DOI: https://doi.org/10.26565/2304-6201-2025-67-05

УДК (UDC) 004.051

Зінов'єв Дмитро Володимирович *старший викладач кафедри інтелектуальних програмних систем і технологій, ННІ комп'ютерних наук та штучного інтелекту, Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, м. Харків, Україна, 61022*
e-mail: zinoviev@karazin.ua
https://orcid.org/0000-0003-1862-9803

Ткачук Микола Вячеславович *д.т.н., професор; професор кафедри інтелектуальних програмних систем і технологій, ННІ комп'ютерних наук та штучного інтелекту, Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, м. Харків, Україна, 61022*
e-mail: mykola.tkachuk@karazin.ua
https://orcid.org/0000-0003-0852-1081

Архітектура, програмна реалізація та аналіз результатів застосування інтелектуального інструментального засобу для конфігурування мікросервісних застосунків

Актуальність. Розробка застосунків з мікросервісною архітектурою потребує ефективного управління конфігураціями в умовах змінного навантаження, вимог до надійності, відмовостійкості й масштабованості. Це зумовлює потребу в інтелектуальних засобах адаптивного конфігурування, здатних працювати в режимі, близькому до реального часу.
Мета. Створити інтелектуальний інструментальний засіб для адаптивного управління конфігураціями МСА з модулем прийняття рішень на основі Case-Based Reasoning (CBR), спроєктувати його архітектуру, зробити програмну реалізацію, а також експериментально оцінити роботу на тестовому полігоні й порівняти кілька CBR-методів.
Методи дослідження. Уточнено базові поняття процесів конфігурування МСА; спроектовано полігон із трьома сервісами (auth, product, order) і вимогами до продуктивності (≤1000 одночасних запитів, середня затримка ≤200 мс). Адаптивне управління конфігураціями мікросервісів реалізовано як мікросервіс із REST API (FastAPI) та сховищем прецедентів (PostgreSQL); використовуються метрики QoS, ресурсні, «вартісні» та адаптивності. Досліджено п'ять CBR-методів: K-Nearest Neighbors, Weighted KNN, Feature-Based Retrieval, Cluster-Based Retrieval, Indexing & Hashing. Проведено серію вимірювань часу підбору конфігурації для бази прецедентів у 50–1000 записів із усередненням по 100 прогонах.
Результати. Підсистема коректно ідентифікує стани та застосовує релевантні конфігурації для різних сценаріїв (low/medium/high/peak), відповідаючи вимозі часу підбору ≤0,5 с. Найвищу швидкодію продемонстрував метод Indexing & Hashing (≈27,6–50,3 мс для 50–1000 кейсів); KNN має лінійне зростання часу, а Weighted KNN дає керованість за рахунок ваг метрик. Реалізований веб-інтерфейс забезпечує моніторинг і ручний/автоматичний режим застосування змін у реальному часі.
Висновки. Запропонована архітектура та програмна реалізація інструментального засобу з CBR підтверджують практичну доцільність адаптивного конфігурування МСА й створюють підґрунтя для масштабованих даними керованих рішень. Окреслено подальші напрями: еволюція кейс-бази з онлайн навчанням, багатокритеріальна оптимізація (продуктивність/надійність/вартість/енергоефективність), глибша інтеграція з оркестраторами та service mesh, підвищення пояснюваності рішень.

Ключові слова: *програмний мікросервіс, архітектура, управління конфігураціями, інтелектуальний підхід, метод аналізу прецедентів, CBR, інтелектуальний інструментальний засіб, тестування, якість, метрика, модель.*

Як цитувати: Зінов'єв Д.В., Ткачук М.В. Архітектура, програмна реалізація та аналіз результатів застосування інтелектуального інструментального засобу для конфігурування мікросервісних застосунків. *Вісник Харківського національного університету імені В.Н.Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2025. вип. 67. С.56-65. https://doi.org/10.26565/2304-6201-2025-67-05

How to quote: D.V. Zinov'ev, M.V. Tkachuk, "Architecture, software implementation and results analyzing of the usage an intelligent tool for configuring microservice applications". *Bulletin of V.N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems,* 2025. vol. 67, pp.56-65. https://doi.org/10.26565/2304-6201-2025-67-05 [in Ukrainian]

Вступ. Актуальність і мета дослідження.

В сучасних умовах стрімкого розвитку інформаційних технологій мікросервісна архітектура (МСА) стала провідною парадигмою проектування розподілених програмних систем. Цей підхід

передбачає розбиття великої та складної системи на набір невеликих, автономних сервісів, кожен із яких виконує свою чітко визначену функцію та взаємодіє з іншими через стандартизовані інтерфейси, такі як REST або gRPC. Це дозволяє створювати масштабовані, гнучкі та легко підтримувані застосунки, що складаються з незалежних сервісів, кожен із яких виконує окрему функцію. Слабка зв'язність між компонентами в МСА забезпечує незалежне розгортання, масштабування та оновлення кожного мікросервісу, що надає суттєві переваги по відношенню до монолітних архітектур [1]. Конфігурування мікросервісів являє собою складний процес, що охоплює налаштування численних параметрів (мережеві адреси, ліміти ресурсів, ключі доступу, налаштування безпеки, тощо) для кожного сервісу. У реальних системах ці параметри можуть часто змінюватися в залежності від навантаження, типу запитів, відмов компонентів або зовнішніх умов. Некоректне конфігурування окремих сервісів може призвести до критичних збоїв у роботі всієї системи - від затримок у відповіді на деякі запити до повної недоступності її функціоналу. Проблеми ще більше ускладнюються, коли система працює в хмарному середовищі з автоматичним масштабуванням, де умови змінюються в режимі реального часу. У такому контексті конфігурації не можуть бути статичними, а мають постійно адаптуватися до змін обчислювального навантаження, доступності сервісів і зовнішніх залежностей [2].

