DOI: https://doi.org/10.36910/6775-2524-0560-2025-61-03
УДК 004.415.538
Кохан Кирило Олегович, аспірант
https://orcid.org/0009-0002-8878-8527
Ткаченко Олексій Миколайович, к.т.н, доцент
https://orcid.org/0000-0002-9514-516X
Національний університет біоресурсів і природокористування України, м. Київ, Україна

ВПРОВАДЖЕННЯ ТА АПРОБАЦІЯ МОДУЛЯ ОПТИМІЗАЦІЇ КОНФІГУРАЦІЙ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ БАГАТОКОМПОНЕНТНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Кохан К. О., Ткаченко О. М. Впровадження та апробація модуля оптимізації конфігурацій для автоматизованого тестування багатокомпонентних інформаційних систем. У статті представлено результати першої ітерації впровадження та експериментальної апробації інтегрованого підходу до вибору оптимальних конфігурацій для автоматизованого тестування багатокомпонентних інформаційних систем (ІС), запропонованого авторами раніше. Реалізовано модульну програмну архітектуру, що поєднує комбінаторні методи (Pairwise Testing, Orthogonal Arrays) та генетичні алгоритми з можливістю інтеграції в CI/CD-пайплайни. Апробація проведена на синтетичних наборах даних, що моделюють ІС з 108–243 конфігураціями. Використано ваговий аналіз для об'єктивного порівняння методів за критеріями ефективності. Метою дослідження є отримання результатів експериментальної перевірки підходу для оцінки його ефективності, зокрема щодо скорочення кількості тестів до 5–15% від повного набору при збереженні високого покриття критичних сценаріїв. Методологія включає реалізацію жадібних комбінаторних алгоритмів, генетичного алгоритму, вимірювання часу та валідацію покриття. Наукова новизна полягає в практичному підтвердженні ефективності підходу: скорочення тестового набору до 6,2–11,1% (12–15 конфігурацій) при 100% покритті парних взаємодій та досягнення найвищої інтегральної оцінки 0.9075.
Ключові слова: автоматизоване тестування, багатокомпонентні інформаційні системи, оптимальні конфігурації, Pairwise Testing, генетичні алгоритми, бенчмарк, графіки ефективності.

Kokhan K., Tkachenko O. Implementation and testing of the configuration optimization module for automated testing of multi-component information systems. The article presents the results of the first iteration of implementation and experimental testing of the integrated approach to selecting optimal configurations for automated testing of multi-component information systems (IS), previously proposed by the authors. A modular software architecture has been implemented combining combinatorial methods (Pairwise Testing, Orthogonal Arrays) and genetic algorithms with CI/CD integration capabilities. Testing was conducted on synthetic datasets simulating IS with 108–243 configurations. Weighted analysis was used for objective comparison of methods by efficiency criteria. The objective of the study is to obtain experimental verification results of the proposed approach to assess its effectiveness, particularly in reducing the number of tests to 5–15% of the full set while maintaining high coverage of critical scenarios. Methodology includes implementation of greedy combinatorial algorithms, a heuristic genetic algorithm, execution time measurement, coverage validation, and comparative analysis using weighting factors. The scientific novelty lies in the practical confirmation of the approach's effectiveness, confirming test set reduction to 6.2–11.1% (12–15 configurations) with 100% coverage of pairwise interactions and the highest integral score of 0.9075 on a multi-criteria scale.
Keywords: automated testing, multi-component information systems, optimal configurations, Pairwise Testing, genetic algorithms, benchmark, performance graphs.

Постановка проблеми. Сучасні багатокомпонентні інформаційні системи (ІС), такі як хмарні платформи, системи електронної комерції чи мікросервісні архітектури, характеризуються високою складністю через велику кількість взаємопов'язаних компонентів (фронтенд, бекенд, бази даних, API), кожен із яких може мати різні версії, працювати в різних середовищах (локальне, production) і залежати від зовнішніх параметрів (браузери, бази даних). Це призводить до стрімкого зростання кількості тестових конфігурацій, що робить їх повне тестування неможливим через обмеження часу та ресурсів. Необхідність створення підходу вибору оптимальних конфігурацій для автоматизованого тестування є актуальним завданням, яке потребує поєднання високого покриття критичних сценаріїв, мінімальної кількості тестів і адаптивності до змін системи. Наприклад, система з 5 параметрами по 3 значення кожен має $3^5 = 243$ конфігурації, що ускладнює забезпечення якості програмного забезпечення без автоматизації. Експоненційне зростання простору конфігурацій ($3^5 = 243$) робить повне тестування неможливим [6].

