DOI: https://doi.org/10.26565/2304-6201-2023-57-03

УДК (UDC) 004.051

**Зинов'єв Дмитро Володимирович** *старший викладач кафедри моделювання систем і технологій 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 моделювання систем і технологій 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

# Аналіз, класифікація та тестування інструментальних засобів для управління конфігураціями програмних мікросервісів

**Актуальність.** Розробка програмних застосунків з мікросервісною архітектурою (МСА) потребує вирішення проблем їх проектування, інтеграції, масштабування, адаптації, надійності, відмовостійкості та, в цілому, підвищення показників якості таких систем, і тому питання ефективного управління конфігураціями МСА є актуальної науково-технічною задачею.
**Мета.** Метою цього дослідження є аналіз функціональних особливостей існуючих інструментальних засобів для конфігурування МСА, а також їх класифікація та тестування, що уможливлює їх вдосконалення шляхом розробки модельно-технологічних рішень для забезпечення адаптивного управління в таких застосунках.
**Методи дослідження.** Для досягнення мети дослідження визначені базові поняття та сутність процесів управління конфігураціями МСА, проведено аналіз функціональних можливостей деяких сучасних інструментальних засобів для конфігурування МСА та побудована їх класифікація. Для подальшого розгляду мотивовано обрано фреймворк Microconfig.io, програмно реалізовано конкретний приклад відповідного МСА застосунку, проведено його тестування та проаналізовано отримані результати.
**Результати.** Зроблено обґрунтований висновок про можливість та доцільність підвищення ефективності інструментального засобу MicroConfig.io шляхом його використання у складі перспективної інформаційної технології адаптивного управління процесом конфігурування МСА, для якої розроблена UML діаграма розміщення компонентів.
**Висновки.** Проаналізовані функціональні особливості сучасних інструментальних засобів для конфігурування МСА, побудована їх можлива класифікація та проведено програмне тестування однієї з типових таких систем, а саме, фреймворку MicroConfig.io. Запропоновано перспективна інформаційна технологія адаптивного управління процесом конфігурування МСА у вигляді компонентної діаграми та сформульовані напрямки подальших досліджень.

**Ключові слова:** програмні мікросервіси, управління конфігураціями, класифікація, тестування, адаптація, якість, метрика, інформаційна технологія, UML, діаграма компонентів, модель, засіб

**Як цитувати:** 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. 32-41. https://doi.org/10.26565/2304-6201-2023-57-03

## 1 Вступ. Мета та актуальність дослідження

Сучасні процеси, що пов'язані з життєвим циклом програмних застосунків зокрема з мікросервісною архітектурою (МСА), ставлять перед розробниками питання щодо їх проектування, інтеграції, масштабування, адаптації, надійності, відмовостійкості та, в цілому, підвищення показників якості таких систем. Зростання функціональності, великі об'єми даних, вимоги до продуктивності, інтеграція з іншими системами, високий темп розвитку технологій та багато інших факторів значно ускладнюють процеси розробки, впровадження та підтримки програмного забезпечення (ПЗ) [1, 2].

Для вирішення багатьох з цих проблем при розробці використовуються методи та технології управління конфігураціями програмного продукту (Configuration Management). На даний час існує

декілька типів інструментальних засобів для управління конфігураціями ПЗ, це, наприклад системи [3, 4]: контролю версій (Version Control System); автоматизованого розгортання (Deployment Automation); управління залежностями (Dependency Management); конфігурації (Configuration Management); управління змінами (Change Management) та інш. Вони допомагають ефективно керувати різними аспектами програмних продуктів, включаючи версіонування, розгортання, масштабування, залежності, конфігурацію та контроль змін. Ці методики та інструменти досить давно використовуються для підвищення якості ПП із монолітною архітектурою. Для розподілених систем, а саме для ПЗ з мікросервісною архітектурою, де застосунки розгортаються, як набір незалежних компонентів, ефективне управління конфігураціями стає ключовим аспектом забезпечення стабільності, масштабованості та надійності систем [5].

Питання ефективності використання інструментів для УКПП для розподілених систем на базі МСА ще до кінця не вивчені. Тому основною задачею цього дослідження є аналіз функціональних особливостей існуючих інструментальних засобів для конфігурування програмних мікросервісів, а також їх класифікація та тестування, з метою подальшого вдосконалення шляхом розробки модельно-технологічних рішень для забезпечення можливостей адаптивного управління в таких застосунках.

