. 3).

Такий підхід не тільки підвищує надійність та якість програмного продукту, але й сприяє формуванню зрілої інженерної культури, яка проактивно запобігає проблемам.

Проведені паралелі свідчать про те, що:

– модульне тестування (Unit testing) є аналогом мікроексперименту, де кожен тест перевіряє найменшу гіпотезу щодо поведінки окремого компонента;

– інтеграційне тестування (Integration testing) можна порівняти з підходом системної біології, що досліджує взаємодію між елементами складної системи;

– рецензування коду (Code review) є формою академічного рецензування (peer review), що забезпечує колективну верифікацію та підвищення достовірності програмного коду;

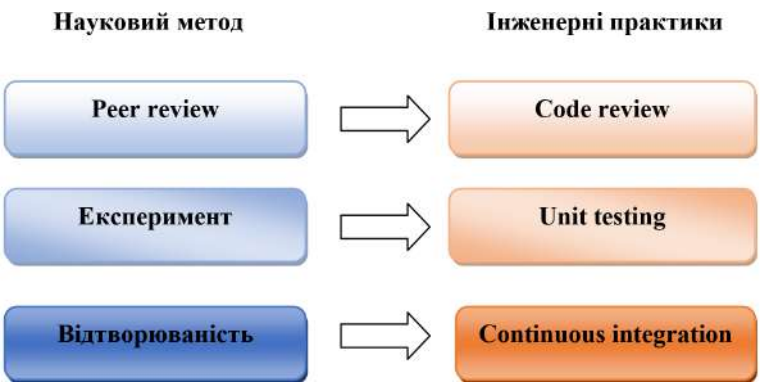

Рис. 3. Методологічні паралелі між науковим методом і практиками розробки програмного забезпечення

Джерело: сформовано автором

– безперервна інтеграція (Continuous integration) є втіленням принципу відтворюваності, що гарантує стабільність результатів незалежно від середовища.

Кожна з окреслених практик є невіддільним компонентом цілісного процесу, який дає змогу командам розробників не просто реагувати на збої, а системно вимірювати та покращувати ефективність своїх інженерних рішень.

Переходячи від окремих інженерних практик до їхньої сукупної дії, стає очевидним, що успішна розробка програмного забезпечення залежить не лише від технічної досконалості, але й від зрілості командної культури. Саме культура, зокрема в контексті рецензування коду (code review), є ключовим важелем, що перетворює індивідуальну роботу на спільне досягнення й безпосередньо впливає на надійність кінцевого продукту. Для візуалізації цих взаємозв'язків ми створили концептуальну модель, яка представлена на рис. 4.

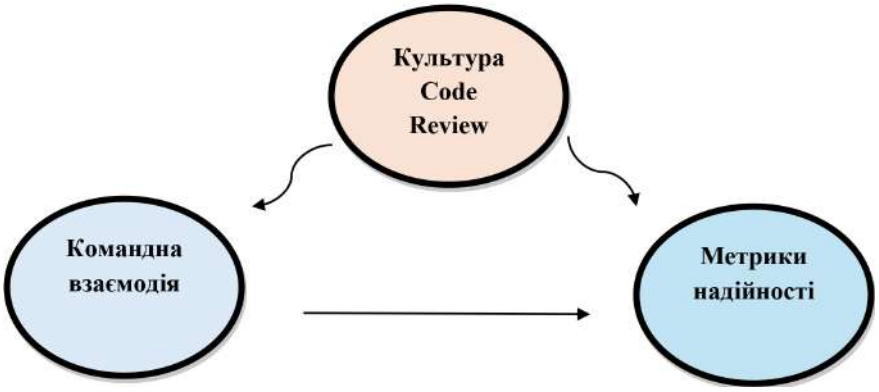

Рис. 4. Взаємозв'язок між інженерними практиками та надійністю програмного забезпечення

Джерело: сформовано автором

Як видно зі схеми, культура рецензування коду має подвійну дію на метрики надійності. По-перше, вона забезпечує прямий технічний вплив через виявлення дефектів та їх усунення ще до потрапляння в реліз. По-друге, має культурне значення, формуючи здорову командну взаємодію, що характеризується довірою, відкритою комунікацією та колективною відповідальністю. Ці два потоки сукупно визначають такі ключові показники надійності, як кількість багів після релізу, частоту відкатів та чисельність критичних інцидентів. Тож модель підкреслює, що інвестиції в становлення зрілої культури code review є дуже вагомими для довгострокової стабільності та якості продукту.

