УДК 004.02
DOI https://doi.org/10.32782/2663-5941/2024.2/06

Бейрак Д.Я.
Державний університет «Житомирська політехніка»

Вакалюк Т.А.
Державний університет «Житомирська політехніка»

ПІДХОДИ ДО ДЕКОМПОЗИЦІЇ ПРЕДМЕТНОЇ ОБЛАСТІ У ПОБУДОВІ МІКРОСЕРВІСНИХ СИСТЕМ В НАУКОВІЙ ЛІТЕРАТУРІ

На сьогоднішній день продовжує актуалізовуватись питання побудови мікросервісних систем, яке є комплексним та багатоаспектним. Мікросервісна архітектура дозволяє будувати системи з низькою зв'язністю, що, завдяки можливості горизонтального масштабування та за рахунок того, що менші окремі сервіси простіше розуміти та змінювати, ніж складні великі моноліти, покращує такі архітектурні параметри як масштабованість, можливість підтримки, ефективність використання ресурсів тощо. Ці та інші причини впливають на вибір спеціалістів в індустрії, які все частіше роблять його на користь саме цього типу архітектури при розробці нових систем, а також рухаються у напрямку міграції наявних успадкованих монолітів на мікросервіси. Важливими питаннями, що постають перед інженерами та архітекторами, які обирають мікросервісну архітектуру, є декомпозиція предметної області при розробці нових мікросервісів та декомпозиція вже наявної монолітної системи при переведенні її на мікросервісну архітектуру. Так як саме якість декомпозиції має первинний суттєвий вплив на весь подальший розвиток системи, виникає потреба у виборі відповідних методів та інструментів для виконання даного процесу. В даній статті розглядаються методи та інструменти, призначені для розв'язання задач декомпозиції предметної області при проектуванні мікросервісних систем «з нуля» та успадкованих монолітних систем при переведенні їх на мікросервісну архітектуру. Для кожного з цих аспектів міститься як висвітлення усталених в індустрії підходів та патернів, так і огляд методів та інструментів, що наразі здебільшого розглядаються в наукових публікаціях. Окрім поділу методів на ті, що застосовуються при побудові систем «з нуля», та ті, що використовуються при роботі з успадкованими монолітами, розглянуті у статті підходи та інструментарій також можна умовно розділити на мануальні (ручні), напівавтоматизовані та автоматизовані. Крім того, в основу роботи розглянутих методів та інструментів покладено різні варіанти декомпозиції, застосування метрик, використання історій користувачів (user stories), сценаріїв використання (use cases) та діаграм потоків, застосування кластеризації та аналіз графів, аналіз даних та пов'язаного з ними функціоналу системи тощо.

Ключові слова: *мікросервісна архітектура, декомпозиція, моделювання, міграція, патерни.*

Постановка проблеми. Сьогодні все більше нових проектів створюється на основі мікросервісної архітектури, а також все більше компаній намагаються перевести на дану архітектуру успадковані монолітні системи задля підвищення конкурентоспроможності своїх продуктів на ринку. Від рішень, прийнятих на етапі декомпозиції, у значній мірі залежить подальший розвиток мікросервісної системи, її спроможність відповідати узгодженим із стейкхолдерами показникам та характеристикам, вартість подальшої підтримки та реалізації нового функціоналу тощо. Дані фактори роблять актуальними дослідження наявних рішень декомпозиції предметної області, як при написанні мікросервісних систем «з нуля», так і декомпозиції успадкованих монолітних систем у мікросервісні.

Аналіз останніх досліджень і публікацій. Проблеми декомпозиції у контексті мікросервісної архітектури розглядались в науковій літературі та публікаціях багатьма авторами та вченими. Зокрема, такі підходи розглядали Кріс Річардсон (Chris Richardson) [1], Сем Ньюман (Sam Newman) [2], Марк Річардс (Mark Richards) та Ніл Форд (Neal Ford) [3]. Інші, менш відомі поза академічним колом підходи, розглядали Чайтанья К. Рудрабхатла (Chaitanya K. Rudrabhatla) [4], Фреді Вера-Рівера (Fredy H. Vera-Rivera) [5], Мохаммед Дауд (Mohamed Daoud) [6], Седігена Хошневіса (Sedigheh Khoshnevis) [7], Шаньшань Лі (Shanshan Li) [8], Діпалі Баджадж (Deepali Bajaj) [9], Аніта Гоель (Anita Goel) [9], Суреш Гупта (Suresh Gupta) [10], Манабу Камімура (Manabu Kamimura) [11], Мігель Бріто (Miguel Brito) [12],