## 2 Основні поняття, функціональність та класифікація інструментальних засобів управління конфігураціями мікросервісів

### 2.1 Управління конфігураціями програмного забезпечення

Управління конфігураціями ПЗ включає такі основні аспекти:

- управління версіями, яке забезпечує відстеження і контроль за різними версіями компонентів програмного продукту, включи збереження вихідного коду, бібліотек, конфігураційних файлів і документації для кожної версії;
- управління змінами, яке визначає процес заявки, оцінки, впровадження та відстеження змін, що вносяться до програмного продукту, включи управління проблемами, заявками на виправлення помилок, нововведеннями і іншими змінами;
- конфігураційне управління, яке встановлює і підтримує правильну конфігурацію програмного продукту та його компонентів, включи управління залежностями, конфігураційними файлами, налаштуваннями середовища і іншими конфігураційними елементами [2];
- забезпечення якості, до якого включені процеси тестування, перевірки та перевірки програмного забезпечення для забезпечення відповідності вимогам, стандартам і очікуванням;
- відстеження залежностей, що забезпечує відстеження залежностей між різними компонентами програмного продукту;
- управління розгортанням, що визначає процес підготовки і розгортання програмного забезпечення в різних середовищах, включаючи тестове і виробниче середовища та забезпечує контрольоване і надійне розгортання змін та версій програмного продукту.
- аудит і звітність, де забезпечуються здатність відстежувати всі зміни, вносити аудит і генерувати звіти про конфігурацію, версії та зміни в програмному продукті;
- інтеграцію з іншими процесами, що сприяє ефективному функціонуванню і успішному виконанню проекту.

### 2.2 Конфігураційні файли та формати

Управління конфігураціями програмного забезпечення включає в себе використання конфігураційних файлів та форматів для збереження налаштувань та параметрів програми. У роботі розглянуті деякі з популярних конфігураційних файлів, а саме:

- YAML (YAML Ain't Markup Language) –текстовий формат, який має простий для читання синтаксис, що дозволяє зберігати дані у вигляді списків, словників та примітивних типів даних [6];
- JSON (JavaScript Object Notation) – легкий текстовий формат обміну даними, який широко використовується для конфігураційних файлів і базується на синтаксисі JavaScript та дозволяє зберігати дані у вигляді пар "ключ-значення" [7];

- XML (Extensible Markup Language) – є розширюваним мовним форматом розмітки, який використовує власну синтаксичну структуру з відкриваючими та закриваючими тегами для представлення даних [8];
- INI (Initialization) файли – є простими текстовими файлами, що використовуються для зберігання конфігураційних даних у вигляді секцій та параметрів, де кожна секція містить набір ключів-значень;
- Properties файл – формат властивостей, часто використовуваний у Java-проектах, є простим текстовим форматом, в якому дані зберігаються у вигляді пар "ключ=значення";
- TOML (Tom's Obvious, Minimal Language) – є простим форматом конфігураційних файлів, який нагадує INI-файли, але має більш розширений синтаксис та підтримку більш складених структур даних;
- HCL (HashiCorp Configuration Language) – використовується в інструментах інфраструктури коду, таких як Terraform, для опису конфігурації ресурсів. HCL має простий та декларативний синтаксис, який легко зрозуміти і підтримує розширення;
- ENV (Environment Variables) – ENV-змінні зазвичай використовуються для передачі конфігураційних параметрів через зовнішнє середовище, наприклад, операційну систему або контейнер. Це гнучкий підхід, оскільки дозволяє легко змінювати параметри без зміни конфігураційного файлу;
- конфігурація бази даних оскільки деякі програми зберігають конфігураційні параметри безпосередньо в базі даних, де можна зберігати та керувати параметрами за допомогою запитів SQL або ORM (об'єктно-реляційне відображення).

### 2.3 Класифікація інструментальних засобів для управління конфігураціями програмних застосунків

На рисунку 2.1 наведено схему класифікації інструментальних засобів які можна використовувати для управління конфігураціями мікросервісів.

Інструменти контролю версій, такі як Git, Subversion та Mercurial, дозволяють відстежувати зміни в програмному забезпеченні та його компонентах, зберігати історію змін, відстежувати версії коду та спільно працювати над проектом у команді [9-11]. Ці інструменти допомагають ефективно керувати розробкою та забезпечувати зручну спільну роботу над програмними мікросервісами.

Інструменти, такі як Ansible, Puppet, Chef, Terraform та Kubernetes, надають засоби для автоматизації процесів розгортання та управління інфраструктурою програмних мікросервісів [12, 13]. Вони дозволяють швидко та надійно розгортати сервіси, керувати їх налаштуваннями та забезпечувати масштабованість та відновлюваність.