Більшість сучасних рішень у цій сфері, такі як Kubernetes ConfigMaps, HashiCorp Consul або Spring Cloud Config [3], пропонують статичні або шаблонізовані підходи, які не враховують поточний стан системи та вимагають ручного втручання адміністратора. Таким чином, актуальним є науково-технічне завдання розробки інтелектуальних засобів адаптивного управління конфігураціями мікросервісних застосунків. Ця задача стикається з високою складністю технологічних процесів, великою кількістю параметрів та слабкою формалізацією їх взаємозв'язків. Для вирішення цих проблем пропонується використовувати підходи штучного інтелекту, зокрема метод аналізу прецедентів (Case-Based Reasoning - CBR), і розроблена алгоритмічна модель адаптивного управління конфігураціями МСА з його використанням [4].

Мета роботи

Метою цієї роботи є розробка архітектури, програмна реалізація та дослідження результатів застосування інтелектуального інструментального засобу для автоматизованого конфігурування застосунків з МСА на основі методу аналізу прецедентів, що забезпечує пошук та адаптацію конфігурацій окремих мікросервісів з урахуванням змін у середовищі їх функціонування.

2. Архітектура та особливості програмної реалізації інструментального засобу адаптивного управління конфігураціями мікросервісів (АУКМ)

Для організації процесу АУКМ в [3] була запропонована структурно-функціональна схема інструментального засобу, яка містить наступні шари компонентів:

- основний функціонал системи, що передбачає розгортання кожного окремого мікросервісу у відповідному Docker-контейнері, множина яких оркеструється засобами технології Kubernetes;
- модуль прийняття рішень, де спеціальний керуючий мікросервіс, що реалізує методи CBR, приймає метрики через API, обирає адаптивну конфігурацію та формує відповідь;
- компоненти для моніторингу стану МСА, які забезпечують збір даних та розрахунок ключових показників (метрик): завантаження CPU, розмір оперативної пам'яті, час затримки відповідей та ін.) після застосування змін, що забезпечує замкнений цикл адаптації;
- базу даних (БД) прецедентів, кожний з яких включає значення параметрів конфігурації та метрики продуктивності МСА при застосуванні певного алгоритму CBR.

2.1. Функціональні та нефункціональні вимоги до розробки засобу АУКМ

Для визначення вимог до засобу АУКМ був розроблений програмний тестовий полігон, що складався з трьох мікросервісів, які є типовими для застосунку з МСА у предметній області «*Обробка замовлень користувачів при роботі з каталогом продуктів*», а саме: *auth_service* для автентифікації користувачів; *product_service* для керування каталогом продуктів та *order_service* для обробки замовлень. Його функціонал включав:

– генерацію навантаження різного рівня (низьке, середнє, високе) з можливістю ручного налаштування параметрів, таких як розмір пулу підключень до БД та обмеження на кількість запитів;
– API-взаємодію з мікросервісами для отримання метрик (CPU, пам'ять, затримка відповіді, доступність) та застосування змін конфігурації у реальному часі.

Підсистема АУКМ мала виконувати наступні задачі (тобто, функціональні вимоги):
– автоматичний підбір потрібних конфігурацій МСА на основі отриманих метрик за допомогою методів CBR (K-Nearest Neighbors, Cluster-Based Retrieval тощо);
– повний цикл обробки запитів до БД прецедентів: пошук релевантних рішень, адаптація до поточного стану системи, збереження нових випадків для навчання;
– інтеграцію з тестовим полігоном через API для внесення змін без зупинки сервісів**;**
– інтерфейс адміністратора для налаштування параметрів алгоритмів для окремих методів CBR та перегляду результатів знайдених рішень.

Також шляхом вивчення деяких існуючих рішень [5-8], а також експертним шляхом були визначені наступні нефункціональні вимоги до тестового полігону та системи АУКМ, які представлені у Таблиці 1.

Таблиця 1. Узагальнені функціональні та нефункціональні вимоги
Table 1 Generalized functional and non-functional requirements

Вимога	**Тестовий полігон**	**Інструментарій АУКМ**
Функціональні	Симуляція навантаження, збір метрик, API для конфігурування	Автоматичний підбір конфігурацій через CBR, інтеграція з API, збереження прецедентів
Продуктивність	≤200 мс затримки, 1000 запитів	≤0,5 с на підбір конфігурації
Масштабованість	До 10 мікросервісів	До 2000 прецедентів
Надійність	≥95% доступності	Стійкість до збоїв одного мікросервісу
Безпека	OAuth 2.0, AES-256	OAuth 2.0, AES-256
Інтеграція/Розширюваність	RESTful API, модульна архітектура	RESTful API, підтримка нових методів CBR

2.2. Особливості реалізації тестового полігону для дослідження засобу АУКМ

Для ізоляції даних при роботі мікросервісів тестового полігону передбачені окремі таблиці в базі даних PostgreSQL, взаємодія між сервісами здійснюється за допомогою архітектурного стилю RESTful API, і на рисунку 1 наведена ER-діаграма бази даних тестового полігону.