Луїс Карвальо (Luiz Carvalho) [13], Омар Аль-Дебагі (Omar Al-Debagy) [14], Лулай Чжу (Lulai Zhu) [15], Діпалі Баджадж (Deepali Bajaj) [9,10], Сімоне Стаффа (Simone Staffa) [16,17], Джованні Кватроккі (Giovanni Quattrocchi) [16,17] та інші.

Постановка завдання. Метою статті є детальний огляд підходів до декомпозиції предметної області та наявних монолітних систем у контексті побудови мікросервісної архітектури.

Виклад основного матеріалу. Здебільшого, інженери та науковці галузі розглядають питання декомпозиції предметної області з двох позицій: проектування мікросервісних систем «з нуля» (greenfield) та міграції монолітних систем у мікросервісні (brownfield). Варто зауважити, що ряд патернів, які розглядаються у даному контексті, універсальні, та можуть бути застосовані для будь-якого з цих сценаріїв.

В індустрії найбільш поширеними з позиції проектування «з нуля» є варіанти декомпозиції за бізнес-можливостями (business capabilities) та за піддоменами (subdomains), які, зокрема, розглядаються у книзі Кріса Річардсона (Chris Richardson) [1, с. 33]. Обидва варіанти передбачають створення компонентів (сервісів) базуючись на ідеях, або смислових складових, що лежать в основі бізнес-логіки, абстрагуючись при цьому від варіанту їх конкретної реалізації. Наприклад, наявність в системі частини, що відповідає за приймання платежів, виноситиметься в окремий сервіс, незалежно від того, які платіжні методи або пеймент-провайдери використовуються в рамках даної загальної складової. Декомпозиція за піддоменами [1, с. 54], що має корні в предметно-орієнтованому проектуванні (domain-driven design, DDD), відрізняється підходом до роботи з моделями (сутностями). Якщо декомпозиція за бізнес-можливостями [1, с. 51], передбачає створення єдиної моделі для кожної бізнес-сутності, що може бути незручно, враховуючи масштаб великих розподілених систем, то декомпозиція за піддоменами передбачає використання проекції необхідної частини моделі у кожному окремому контексті предметної області.

Марк Річардс (Mark Richards) та Ніл Форд (Neal Ford) розглядають питання гранулярності в контексті декомпозиції за піддоменами, зазначаючи, що не завжди можна легко визначити обсяги та межі того чи іншого сервісу [3, с. 110]. Чайтанья К. Рудрабхатла (Chaitanya K. Rudrabhatla) у своєму дослідженні зазначає, що декомпозиція за бізнес-можливостями може призводити до неефективного дизайну з позиції продуктивності системи, і пропонує гібридний підхід, який об'єднує переваги декомпозицій за бізнес-можливостями та за піддоменами [4]. Більш детально цей підхід описано в документації Amazon AWS [18] та в роботі колективу авторів (Мухаммад Васім (Muhammad Waseem) та ін.) [19], де він отримав назву декомпозиції за транзакціями. Суть даного патерну в тому, щоб транзакції системи проходили в межах одного сервісу, а не розподілялися між кількома, що дозволяє зменшити час відповіді в системі, покращує доступність (availability) і дозволяє уникнути проблем з узгодженістю даних (data consistency), хоча, з іншого боку, добитися цих характеристик важче, так як декомпозована таким чином система починає тяжіти до монолітної структури.

Метод, заснований на генетичному алгоритмі та використанні історій користувачів (user stories), що фокусується на гнучких методологіях розробки, запропонувала група вчених (Фреді Вера-Рівера (Fredy H. Vera-Rivera) та ін.) [5]. Даний метод передбачає завантаження історій користувачів, знаходження між ними залежностей, ідентифікацію сутностей, визначення агрегатів, визначення контекстів та розрахунок метрик. Даний підхід отримав розвиток у роботі [20], де автори додали аналіз семантичної схожості між сутностями сценаріїв користувачів та мікросервісів, метрику когнітивної складності (cognitive complexity metric) для оцінки запропонованих декомпозицій, математичну формалізацію мікросервісних додатків з позиції сценаріїв користувачів та метрик, та оновили генетичний алгоритм.