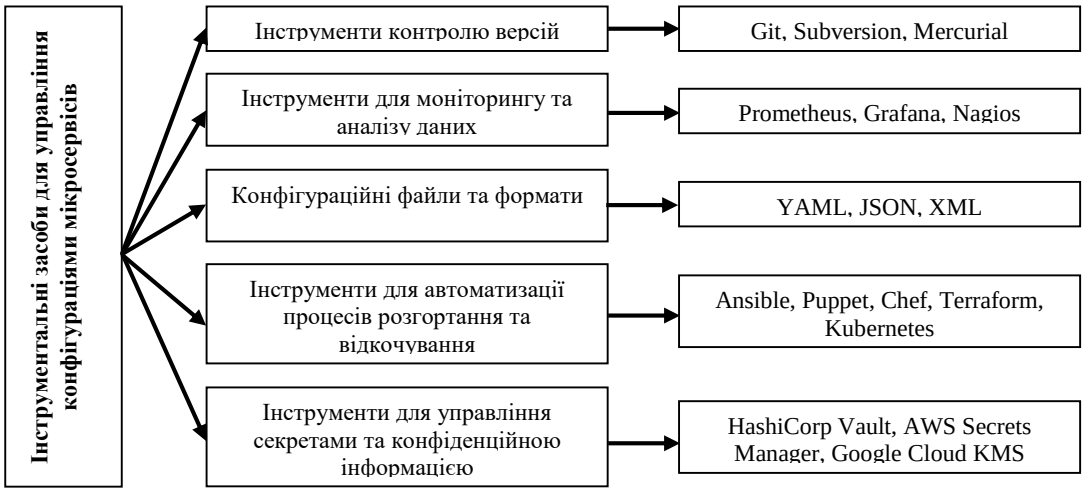


*Рис. 2.1 Класифікація інструментальних засобів управління конфігураціями ПП*

Інструменти Prometheus, Grafana та Nagios дозволяють здійснювати моніторинг та аналіз стану програмних мікросервісів [14, 15]. Вони забезпечують збір, візуалізацію та аналіз метрик,

журналів та інших даних, що дозволяє оперативно виявляти проблеми та оптимізувати роботу системи.

Інструменти HashiCorp Vault, AWS Secrets Manager, Google Cloud KMS допомагають забезпечувати безпеку та управління секретами, такими як паролі, ключі API та інша конфіденційна інформація, яка використовується в програмних мікросервісах [16, 17]. Вони дозволяють зберігати, керувати та забезпечувати безпеку доступу до цих секретів.

Конфігураційні файли використовуються для збереження налаштувань та параметрів програмного забезпечення або його компонентів. Формати, такі як YAML, JSON або XML, часто використовуються для представлення цих даних у конфігураційних файлах.

## 3 Тестування та аналіз результатів проведених програмних експериментів

### 3.1 Мотивований вибір та опис функціональних можливостей інструментального засобу Microconfig.io

Для вибору інструменту для управління конфігураціями програмних мікросервісів з урахуванням потреб і вимог досліджуваної системи були визначені наступні критерії: функціональні можливості, інтеграція з іншими інструментами, простота використання, розширюваність, гнучкість, надійність та безпека.

Серед існуючих інструментальних засобів для управління конфігураціями програмних мікросервісів був вибраний новий фреймворк Microconfig.io, який дозволяє вирішувати рутинні завдання з налаштування програм. Основне завдання, яке вирішує - це звести до мінімуму дублюючі параметри налаштування та спростити доставку конфігурації до мікросервісів, тим самим зменшити ймовірність додавання помилок. Цей засіб використовує підхід "конфігурація-як-код" для управління МСА, тобто дозволяє описувати конфігурації у вигляді програмного коду, що забезпечує більшу гнучкість та автоматизацію. На рисунку 3.1 показана стартова сторінка при запуску фреймворку Microconfig.io.

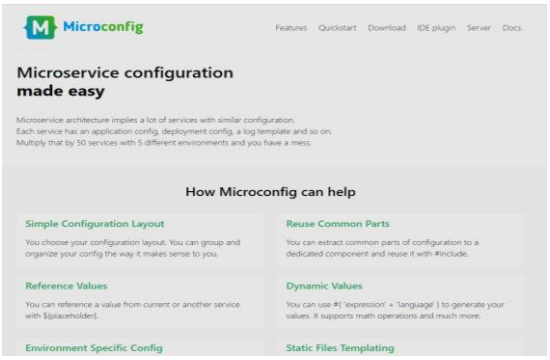


*Рис. 3.1 Стартова сторінка фреймворку MicroConfig.io*

До основних функцій та можливостей програми Microconfig.io відносяться:

- централізоване збереження конфігурацій, що дозволяє зберігати всі конфігураційні файли в одному місці з метою полегшення управління ними та забезпечує їх однорідність;
- версіонування та контроль доступу, що дозволяє стежити за їхніми змінами у програмі та дає можливість відновлювати попередні версії;
- інтерфейс користувача надає інтуїтивно зрозумілий інтерфейс для управління конфігураціями, що дозволяє легко переглядати, редагувати та виконувати пошук конфігураційних параметрів;
- інтеграція з автоматизованими процесами, коли є можливість легко інтегруватися з автоматизованими процесами розгортання, неперервної поставки та іншими інструментами розробки, що дозволяє автоматизувати процеси налаштування та забезпечує їхню стабільність.