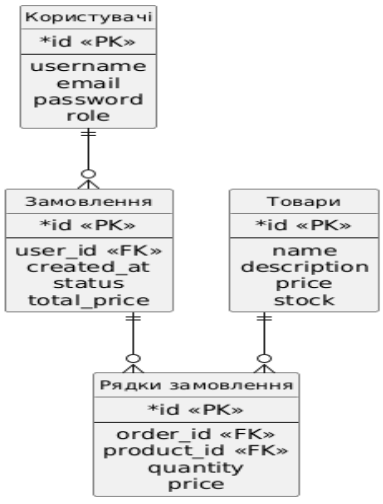

Рис. 1. ER-діаграма бази даних тестового полігону
Fig. 1 ER diagram of the testing ground database

Кожний мікросервіс у складі полігону має набір своїх конфігураційних параметрів, які система АУКМ повинна адаптивно змінювати на основі метрик системи, таких як завантаження процесора чи затримка відповіді мікросервіса. Ці параметри дозволяють адаптувати сервіси до різних сценаріїв навантаження: низьке (до 100 запитів/с), середнє (100-500 запитів/с), високе (500-1000 запитів/с) і пікове (понад 1000 запитів/с). Зміна параметрів відбувається через захищені ендпоінти POST /config/update із подальшим перезапуском бази даних у разі зміни db_connection_type або db_pool_size.

2.3. Розробка підсистеми адаптивного управління конфігураціями мікросервісів

На рисунку 2 наведена діаграма компонентів системи АУКМ.

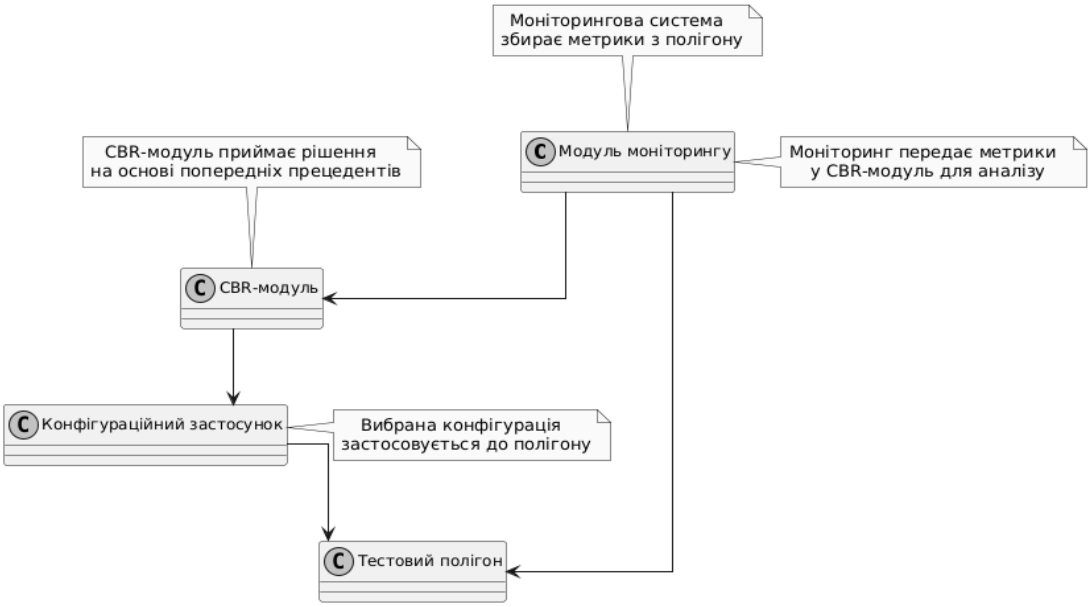

Рис. 2. Діаграма компонентів системи АУКМ
Fig. 2 Diagram of the components of the AMCM system

Система АУКМ визначає потрібні конфігурації МСА тестового полігону за поточними метриками, використовуючи відповідні методи CBR, і має наступні компоненти:

- *модуль моніторингу* збирає наступні метрики з тестового полігону: середня затримка (performance_metric); час відповіді (response_time); використання пам'яті (memory_usage); використання дискового простору (disk_usage); завантаження процесора (cpu_load), мережевий трафік (network_usage), операційні витрати (operational_costs) і доступність (availability). через ендпоінти /health мікросервісів по HTTP.
- *CBR-модуль* зіставляє поточний стан із базою прецедентів і обирає конфігурацію на основі одного з наявних CBR методів: KNN (звичайний та з вагами), Feature-Based, Cluster-Based або Indexing & Hashing.
- *Модуль конфігурування* замінює конфігурації мікросервісів auth_service, product_service, order_service на наново обрані через ендпоінти POST /config/update; також, цей модуль передбачає збереження нових прецедентів у базі даних.

