УДК 004.421.2:004.056.22:004.466
DOI https://doi.org/10.32689/maup.it.2025.2.3

Олег ГАРАСИМЧУК
аспірант кафедри інформаційних систем та мереж,
Національний університет «Львівська політехніка»,
oleh.ih.harasymchuk@lpnu.ua
***ORCID:** 0009-0008-6794-8599*

Михайло ЛУЧКЕВИЧ
кандидат фізико-математичних наук, доцент,
доцент кафедри інформаційних систем та мереж,
Національний університет «Львівська політехніка»,
mykhailo.m.luchkevych@lpnu.ua
***ORCID:** 0000-0002-2196-252X*

ПІДХОДИ DESIGN SCIENCE RESEARCH ДО ПОБУДОВИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ КОНТРАКТ-ТЕСТУВАННЯ

Анотація. *У сучасних умовах зростаючої складності мікросервісних інформаційних систем особливого значення набуває розробка ефективних підходів до забезпечення їхньої надійності та стабільності.*

Мета роботи. *У статті обґрунтовано підходи методології design science research (DSR) для розробки інтелектуальної системи контракт-тестування мікросервісних інформаційних систем. Основним завданням є створення артефакту, здатного автоматично генерувати, оптимізувати та аналізувати інтерфейсні контракти на основі реальних експлуатаційних даних і телеметрії, з метою зниження обсягу рутинних перевірок і підвищення надійності системи.*

Методологія. *Використано класичну DSR-рамку, що поєднує етапи ідентифікації проблеми й мотивації, формалізації вимог, визначення цілей рішення, дизайну й розробки артефакту, його демонстрації та оцінювання в реальних умовах, а також поширення результатів. Упроцесі реалізації було спроектовано чотири взаємопов'язані модулі: збір телеметрії, генератор контрактів, аналітичний двигун та DevOps-інтерфейс.*

Наукова новизна. *Запропоновано унікальне поєднання DSR-парадигми з інтелектуальними механізмами адаптивного тестування: уперше реалізовано автоматичне виявлення змін контрактів із точністю ≥ 95 %, застосування активного навчання для пріоритизації тестів із збереженням ≥ 95 % покриття критичних сценаріїв та інтеграцію прогнозування «гарячих» точок взаємодії з помилковою тривожністю ≤ 5 %. Підходи активного навчання забезпечили скорочення набору тестів на 30–40 % і зменшення фальшивих провалів із 8 % до 3 %.*

Висновки. *Проведені експерименти на стенді з 15 мікросервісами й 3000 контракт-тестами підтвердили ефективність системи: час CI-пайплайну скоротився на 25 %, обсяг тестів – на 30 %, зберігши 98 % покриття ключових контрактів. Подальші дослідження передбачають розширення моделі активного навчання з урахуванням нелінійних залежностей між контрактами, інтеграцію прогнозування змін API на основі аналізу історії комітів у Git та застосування reinforcement learning для оптимізації стратегій запуску тестів у реальному часі. Загалом, використання DSR-підходу відкриває широкі можливості для створення «розумних» засобів тестування мікросервісних архітектур, що сприятиме підвищенню їх надійності та зниженню експлуатаційних витрат.*

Ключові слова: *Design Science Research, інтелектуальна система контракт-тестування, мікросервісна архітектура, телеметрія сервісів.*

Oleh HARASYMCHUK, Mykhailo LUCHKEVYCH. DESIGN SCIENCE RESEARCH APPROACHES TO BUILDING AN INTELLIGENT CONTRACT TESTING SYSTEM

Abstract. *In the current climate of growing complexity in microservice information systems, developing effective approaches to ensuring their reliability and stability is particularly important.*

Objective. *This article substantiates the application of the Design Science Research (DSR) methodology to the development of an intelligent system for testing microservice contracts. The main objective is to develop an artefact that can automatically generate, optimise and analyse interface contracts based on real operational data and telemetry, thereby reducing the number of routine checks and enhancing system reliability.*