Одною з головних переваг Microconfig.io у порівнянні з аналогічними застосунками є можливість інтеграції з різними інструментами та технологіями розробки, наприклад такими як:

- інструментами для автоматизованого розгортання (Ansible, Chef або Puppet);
- інструментами неперервної поставки (Jenkins, GitLab CI/CD або Travis CI);
- інструментами управління контейнерами ( Docker або Kubernetes);

- інструментами системного моніторингу ( Prometheus або Nagios).

Це дозволяє зручно налаштовувати кожен програмний мікросервіс незалежно, забезпечуючи гнучкість та масштабованість всієї програмної системи.

### 3.2 Розробка тестового мікросервісного застосунку

Для дослідження особливостей процесів конфігурування програмних застосунків за допомогою Microconfig.io був розроблений тестовий застосунок з МСА, при цьому не розглядалися питання його безпосереднього використання як програмного продукту, а основною задачею створення було використання його у якості стенда для тестування наявних засобів конфігурування.

Застосунок складався з 3 окремих мікросервісів:

- User-service відповідав за керування користувачами;
- Product-service відповідав за керування товарами;
- Order-service відповідав за керування замовленнями.

На рисунку 3.2 наведена компонентна структура тестового застосунку, з якої видно що Order-sevice взаємодіє з User-service та Product-service, а локальні дані від кожного сервісу зберігаються в локальну базу даних.

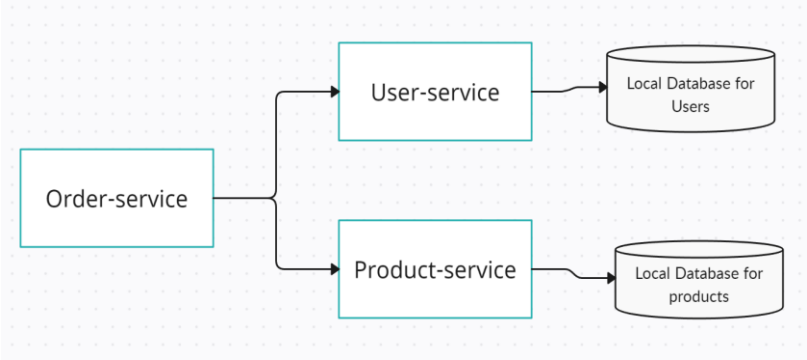


*Рис. 3.2 Компонентна структура тестового МСА застосунку*

На рисунку 3.3 наведена діаграма класів застосунку.

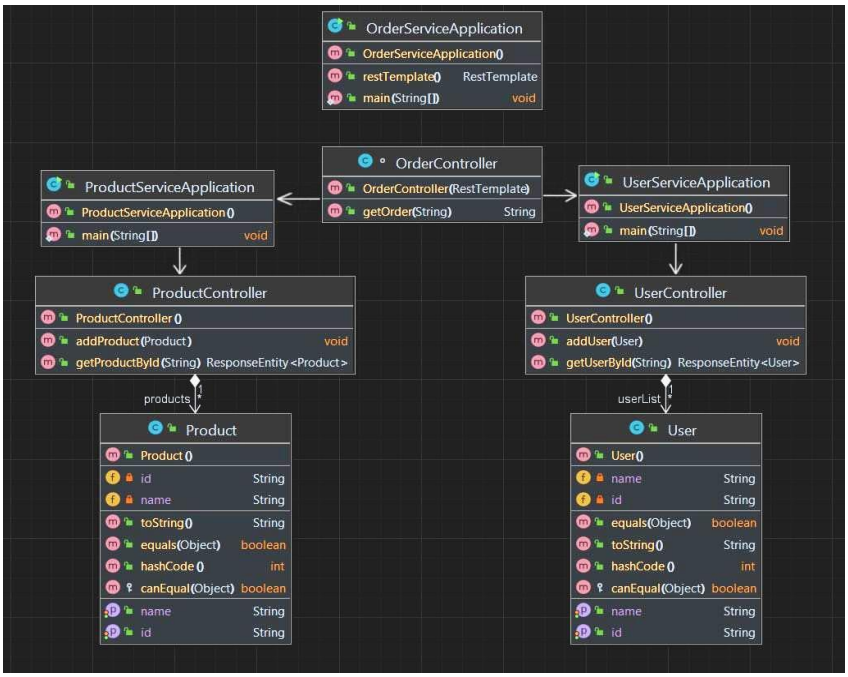


*Рис. 3.3 UML діаграма класів тестового застосунку з МСА*

Обрана архітектура забезпечувала модульність та розділення обов'язків між мікросервісами, що дозволяло ефективно масштабувати систему. Кожен мікросервіс міг функціонувати незалежно, що полегшувало розгортання та підтримку системи. Крім того, мікросервіси мали змогу взаємодіяти між собою для отримання інформації або виконання певних операцій.