3. Методика та аналіз результатів застосування запропонованого інструментального засобу для конфігурування мікросервісів

3.1. Методика проведення експериментів на тестовому програмному полігоні АУКМ

Стан кожного з мікросервісів тестового полігону описувався вектором метрик: завантаження процесора (CPU), використання оперативної пам'яті (RAM) і дискової пам'яті (DISK), мережева активність, середня затримка/час відповіді s доступність відповідного сервісу. Для відтворення сценаріїв low/medium/high/peak/suboptimal використовувався скрипт генерації значень параметрів конфігурацій МСА у заданих діапазонах. У процедурі порівняння для кожного CBR-методу час обчислення вимірювався 100 разів і потім осереднювався. Експерименти проводилися з різними

розмірами бази прецедентів - 50, 100, 200, 500 і 1000 записів. В якості тестового середовища використовувався Windows 10 22H2; RAM 16 GB DDR4 3200 MHz; CPU Ryzen 5 4600H (6C/12T). Додатково перевірялися результати коригування конфігурацій через інтерфейс системи і їх фактичне застосування до сервісів (див. нижче).

3.2. Інтерфейс користувача підсистеми АУКМ

Підсистема АУКМ реалізована як односторінковий веб-застосунок (single-page application) із двома вкладками. Перша вкладка “Monitoring & Manual Configuration” відображає поточні показники параметрів конфігурацій МСА за різними метриками, які згруповані у такі категорії:

- QoS Metrics - середня затримка, час відповіді, доступність сервісів;
- System Resources - завантаження CPU/RAM/диску та мережі;
- Cost Efficiency Metrics - умовні операційні витрати, що обчислюються як зважена функція ресурсних показників з обраними ваговими коефіцієнтами, які визначалися експертним шляхом: (OC = 0.4·CPU + 0.3·RAM + 0.2·DISK + 0.1·NETWORK);
- Adaptability Metrics - час адаптації та кількість виконаних реконфігурацій.

На рисунку 3 наведений приклад метрик категорії «System Resources», а на рисунку 4 показані поточні конфігурації “Current Configurations” для всіх 3-х мікросервісів тестового полігону.

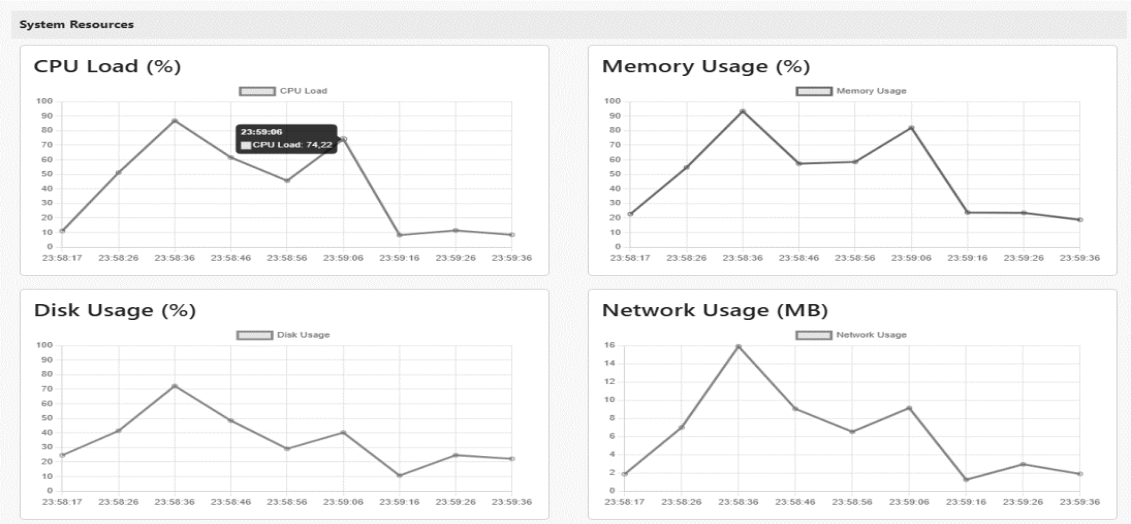

Рис. 3. Категорія «System Resources»
Fig. 3 “System Resources” category

Current Configurations

Service	Config
auth_service	{ "access_token_expire_minutes": 30, "algorithm": "HS256", "db_connection_type": "sync", "db_pool_size": 5, "rate_limit_requests": 100, "response_delay": 0 }
product_service	{ "db_connection_type": "sync", "db_pool_size": 5, "rate_limit_requests": 100, "response_delay": 0, "max_products_per_page": 20 }
order_service	{ "db_connection_type": "sync", "db_pool_size": 5, "rate_limit_requests": 100, "response_delay": 0, "max_orders_per_page": 20, "default_order_status": "pending" }

Рис. 4. Поточні конфігурації мікросервісів тестового полігону
Fig. 4 Current configurations of testing ground microservices

Друга вкладка “Automatic Configuration” (див. рисунки 5 та 6) надає можливість обрати відповідний CBR-метод, увімкнути його авто-конфігурацію параметрів та налаштувати період їх оновлення. Наприклад, у методі Weighted KNN задаються ваги метрик; у методі Feature-Based - набір ознак для поточного прецеденту; у методі Cluster-Based - кількість кластерів) і т.п.

