

DOI: 10.18372/2310-5461.71.21421
УДК 004(045)

А. І. Гізун, доцент

Київський національний університет імені Тараса Шевченка, Київ

orcid.org/0000-0002-2974-6987

e-mail: andriy.gizun@gmail.com

А. В. Моргун, аспірант

Національний університет «Київський авіаційний інститут», Київ

orcid.org/0009-0002-0027-1225

e-mail: 2175566@stud.kai.edu.ua

КРИТЕРІЇ, ВИМОГИ ТА МЕТРИКИ ЕТАЛОННОГО БЕНЧМАРКУ RefMSBench ДЛЯ РЕКОНСТРУКЦІЇ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Вступ

Мікросервісна архітектура (MSA) є поширеним підходом до побудови розподілених програмних систем, у яких окремі сервіси розробляються, розгортаються й масштабуються незалежно [1, 2]. Така організація полегшує еволюцію великих систем, однак не усуває складність: вона переносить її з внутрішньої структури модулів у мережу сервісів, контрактів, подій, сховищ даних і способів розгортання. Через автономність сервісів і команд, а також розподілений характер виконання, фактичні залежності мікросервісної системи важко простежувати лише за описами окремих компонентів [3, 4]. Це посилює розрив між запланованою та реалізованою архітектурою, що є однією з причин застосування методів реконструкції [5, 6].

Реконструкція архітектури спрямована на відновлення фактичної структури системи: її сервісів, залежностей, типів взаємодій і контрактів. Такі результати потрібні дослідникам та інженерним командам для оцінювання технічного боргу, планування рефакторингу, перевірки відповідності реалізації проєктному задуму та підтримки DevOps-процесів [5]. Повніше уявлення про мікросервісну систему можуть давати гібридні підходи, які поєднують статичні джерела, зокрема код, контракти й конфігурації розгортання, з динамічними даними виконання: логами, метриками та трасуваннями. Такі підходи складніше оцінювати, оскільки для них потрібна експериментальна система з доступними артефактами обох типів і еталонною моделлю для перевірки результатів. Проте досі важко порівнювати, наскільки повно різні методи реконструкції відновлюють архітектуру. Дослідники використовують різні цільові системи, працюють із різними джерелами даних і по-різному визначають еталонну архітектуру. За відсутності спільного еталона результат оцінювання залежить не лише від методу, а й від обраної експериментальної системи, доступних

даних і правил інтерпретації результатів.

Аналіз останніх досліджень і публікацій

Для експериментів із мікросервісними системами зручно використовувати бенчмарки з відкритим вихідним кодом, які можна знайти, розгорнути та використати для досліджень. Train Ticket було запропоновано як бенчмарк для досліджень проблем програмної інженерії [7]; Sock Shop і TeaStore розглядають у контексті тестування мікросервісів [8], а DeathStarBench запропоновано як бенчмарк для дослідження продуктивності мікросервісних систем у хмарних середовищах [9]. Попри це, огляди досліджень MSA підкреслюють важливість експериментальних середовищ, придатних для відтворюваного порівняння підходів і результатів [2]. У літературі також розглянуто кілька десятків інструментів реконструкції MSA: MicroART, статичні підходи на основі коду, контрактів і конфігурацій розгортання, а також динамічні підходи на основі телеметрії [5, 10]. Динамічні методи використовують логи, метрики й розподілені трасування [11, 12]; статичні підходи аналізують артефакти, доступні без запуску системи, зокрема вихідний код, байткод і структуру репозиторіїв [13, 14]. Окремі роботи також використовують контракти API та конфігурації розгортання, зокрема Docker-конфігурації й інші інфраструктурні артефакти [5]. Порівняння статичних інструментів показує, що навіть для одного класу методів потрібен окремий набір еталонних моделей, створених і перевірених вручну; саме з ним зіставляють відновлені залежності [15]. На якість оцінювання, крім самого алгоритму, впливають тип вхідних даних, обрана система та спосіб визначення еталона.

Частина робіт зосереджена на керуванні топологією мікросервісних систем. HydraGen генерує варіанти мікросервісних систем, а Adaptable TeaStore розширює конфігураційні можливості TeaStore як мікросервісного бенчмарку [16, 17].

Для реконструкції цього недостатньо: змінювана топологія має супроводжуватися еталонним графом сервісів, очікуваними взаємодіями для сценаріїв навантаження, описом збору телеметрії та метриками.

Проблема еталонного оцінювання виходить за межі порівняння інструментів реконструкції. Якщо відновлена модель використовується як вхід для автоматизованого проєктування MSA із застосуванням штучного інтелекту, її неточність робить подальші архітектурні рекомендації ненадійними [18]. У такому випадку еталон для перевірки реконструкції набуває практичної ваги, оскільки від нього залежить, наскільки можна довіряти автоматизованим рекомендаціям. Наявні дослідження достатньо повно описують інструменти реконструкції архітектури, однак не формують узгодженої основи для порівняння результатів реконструкції. Ця робота спрямована на визначення критеріїв, вимог і метрик еталонного мікросервісного бенчмарку.