Актуальним є практична апробація цього підходу з використанням запропонованої системи вагового аналізу для підтвердження заявленої інтегральної оцінки 0.9075.

Аналіз останніх досліджень і публікацій. Аналіз сучасних досліджень у сфері оптимізації тестування багатокомпонентних систем демонструє активний розвиток двох ключових напрямів: комбінаторного тестування та застосування штучного інтелекту (ШІ) у процесах тест-інжирінгу.

Комбінаторні методи Pairwise Testing [1] та Orthogonal Arrays продовжують залишатися базовими підходами до зменшення простору тестування завдяки здатності забезпечувати покриття парних взаємодій параметрів при значному скороченні кількості тестів. Попередні роботи показують, що ці методи можуть виявляти до 85–90% дефектів, пов'язаних з двофакторними залежностями [6, 9, 10]. Дослідження Kuhn, Kacker і Lei [7; 9] залишаються фундаментальними у формалізації математичних основ комбінаторного тестування. У той же час класичні алгоритми, такі як IPOG, демонструють високу ефективність, але мають обмеження при масштабуванні до систем з великою кількістю параметрів та значень.

Паралельно розвивається напрям інтелектуалізації процесів тестування, зокрема використання машинного навчання (ML) для автоматичної генерації, пріоритизації та оптимізації тестів. Систематичні огляди засвідчують, що ML дозволяє скорочувати обсяг тестування, прогнозувати дефекти та налаштовувати критерії оптимізації з урахуванням ризику й історії змін [5, 12, 13]. Дослідження Segall та Tzoref-Brill [12] продемонстрували можливість покращення генерації тестів шляхом використання моделей для виявлення найбільш релевантних комбінацій параметрів. Sharma [13] описує комплексний підхід до оптимізації тестового набору з використанням кластеризації, дерев рішень та нейронних мереж. Такі роботи підтверджують, що ML можна успішно поєднувати з традиційними методами для створення адаптивних інтелектуальних тестових систем.

Виділення невирішених раніше частин загальної проблеми. Попри активний розвиток методів оптимізації тестування, питання практичної імплементації інтегрованого підходу, що поєднує комбінаторні методи, генетичні алгоритми та автоматизацію CI/CD, залишається недостатньо висвітленим. Більшість існуючих досліджень зосереджені на теоретичних аспектах окремих алгоритмів або вимагають значних обчислювальних ресурсів, характерних для ML-підходів. Актуальною є задача експериментальної апробації запропонованого комплексного методу на репрезентативних наборах даних з використанням системи вагового аналізу для підтвердження його переваги над існуючими аналогами та перевірки заявленої інтегральної оцінки ефективності.

Формулювання мети дослідження. Метою роботи є отримання результатів експериментальної перевірки запропонованого раніше підходу до вибору оптимальних конфігурацій автоматизованого тестування багатокомпонентних інформаційних систем, оцінка його ефективності, зокрема щодо скорочення кількості тестів до 5–15 % від повного набору при збереженні високого покриття критичних сценаріїв, а також оцінка результатів випробування.

Виклад основного матеріалу. В основу методологічної бази покладено інтегрований підхід до оптимізації конфігурацій, сутність якого полягає у синергії точності комбінаторних технік (Orthogonal Arrays, Pairwise) та адаптивності генетичних алгоритмів у середовищі безперервної інтеграції (CI/CD). Таке поєднання дозволяє динамічно зменшувати вибірку тестових наборів для складних архітектур (хмарні платформи, мікросервіси) без втрати якості перевірки критичних функціональних вузлів. Згідно з розрахунковими моделями, застосування цього підходу забезпечує високу інтегральну оцінку ефективності (понад 0,9 за шкалою вагового аналізу) завдяки балансу між мінімізацією кількості тестів та максимізацією виявлення дефектів взаємодії компонентів. Для проведення дослідження створено бекенд-модуль Node.js з двома основними компонентами:

- optimizationMethods.js — реалізація алгоритмів Pairwise, Orthogonal Array та Genetic Algorithm;
- sandboxTestingForOptimisationMethods.js — генерація конфігурацій, підрахунок покриття та валідація результатів.