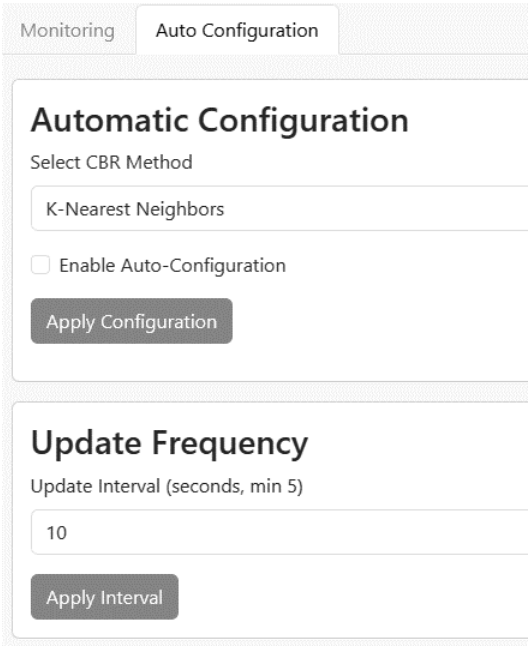

Рис.5. Вкладка «Automatic Configuration»
Fig. 5 Automatic Configuration tab

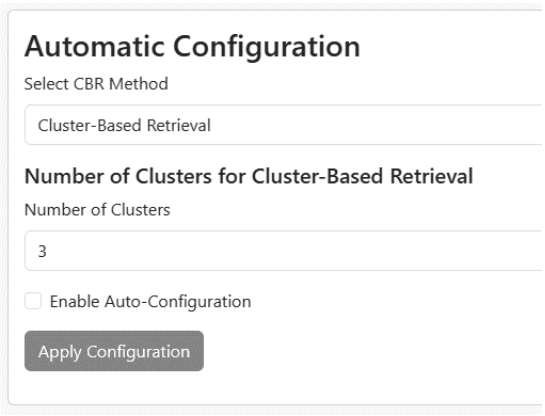

Рис. 6. Параметри для методу Cluster-Based Retrieval

3.3. Результати проведення програмних експериментів та їх аналіз

Для проведення досліджень був розроблений скрипт на мові Python, який генерував значення для параметрів конфігурацій МСА у діапазонах, які відповідали різним типам обчислювального навантаження полігону (вони представлені у Таблиці 2).

Таблиця 2. Інтервали значень конфігураційних параметрів для різних діапазонів навантаження
Table 2 Configuration parameter value ranges for different load ranges

Параметр конфігурації мікросервіса	**Навантаження**				
	Low Load	**Medium Load**	**High Load**	**Peak Load**	**Suboptimal**
cpu_load	5.0-20.0	30.0-50.0	60.0-80.0	80.0-95.0	40.0-70.0
memory_usage	15.0-30.0	40.0-60.0	65.0-85.0	85.0-95.0	50.0-80.0
disk_usage	10.0-25.0	25.0-40.0	40.0-60.0	60.0-75.0	30.0-50.0
network_usage	0.5-3.0 MB/s	3.0-8.0 MB/s	8.0-15.0 MB/s	15.0-25.0 MB/s	5.0-10.0 MB/s
performance_metric	10.0-30.0 ms	30.0-80.0 ms	80.0-150.0 ms	150.0-300.0 ms	200.0-400.0 ms
response_time	5.0-20.0 ms	20.0-50.0 ms	50.0-100.0 ms	100.0-200.0 ms	150.0-300.0 ms
operational_costs	5.0-15.0 units	15.0-30.0 units	30.0-50.0 units	50.0-80.0 units	25.0-60.0 units
availability	95.0-100.0%	90.0-100.0%	85.0-95.0%	80.0-90.0%	70.0-85.0%
adaptation_time	0.1-0.5 s	0.5-1.0 s	1.0-2.0 s	2.0-5.0 s	1.5-3.0 s
reconfigurations	0-2	1-5	3-8	5-10	2-6

Спочатку система АУКМ генерувала поточні значення параметрів мікросервісів тестового полігону, а потім за допомогою одного з CBR методів знаходила в БД прецедент для поточних параметрів системи. На рисунку 7 показаний результат вибору із БД прецедентів нового сценарію на основі поточних параметрів системи.

```
2025-06-05 09:21:54,363 - INFO - Best case selected: performance_metric=50.36
2025-06-05 09:21:54,363 - INFO - Retrieved config: {
  "auth_service": {
    "access_token_expire_minutes": 60,
    "algorithm": "HS256",
    "db_connection_type": "sync",
    "db_pool_size": 7,
    "rate_limit_requests": 150,
    "response_delay": 0.1
  },
  "product_service": {
    "db_connection_type": "sync",
    "db_pool_size": 7,
    "rate_limit_requests": 150,
    "response_delay": 0.1,
    "max_products_per_page": 30
  },
  "order_service": {
    "db_connection_type": "sync",
    "db_pool_size": 5,
    "rate_limit_requests": 100,
    "response_delay": 0.0,
    "max_orders_per_page": 20,
    "default_order_status": "pending"
  }
```