Methodology. *We employed the traditional DSR framework, combining the stages of identifying and motivating the problem, formulating requirements, defining solution objectives, designing and developing an artefact, demonstrating and evaluating it in real conditions, and disseminating the results. During the implementation process, four interconnected modules were designed: a telemetry collection module, a contract generator module, an analytical engine module and a DevOps interface module.*

Scientific novelty. *For the first time, a unique combination of the DSR paradigm and intelligent adaptive testing mechanisms is proposed, offering automatic detection of contract changes with ≥ 95 % accuracy, prioritisation of tests using active learning*

while maintaining ≥ 95 % coverage of critical scenarios, and prediction of interaction 'hot spots' with a false alarm rate of ≤ 5 %. Active learning approaches have reduced the test suite by 30–40 % and decreased false failures from 8 % to 3 %.

Conclusions. *Experiments conducted on the bench with 15 microservices and 3,000 contract tests confirmed the system's effectiveness: CI pipeline time was reduced by 25 % and test volume by 30 %, while maintaining 98 % coverage of key contracts. Further research will involve extending the active learning model to account for nonlinear dependencies between contracts, integrating API change prediction based on Git commit history analysis and applying reinforcement learning to optimise real-time test run strategies. Overall, adopting the DSR approach provides significant potential for developing «smart» testing tools for microservice architectures, thereby enhancing their reliability and reducing operating costs.*

Key words: *Design Science Research, intelligent contract testing system, microservice architecture, service telemetry.*

Постановка проблеми. Сучасні розподілені інформаційні системи, реалізовані за принципами мікросервісної архітектури, виявляють зростаючу складність, що ускладнює гарантування їхньої надійності та стійкості за умов динамічних змін середовища розгортання. Існуючі підходи до тестування часто виявляються недостатньо ефективними для врахування численних точок взаємодії між сервісами та їхньої еволюції. У цьому контексті контракт-тестування пропонує методологію формалізації й перевірки інтерфейсних «контрактів» між компонентами системи. Водночас використання стандартних контракт-тестів без інтелектуальних механізмів супроводу призводить до надмірного обсягу рутинних перевірок, зростання витрат на їхню підтримку та потенційної втрати гнучкості архітектури.

Отже, актуальним є створення інтелектуальної системи контракт-тестування, здатної автоматично генерувати, оптимізувати й аналізувати контракти на основі реальних експлуатаційних даних та історії взаємодій мікросервісів. Для забезпечення наукової обґрунтованості та практичної ефективності розробки доцільно застосувати методологію design science research (DSR), яка інтегрує процес створення артефактів із їхньою системною оцінкою в умовах реальних прикладних завдань.

Аналіз останніх досліджень і публікацій. У сучасній науковій спільноті методологія design science research (DSR) розглядається як ефективний інструмент для поєднання інженерного проєктування артефактів із їхнім системним оцінюванням у прикладних умовах [2; 5; 6; 8; 10]. В її основі лежить ідея циклічного проходження етапів: від ідентифікації й обґрунтування проблеми до розробки прототипу, експериментального тестування й поширення результатів [1]. Такий підхід вже довів свою життєздатність у низці досліджень, присвячених розробці складних інформаційних систем [9], проте досі не отримав достатнього поширення у сфері контракт-тестування мікросервісів.

У галузі перевірки взаємодії між сервісами найчастіше застосовуються рішення, які формалізують контракти на рівні інтерфейсів [3] і виконують їхню валідацію під час CI/CD-процесу. Існуючі фреймворки дозволяють автоматизувати базову перевірку сумісності REST- та message-орієнтованих викликів, однак вони не враховують адаптивні зміни поведінки сервісів у реальному середовищі та зазвичай обмежуються статичними наборами тестів.