Постановка задач дослідження

Метою дослідження є формування основи еталонного мікросервісного бенчмарку RefMSBench для відтворюваного оцінювання методів реконструкції архітектури. Для досягнення цієї мети поставлено такі часткові завдання:

- визначити критерії придатності мікросервісного бенчмарку до задач реконструкції архітектури;
- оцінити за визначеними критеріями наявні мікросервісні системи Sock Shop, TeaStore, DeathStarBench і Train Ticket;
- на основі проведеного аналізу сформулювати функціональні й нефункціональні вимоги до RefMSBench;
- визначити склад еталонної моделі, метрики оцінювання та правила зіставлення результатів реконструкції з еталонною моделлю;
- окреслити межі дослідження, відокрепивши формування критеріїв, вимог і метрик від подальшої реалізації та експериментальної перевірки RefMSBench.

Виклад основного матеріалу

Для оцінювання методів реконструкції бенчмарк має забезпечувати розгалужений граф залежностей, збирання й представлення телеметрії, а також еталонну модель для порівняння. Тому критерії сформульовані як вимоги до експериментального середовища, а не як універсальні показники якості мікросервісного бенчмарку. Оцінюваність відображає наявність еталонної моделі, контрактів і метрик, тому системи, не призначені для такого оцінювання, можуть мати нижчі рівні

за цим критерієм. Визначення критеріїв спирається на проблеми, описані в літературі з реконструкції MSA: огляди інструментів реконструкції [5], порівняння статичних інструментів з еталонними моделями, створеними вручну [15], дослідження динамічного аналізу та практик спостережуваності [11, 12]. Підхід до формалізації архітектурної моделі враховано за [18], а самі критерії узгоджено з процес-орієнтованою таксономією відновлення архітектури, що розрізняє процеси, цілі, вхідні дані, техніки та результати реконструкції [6]. Критерії поділено на два рівні: основні C1–C5 визначають мінімальні умови для порівняння методів реконструкції, а розширені E1–E6 описують властивості, потрібні для серії експериментів. Для основних критеріїв наведено підкритерії (табл. 1), щоб зробити оцінювання менш залежним від узагальненої експертної оцінки.

Архітектурна складність (C1) визначає, наскільки система має складний граф залежностей: поряд із кількістю сервісів враховують типи взаємодій і щільність зв'язків. Телеметрія (C2) визначає придатність системи до динамічного аналізу, оскільки без логів, метрик і трасувань поведінку під час виконання можна відтворити лише частково. Відтворюваність (C3) охоплює можливість повторити експеримент із тими самими версіями компонентів, ресурсними обмеженнями та конфігурацією середовища. Реалістичність (C4) показує, наскільки обраний домен і сценарії відповідають реальним мікросервісним системам. Оцінюваність (C5) визначає, чи можна зіставити результат реконструкції з еталонною моделлю та обчислити похибку методу.

Розширені критерії визначають придатність бенчмарку для серії експериментів і уточнюють окремі основні критерії. Зокрема, E5 деталізує наявність еталонної моделі в межах оцінюваності (C5), а E6 уточнює типи взаємодій у межах архітектурної складності (C1). Щоб зменшити залежність оцінювання від експертної думки, для кожного розширеного критерію визначено спостережувані ознаки. Для E1 це актуальність залежностей, контейнерних образів і документації; для E2 – можливість змінювати масштаб і топологію через конфігурацію; для E3 – використання різних мов і фреймворків. Для E4 враховується наявність кількох версій топології або задокументованих архітектурних змін, для E5 – придатність еталонної моделі до автоматичної перевірки, а для E6 – розмежування синхронних викликів, асинхронних команд-відповідей і подій. Володіння даними фіксується окремо як атрибут еталонної моделі, оскільки воно не утворює міжсервісного ребра.

Системи оцінюються за п'ятирівневою порядковою шкалою: 1 – низький, 2 – середньо-низький, 3 – середній, 4 – високий, 5 – дуже високий. Для основних критеріїв C1–C5, кожен з яких має три підкритерії, рівень прив'язано до кількості виконаних підкритеріїв. Низький рівень означає відсутність відповідності підкритеріям; середньо-низький, часткову відповідність принаймні одному підкритерію; середній, повну відповідність одному підкритерію; високий, повну відповідність двом підкритеріям; дуже високий, повну відповідність усім трьом. Розширені критерії E1–E6

є цілісними, тому та сама шкала застосовується до ступеня прояву відповідної властивості: від її відсутності або непідтвердженості в доступних джерелах до повної, документованої й відтвореної підтримки. Оцінки спираються на документацію, вихідний код або опубліковані дослідження відповідних систем, на які є посилання в подальшому аналізі. Для RefMSBench така шкала визначає цільові рівні властивостей, які має забезпечити майбутня реалізація.

