EUROPEAN CONFERENCE

Conference Proceedings

V International Science Conference «Modern scientific problems and ways to solve them»

September 29 - October 01, 2025

Munich, Germany

MODERN SCIENTIFIC PROBLEMS AND WAYS TO SOLVE THEM

Abstracts of V International Scientific and Practical Conference

Munich, Germany
(September 29 – October 01, 2025)

UDC 01.1

ISBN – 979-8-89814-231-5

The V International scientific and practical conference «Modern scientific problems and ways to solve them», September 29 – October 01, 2025, Munich, Germany, 146 p.

 Collection of scientific articles published is the scientific and practical publication, which contains scientific articles of students, graduate students, Candidates and Doctors of Sciences, research workers and practitioners from Europe, Ukraine and from neighboring countries and beyond. The articles contain the study, reflecting the processes and changes in the structure of modern science. The collection of scientific articles is for students, postgraduate students, doctoral candidates, teachers, researchers, practitioners and people interested in the trends of modern science development.

The recommended citation for this publication is: Synytsia O.V., Shlapak H.V., Yarovenko D.V. Influence of electroactivated water fractions on the microbiological condition of poultry-based semi-finished products. Abstracts of V International Scientific and Practical Conference. Munich, Germany. Pp. 8-10.

URL: https://eu-conf.com/en/events/modern-scientific-problems-and-ways-to-solve-them/

СТРАТЕГІЇ РЕІНЖИНІРИНГУ ВІД МОНОЛІТНОЇ ДО МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ: АНАЛІЗ ЕФЕКТИВНОСТІ ТА РИЗИКІВ

Калашник Валерій Юрійович
к.т.н., старший викладач
Київський Національний Університет Технологій та Дизайну

Анотація

У сучасному світі, де динамічний розвиток програмного забезпечення є критично важливим, багато організацій стикаються з обмеженнями застарілих монолітних архітектур. Дана робота аналізує ключові проблеми, пов’язані з технічним боргом та масштабуванням таких систем, і представляє систематизований підхід до їх модернізації. В основу дослідження покладено стратегію реінжинірингу з використанням патерну «Strangler Fig» для плавного переходу до мікросервісної архітектури. Розглянуто роль API Gateway та Service Discovery як інструментів, що забезпечують керованість та гнучкість розподілених систем. Особлива увага приділяється практичним аспектам впровадження моніторингу (Prometheus, Grafana) як засобу для об'єктивної оцінки ефективності процесу та мінімізації ризиків. Результати дослідження показують, що правильно спланований та інструментально підкріплений реінжиніринг дозволяє значно підвищити швидкість розробки, масштабованість та відмовостійкість програмних систем, що є ключовим для вирішення сучасних наукових проблем у сфері програмної інженерії.

Вступ

Стрімкий розвиток інформаційних технологій і зростаючі вимоги ринку до швидкості випуску нових продуктів ставлять перед розробниками та архітекторами програмного забезпечення (ПЗ) нові виклики. Системи, що починали свій шлях як монолітні додатки, з часом стають неефективними. Зростання кодової бази, складність її підтримки, висока взаємозалежність компонентів та труднощі з масштабуванням призводять до накопичення технічного боргу. Це гальмує інновації, збільшує витрати на інфраструктуру та знижує загальну відмовостійкість.

Вирішення цих проблем вимагає системного підходу до реінжинірингу. Ця робота присвячена аналізу та практичному застосуванню стратегій реінжинірингу, що дозволяють модернізувати застарілі архітектури, переходячи від моноліту до мікросервісів. Метою є не лише демонстрація теоретичних підходів, а й їхнє практичне обґрунтування на прикладі навчального проекту, що дозволяє оцінити їх ефективність.

Основні положення та результати

На основі аналізу проблем монолітної архітектури та сучасних індустріальних практик, ми розробили та апробували методологію реінжинірингу, яка складається з наступних етапів:

1. **Виявлення та діагностика проблем.** Застосування стратегій тестування продуктивності, як-от використання інструментів JMeter та k6, дозволяє ідентифікувати «вузькі місця» моноліту. Аналіз результатів тестування дає змогу виявити неоптимальні алгоритми та архітектурні дефекти, що гальмують роботу системи. Наприклад, тривалість відгуку API на запит бронювання може значно зростати при одночасному навантаженні великої кількості користувачів. Інструменти, такі як JMeter, дозволяють вимірювати не лише час відгуку, а й кількість запитів на секунду (throughput) та рівень використання ресурсів сервера (CPU, пам'ять). Детальне вивчення цих метрик у поєднанні з аналізом логів дозволяє точно встановити причину проблеми: чи це неоптимальний запит до бази даних, чи занадто складна бізнес-логіка в єдиному потоці.