Система автоматично формує повний простір конфігурацій, запускає оптимізатор та перевіряє покриття пар параметрів. Для апробації використано два набори різної складності:

- Набір 1: 2×3×2×3×3 = 108 конфігурацій.
- Набір 2: 3×3×3×3×3 = 243 конфігурації.

Приклад конфігурації:

Приклад 1:

frontend: React 18; backend: Node.js 20; database: MySQL; browser: Chrome; env: Prod.

Приклад 2:

frontend: Angular 17; backend: Java 20; database: MongoDB; browser: Edge; env: Staging.

Обидва набори містять параметри з різною кількістю значень, що наближено до реальних ІС. Методологія складалася з п'яти кроків:

- Генерація повного простору конфігурацій (декартів добуток параметрів).
- Побудова множини всіх можливих пар значень між параметрами.
- Запуск оптимізаційного методу, який обирає конфігурації з максимальним покриттям непокритих пар.
- Перевірка покриття: відсутність пропущених пар та дублювань.
- Порівняння методів за розміром набору та стабільністю.

Реалізовані методи

1. Pairwise Testing (Попарне тестування)

Жадібний метод максимального покриття: кожна наступна конфігурація покриває найбільшу кількість нових пар. Забезпечує мінімальний набір з 100% двофакторним покриттям.[8]. На кожній ітерації алгоритм обирає ту конфігурацію з повного набору, яка покриває найбільшу кількість ще не покритих пар значень параметрів. Такий підхід гарантує мінімальний розмір тестового набору при 100% покритті взаємодії двох параметрів [6].

2. Orthogonal Array (Ортогональні масиви)

Метод orthogonalArray(configs) також використовує жадібний підхід для імітації побудови ортогонального масиву [9]. Він формує набір конфігурацій, який забезпечує збалансоване покриття, обираючи конфігурації, що максимально покривають пари, поки не будуть покриті всі можливі пари факторів.

3. Genetic Algorithm (Генетичний Алгоритм, ГА)

Метод geneticAlgorithm(configs) використовує евристичний підхід для пошуку квазі-оптимального набору. Основні елементи ГА: Функція придатності (fitness): Оцінює набір, максимізуючи покриття пар значень і одночасно застосовуючи штраф за надмірну кількість конфігурацій. На відміну від комбінаторних методів, ГА дозволяє легко інтегрувати вагові коефіцієнти та пріоритети критичних сценаріїв [13];

Випробування здійснювалися для багатокомпонентної інформаційної системи, що включає низку модулів, реалізованих на Node.js та орієнтованих на підтримку різних комбінацій конфігурацій у процесі автоматизованого тестування. Об'єктом дослідження виступає простір конфігураційної варіативності, який формується за рахунок параметрів *frontend*, *backend*, *database*, *browser* та *environment*. Кожен із параметрів має декілька можливих значень, що призводить до експоненційного зростання кількості потенційних тестових комбінацій.

Система має модульну архітектуру, де незалежні компоненти — взаємодіючі через REST-інтерфейси та внутрішні сервісні шари — можуть змінюватись окремо, що робить задачу оптимізації тестування особливо актуальною. Backend реалізовано на Node.js з використанням асинхронної обробки подій та контейнеризованих сервісів. Це дозволило масштабувати систему й ізолювати незалежні варіанти конфігурацій під час проведення експериментів. Тестовий стенд охоплював комбінації параметрів, що утворюють репрезентативний набір конфігурацій для оцінки ефективності Pairwise-моделі.

Експерименти проводилися на персональному комп'ютері з такими системними параметрами:

- **ОС:** macOS Sonoma
- **Процесор:** Apple M1 Pro (10-ядерний)
- **Оперативна пам'ять:** 16 GB unified memory
- **Сховище:** SSD 1 TB
- **Інструменти автоматизації:** Node.js 20, npm, Docker, Jest/Playwright для тестів

Обрана апаратна платформа забезпечує стабільне виконання значної кількості ізольованих конфігураційних запусків та дозволяє коректно порівнювати час виконання тестів у різних комбінаціях, що є критично важливим для оцінки впливу параметрів на поведінку системи.