Таблиця 1

Підкритерії основних критеріїв C1–C5

Критерій	Підкритерій	Опис / показник
Архітектурна складність C1	Кількість сервісів	Число незалежно розгорнутих мікросервісів у системі
	Типи взаємодій	Наявність різних способів обміну запитами, повідомленнями та подіями між сервісами
	Щільність графу	Середня кількість залежностей на один сервіс
Телеметрія C2	Логи	Наявність та повнота структурованих записів подій, помилок і ключових операцій сервісів
	Метрики	Числові показники виконання операцій в системі. Включає затримку, пропускну здатність і помилки
	Трасування	Наявність та повнота розподілених трасувань міжсервісних викликів
Відтворюваність C3	Час розгортання	Тривалість і складність розгортання системи
	Ресурси	Наявність задокументованих вимог до обчислювальних ресурсів і середовища розгортання
	Наявність фіксованих версій	Наявність фіксованих версій контейнерних образів та їх залежностей
Реалістичність C4	Предметна область	Наскільки обраний домен підходить до задач, які вирішуються за допомогою мікросервісної архітектури
	Сценарії	Кількість основних процесів домену, представлені сценаріями взаємодії сервісів
	Операційні шаблони	Наявність типових механізмів, що відображають реальну експлуатацію мікросервісної системи
Оцінюваність C5	Еталонна модель	Наявність стандартизованого опису сервісів, залежностей і типів взаємодій
	Контракти	Кількість та повнота формалізованих описів інтерфейсів і повідомлень між сервісами
	Метрики оцінювання	Наявність визначених метрик для порівняння результатів реконструкції з еталоном

Для порівняння обрано чотири відкриті системи, які використовують у дослідженнях мікросервісів: Sock Shop, TeaStore, DeathStarBench і Train Ticket [7, 8]. Їх обрано за відкритим вихідним кодом, доступною документацією та наявністю публікацій, що дає змогу оцінювати системи за визначеними критеріями. Системи відрізняються масштабом, доменом, типами взаємодій і

складністю розгортання. Кількісні характеристики взяті з першоджерел: для TeaStore, з публікації авторів [19]; для DeathStarBench, зі сценарію socialNetwork [9]; для Train Ticket, з офіційного репозиторію та супровідної статті [7]. Результати оцінювання за критеріями C1–C5 та E1–E6 подано в таблиці 2.

Sock Shop використовує типову предметну область інтернет-магазину, має підтримку логування та окремі архітектурні описи, однак його

підтримка обмежена: репозиторій Weaveworks позначено як застарілий і перенесено до архіву [8]. TeaStore є компактним і добре задокументованим бенчмарком із п'ятьма доменними сервісами та реєстром [19], але має простий граф залежностей, переважно синхронні HTTP-взаємодії та низьку технологічну гетерогенність. DeathStarBench, зокрема сценарій socialNetwork, містить 36 унікальних сервісів, кілька мов реалізації та Thrift IDL [9], що робить його придатним для перевірки методів реконструкції на складнішій топології. Водночас вибіркове збирання трасувань може не відображати повного графа міжсервісних викликів. Train Ticket має найбільший масштаб серед розглянутих систем: 41 мікросервіс, 4 мови реалізації та асинхронна взаємодія через шину повідомлень. Його обмеженнями є відсутність розподіленого трасування в базовій конфігурації, високі ресурсні вимоги до розгортання та відсутність формальної еталонної моделі.

Таблиця 2

Оцінка наявних систем за критеріями C1–C5 та E1–E6

Критерій	Sock Shop	TeaStore	DeathStar Bench	Train Ticket
C1	3	2	4	5
C2	4	5	3	2
C3	2	4	3	3
C4	4	3	3	4
C5	2	4	3	2
E1	1	1	3	3
E2	1	3	1	1
E3	3	1	4	4
E4	1	4	1	1
E5	1	3	1	1
E6	3	3	3	3

Розширені критерії уточнюють цю різницю. DeathStarBench і Train Ticket мають вищу підтримуваність і гетерогенність, а TeaStore краще підтримує налаштування та дослідження еволюції. Однак, жодна з розглянутих систем не надає повної еталонної моделі для задач реконструкції, придатної для автоматичної перевірки, і не розмежовує формально синхронні виклики, асинхронні команди-відповіді, події та володіння даними.