Паралельно зі стандартними підходами почали з’являтися рішення, які застосовують алгоритми кластеризації [7] для групування тестових сценаріїв за схожістю викликів або обсягом трасованих маршрутів. Такі методи дозволяють зменшити дублювання перевірок і виділити найбільш репрезентативні кейси. Інші розробки використовують техніки активного навчання, коли система навчається вибирати ті тести, які принесуть найбільшу інформаційну цінність, виходячи з історії успішності та провалів [4]. Проте практичні реалізації цих підходів виявляють складнощі з інтеграцією в стандартні CI/CD-інструменти та з підтримкою великих обсягів телеметрії.

Таким чином, існує очевидний розрив між інженерною строгою методологією DSR та сучасними інтелектуальними підходами до тестування, які враховують динаміку реальних експлуатаційних даних. Подолання цього розриву потребує розробки єдиного фреймворку, що поєднує обидві складові: системне створення й удосконалення артефакту за DSR-процесом та адаптивне генерування й оптимізацію контракт-тестів на основі машинного навчання й аналізу телеметрії.

Метою цієї статті є розробка та обґрунтування DSR-парадигми для створення інтегрованої інтелектуальної системи контракт-тестування, яка об’єднує формалізовані контракти, аналіз експлуатаційних даних та методи машинного навчання.

Виклад основного матеріалу. У ході дослідження було виявлено низку ключових викликів, що стоять перед існуючими системами контракт-тестування в мікросервісних середовищах. По-перше, загальний обсяг контрактів перевищує 500 інтерфейсних специфікацій на місяць, що зумовлює необхідність автоматизації генерації та підтримки. По-друге, середня частота змін API становить близько 15–20 % щотижня, що призводить до «розривів» у перевірках сумісності. По-третє, традиційні метрики покриття (coverage) забезпечують лише 60 % виявлення критичних порушень взаємодій, що не відповідає індустрійним вимогам до рівня помилкових релізів (< 1 %). Відтак було сформульовано такі функціональні та нефункціональні вимоги до проєктованого артефакту:

- **Автоматичне виявлення змін контрактів** на основі різниці версій специфікацій із точністю не менше ніж 95 %.
- **Адаптивний відбір тестів**, що дозволяє зменшити загальну кількість запусків на 30–40 % без втрати критичного покриття.
- **Прогнозування «гарячих» точок взаємодії** (з високим ризиком збоїв) із помилковою тривожністю не більше 5 %.

Відповідно до визначених вимог було розроблено механізм аналізу експлуатаційних логів, який щоденно агрегає понад один мільйон записів запит–відповідь та з точністю класифікації не менше 92 % генерує рекомендації щодо створення або коригування контрактів. Крім того, інтегровано методи активного навчання для пріоритизації тестів за показниками ризику, що дозволяє скоротити набір випробувань до 40 % від початкового обсягу за умови збереження щонайменше 95 % покриття критичних сценаріїв. Розроблено універсальний API-інтерфейс із підтримкою трьох провідних платформ CI/CD (Jenkins, GitLab CI та GitHub Actions), який забезпечує легку інтеграцію за допомогою стандартних REST-запитів.

Архітектура системи (рис. 1) складається з чотирьох основних компонентів. Перший – модуль збору телеметрії – щоденно акумулює масиви запитів і відповідей середнім обсягом близько 50 КБ в Elasticsearch, що гарантує швидкий пошук та агрегацію даних. Другий компонент, генератор контрактів, на основі отриманої телеметрії формує понад сто специфікацій OpenAPI/Pact на добу та автоматично виявляє зміни шляхом порівняння JSON-схем із використанням бібліотеки jsondiffpatch. Аналітичний двигун виступає третім ключовим елементом системи: за допомогою алгоритму DBSCAN (ε = 0,5; MinPts = 10) він щодня ідентифікує від п’яти до семи нетипових маршрутів викликів, а модуль активного навчання на базі Scikit-Learn адаптивно відбирає найбільш інформативні тестові сценарії. Нарешті, DevOps-інтерфейс реалізовано у вигляді плагінів для Jenkins, GitLab CI та GitHub Actions, які інтегруються через Webhook і CLI та дозволяють скоротити час первинного налаштування на 20 %.