Рис. 7. Результат вибору потрібного прецеденту
Fig. 7 Result of selecting the necessary precedent

На рисунку 8 показаний результат застосування конфігурацій нового сценарію до кожного з мікросервісів тестового полігону.


```
2025-06-05 09:21:54,375 - INFO - Updated configuration for auth_service: {
  "access_token_expire_minutes": 60,
  "algorithm": "HS256",
  "db_connection_type": "sync",
  "db_pool_size": 7,
  "rate_limit_requests": 150,
  "response_delay": 0.1
}
2025-06-05 09:21:54,503 - INFO - Updated configuration for product_service: {
  "db_connection_type": "sync",
  "db_pool_size": 7,
  "rate_limit_requests": 150,
  "response_delay": 0.1,
  "max_products_per_page": 30
}
2025-06-05 09:21:54,618 - INFO - Updated configuration for order_service: {
  "db_connection_type": "sync",
  "db_pool_size": 5,
  "rate_limit_requests": 100,
  "response_delay": 0.0,
  "max_orders_per_page": 20,
  "default_order_status": "pending"
}
```


Рис. 8. Результат застосування конфігурацій нового сценарію до кожного з мікросервісів
Fig. 8 The result of applying the new scenario configurations to each of the microservices

У всіх експериментах з поточними конфігураціями засіб АУКМ коректно ідентифікував поточний стан мікросервісів та застосовував відповідні параметри конфігурації, що підтверджує його працездатність. Важливим чинником для застосування цього підходу на практиці є час, який потрібен для пошуку такого рішення, і у Таблиці 3 та на рисунку 9 наведеніані результати досліджень з порівняння часу виконання алгоритмів п'яти CBR методів в залежності від об'єму бази прецедентів.

Таблиця 3. Порівняння часу виконання CBR-методів в залежності від об'єму бази прецедентів
Table 3. Comparison of execution time of CBR methods depending on the volume of the precedent databases

Метод	Час виконання алгоритмів (мс)				
	для 50 записів	для 100 записів	для 200 записів	для 500 записів	для 1000 записів
K-Nearest Neighbors	44,8	58,3	68,9	105,2	152,4
Weighted K-Nearest Neighbors	49,6	64,1	75,8	112,7	165,9
Feature-Based Retrieval	47,3	61,9	72,5	109,8	159,2
Cluster-Based Retrieval	53,9	69,2	82,4	126,5	190,8
Indexing and Hashing	27,6	32,1	36,9	43,2	50,3

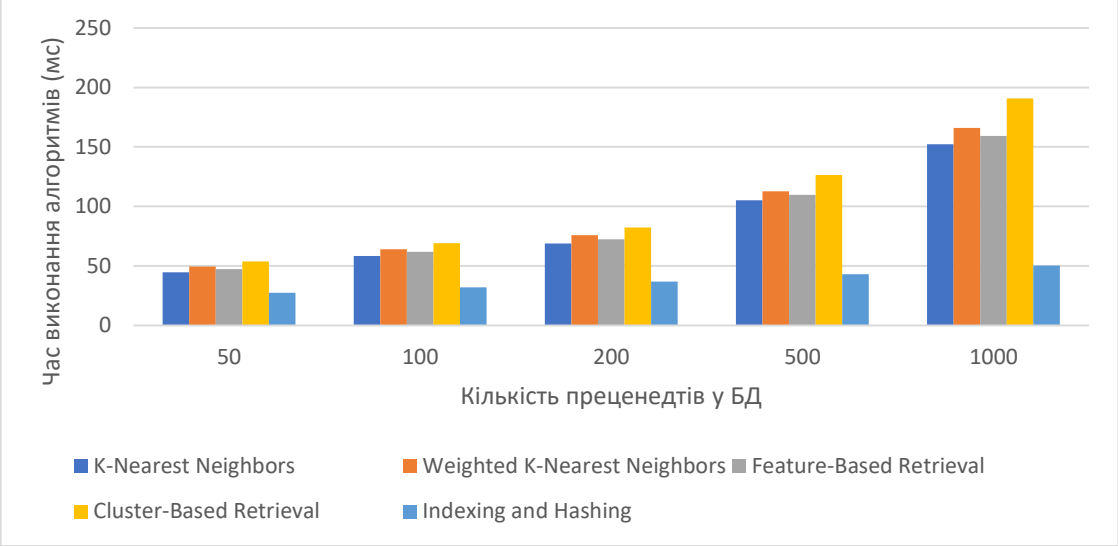

Рис. 9. Стовпчаста діаграма порівняння швидкодії всіх CBR-методів
Fig. 9 Bar chart comparing the speed of all CBR methods

Аналізуючи результати дослідження можна зробити висновки, що за критерієм швидкодії оновлення конфігурацій найкращим виявився CBR метод Indexing and Hashing завдяки попередній індексації БД прецедентів, що суттєво скорочує час пошуку. Метод KNN демонструє лінійне зростання часу з обсягом бази прецедентів і забезпечує стабільність при малих даних, а метод Weighted KNN надає гнучкість за рахунок вагових коефіцієнтів, але працює повільніше за інших. Методи Feature-Based та Cluster-Based є компромісом між точністю, масштабованістю та обчислювальними витратами (див. таблицю 3).