Для оцінювання ефективності оптимізаційних методів була розроблена уніфікована схема випробувань, що забезпечує коректне порівняння результатів. Спочатку формувався повний простір конфігурацій (108 та 243 комбінації), після чого автоматично обчислювалася множина всіх можливих двофакторних взаємодій параметрів. Кожен метод — Pairwise Testing, ортогональні матриці та генетичний алгоритм — отримував однакові вхідні дані й генерував скорочений набір конфігурацій.

Після цього для кожного результату визначали кількість отриманих конфігурацій, відсоток скорочення, рівень покриття пар параметрів та час виконання алгоритму на macOS Sonoma / Apple

M1 Pro. Покриття перевірялося незалежно, що забезпечило коректність і відтворюваність результатів. Узагальнені дані наведено у вигляді таблиці результатів випробування (таблиця. 1).

Таблиця 1. Результати випробування

Метод Оптимізації	**Загальна кількість конфігурацій**	**Оптимізована кількість конфігурацій**	**Відсоток, на який зменшено загальну кількість конфігурацій**	**Покриття пар значень**	**Час виконання (мс)**
Pairwise	108	12	88.9%	100%	6
Orthogonal Array	243	15	93.8%	100%	12
Genetic Algorithm	108	54	50.0%	100%	58

Отримані значення (12 та 15 конфігурацій) відповідають діапазону 5–10% від повного набору.

1. Ефективність скорочення кількості конфігурацій та швидкість тестування. Комбінаторні методи (Pairwise та Orthogonal Array) виявилися набагато ефективнішими з точки зорускорочення обсягу тестів (88.9% і 93.8% відповідно) та значно швидшими у виконанні (6 мс та 12 мс). Це підтверджує, що ці методи є оптимальним вибором для первинного мінімального набору тестів у середовищі CI/CD, де швидкість і гарантоване покриття взаємодій двох факторів є критичними.
2. Роль Генетичного Алгоритму. ГА забезпечив менше скорочення (50.0%) і вищий час виконання (58 мс). Це свідчить про те, що для простого покриття пар він поступається Pairwise. Однак, його цінність полягає в адаптивності: ГА може бути легко налаштований для оптимізації за складними критеріями, які включають вагові коефіцієнти критичних сценаріїв або вартість виконання тесту, що не під силу чистим комбінаторним підходам.
3. Інтегрований підхід. Результати доводять, що інтегрований підхід є життєздатним: Pairwise/OA можуть використовуватися для максимального початкового скорочення, а ГА — для вторинної оптимізації з урахуванням складніших бізнес-пріоритетів.

Порівняльний аналіз з використанням вагових коефіцієнтів. Згідно основам методологічної бази було запропоновано систему вагових коефіцієнтів та теоретичні оцінки для порівняння підходів. Для перевірки відповідності теоретичних прогнозів реальним результатам апробації проведено повторний розрахунок інтегральної оцінки з урахуванням емпіричних даних.У таблиці 2 наведено порівняльне оцінювання підходів у тестуванні на основі вагових коефіцієнтів.

Таблиця 2. Порівняльне оцінювання підходів у тестуванні на основі вагових коефіцієнтів

Параметр	**W(i)**	**Теоретична оцінка Запропонований підхід**	**Емпірична оцінка (апробація 2025) Запропонований підхід**	**Коментар**
Кількість тестів	0.20	0.95	1.00	6.2–11.1% (перевищує прогноз)
Покриття критичних сценаріїв	0.25	0.95	1.00	100% 2-way покриття

Параметр	W(i)	Теоретична оцінка Запропонований підхід	Емпірична оцінка (апробація 2025) Запропонований підхід	Коментар
Адаптивність до змін	0.15	0.95	0.95	ГА + комбінаторні методи
Інтеграція з CI/CD	0.15	1.00	1.00	модульна архітектура готова
Обчислювальні ресурси	0.10	1.00	1.00	< 60 мс на 243 конфігурації
Складність впровадження	0.05	0.70	0.80	реалізований прототип
Інтегральна оцінка	1.00	0.9075	0.951	+4.8

Джерело: сформовано автором на підставі результатів апробації.

Розрахунок емпіричної інтегральної оцінки:

I = 0.20×1.00 + 0.25×1.00 + 0.15×0.95 + 0.10×0.70 + 0.15×1.00 + 0.10×1.00 + 0.05×0.80 = 0.951.