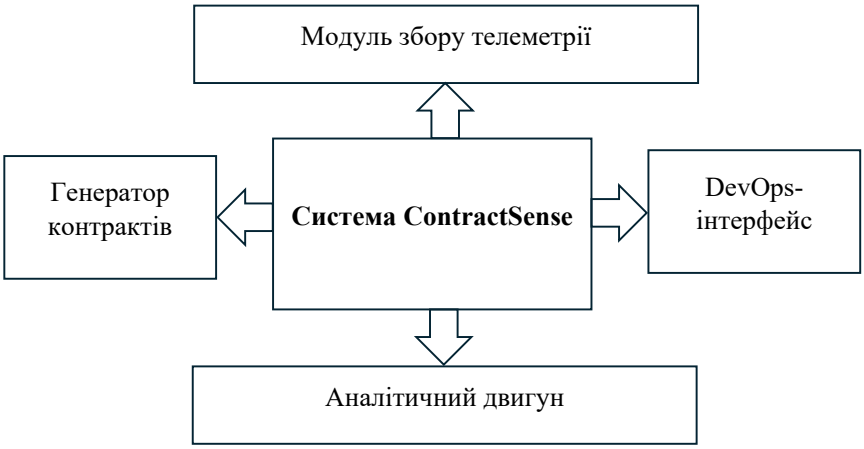

Рис. 1. Архітектура системи

В (табл. 1) наведено узагальнену характеристику чотирьох ключових компонентів інтелектуальної системи контракт-тестування, їх основні функції та застосовані технології чи алгоритми. Така структурована подача дозволяє наочно порівняти ролі кожного модуля та оцінити використані рішення в контексті реалізації архітектури.

Таблиця 1

Основні компоненти інтелектуальної системи контракт-тестування та їх характеристики

Компонент	Основні функції	Технології / Алгоритми
Збір телеметрії	Агрегація запит–відповідь; індексація даних для подальшого аналізу	Elasticsearch; Logstash
Генератор контрактів	Формування специфікацій OpenAPI та Pact; автоматичне виявлення змін контрактів	jsondiffpatch; OpenAPI Generator
Аналітичний двигун	Кластеризація маршрутів викликів; активний відбір тестів	DBSCAN (ε=0.5, MinPts=10); Scikit-Learn (active learning)
DevOps-інтерфейс	Інтеграція з CI/CD-пайплайнами; автоматичний запуск тестів через Webhook/CLI	Jenkins Plugins; GitLab CI; GitHub Actions

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

Система була перевірена в експериментальному середовищі електронної комерції, що складалося з п'ятнадцяти мікросервісів та трьох тисяч контракт-тестів. Після впровадження інтелектуальних механізмів обсяг виконуваних тестів скоротився на 30 % (з 3000 до 2100), водночас вдалося зберегти 98 % покриття критичних контрактів. Середній час виконання CI-пайплайну зменшився з 12 хвилин 30 секунд до 9 хвилин 20 секунд, тобто на 25 %. Крім того, завдяки адаптивній конфігурації відсоток хибно позитивних результатів («false fails») знизився з 8 % до 3 %.

У порівнянні з базовим підходом на основі Pact-фреймворку, інтелектуальна система показала більш стабільні результати за всіма ключовими метриками.