4. Висновки та напрямки подальших досліджень

У цьому дослідженні розв'язано актуальну науково-технічну задачу автоматизації адаптивного конфігурування застосунків з МСА шляхом створення інтелектуального інструментального засобу з модулем прийняття рішень на основі методів CBR. Реалізовано програмний тестовий полігон та розроблено інструментальний засіб АУКМ із чітко визначеними вимогами до

продуктивності, масштабованості, безпеки та інтеграції. Побудовано ER-модель БД прецедентів і реалізовано веб-інтерфейс для моніторингу/ручного та автоматичного конфігурування МСА.

Також в роботі оцінено часові витрати п’яти різних CBR-методів у діапазоні розмірів БД прецедентів у діапазоні [50-1000] записів. Найвищу швидкодію продемонстрував Indexing and Hashing (орієнтовно 27,6–50,3 мс), KNN показав лінійне зростання часу з обсягом кейс-бази, а Weighted KNN забезпечив більшу керованість за рахунок вагових коефіцієнтів для окремих параметрів опису прецедентів.

СПИСОК ЛІТЕРАТУРИ

1. Su R., Li X., Taibi D. From Microservice to Monolith: A Multivocal Literature Review. Electronics. 2024. Vol. 13, No. 8. Art. 1452. DOI: 10.3390/electronics13081452.
URL: https://www.mdpi.com/2079-9292/13/8/1452
2. Pozdniakova O.; Mažeika D.; Cholomskis A. SLA-Adaptive Threshold Adjustment for a Kubernetes Horizontal Pod Autoscaler. *Electronics*. 2024. Т. 13, № 7. 1242. DOI: 10.3390/electronics13071242.
URL: https://www.mdpi.com/2079-9292/13/7/1242 // [2]
3. Зінов’єв Д.В., Ткачук М.В. Аналіз, класифікація та тестування інструментальних засобів для управління конфігураціями програмних мікросервісів. Вісник Харківського національного університету імені В.Н.Каразіна, сер. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». 2023. вип. 57. С.33-42, DOI: 10.26565/2304-6201-2023-57-03
URL: https://periodicals.karazin.ua/mia/article/view/23251 // [3]
4. Ткачук М. В., Зінов’єв Д.В. Розробка та дослідження алгоритмічної моделі для адаптивного управління конфігураціями програмних мікросервісів. Системи обробки інформації. 2024. № 2(177). - С. 116 –120, DOI:10.30748/soi.2024.177.13
URL: https://doi.org/10.30748/soi.2024.177.12 // [4]
5. Figueira J.; Coutinho C. Developing Self-Adaptive Microservices. *Procedia Computer Science*. 2024. Т. 232. С. 264–273. DOI: 10.1016/j.procs.2024.01.026. // [5]
URL: https://www.sciencedirect.com/science/article/pii/S1877050924000267
6. Ma W.; Wang R.; Gu Y.; Meng Q.; Huang H.; Deng S.; Wu Y. Multi-objective microservice deployment optimization via a knowledge-driven evolutionary algorithm. *Complex & Intelligent Systems*. 2021. Т. 7. С. 1153–1171. DOI: 10.1007/s40747-020-00180-1. // [6]
URL: https://link.springer.com/article/10.1007/s40747-020-00180-1
7. Niswar M.; Safruddin R. A.; Bustamin A.; Aswad I. Performance Evaluation of Microservices Communication with REST, GraphQL, and gRPC. *International Journal of Electronics and Telecommunications*. 2024. Т. 70, № 2. С. 429–436. DOI: 10.24425/ijet.2024.149562.
URL: https://ijet.ise.pw.edu.pl/index.php/ijet/article/view/10.24425-ijet.2024.149562 /
8. Yan A.; Cheng Z. A Review of the Development and Future Challenges of Case-Based Reasoning. *Applied Sciences*. 2024. Т. 14, № 16. 7130. DOI: 10.3390/app14167130.
URL: https://www.mdpi.com/2076-3417/14/16/7130

REFERENCES

1. R. Su, X. Li, and D. Taibi, “From Microservice to Monolith: A Multivocal Literature Review,” *Electronics*, vol. 13, no. 8, p. 1452, Apr. 2024, doi: 10.3390/electronics13081452.
Available: https://www.mdpi.com/2079-9292/13/8/1452
2. O. Pozdniakova, D. Mažeika, and A. Cholomskis, “SLA-Adaptive Threshold Adjustment for a Kubernetes Horizontal Pod Autoscaler,” *Electronics*, vol. 13, no. 7, 1242, 2024, doi: 10.3390/electronics13071242.
Available: https://www.mdpi.com/2079-9292/13/7/1242
3. Zinov’ev, D.V. and Tkachuk, M.V. (2025), “Rozrobka ta doslidzhenniy algoritmichnoi modeli gla adaptivnogo upravlinnya konfiguratsiyami programnuh mikroservisiv” [Development and research of an algorithmic model for adaptive configuration management of software microservices], *Information processing systems*. 2024. № 2(177). - P. 116 –120. [in Ukrainian]
Available: https://doi.org/10.30748/soi.2024.177.13
4. Zinov’ev, D., & Tkachuk, M. (2023). “Analiz, klasyfikatsiia ta testuvannia instrumentiv upravlinnia konfiguratsiiemi dlia programnykh mikroservisiv” [Analysis, classification and testing of