Варіанти декомпозиції, що базуються на ідентифікації мікросервісів з набору бізнес-процесів, запропоновані у роботах групи вчених (Мохаммед Дауд (Mohamed Daoud) та ін.) [6] та Седігена Хошневіса (Sedigheh Khoshnevis) [7]. У першій роботі запропоновано багатомодельний підхід, у якому розглядаються залежності між бізнес-процесами з певної однієї точки зору (керування, семантики, даних тощо). Результуючі дані кластеризуються для знаходження груп зв'язаних активностей, які можуть стати потенційними мікросервісами [6]. У другій роботі розглядається підхід, який не тільки використовує у якості основи бізнес-процеси, але й застосовує інформацію про хмарних тенантів (cloud tenants), що спеціалізує даний підхід саме в контексті проектування мікросервісів для корпоративних SaaS-додатків, що розгортаються у хмарних середовищах [7].

Наступним варіантом декомпозиції, який підходить для проектування мікросервісів «з нуля»

є спосіб, розглянутий групою вчених (Шаньшань Лі (Shanshan Li) та ін.), заснований на напівавтоматичному підході декомпозиції за діаграмами потоків [8]. Цей патерн являє собою чотирьохетапний процес з проведення аналізу бізнес-вимог та створення специфікацій, побудови детальних діаграм потоків даних для бізнес-логіки та даних, виявлення залежностей між бізнес-логікою та даними, та ідентифікації мікросервісів- кандидатів шляхом кластеризації [8].

Діпалі Баджадж (Deepali Bajaj), Аніта Гоел (Anita Goel) та С. Гупта (S. C. Gupta) у своїй роботі пропонують автоматизований підхід під назвою GreenMicro, в основі якого лежить робота зі сценаріями використання (use cases) та сутностями бази даних (database entities) [9]. Він складається з п'яти етапів: складання матриці відношення сценаріїв використання та сутностей бази даних (Use Case-Database Entities Relationship Matrix), що базується на припущенні, що різний функціонал, який звертається до одних і тих самих даних є пов'язаним між собою; складання матриці відношення сценаріїв використання (Use Case-Use Case Relationship Matrix), що показує ступінь залежності між усіма частинами функціоналу крізь призму понять «включення», «наслідування» та «узагальнення»; складання матриці подібності (Constructing a Resemblance Matrix) для матриці відношення сценаріїв використання та сутностей бази даних, що показує пов'язаність двох одиниць функціоналу з точки зору доступу до даних; створення комбінованої матриці подібності (Combined Similarity Matrix) для сценаріїв використання, що є агрегацією другої та третьої матриць; кластеризації даних четвертої матриці та отримання зв'язаного набору сценаріїв використання (одиниць функціональності), які можуть бути об'єднані в рамках єдиного мікросервісу [9].

З позиції міграції монолітних систем у мікросервісні, в індустрії здебільшого розглядаються патерни, що передбачають поступовий перехід від монолітів до мікросервісів. Одним з таких патернів є додаток-душитель (strangler application), відомий також як фікус-душитель (strangler fig). В його основі лежить процес побудови нової мікросервісної системи навколо старої монолітної, поступово переводячи все більшу кількість функцій в новий додаток, тим самим поступово зводячи старий нанівець. Це досягається організацією шару перехоплювання, який відповідним чином адресує користувацькі запити в ту, чи іншу систему [1, с. 430; 2, с. 79].

Наступний патерн, який використовується при переході від монолітної до мікросервісної архітектури, називається запобіжний шар (anti-corruption layer). Його завданням є перетворення моделей між громіздкою монолітною та контекстно-орієнтованою за принципами DDD, що запобігає проникненню зайвих членів застарілої моделі у нову, а також дозволяє моноліту звертатися до мікросервісу [1, с. 447].

Сем Ньюман (Sam Newman) також розглядає такі патерни, як паралельне виконання (parallel run), що дозволяє одночасно використовувати обидві системи з метою порівняння їх роботи, та вимикачі функціональності (feature toggles), що впроваджує механізм включення та відключення певного функціоналу системи, включення певної реалізації цього функціоналу тощо. Проте, ці патерни можна віднести і до більш загальних випадків застосування, а не лише до декомпозиції [2, с. 80].