Беручі до уваги те, що полігон розроблявся виключно для тестування, було прийняте рішення не використовувати окремі бази даних, такі як SQL або NoSQL. Був обраний варіантів локального збереження файлів на комп'ютері.

### 3.3. Основні етапи тестування та аналіз отриманих результатів

Для тестування був створений єдиний проект з конфігураціями який включав спільні конфігурації для логування та моніторингу в Microconfig.io. Основні етапи роботи з плагіном це:

- перехід до компонента;
- вирішення заповнювача;
- попередній перегляд конфігурації;
- конфігурація збірки (через «Запуск конфігурації»).

Для генерування конфігурації для user сервісу в певному оточенні в Microconfig.io необхідно виконати команду, яка зображена на рисунку 3.4.


*Рис. 3.4 Створення конфігурації для user сервісу*

Створення конфігурацій для product та order сервісів показано на рисунку 3.5.


*Рис. 3.5 Створення конфігурацій для product та order сервісів*

Результат створення папки збірки з конфігураціями для середовища розробки (dev) для сервісів user, product і orders за допомогою застосунку Microconfig.io показаний на рисунку 3.6. Кожен сервіс має свою власну підпапку з конфігураціями, які відповідають середовищу розробки.


*Рис. 3.6 Результат автоматичного створення конфігураційних файлів за допомогою Microconfig.io*

Таким чином, в результаті тестування засобу Microconfig.io була показана принципова можливість створення конфігураційних файлів, які включають налаштування як для кожного мікросервісу проекту, так і для всіх сервісів одночасно. Це дозволяє значно спростити процес конфігурування проекту, підвищити швидкість впровадження змін в конфігурації, забезпечує надійність у роботі проекту.

**4 Перспективна технологія використання інструментального засобу Microconfig.io з можливостями адаптивного управління та моніторингу показників якості конфігурацій мікросервісів**

На підставі проведеного аналізу функціональних властивостей існуючих інструментальних засобів для конфігурування програмних мікросервісів, можна стверджувати, що на теперішній час переважна їх більшість лише забезпечує автоматизацію основних рутинних операцій у цих процесах, і не підтримує такі більш складні можливості їх налаштування, зокрема, адаптацію параметрів нових конфігурацій з урахуванням стану операційного середовища виконання відповідних застосунків з МСА.

Те, що вирішення подібних проблем розробки більш ефективних засобів конфігурування та подальшої координації застосунків з МСА є актуальною науково-технічною задачею, підтверджує поява новітніх публікацій з цих питань, зокрема, в роботах [18, 19]. Для цього в них розробляються підходи із застосуванням алгоритмічних моделей динамічного управління роботою мікросервісів на основі накопичення та обробка інформації про їх взаємозв'язки (inter-service dependencies) та ланцюжки викликів (service call chains), що дозволяє, у кінцевому рахунку, зменшити обсяг необхідної пам'яті для функціонування кластерів мікросервісів та, в цілому, підняти продуктивність роботи усього МСА застосунку. Враховуючи такі тенденції у проблематиці вирішення задач підвищення якості функціонування мікросервісів, у цій роботі пропонується розглянути можливість підвищення ефективності таких інструментальних засобів як Microconfig.io шляхом їх використання у складі перспективної інформаційної технології адаптивного управління процесом конфігурування МСА. Ця технологія представлена у вигляді спрощеної компонентної діаграми в нотації UML на рисунку 4.1.

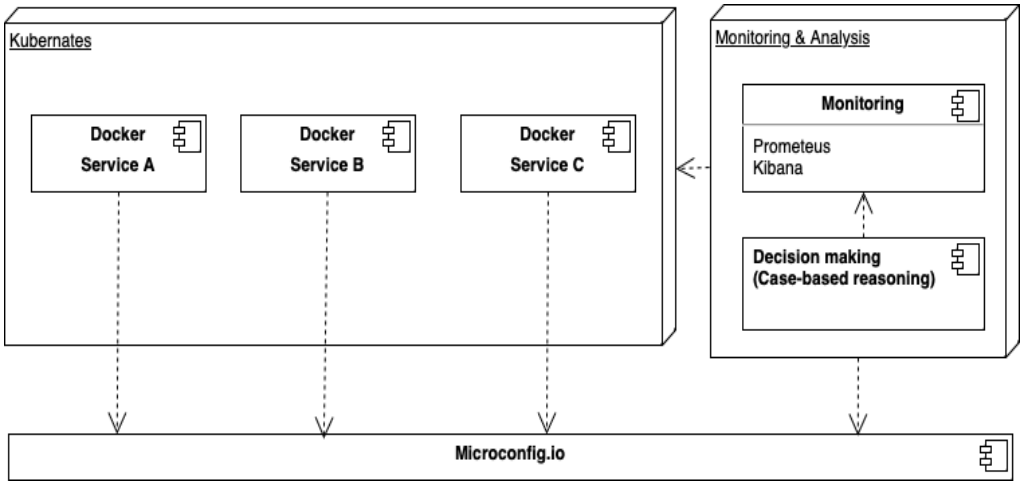


*Рис 4.1 Компонентна діаграма адаптивного управління процесом конфігурування МСА*