До ідеї формального еталона для реконструкції архітектури найближчі є окремо опубліковані набори перевірених еталонних моделей. Зокрема, для порівняння статичних інструментів реконструкції використано набір із 17 програмних рішень з відкритим кодом, для кожного з яких подано перевірену людиною модель архітектури у формі сервісів і зв'язків між ними [15]. Такий набір є цінним, однак не розв'язує задачу оцінювання динамічних і гібридних методів реконструкції: він не містить телеметрії, метрик і трасувань,

а його ребра подано як узагальнені зв'язки без розмежування синхронних викликів, асинхронних команд-відповідей, подій і володіння даними. Тому за таким еталоном неможливо оцінити правильність типізації взаємодій. Додатковими обмеженнями є переважно синхронна REST-взаємодія, один технологічний стек, фіксована топологія та відсутність конфігурованого масштабування, сценаріїв навантаження й контрольованої еволюції.

Розширення наявної системи також не є простим. Для TeaStore це вимагало б суттєвої зміни архітектури, зокрема додавання шини подій, технологічної гетерогенності та формальної еталонної моделі; Adaptable TeaStore розширює можливості налаштування, але не змінює склад взаємодій і не вводить еталонну модель для оцінювання реконструкції [17]. Для Train Ticket повна інструментація й супровід кодової бази з 41 сервісу, реалізованого чотирма мовами, потребують значних зусиль. Використання реальної системи з вручну побудованим еталоном підвищує реалістичність, але обмежує контроль над топологією та ускладнює відтворення експерименту поза межами конкретного дослідження [15].

Домен і склад артефактів RefMSBench

RefMSBench (Reference Microservice Benchmark) специфікується як компактний бенчмарк для оцінювання реконструкції MSA за заздалегідь заданою еталонною моделлю сервісів, залежностей і типів взаємодій. Як предметну область обрано міські перевезення: запити на поїздку, призначення водія, оплату, сповіщення, рейтинг і відстеження місця водія. Така предметна область поєднує синхронні виклики через API Gateway, асинхронну взаємодію, події, окреме володіння даними та кешування, але залишається достатньо компактною для ручної й автоматичної перевірки.

Вимоги до RefMSBench охоплюють доменні сервіси, API Gateway, телеметрію, еталонну модель, сценарії навантаження та конфігурації розгортання. Еталонна модель спирається на ідею формального опису архітектури у вигляді компонентів і конекторів [20] та конкретизує її для MSA через сервіси, залежності, контракти, типи взаємодій і володіння даними. Взаємодії подаються як типізовані ребра, а володіння даними фіксується окремо. Перед експериментами модель перевіряється на внутрішню узгодженість, зокрема на цикли синхронних викликів, подвійне володіння сховищем і некоректну класифікацію взаємодій. Конкретний склад сервісів, топологію взаємодій і процедуру перевірки має зафіксувати подальша реалізація.

Система вимог і метрик оцінювання

Вимоги до RefMSBench пов'язують критерії C1–C5 та E1–E6 з інженерними властивостями платформи. Функціональні вимоги (табл. 3) задають мінімальну поведінку домену міських перевезень, а нефункціональні (табл. 4) визначають

складність, телеметрію, відтворюваність, еталонну модель, конфігурованість і підтримку еволюції. Міжсервісні взаємодії з функціональних вимог фіксуються в еталонній моделі як типізовані залежності, а склад сервісів і мінімальна кількість зв'язків визначаються за NFR-01.

Таблиця 3

Функціональні вимоги до RefMSBench

Вимога	Опис	Критерії
FR-01	Система має забезпечувати реєстрацію та автентифікацію пасажирів і водіїв	C4
FR-02	Система має забезпечувати створення запиту на поїздку із зазначенням початкової та кінцевої точки маршруту	C4
FR-03	Система має забезпечувати пошук доступного водія та призначення його до поїздки	C1, C4
FR-04	Система має підтримувати зміну статусу поїздки за визначеним життєвим циклом: створена, прийнята, в процесі, завершена або скасована	C1, C4
FR-05	Система має забезпечувати синхронну авторизацію оплати після завершення поїздки з подальшим асинхронним підтвердженням	C1, E6
FR-06	Система має надсилати пасажиру й водію сповіщення про ключові події поїздки	C1, E6
FR-07	Система має забезпечувати оцінювання поїздки після її завершення	C4
FR-08	Система має оновлювати поточне місце водія та використовувати ці дані під час призначення поїздки	C1, C4
FR-09	Система має містити сценарії навантаження та еталонні описи очікуваних взаємодій для зіставлення з результатами реконструкції	C2, C4, C5

Кількісні пороги NFR-01 запроваджено, щоб оцінювання не ґрунтувалося на поодиноких зв'язках. Щонайменше 15 залежностей зменшують вплив поодиноких помилок на значення Edge-F1, а кілька ребер кожного типу дають змогу оцінювати Туре-ассигасу для різних класів взаємодій. Наведені значення слід розглядати як орієнтовні нижні межі, а не як фіксовані вимоги для будь-яких експериментів. Еталонна модель зберігає контракти у стандартизованих форматах, зокрема

OpenAPI для REST-інтерфейсів, Protocol Buffers для gRPC-викликів та AsyncAPI для асинхронних команд, відповідей і подій. Для кожної версії топології зберігаються відповідні версії еталона, сценарії навантаження та конфігурації розгортання. У базових експериментах трасування збираються для всіх запитів, а вибіркоче збирання трасувань розглядається як окремий режим перевірки стійкості методів до неповних спостережень.