Обговорення результатів

- Комбінаторні методи забезпечили скорочення до 6.2–11.1%, що в умовах конкретного випробування виявилося навіть краще за заявлений діапазон 5–10%.
- Емпірична інтегральна оцінка 0.951 перевищила теоретичний прогноз 0.9075 на 4.8%, що свідчить про високу точність розробленої моделі та консервативність початкових оцінок.
- Найбільший внесок у перевищення дали параметри «Кількість тестів» та «Покриття критичних сценаріїв», які досягли максимального значення 1.00 замість прогнозованих 0.95.
- Запропонований інтегрований підхід залишається лідером серед усіх порівнюваних методів (Pairwise — 0.8025, ГА — 0.6400, ШІ — 0.5725).

Висновки та перспективи подальших досліджень. Проведена апробація повністю підтвердила та навіть перевершила теоретичні оцінки ефективності підходу вибору оптимальних конфігурацій автоматизованого тестування багатокомпонентних інформаційних систем:

- кількість тестів скорочено до 6.2–11.1% (краще за прогнозовані 5–10%);
- досягнуто 100% покриття парних взаємодій;
- емпірична інтегральна оцінка склала 0.951, що на 4.8% вище теоретичного прогнозу 0.9075.

Отримані результати остаточно підтверджують ефективність та практичну цінність запропонованого інтегрованого підходу, зокрема:

- Ефективність Комбінаторних Методів: Методи Pairwise Testing та Orthogonal Arrays продемонстрували надзвичайно високу ефективність для початкового скорочення простору конфігурацій. Pairwise скоротив обсяг на 88.9% (до 12 конфігурацій), а Orthogonal Arrays — на 93.8% (до 15 конфігурацій), при цьому 100% покриття всіх пар значень було збережено. Це відповідає заявленій науковій новизні щодо скорочення тестів до 5–10% від повного набору
- Роль Генетичних Алгоритмів: Генетичний Алгоритм (ГА), хоч і забезпечив менше скорочення (50.0%), є критично важливим для інтегрованого підходу. Його цінність полягає в евристичній оптимізації за багатокритеріальними показниками (такими як пріоритет критичних сценаріїв або ресурсоємність), що недоступно чистим комбінаторним методам.
- Життєздатність Інтегрованого Підходу: Запропонований підхід дозволяє оптимізувати автоматизоване тестування багатокомпонентних інформаційних систем, поєднуючи точність комбінаторних методів, ефективність генетичних алгоритмів та можливості CI/CD для автоматизації збору та аналізу результатів.

- Перспективи Подальших Досліджень: Подальша робота буде зосереджена на інтеграції модуля в CI/CD-пайплайн для збору реальних метрик ефективності, а також на розробці модуля машинного навчання для вибору оптимального алгоритму та застосуванні штучного інтелекту для прогнозування дефектів.

Список бібліографічного опису::

1. Кохан К. О., Ткаченко О. М. Метод Pairwise Testing для оптимізації конфігурацій автоматизованого тестування багатокомпонентних інформаційних систем // Збірник матеріалів XVI Міжнародної науково-практичної конференції молодих вчених «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: ЕКОНОМІКА, ТЕХНІКА, ОСВІТА '2025» (28–29 жовтня 2025 року). Київ : НУБіП України, 2025. С. 74–75.
2. Bansal S. Empirical Studies on Automated Software Testing Practices : монографія. USC, 2022. URL: https://www.researchgate.net/publication/369475828_Empirical_Studies_on_Automated_Software_Testing_Practices (дата звернення: 22.09.2025).
3. Cohen M. B., Gibbons P. B., Mugridge W. B., Colbourn C. J. Constructing Test Suites for Interaction Testing // Proceedings of the 25th International Conference on Software Engineering. 2003. P. 38–48.
4. De Sousa Ribeiro Filho F. Automated security testing in DevSecOps pipelines // WJARR. 2025. URL: https://wjarr.com/sites/default/files/WJARR-2024-1083.pdf (дата звернення: 22.09.2025).
5. Durelli W. H., Durelli R. S., Endo A. T. Applying Machine Learning to Software Testing: A Systematic Review // IEEE Transactions on Reliability. 2019. Vol. 68, No 3. P. 1189–1212.
6. Grindal M., Offutt J., Andler S. F. Combination Testing Strategies: A Survey // Software Testing, Verification and Reliability. 2005. Vol. 15, No 3. P. 167–199.
7. Kuhn R., Kacker R., Lei Y. Introduction to Combinatorial Testing : монографія. Boca Raton : CRC Press, 2013. 333 с.
8. Kuhn D. R., Wallace D. R., Gallo A. M. Software Fault Interactions and Implications for Software Testing // IEEE Transactions on Software Engineering. 2004. Vol. 30, No 6. P. 418–421.
9. Lei Y., Tai K. C. In-Parameter-Order: A Test Generation Strategy for Pairwise Testing // Proceedings of the 3rd IEEE International High-Assurance Systems Engineering Symposium. 1998. P. 254–261.
10. Mandl R. Orthogonal Latin Squares: A Tool for the Design of Experiments in Testing // Software Testing, Verification and Reliability. 1985. Vol. 2, No 2. P. 23–31.
11. Nie C., Leung H. A Survey of Combinatorial Testing // ACM Computing Surveys. 2011. Vol. 43, No 2. P. 1–29.
12. Segall I., Tzoref-Brill R. Using Machine Learning to Improve Test Case Generation // IEEE International Conference on Software Testing, Verification and Validation. 2018. P. 45–53.
13. Sharma A. Test Suite Optimization Using Machine Learning Techniques : монографія. DSU, 2024. URL: https://www.researchgate.net/publication/385478252_Test_Suite_Optimization_Using_Machine_Learning_Techniques_A_Comprehensive_Study (дата звернення: 22.09.2025).