На цій діаграмі операційна платформа Kubernetes відіграє ключову роль у розгортанні та оркестровці множини контейнерів Docker, у яких розміщуються окремі мікросервіси, Кожен мікросервіс, розміщений у своєму власному Docker контейнері, може бути налаштований з використанням інструменту Microconfig.io, який спрощує процес конфігурації через можливість генерації конфігураційних файлів для різноманітних технологій та платформ, включно із Spring, Ansible, тощо (див. Розділ 3). Моніторинг та аналіз параметрів функціонування системи здійснюється завдяки інтеграції таких інструментів як Prometheus [20] та Kibana [21]. Prometheus є інструментом моніторингу з відкритим кодом, який збирає значення певних метрик з наявних конфігурацій МСА на заданих інтервалах, виводить результати та може генерувати системні повідомлення, якщо щось йде не за планом. Ці метрики включають показники від окремих

мікросервісів (Docker) та від інфраструктури їх функціонального середовища (Kubernetes), які відображають стан системи в реальному часі. В свою чергу, компонент Kibana, що є частиною фреймворку Elastic Stack, дозволяє користувачам візуалізувати дані з ElasticSearch, які потім компонент Prometheus може збирати і створювати графіки та візуальні схеми, що полегшує розуміння великої кількості інформації та прискорює процес прийняття рішень щодо процесу адаптивного конфігурування МСА застосунку. Компонент 'Decision making' у цій інформаційній технології, який має бути розробленим додатково, може використовувати методи CBR (Case-based Reasoning) для аналізу отриманих даних і, відповідно до певних правил та алгоритмів, може запропонувати коригування поточних конфігурацій окремих мікросервісів [22], що забезпечує підвищення продуктивності та надійності середовища їх функціонування.

**5 Висновки та напрямки подальших досліджень**

У цьому дослідженні розглянута актуальна науково-технічні задача підвищення ефективності використання існуючих інструментальних засобів для конфігурування застосунків з мікросервісною архітектурою (МСА) шляхом розробки модельно-технологічних рішень для забезпечення можливостей адаптивного управління в таких застосунках. Визначені базові поняття та сутність процесів управління конфігураціями програмних мікросервісів, проведено аналіз функціональних можливостей інструментальних засобів для конфігурування МСА і побудована їх класифікація. Для подальшого дослідження обрано систему Microconfig.io, яка представляє собою сучасний інструментальний фреймворк для автоматизації усіх основних рутинних операція в процесах конфігурування МСА. На основі проведеної аналітичної та експериментальної роботи зроблено обґрунтований висновок про можливість та доцільність підвищення ефективності інструментального засобу Microconfig.io шляхом його використання у складі перспективної інформаційної технології адаптивного управління процесом конфігурування МСА, для якої розроблена відповідна UML діаграма компонентів.

У якості подальших досліджень у цій проблематиці можливо запропонувати розробку інформаційної технології з використанням інтелектуальних методів прийняття рішень для координації взаємодії окремих мікросервісів також на етапі їх виконання за умов обмежених операційних ресурсів, таких як оперативна пам’ять, що у кінцевому рахунку може забезпечити підвищення продуктивності МСА застосунків.

ЛІТЕРАТУРА

1. Зінов’єв Д.В., Ткачук М.В.. Тріщенко І.В. Моделі та технології забезпечення якості сервіс-орієнтованих програмних систем: сучасний стан та перспективні напрямки досліджень // Матеріали міжн. науков.-техн. конференції КМНТ-2021 (м. Харків, 23-25 квітня 2021 року) – Х.: ХНУ імені В.Н. Каразіна, 2021. – С. 166-169.
2. Зінов’єв Д.В., Ткачук М.В. Розробка інструментального засобу для автоматизованої оцінки показників якості мікросервісних застосунків // «Стан, досягнення та перспективи інформаційних систем і технологій», XXIII Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів. Одеса, 20-21 квітня 2023 р. - Одеса, ОНТУ, 2023 р. – с. 239-240.
3. G. Singh, B. Singh, S. Srivastva. Comparative Analysis of various Software Configuration Management Tools. International Journal of Latest Engineering and Management Research (IJLEMR). Volume 1. Issue 1.November 2015. PP.01-07
4. Site: SimpliLearn. Top 12 Effective & Popular Software Configuration Management Tools. Last updated on Feb 17, 2023. [Електронний ресурс] URL: https://www.simplilearn.com/configuration-management-tools-article (дата звернення: 23.10 .2023)
5. Chris Richardson "Microservices Patterns: With examples in Java" pp. 1–65. Manning (2019)
6. Redhat сайт. YAML. [Електронний ресурс] URL: https://www.redhat.com/en/topics/automation/what-is-yaml#:~:text=YAML%20is%20a%20human%2Dreadable,is%20for%20data%2C%20not%20documents. (дата звернення: 15.05.2023).
7. Hubspot сайт. JSON [Електронний ресурс] URL: https://blog.hubspot.com/website/json-files#:~:text=What%20is%20a%20JSON%20file,the%20values%20containing%20related%20data. (дата звернення: 16.10.2023).