Таблиця 4

Нефункціональні вимоги до RefMSBench

Вимога	Опис	Критерії
NFR-01	Бенчмарк має містити не менше шести доменних сервісів, не менше п'ятнадцяти залежностей і не менше трьох типів взаємодій.	C1
NFR-02	Бенчмарк має підтримувати різні типи архітектурних взаємодій: синхронні виклики, асинхронні команди-відповіді та події, а також окремо фіксувати володіння даними.	C1, E6
NFR-03	Кожен доменний сервіс має надавати телеметрію, достатню для збирання логів, метрик і розподілених трасувань.	C2
NFR-04	Бенчмарк має підтримувати збирання трасувань для всіх запитів під час контрольованих сценаріїв навантаження.	C2, E5
NFR-05	Бенчмарк має забезпечувати відтворюване локальне або серверне розгортання з фіксованими версіями компонентів.	C3
NFR-06	Бенчмарк має містити документовані ресурсні вимоги, перевірку працездатності сервісів і базову перевірку телеметрії після розгортання.	C3
NFR-07	Предметна область бенчмарку має бути достатньо реалістичною та містити кілька типових сценаріїв виконання	C4
NFR-08	Архітектура бенчмарку має містити API Gateway, окремі сховища для сервісів і брокер повідомлень для асинхронних команд та подій.	C4, E6
NFR-09	Бенчмарк має містити еталонну модель сервісів, залежностей і типів взаємодій, а також атрибути володіння даними.	C5, E5

Закінчення табл. 4

Вимога	Опис		Критерії
NFR-10	Бенчмарк має містити задокументовані контракти для синхронних API, асинхронних команд-відповідей і подій.		C5
NFR-11	Еталонна модель має бути придатною до автоматичної перевірки на внутрішню узгодженість.		E5
NFR-12	Бенчмарк має спиратися на підтримувані версії основних технологічних залежностей, щоб забезпечити його подальшу відтворюваність і супровід.		E1
NFR-13	Топологія бенчмарку має підтримувати масштабування без зміни основної архітектури.		E2
NFR-14	Доменні сервіси мають використовувати щонайменше дві мови програмування або два технологічні стеки.		E3
NFR-15	Бенчмарк має забезпечувати кілька версій топології та підтримувати сценарії контрольованої архітектурної еволюції.		E4

Результат реконструкції подається як граф сервісів і взаємодій та порівнюється з еталонною моделлю за трьома метриками: Service-F1 оцінює відновлення множини сервісів, Edge-F1 оцінює залежності між ними, а Type-accracy визначає правильність типізації знайдених ребер. Правила зіставлення фіксуються наперед: вузлами вважаються незалежно розгорнуті сервіси, ребрами є спрямовані безпосередні взаємодії між ними, а транзитивні залежності не додаються. Для подій ребро спрямоване від видавця до споживача, для асинхронних команд-відповідей від ініціатора до виконавця, а брокер повідомлень розглядається як транспорт, а не як вузол графа. Звернення сервісу до власного сховища фіксується як атрибут володіння даними й не утворює міжсервісного ребра. Вузли зіставляються за канонічними іменами розгортання; ребро вважається правильно відновленим за збігом ініціатора, адресата та напрямку, а його тип оцінюється окремо за Type-accracy. Імена вузлів нормалізуються за наперед визначеним правилом. Об'єднання кількох сервісів в один вузол або поділ одного сервісу на кілька вузлів розглядається як помилка відновлення множини сервісів і враховується під час обчислення Service-F1.