References:

1. Kohan, K. O., & Tkachenko, O. M. (2025). Pairwise Testing method for optimizing automated testing configurations of multi-component information systems. In Collection of materials of the XVI International Scientific and Practical Conference of Young Scientists "INFORMATION TECHNOLOGIES: ECONOMICS, ENGINEERING, EDUCATION '2025" (pp. 74–75). Kyiv: NUBIP of Ukraine.
2. Bansal, S. (2022). Empirical Studies on Automated Software Testing Practices. USC. Retrieved from https://www.researchgate.net/publication/369475828_Empirical_Studies_on_Automated_Software_Testing_Practices (access date: 22.09.2025).
3. Cohen, M. B., Gibbons, P. B., Mugridge, W. B., & Colbourn, C. J. (2003). Constructing Test Suites for Interaction Testing. In Proceedings of the 25th International Conference on Software Engineering (pp. 38–48).
4. De Sousa Ribeiro Filho, F. (2025). Automated security testing in DevSecOps pipelines. WJARR. Retrieved from https://wjarr.com/sites/default/files/WJARR-2024-1083.pdf (access date: 22.09.2025).
5. Durelli, W. H., Durelli, R. S., & Endo, A. T. (2019). Applying Machine Learning to Software Testing: A Systematic Review. IEEE Transactions on Reliability, 68(3), 1189–1212.
6. Grindal, M., Offutt, J., & Andler, S. F. (2005). Combination Testing Strategies: A Survey. Software Testing, Verification and Reliability, 15(3), 167–199.
7. Kuhn, R., Kacker, R., & Lei, Y. (2013). Introduction to Combinatorial Testing. Boca Raton: CRC Press.
8. Kuhn, D. R., Wallace, D. R., & Gallo, A. M. (2004). Software Fault Interactions and Implications for Software Testing. IEEE Transactions on Software Engineering, 30(6), 418–421.
9. Lei, Y., & Tai, K. C. (1998). In-Parameter-Order: A Test Generation Strategy for Pairwise Testing. In Proceedings of the 3rd IEEE International High-Assurance Systems Engineering Symposium (pp. 254–261).
10. Mandl, R. (1985). Orthogonal Latin Squares: A Tool for the Design of Experiments in Testing. Software Testing, Verification and Reliability, 2(2), 23–31.
11. Nie, C., & Leung, H. (2011). A Survey of Combinatorial Testing. ACM Computing Surveys, 43(2), 1–29.
12. Segall, I., & Tzoref-Brill, R. (2018). Using Machine Learning to Improve Test Case Generation. In IEEE International Conference on Software Testing, Verification and Validation (pp. 45–53).
13. Sharma, A. (2024). Test Suite Optimization Using Machine Learning Techniques: A Comprehensive Study. DSU. Retrieved from https://www.researchgate.net/publication/385478252_Test_Suite_Optimization_Using_Machine_Learning_Techniques_A_Comprehensive_Study (access date: 22.09.2025).