8. Indeed сайт. XML [Електронний ресурс] URL: https://www.indeed.com/career-advice/career-development/xml-file#:~:text=An%20XML%20file%20is%20an,that%20you%20wish%20to%20store. (дата звернення: 24.10.2023).
9. Git документація. [Електронний ресурс] URL: https://git-scm.com/ (дата звернення: 17.05.2023).
10. Subversion сайт. [Електронний ресурс] URL: https://subversion.apache.org/ (дата звернення: 24.10.2023).
11. Mercurial сайт. [Електронний ресурс] URL: https://www.mercurial-scm.org/ (дата звернення: 24.10.2023).
12. Ansible сайт. [Електронний ресурс] URL: https://www.ansible.com/ (дата звернення: 25.10.2023).
13. Puppet сайт. [Електронний ресурс] URL: https://www.puppet.com/ (дата звернення: 26.10.2023).
14. Grafana сайт. [Електронний ресурс] URL: https://grafana.com/ (дата звернення: 24.10.2023).
15. Nagios сайт. [Електронний ресурс] URL: https://www.nagios.org/ (дата звернення: 26.10.2023).
16. AWS Secrets Manager сайт. [Електронний ресурс] URL: https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html (дата звернення: 21.05.2023).
17. Cornelia Davis "Cloud Native Patterns: Designing Change-Tolerant Software" pp. 1–46. Manning (2019)
18. Wang, N.; Wang, L.; Li, X.; Qin, X. Fine-Grained Management for Microservice Applications with Lazy Configuration Distribution. Electronics 2023, 12, 3404. https://doi.org/10.3390/electronics12163404
19. Shangguang Wang , Yan Guo , Ning Zhang Peng Yang et.al. Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach // IEEE Transactions on mobile computing, Vol. 20, No. 3, 2021 pp. 939-951
20. Pivotto, Julien, and Brian Brazil. Prometheus: Up & Running. 2nd ed., O'Reilly Media, 2023
21. Konda, Madhusudhan. Elasticsearch in Action. 2nd ed., Manning Publications, September 2023
22. Tkachuk M.V., Zinoviev D.V. A case-based reasoning approach to quality assurance in microservice software systems // Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: Тези доповідей XXXI міжнародної науково-практичної конференції MicroCAD-2023, 17-20 травня 2023р., / за ред. проф. Сокола Є.І. – Харків, НТУ «ХПІ». – С.1034.

## REFERENCES

1. Zinov'ev D.V., Tkachuk M.V.. Trishchenko I.V. Models and technologies for ensuring the quality of service-oriented software systems: the current state and promising directions of research // Materials of the International. scientific and technical of the KMNT-2021 conference (Kharkiv, April 23-25, 2021) - Kh.: KhNU by V.N. Karazin, 2021. - P. 166-169. [in Ukrainian]
2. Zinov'ev D.V., Tkachuk M.V. Development of a tool for automated assessment of quality indicators of microservice applications // "State, achievements and prospects of information systems and technologies", XXIII All-Ukrainian scientific and technical conference of young scientists, graduate students and students. Odesa, April 20-21, 2023 - Odesa, ONTU, 2023 - p. 239-240. [in Ukrainian]
3. G. Singh, B. Singh, S. Srivastva. Comparative Analysis of various Software Configuration Management Tools. International Journal of Latest Engineering and Management Research (IJLEMR). Volume 1. Issue 1. November 2015. PP.01-07
4. Site: SimpliLearn. Top 12 Effective & Popular Software Configuration Management Tools. Last updated on Feb 17, 2023. URL: https://www.simplilearn.com/configuration-management-tools-article [date of access: 25.10.2023]
5. Chris Richardson "Microservices Patterns: With examples in Java" pp. 1–65. Manning (2019)
6. Site "Redhat". What is YAML? [Електронний ресурс] URL: https://www.redhat.com/en/topics/automation/what-is-yaml#:~:text=YAML%20is%20a%20human%2Dreadable,is%20for%20data%2C%20not%20documents [date of access: 25.10.2023]
7. Site "Hubspot". What Are JSON Files & How Do You Use Them? URL: https://blog.hubspot.com/website/json-files#:~:text=What%20is%20a%20JSON%20file,the%20values%20containing%20related%20data [date of access: 25.10.2023]
8. Site "Indeed". What Is an XML File? (Definition, Benefits and How To Open). URL: https://www.indeed.com/career-advice/career-development/xml-file#:~:text=An%20XML%20file%20is%20an,that%20you%20wish%20to%20store [date of access: 25.10.2023]
9. Site "Git. Documentation". URL: https://git-scm.com/doc [date of access: 25.10.2023]