configuration management tools for software microservices] *Bulletin of V.N. Karazin Kharkiv National University, Series «Mathematical Modeling. Information Technology. Automated Control Systems»*, *57*, 32-41, doi: 10.26565/2304-6201-2023-57-03
Available: https://periodicals.karazin.ua/mia/article/view/23251
5. J. Figueira and C. Coutinho, "Developing Self-Adaptive Microservices," *Procedia Computer Science*, vol. 232, pp. 264–273, 2024, doi: 10.1016/j.procs.2024.01.026.
Available: https://www.sciencedirect.com/science/article/pii/S1877050924000267
6. W. Ma, R. Wang, Y. Gu, Q. Meng, H. Huang, S. Deng, and Y. Wu, "Multi-objective microservice deployment optimization via a knowledge-driven evolutionary algorithm," *Complex & Intelligent Systems*, vol. 7, pp. 1153–1171, 2021, doi: 10.1007/s40747-020-00180-1.
Available: https://link.springer.com/article/10.1007/s40747-020-00180-1
7. M. Niswar, R. A. Safruddin, A. Bustamin, and I. Aswad, "Performance Evaluation of Microservices Communication with REST, GraphQL, and gRPC," *International Journal of Electronics and Telecommunications*, vol. 70, no. 2, pp. 429–436, Jun. 2024, doi: 10.24425/ijet.2024.149562.
Available: https://ijet.ise.pw.edu.pl/index.php/ijet/article/view/10.24425-ijet.2024.149562
8. Yan and Z. Cheng, "A Review of the Development and Future Challenges of Case-Based Reasoning," Applied Sciences, vol. 14, no. 16, art. 7130, 2024, doi: 10.3390/app14167130.
Available: https://www.mdpi.com/2076-3417/14/16/7130

Zinov'ev Dmytro *Senior lecturer of the Department of Intelligent Software Systems and Technologies, Education and Research Institute of Computer Sciences and Artificial Intelligence, V. N. Karazin Kharkiv National University, 4 Svobody Sq., Kharkiv, 61022, Ukraine*

Tkachuk Mykola *Doctor of technical sciences, Professor; Professor of the Department of Intelligent Software Systems and Technologies, Education and Research Institute of Computer Sciences and Artificial Intelligence, V. N. Karazin Kharkiv National University, 4 Svobody Sq., Kharkiv, 61022, Ukraine*

Architecture, software implementation and results analyzing of the usage an intelligent tool for configuring microservice applications

Actuality. Developing applications with a microservice architecture requires effective configuration management under varying load conditions, reliability, fault tolerance, and scalability requirements. This creates a need for intelligent adaptive configuration tools that can operate in near-real time mode.
Goal. To create an intelligent tool for adaptive management of MCA configurations with a decision-making module based on Case-Based Reasoning (CBR), design its architecture, make a software implementation, as well as experimentally evaluate the work on a test site and compare several CBR methods.
Research methods. The basic concepts of MSA configuration processes are clarified; a polygon with three services (auth, product, order) and performance requirements (≤1000 simultaneous requests, average latency ≤200 ms) is designed. Adaptive microservice configuration management is implemented as a microservice with REST API (FastAPI) and a precedent database (PostgreSQL); QoS, resource, "cost" and adaptability metrics are used. Five CBR methods are investigated: K-Nearest Neighbors, Weighted KNN, Feature-Based Retrieval, Cluster-Based Retrieval, Indexing & Hashing. A series of measurements of configuration selection time for a precedent database of 50–1000 records with averaging over 100 runs is conducted.
Results. The subsystem correctly identifies states and applies relevant configurations for different scenarios (low/medium/high/peak), meeting the requirement of a matching time of ≤0.5 s. The Indexing & Hashing method demonstrated the highest performance (≈27.6–50.3 ms for 50–1000 precedents); KNN has a linear time growth, and Weighted KNN provides controllability due to metric weights. The implemented web interface provides monitoring and manual/automatic mode of applying changes in real time.
Conclusions. The proposed architecture and software implementation of the CBR tool confirm the practical feasibility of adaptive configuration of the MCA and create a basis for managed solutions that are scaled by data. Further directions are outlined: evolution of the case base with online learning, multi-criteria optimization (performance/reliability/cost/energy efficiency), deeper integration with orchestrators and service mesh and increased explainability of solutions.

Keywords: *microservice, architecture, configuration management, intelligent approach, Case Based Reasoning, CBR, intelligent tool, testing, quality, metrics, model.*

Надійшла у першій редакції 30.08.2025, в останній - 01.10.2025.
The first version has been received on 30.08.2025, the final version – on 01.10.2025.