Таке розділення метрик важливе для MSA, оскільки зайвий або пропущений сервіс, помилкова залежність і неправильна типізація взаємодії є різними класами помилок. Семантика ребра може бути не менш суттєвою, ніж сам факт залежності, адже синхронний виклик, асинхронна команда-відповідь і подія мають різні архітектурні наслідки. Метрики F1 обрано тому, що сервіси й ребра природно подаються множинами, для яких точність і повнота обчислюються безпосередньо; такий підхід узгоджується з практикою емпіричного порівняння методів реконструкції архітектури [21, 15]. Узагальнені метрики відмінності

між графами менш придатні для цієї задачі, оскільки поєднують різні типи помилок в одному числовому значенні та залежать від попередньо заданих ваг операцій [22]. Натомість Service-F1, Edge-F1 і Type-accracy окремо оцінюють відновлення сервісів, залежностей і типів взаємодій. Type-accracy обчислюється лише для правильно відновлених ребер, тому її слід інтерпретувати разом з Edge-F1. Щоб зменшити вплив найчастішого типу взаємодії, Type-accracy доцільно подавати як незважене середнє за класами взаємодій (macro-averaged Type-accracy), додатково наводячи матрицю помилок.

Реалізацію RefMSBench доцільно починати з мінімальної топології, що охоплює еталонну модель, контракти, сценарії навантаження та базову телеметрію. Надалі платформу можна розширити до повного складу сервісів і використати для експериментального порівняння інструментів реконструкції за Service-F1, Edge-F1 і Type-accracy [5, 10, 15].

Обмеження запропонованого підходу

Перше обмеження дослідження пов'язане з його проектним характером. У статті визначено критерії, вимоги та метрики RefMSBench, однак конкретну архітектуру бенчмарку, його реалізацію та експериментальне застосування залишено на подальший етап. Емпіричне порівняння інструментів реконструкції на цій платформі також потребує окремого дослідження.

Друге обмеження стосується оцінювання наявних систем: рівні в таблиці 2 визначено одним дослідником. Прив'язка оцінок до підкритеріїв і джерел зменшує суб'єктивність, однак не усуває її повністю; тому подальша робота має передбачати незалежне повторне оцінювання. Окремої перевірки також потребує чутливість методів реконструкції до масштабу системи, зокрема через

контрольоване ускладнення топології, параметризації та сценарії архітектурної еволюції.

Висновки

У статті сформовано основу еталонного мікро-сервісного бенчмарку RefMSBench для оцінювання методів реконструкції програмної архітектури. Основним результатом є узгоджена схема оцінювання, що поєднує дворівневу систему критеріїв, функціональні й нефункціональні вимоги, метрики Service-F1, Edge-F1 і Type-accuracy, а також правила зіставлення графів.

Оцінювання Sock Shop, TeaStore, DeathStarBench і Train Ticket показало, що наявні системи охоплюють лише окремі складові задачі: телеметрію, формальні описи, контрольовану топологію або стабільні умови розгортання. Водночас жодна з них не поєднує ці властивості в межах однієї підтримуваної платформи з еталонною моделлю, придатною для автоматичної перевірки, і формальним розмежуванням типів взаємодій. Це дає підстави вважати, що адаптація наявної системи до повного набору артефактів може бути за складністю співмірною з проектуванням компактного бенчмарку з наперед визначеними цілями.

Подальша робота має охопити проектування архітектури й реалізацію RefMSBench, публікацію еталонної моделі, незалежне повторне оцінювання наявних систем та експериментальне порівняння інструментів реконструкції [5, 15]. Окремим напрямом є перевірка RefMSBench у сценаріях архітектурної еволюції та його використання як контрольованого еталона для автоматизованого проектування MSA із застосуванням штучного інтелекту [18].