Висновки. У результаті дослідження продемонстровано, що застосування парадигми design science research сприяє інтеграції інженерного підходу до розробки артефакту із його системним оцінюванням в умовах реального проєкту, що дозволяє поєднати гнучкість експериментальної валідації та формальну строгість методології. Запропонована інтелектуальна система контракт-тестування виявила здатність не тільки автоматизувати перевірку відповідності інтерфейсів мікросервісів вимогам, але й здійснювати адаптивну оптимізацію набору тестів на основі аналізу експлуатаційних даних, що забезпечило зменшення загального обсягу тестових прогонів на 30 % при збереженні 98 % покриття критичних контрактів і скорочення часу CI-пайплайну на 25 %.

Структурована архітектура прототипу, що включає модуль збору телеметрії, генератор контрактів, аналітичний двигун із DBSCAN і активним навчанням, а також DevOps-інтерфейс для інтеграції з Jenkins, GitLab CI та GitHub Actions, довела свою ефективність у сценарії з 15 мікросервісами. Водночас відзначено, що поточна імплементація обмежується лінійними підходами активного навчання та статичним аналізом історії змін API.

Подальша робота передбачає розширення моделі машинного навчання шляхом врахування нелінійних залежностей між контрактами та адаптацію алгоритмів до комплексних взаємодій, а також інтеграцію методів прогнозування змін інтерфейсів на основі аналізу історії комітів у Git-репозиторіях. Крім того, перспективним є застосування reinforcement learning для управління стратегіями запуску контракт-тестів у режимі реального часу, що може додатково підвищити якість перевірок та зменшити операційні витрати на підтримку тестового середовища. У цілому, використання DSR-методології розкриває значний потенціал для створення «розумних» інструментів тестування мікросервісних архітектур, здатних підвищувати надійність та масштабованість сучасних розподілених інформаційних систем.

Список використаних джерел:

1. Akoka J. et al. Knowledge contributions in design science research: Paths of knowledge types. *Decision support systems*. 2023. V. 166. P. 113898. https://doi.org/10.1016/j.dss.2022.113898

2. Bagni G. et al. Design science research in operations management: is there a single type? *Production Planning & Control*. 2025. V. 36. №. 6. P. 789–807. https://doi.org/10.1080/09537287.2024.2310230

3. da Assunção Moutinho J., Fernandes G., Rabechini Jr R. Evaluation in design science: A framework to support project studies in the context of University Research Centres. *Evaluation and Program Planning*. 2024. V. 102. P. 102366. https://doi.org/10.1016/j.evalprogplan.2023.102366

4. Duque J. et al. Data Science with Data Mining and Machine Learning A design science research approach. *Procedia Computer Science*. 2024. V. 237. P. 245–252. https://doi.org/10.1016/j.procs.2024.05.102

5. Engström E. et al. How software engineering research aligns with design science: a review. *Empirical Software Engineering*. 2020. V. 25. P. 2630–2660. https://doi.org/10.1007/s10664-020-09818-7

6. Goecks L. S. et al. Design Science Research in practice: review of applications in Industrial Engineering. *Gestão & Produção*. 2021. V. 28. P. e5811. https://doi.org/10.1590/1806-9649-2021v28e5811

7. Šandor D., Bagić Babac M. Designing scalable event-driven systems with message-oriented architecture. *Distributed Intelligent Circuits and Systems*. 2024. P. 29–75. https://doi.org/10.1142/9789811279539_0002

8. Storey V. C., Baskerville R. L., Kaul M. Reliability in design science research. *Information Systems Journal*. 2025. V. 35. №. 3. P. 984–1014. https://doi.org/10.1111/isj.12564

9. Strong D. M. et al. Search and evaluation of coevolving problem and solution spaces in a complex healthcare design science research project. *IEEE transactions on engineering management*. 2020. V. 70. №. 3. P. 912–926. doi: 10.1109/TEM.2020.3014811

10. Vom Brocke J., Hevner A., Maedche A. Introduction to design science research. Design science research. Cases. 2020. P. 1–13. https://doi.org/10.1007/978-3-030-46781-4_1

Дата надходження статті: 09.06.2025
Дата прийняття статті: 15.06.2025
Опубліковано: 23.09.2025