10. Site “Subversion”. URL: https://subversion.apache.org/ [date of access: 25.10.2023]
11. Site “Mercurial”. URL: https://www.mercurial-scm.org/ [date of access: 25.10.2023]
12. Site “Ansible”. URL: https://www.ansible.com/ [date of access: 25.10.2023]
13. Site “Puppet”. URL: https://www.puppet.com/ [date of access: 25.10.2023]
14. Site “Grafana”. URL: https://grafana.com/ [date of access: 25.10.2023]
15. Site “Nagios”. URL: https://www.nagios.org/ [date of access: 25.10.2023]
16. AWS Secrets Manager”. URL: https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html [date of access: 25.10.2023]
17. Cornelia Davis "Cloud Native Patterns: Designing Change-Tolerant Software" pp. 1–46. Manning (2019)
18. Wang, N.; Wang, L.; Li, X.; Qin, X. Fine-Grained Management for Microservice Applications with Lazy Configuration Distribution. Electronics 2023, 12, 3404. https:// doi.org/10.3390/electronics12163404
19. Shangguang Wang, Yan Guo , Ning Zhang Peng Yang et.al. Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach // IEEE Transactions on mobile computing, Vol. 20, No. 3, 2021 pp. 939-951
20. Pivotto, Julien, and Brian Brazil. Prometheus: Up & Running. 2nd ed., O'Reilly Media, 2023
21. Konda, Madhusudhan. Elasticsearch in Action. 2nd ed., Manning Publications, September 2023
22. Tkachuk M.V., Zinoviev D.V. A case-based reasoning approach to quality assurance in microservice software systems // Information technologies: science, technology, technology, education, health: abstracts of reports of XXI international scientific and practical conference MicroCAD-2023, May 17-20, 2023, / edited by Prof. E.I. Sokol - Kharkiv, NTU "KhPI". - P.1034. [in Ukrainian]

**Zinov’ev Dmitry** *Senior lecturer of the Department of Systems and Technology Modeling, V. N. Karazin Kharkiv National University, 4 Svobody Sq., Kharkiv, 61022, Ukraine;*

**Tkachuk Mykola** *Doctor of Technical Sciences, Professor; Professor of the Department of Systems and Technology Modeling, V. N. Karazin Kharkiv National University, 4 Svobody Sq., Kharkiv, 61022, Ukraine.*

# Analysis, classification and testing of configuration management tools for software microservices

**Actuality.** The development of software applications with microservice architecture (MSA) requires to solve problems of their design, integration, scaling, adaptation, reliability, fault tolerance and, in general, an improving the quality indicators of such systems, and therefore the issue of effective management of MSA configurations is an urgent scientific and technical task.
**Goal.** The purpose of this study is to analyze the functional features of the existing tools for configuring MSA, as well as their classification and testing, which enables their improvement by developing of model-technological solutions to ensure adaptive management in such applications.
**Research methods.** In order to achieve the goal of the research, the basic concepts and the essence of the management processes of MSA configurations were determined, the functional capabilities of some modern tools for MSA configurating were analyzed and their classification is elaborated. For the further research, the Microconfog.io framework was chosen, for which the specific example of the MSA application was developed, its testing has been carried out, and the obtained results were analyzed.
**Results.** A well-founded conclusion was made about the possibility and expediency of increasing the effectiveness of the MicroConfig.io tool by using it as a part of the perspective information technology for adaptive management of the MSA configuration process, for which a corresponding UML component deployment diagram has been developed.
**Conclusions.** The functional features of modern tools for MSA configuring were analyzed, their possible classification was built, and the software testing of one of the typical such systems, namely, the Microconfig.io framework, was provided. A promising information technology of adaptive management of the MSA configuration process is proposed in the form of a component deployment diagram, and the directions for further research are formulated.

**Keywords:** software microservices, configuration management, classification, testing, adaptation, quality, metric, information technology, UML, component diagram, model, tool

**How to quote:** D.V. Zinov’ev, M.V. Tkachuk, “Analysis, classification and testing of configuration management tools for software microservices.” *Bulletin of V.N. Karazin Kharkiv National University, series “Mathematical modelling. Information technology. Automated control systems*, vol. 57, pp.32-41, 2023. https://doi.org/10.26565/2304-6201-2023-57-03

Надійшла у першій редакції 14.02.2023, в останній - 11.04.2023.