2. **Стратегічний поділ та рефакторинг.** Ми використали патерн «Strangler Fig» (Задушливий фікус), що дозволяє поступово виділяти окремі функціональні модулі з моноліту та реалізовувати їх у вигляді незалежних мікросервісів. Цей підхід мінімізує ризики, пов'язані з глобальною міграцією, і дає можливість перевіряти кожен новий сервіс в ізольованому середовищі. На прикладі нашої лабораторної роботи, ми розділили «Сервіс бронювання» на окремі компоненти. Назва патерну походить від тропічної рослини, яка повільно обплітає дерево-господар, поки воно не загине. У нашому випадку, моноліт є деревом-господарем, а нові мікросервіси — «фікусом», що поступово перебирає на себе функціональність. Першим кроком є виділення чітко визначених **обмежених контекстів** (Bounded Contexts) — наприклад, сервіс авторизації, сервіс бронювання, сервіс сповіщень. Після цього, для кожної нової функції, замість додавання коду до моноліту, створюється новий мікросервіс. Потім API Gateway перенаправляє запити на цей новий сервіс, а не на моноліт. Цей ітеративний процес дозволяє підтримувати безперервну роботу системи, одночасно зменшуючи технічний борг.

3. **Впровадження API Gateway та Service Discovery.** Після поділу системи на мікросервіси, виникає потреба в ефективному управлінні взаємодією між ними. API Gateway виступає як єдина точка входу, що спрощує клієнтську частину та дозволяє гнучко маршрутизувати запити. Service Discovery, у свою чергу, дозволяє сервісам динамічно знаходити один одного, що є критично важливим для масштабування та відмовостійкості. У нашому експерименті ми інтегрували ці інструменти, що значно підвищило керованість системи. Детальніше, **API Gateway** виконує кілька критично важливих функцій: він агрегує запити від клієнтів, обробляє аутентифікацію та авторизацію, виконує балансування навантаження та кешування, і таким чином ізолює клієнтів від складності мікросервісної архітектури. **Service Discovery** вирішує проблему, де і як сервіси можуть спілкуватися між собою в динамічному середовищі, де адреси сервісів постійно змінюються. Замість жорстко закодованих адрес, сервіси реєструються в центральному каталозі, а інші сервіси можуть знаходити їх за іменем, що забезпечує гнучкість та стійкість системи до збоїв.

4. **Моніторинг та оцінка результатів.** Для об'єктивної оцінки ефективності реінжинірингу ми налаштували систему моніторингу на базі Prometheus та

Grafana. Це дозволило в реальному часі відстежувати ключові метрики продуктивності (затримки, використання ресурсів, кількість запитів) та візуалізувати їх, що дало змогу підтвердити успішність проведених змін. Ми спостерігали значне зниження затримок та збільшення пропускної здатності після міграції. Окрім загальних метрик, ми також впровадили **розподілене трасування** (distributed tracing) з використанням інструментів, таких як Jaeger. Це дозволило нам відстежувати повний шлях запиту через різні мікросервіси, ідентифікуючи точні місця затримок і збоїв. Наша методологія підкреслює, що реінжиніринг має бути не лише технічним, але й **дано-орієнтованим процесом**, де рішення приймаються на основі об'єктивних показників, а не лише інтуїції.

Висновки та практична цінність

Проведений аналіз та практична апробація показали, що реінжиніринг від монолітної до мікросервісної архітектури є ефективним та необхідним шляхом для модернізації програмного забезпечення. Застосування стратегії «Strangler Fig» у поєднанні з інструментами, такими як API Gateway, Service Discovery та сучасні системи моніторингу, дозволяє значно покращити ключові показники якості програмного забезпечення.

Дане дослідження підтверджує, що системний підхід до реінжинірингу дозволяє не лише усунути технічний борг, а й створити основу для подальшого швидкого та стабільного розвитку. Замість хаотичного і ризикованого процесу, модернізація архітектури стає керованою, інкрементальною та вимірюваною. Це дозволяє підприємствам адаптуватися до мінливих вимог ринку, що є запорукою їхньої конкурентоспроможності.

Список літератури:

1. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.
2. Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.
3. Richards, M. (2015). Software Architecture Patterns. O'Reilly Media.
4. Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., ... & Stafford, J. (2010). Documenting Software Architectures: Views and Beyond. Addison-Wesley.
5. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Addison-Wesley.
6. Сарновський, М. А. (2021). Архітектура програмних систем. К.: КПІ ім. Ігоря Сікорського.
7. Григоренко, О. А. (2019). Основи тестування програмного забезпечення. К.: Центр учбової літератури.
8. Kent, B. (2020). Чистий код. Створення, аналіз і рефакторинг. К.: Наш Формат.
9. Shvets, R., Klymenko, A. (2022). Migrating Monoliths to Microservices: A practical approach. Academic Publishing.
10. Klymenko, A. (2021). Data-Driven Quality Assurance: Methods and

Tools. К.: КНУТД.

11. Калашник, В.Ю. (2024). Забезпечення якості програмного забезпечення в agile-середовищах. Збірник наукових праць КНУТД, 2024(1), 45-51.

Scientific publications

MATERIALS

The V International Scientific and Practical Conference
«Modern scientific problems and ways to solve them»

Munich, Germany
(September 29 – October 01, 2025)