ЛІТЕРАТУРА

- [1] Dragoni N., Giallorenzo S., Lafuente A. L., Mazzara M., Montesi F., Mustafin R., Safina L. Microservices: Yesterday, today, and tomorrow // *Present and Ulterior Software Engineering*. 2017. C. 195–216. https://doi.org/10.1007/978-3-319-67425-4_12
- [2] Jamshidi P., Pahl C., Mendonça N. C., Lewis J., Tilkov S. Microservices: *The Journey So Far and Challenges Ahead*. *IEEE Software*. 2018. Vol. 35, No. 3. P. 24–35. <https://doi.org/10.1109/MS.2018.2141039>
- [3] Wang Y., Kadiyala H., Rubin J. Promises and challenges of microservices: an exploratory study // *Empirical Software Engineering*. 2021. Vol. 26, № 4. Article 63. <https://doi.org/10.1007/s10664-020-09910-y>
- [4] Söylemez M., Tekinerdogan B., Kolukısa Tarhan A. Challenges and solution directions of microservice architectures: A systematic literature review // *Applied Sciences*. 2022. Vol. 12, № 11. Article No. 5507. <https://doi.org/10.3390/app12115507>
- [5] Bakhtin A., Li X., Soldani J., Brogi A., Cerny T., Taibi D. Tools Reconstructing Microservice Architecture: A Systematic Mapping Study. *Software Architecture. ECSA 2023 Tracks, Workshops, and Doctoral Symposium*. Cham: Springer, 2024. Lecture Notes in Computer Science. Vol. 14590. P. 3–22. https://doi.org/10.1007/978-3-031-66326-0_1
- [6] Ducasse S., Pollet D. Software Architecture Reconstruction: A Process-Oriented Taxonomy. *IEEE Transactions on Software Engineering*. 2009. Vol. 35, No. 4. P. 573–591. <https://doi.org/10.1109/TSE.2009.19>
- [7] Zhou X., Peng X., Xie T. Benchmarking Microservice Systems for Software Engineering Research. *Proceedings of the 40th ACM/IEEE International Conference on Software Engineering (ICSE 2018)*. New York: Association for Computing Machinery, 2018. P. 323–324. <https://doi.org/10.1145/3183440.3194991>
- [8] Smith S., Robinson E., Frederiksen T., Stevens T., Cerny T., Bures M., Taibi D. Benchmarks for End-to-End Microservices Testing. *Proceedings of the 17th IEEE International Conference on Service-Oriented System Engineering (SOSE 2023)*. Piscataway: IEEE, 2023. P. 60–66. <https://doi.org/10.1109/SOSE58276.2023.00013>
- [9] Gan Y. et al. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems. *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2019)*. New York: ACM, 2019. P. 3–18. <https://doi.org/10.1145/3297858.3304013>
- [10] Granchelli G. et al. MicroART: A Software Architecture Recovery Tool for Maintaining Microservice-Based Systems. *Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops (ICSAW 2017)*. Gothenburg: IEEE, 2017. P. 298–302. <https://doi.org/10.1109/ICSAW.2017.9>
- [11] Al Maruf A., Bakhtin A., Cerny T., Taibi D. Using Microservice Telemetry Data for System Dynamic Analysis. *Proceedings of the 2022 IEEE International Conference on Service-Oriented System Engineering (SOSE 2022)*. Piscataway: IEEE, 2022. P. 29–38. <https://doi.org/10.1109/SOSE55356.2022.00010>
- [12] Waseem M., Liang P., Shahin M., Di Salle A., Márquez G. Design, Monitoring, and Testing of Microservices Systems: The Practitioners' Perspective. *Journal of Systems and Software*. 2021. Vol. 182. Article 111061. <https://doi.org/10.1016/j.jss.2021.111061>

- [13] Hutcheson R., Blanchard A., Lambaria N., Hale J., Kozak D., Abdelfattah A. S., Cerny T. Software Architecture Reconstruction for Microservice Systems Using Static Analysis via GraalVM Native Image. Proceedings of the 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2024). Piscataway: IEEE, 2024. P. 12–22. <https://doi.org/10.1109/SANER60148.2024.00008>
- [14] Bushong V., Das D., Cerny T. Reconstructing the Holistic Architecture of Microservice Systems Using Static Analysis. Proceedings of the 12th International Conference on Cloud Computing and Services Science (CLOSER 2022). Science and Technology Publications, 2022. P. 149–157. <https://doi.org/10.5220/0011032100003200>
- [15] Schneider S., Bakhtin A., Li X., Soldani J., Brogi A., Cerny T., Scandariato R., Taibi D. Comparison of Static Analysis Architecture Recovery Tools for Microservice Applications. *Empirical Software Engineering*. 2025. Vol. 30, No. 5. Article 128. <https://doi.org/10.1007/s10664-025-10686-2>
- [16] Sedghpour M. R. S. et al. HydraGen: A Microservice Benchmark Generator. Proceedings of the 2023 IEEE 16th International Conference on Cloud Computing (CLOUD 2023). Piscataway: IEEE, 2023. P. 189–200. <https://doi.org/10.1109/CLOUD60044.2023.00030>
- [17] Bliudze S. et al. Adaptable TeaStore. Proceedings of the First Workshop on Adaptable Cloud Architectures (WACA 2025). Open Publishing Association, 2025. P. 1–14. <https://doi.org/10.4204/EPTCS.438.1>
- [18] Моргун А., Гізун А. Концепт програмного забезпечення для проєктування мікросервісної архітектури з використанням штучного інтелекту. *Вісник Хмельницького національного університету. Технічні науки*. 2025. № 5 (357). С. 307–313. <https://doi.org/10.31891/2307-5732-2025-357-39>
- [19] Von Kistowski J. et al. TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research. Proceedings of the 26th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2018). IEEE, 2018. P. 223–236. <https://doi.org/10.1109/MASCOTS.2018.00030>
- [20] Garlan D., Monroe R., Wile D. ACME: An Architecture Description Interchange Language. Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON 1997). Toronto: IBM Press, 1997. P. 169–183.
- [21] Lutellier T. et al. Comparing Software Architecture Recovery Techniques Using Accurate Dependencies. Proceedings of the 37th IEEE/ACM International Conference on Software Engineering (ICSE 2015). Piscataway: IEEE, 2015. P. 69–78. <https://doi.org/10.1109/ICSE.2015.136>
- [22] Garcia J., Ivkovic I., Medvidovic N. A Comparative Analysis of Software Architecture Recovery Techniques. Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE 2013). Piscataway: IEEE, 2013. P. 486–496. <https://doi.org/10.1109/ASE.2013.6693106>

Гізун А. І., Моргун А. В.