Висновки

Отже, сучасний підхід до розробки програмного забезпечення варто розглядати крізь призму наукового методу. Це дає змогу не просто виправляти помилки, а проактивно створювати дійсно надійні та якісні продукти. У ході роботи виявлено, що основні інженерні практики, такі як модульне та інтеграційне тестування, насправді мають багато спільного з науковими експериментами та дослідженнями.

Особливу увагу приділено рецензуванню коду, оскільки ця практика має подвійний вплив на надійність. З одного боку, це технічний процес, який допомагає зменшити кількість дефектів у коді. З іншого – потужний культурний інструмент, що сприяє обопільній довірі та поліпшує комунікацію в команді.

Щоб краще зрозуміти вказані співвідношення, продемонстровано модель взаємозв'язків між інженерними практиками та надійністю програмного забезпечення. Вона наочно показує, як зріла культура code review

безпосередньо діє на ключові показники надійності, такі як кількість багів після релізу або частоту відкатів. Отже, можна стверджувати, що інвестування у формування зрілої інженерної культури, яка базується на наукових принципах, є критично важливим для довгострокового успіху будь-якого проєкту.

Напрямами подальших досліджень, що випливають з отриманих висновків, є проведення детального кейс-стаді практик рецензування коду у великих міжнародних компаніях, таких як Google, Microsoft чи EPAM, для виявлення найкращих підходів, що забезпечують високу надійність. Також актуальним завданням є розроблення конкретних метрик для кількісного вимірювання «зрілості культури code review», що дасть змогу об'єктивно оцінювати її значення. Нарешті, дослідження впливу мультикультурності на стиль рецензування коду в розподілених командах надасть практичні рекомендації для подолання культурних бар'єрів та покращення комунікації, що є критично важливим для надійності в умовах глобалізації.

Список використаної літератури

1. Witter dos Santos E., Nunes I. Investigating the effectiveness of peer code review in distributed software development based on objective and subjective data. *Journal of Software Engineering Research and Development*. 2018. № 6(14). P. 1–31. DOI: https://doi.org/10.1186/s40411-018-0058-0

2. De Silva D. I., Pabasara W. A. C., Sangkavi S. V., Wijerathne L. G. A. T. D., Wijesundara W. M. K. H., Reezan. The Effectiveness of Code Reviews on Improving Software Quality: An Empirical Study. *International Journal of Recent Technology and Engineering*. 2023. № 12(2). P. 1-10. DOI: 10.35940/ijrte.B7666.0712223

3. Hoffmann M., Mendez D., Fagerholm F., Luckhardt A. The human side of Software Engineering Teams: an investigation of contemporary challenges. 2022. P. 1-18. URL: https://arxiv.org/abs/2104.03712

4. Welsch D., Mötefindt D., Burk L., Neumann M. Navigating Cultural Diversity: Barriers and Potentials in Multicultural Agile Software Development Teams. 2023. URL: https://arxiv.org/abs/2311.12061

5. Zaghloul B., Riehle D., Zhou M. Communication in Firm-Internal Global Software Development with China. *Software Business*. Vol. 2. P. 132-138. DOI: 10.1007/978-3-319-19593-3_11

6. Miller C., Rodeghero P., Storey M-A., Ford D., Zimmermann T. "How Was Your Weekend?" Software Development Teams Working From Home During COVID-19. 2021. URL: https://arxiv.org/abs/2101.05877

7. Ofem P., Isong B., Lugayizi F. Metrics for Evaluating and Improving Transparency in Software Engineering: An Empirical Study and Improvement Model. *SN Computer Science*. 2024. № 5(1097). P. 1-35. DOI: 10.1007/s42979-024-03471-3

8. Software.com. Engineering Metrics: Mean Time to Recovery. 2025. URL: https://www.software.com/engineering-metrics/mean-time-to-recovery

9. Forbes Councils. Minimizing The Impact Of Customers Finding Bugs. 2023. URL: https://www.forbes.com/councils/forbestechcouncil/2023/09/18/minimizing-the-impact-of-customers-finding-bugs

10. Haindl P., Plösch R. Value-oriented quality metrics in software development: Practical relevance from a software engineering perspective. *IET Software*. 2022. № 16(2). P. 167–184. DOI: 10.1049/sfw2.12051

11. Jureczko M., Kajda Ł., Górecki P. Code review effectiveness: an empirical study on selected factors influence. *IET Software*. 2021. № 14(7). P. 794-805. DOI: 10.1049/iet-sen.2020.0134

12. Li R., Soliman M., Liang P., Avgeriou P. Symptoms of Architecture Erosion in Code Reviews: A Study of Two OpenStack Projects. 2022. URL: https://arxiv.org/abs/2201.01184v1