КРИТЕРІЙ, ВИМОГИ ТА МЕТРИКИ ЕТАЛОННОГО БЕНЧМАРКУ RefMSBench ДЛЯ РЕКОНСТРУКЦІЇ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

У статті розглянуто проблему вибору мікросервісного бенчмарку для оцінювання методів реконструкції програмної архітектури. Для такого оцінювання недостатньо системи, яку можна розгорнути й запустити; її структуру, взаємодії, телеметрію й еталонний опис має бути можливо зіставити з результатами роботи інструментів реконструкції. Запропоновано систему основних і розширених критеріїв придатності MSA-бенчмарку до задач реконструкції. За цими критеріями оцінено *Sock Shop*, *TeaStore*, *DeathStarBench* і *Train Ticket*. Основні критерії охоплюють архітектурну складність, телеметрію, відтворюваність, реалістичність та оцінюваність. Розширені критерії деталізують підтримуваність, конфігурованість, гетерогенність, підтримку еволюції, придатність еталонної моделі до автоматичної перевірки та типізацію взаємодій. Показано, що розглянуті системи мають окремі властивості, корисні для реконструкції архітектури, однак не поєднують їх у межах однієї підтримуваної платформи. Одні системи краще відображають реальні сценарії використання або підтримують збирання телеметрії, інші мають складнішу топологію чи кращу відтворюваність. Водночас жодна з розглянутих систем не забезпечує повного поєднання властивостей, потрібних для оцінювання результатів реконструкції архітектури. На основі аналізу сформульовано дев'ять функціональних і п'ятнадцять нефункціональних вимог до еталонного бенчмарку *RefMSBench*. Вимоги охоплюють доменні сервіси, API Gateway, телеметрію, еталонну модель, сценарії навантаження та конфігурації розгортання. Окремо визначено склад еталонної моделі, що фіксує сервіси, залежності, контракти, типи взаємодій і володіння даними. Для оцінювання результатів реконструкції запропоновано *Service-F1*, *Edge-F1* і *Type-accuracy*, а також правила зіставлення відновленого графа з еталонною моделлю. Ці правила визначають, як порівнювати сервіси, залежності, напрями взаємодій і типи комунікації. Реалізацію *RefMSBench* та експериментальне порівняння інструментів реконструкції залишено для наступного етапу дослідження.

Ключові слова: мікросервісна архітектура, бенчмаркінг, реконструкція програмної архітектури, еталонна модель.

Gizun A., Morhun A.

CRITERIA, REQUIREMENTS, AND METRICS FOR THE RefMSBench BENCHMARK FOR MICROSERVICE ARCHITECTURE RECONSTRUCTION

The paper addresses the problem of selecting a microservice benchmark for evaluating software architecture reconstruction methods. Such evaluation requires more than a deployable and executable system; its structure, interactions, telemetry, and reference model must be comparable with the results produced by reconstruction tools. A set of core and extended criteria is proposed to assess the suitability of MSA benchmarks for reconstruction tasks. The criteria are applied to Sock Shop, TeaStore, DeathStarBench, and Train Ticket. The core criteria cover architectural complexity, telemetry, reproducibility, realism, and evaluability. The extended criteria refine maintainability, configurability, heterogeneity, evolution support, suitability of the reference model for automated verification, and explicit typing of interactions. The analysis shows that the evaluated systems have individual properties useful for architecture reconstruction, but none combines these properties within a single supported evaluation platform. Some benchmarks better represent realistic usage scenarios or provide telemetry, while others offer larger topologies or stronger reproducibility. However, none provides the full set of properties required to evaluate architecture reconstruction results. Based on the analysis, nine functional and fifteen non-functional requirements are formulated for RefMSBench. These requirements cover domain services, an API Gateway, telemetry, the reference model, workload scenarios, and deployment configurations. The structure of the reference model is also defined: it captures services, dependencies, contracts, interaction types, and data ownership. Reconstruction results are evaluated using Service-F1, Edge-F1, and Type-accuracy, together with rules for matching a reconstructed graph against the reference model. The rules specify how services, dependencies, interaction directions, and interaction types should be compared. Future work will focus on the implementation of RefMSBench and the empirical comparison of reconstruction tools.

Keywords: microservice architecture, benchmarking, software architecture reconstruction, reference model.

Дата першого надходження: 18.06.2026 р.

Дата прийняття до друку: 13.08.2026 р.

Дата публікації: 10.09.2026 р.