13. Touré F., Badri M., Lamontagne L. Investigating the prioritization of unit testing effort using software metrics. In Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'17). 2017. P. 69–80. DOI: 10.5220/0006319300690080

14. Silva R. B., Bezerra C. Empirical investigation of the influence of continuous integration bad practices on software quality. *Virtual event*. 2022. URL: https://sol.sbc.org.br/index.php/vem/article/view/22330/22154

15. Touati M. H., Moanassar A. R. Using Computing Containers and Continuous Integration to Improve Numerical Research Reproducibility. *International Journal of Computer*. 2018. № 30(1). P. 27–33. DOI: https://doi.org/10.53896/ijc.v30i1.1249

References

1. Witter dos Santos, E., & Nunes, I. (2018). Investigating the effectiveness of peer code review in distributed software development based on objective and subjective data. *Journal of Software Engineering Research and Development*, 6(14), 1–31. https://doi.org/10.1186/s40411-018-0058-0

2. De Silva, D. I., Pabasara, W. A. C., Sangkavi, S. V., Wijerathne, L. G. A. T. D., Wijesundara, W. M. K. H., & Reezan. (2023). The Effectiveness of Code Reviews on Improving Software Quality: An Empirical Study. *International Journal of Recent Technology and Engineering*, 12(2), 1-10. https://doi.org/10.35940/ijrte.B7666.0712223

3. Hoffmann, M., Mendez, D., Fagerholm, F., & Luckhardt, A. (2022). *The human side of Software Engineering Teams: an investigation of contemporary challenges*. https://arxiv.org/abs/2104.03712

4. Welsch, D., Mötefindt, D., Burk, L., & Neumann, M. (2023). *Navigating Cultural Diversity: Barriers and Potentials in Multicultural Agile Software Development Teams*. https://arxiv.org/abs/2311.12061

5. Zaghloul, B., Riehle, D., & Zhou, M. (2015). Communication in Firm-Internal Global Software Development with China. In *Software Business*, 2, 132-138. Springer. https://doi.org/10.1007/978-3-319-19593-3_11

6. Miller, C., Rodeghero, P., Storey, M-A., Ford, D., & Zimmermann, T. (2021). "*How Was Your Weekend?*" *Software Development Teams Working From Home During COVID-19*. https://arxiv.org/abs/2101.05877

7. Ofem, P., Isong, B., & Lugayizi, F. (2024). Metrics for Evaluating and Improving Transparency in Software Engineering: An Empirical Study and Improvement Model. *SN Computer Science*, 5(1097), 1-35. https://doi.org/10.1007/s42979-024-03471-3

8. Software.com. (2025). *Engineering Metrics: Mean Time to Recovery*. https://www.software.com/engineering-metrics/mean-time-to-recovery

9. Forbes Councils. (2023). *Minimizing The Impact Of Customers Finding Bugs*. https://www.forbes.com/councils/forbestechcouncil/2023/09/18/minimizing-the-impact-of-customers-finding-bugs

10. Haindl, P., & Plösch, R. (2022). Value-oriented quality metrics in software development: Practical relevance from a software engineering perspective. *IET Software*, 16(2), 167–184. https://doi.org/10.1049/sfw2.12051

11. Jureczko, M., Kajda, Ł., & Górecki, P. (2021). Code review effectiveness: an empirical study on selected factors influence. *IET Software*, 14(7), 794-805. https://doi.org/10.1049/iet-sen.2020.0134

12. Li, R., Soliman, M., Liang, P., & Avgeriou, P. (2022). *Symptoms of Architecture Erosion in Code Reviews: A Study of Two OpenStack Projects*. https://arxiv.org/abs/2201.01184v1

13. Touré, F., Badri, M., & Lamontagne, L. (2017). Investigating the prioritization of unit testing effort using software metrics. In *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'17)*, 69–80. https://doi.org/10.5220/0006319300690080

14. Silva, R. B., & Bezerra, C. (2022). Empirical investigation of the influence of continuous integration bad practices on software quality. In *Virtual event*. SBC. https://sol.sbc.org.br/index.php/vem/article/view/22330/22154

15. Touati, M. H., & Moanassar, A. R. (2018). Using Computing Containers and Continuous Integration to Improve Numerical Research Reproducibility. *International Journal of Computer*, 30(1), 27–33. https://doi.org/10.53896/ijc.v30i1.1249

Дата першого надходження рукопису до видання: 20.09.2025
Дата прийнятого до друку рукопису після рецензування: 15.10.2025
Дата публікації: 28.11.2025