Група вчених (Манабу Камімура (Manabu Kamimura) та ін.) розробила метод напівавтоматичного виокремлення сервісів з моноліту на основі методів графової кластеризації та візуалізації [11]. Автоматизація виокремлення сервісів дозволяє зекономити час на кропіткому аналізі вже існуючої системи та отримати результат у графічному вигляді для системи в цілому, не вдаючись до часткового розглядання, яке більше притаманне додатку-душителю. Отримані результати можуть бути використані у поєднанні з іншими патернами [11].

Один із розвитків кластерного підходу запропоновано вченими Мігелем Бріто (Miguel Brito), Хакоме Кунья (Jácome Cunha) та Жоао Сараїва (João Saraiva) [12]. В ньому використовується наступна процедура: виділення лексичної та структурної інформації з вихідного коду моноліту, яка використовується для пошуку тем (topics) та їх розподілу для кожного з компонентів, які потім кластеризуються для виокремлення кандидатів у мікросервіси [12].

Пошуковий підхід на основі багатьох критеріїв запропоновано групою вчених (Луїс Карвальо (Luiz Carvalho) та ін.) [13]. Автори стверджують, що основна маса попередніх підходів базується лише на критеріях зв'язності (coupling) та пов'язаності (cohesion), чого, на їх думку, недостатньо. Вони пропонують підхід, який враховує п'ять факторів: зв'язність, пов'язаність, модульність функціональності (feature modularization), мережеві накладні витрати (network overhead) та повторне використання (reuse). Даний підхід

є графовим, та приймає у якості вхідних параметрів вихідний код успадкованої системи, перелік функціональності та кількість мікросервісів, які необхідно ідентифікувати. Складається граф, кожна вершина якого представляє метод успадкованої системи, призначений її відповідній функції, а кожне ребро містить статичну та динамічну інформацію, що представляє зв'язок між методами – опис передачі даних та виявлені виклики [13].

Метод, що застосовує графову кластеризацію для пошуку залежностей між класами моноліту, запропонували в своїй роботі Омар Аль-Дебагі (Omar Al-Debagy) та Петер Мартінек (Péter Martinek) [14]. Основна ідея методу полягає в тому, щоб, вибравши залежні між собою класи системи, віднайти та згрупувати такі, що мають найсильнішу залежність. Ефективність даного підходу напряму залежить від обраного методу кластеризації, що детально розглядається в роботі вчених [14].

Напівавтоматичний підхід декомпозиції моноліту в безсерверні функції (serverless functions) запропоновано в роботі Лулай Чжу (Lulai Zhu), Даміан Ендрю Тамбуррі (Damian Andrew Tamburri) та Джуліано Казале (Giuliano Casale) [15]. Автоматична частина даного етапу передбачає поступове виокремлення функцій: спочатку відбувається пошук корельованого функціоналу системи на основі API, з виокремленням крупних мікросервісних частин, які потім розбиваються на дрібні функції. Необхідність у ручному втручанні виникає тоді, коли успадкована система написана у форматі архітектури, що застосовує шари. Так як функції є повністю ізольованими одна від іншої, будь-який зв'язок функціональності через шари системи має бути усунутий для фінальної міграції. Даний підхід також може бути використаний як доповнення при міграції монолітної системи у мікросервісну за необхідності виокремлення деякого функціоналу системи в окремі безсерверні функції [15].

Автори підходу GreenMicro Діпалі Баджадж (Deepali Bajaj), Аніта Гоель (Anita Goel) та Суреш Гупта (Suresh Gupta) також запропонували комплексний підхід вилучення мікросервісів, який працює не лише з бізнес-логікою, але й з об'єктами бази даних [10]. Він містить п'ять кроків: статичний аналіз кодової бази для виявлення залежностей між класами та методами; евристичне фільтрування переліки викликів класів та методів, отриманих на попередньому кроці, з метою залишення лише релевантних до основної логіки класів; ідентифікацію сервісів шляхом формування графу залежностей класів та методів; складання матриці контролю доступу мікросервісів з метою визначення баз даних, якими буде володіти той чи інший сервіс; визначення належності сутностей бази даних тому чи іншому мікросервісу (для чого у даній роботі авторами запропоновано вісім критеріїв) [10].

Аналіз джерел показав, що окрім підходів, які працюють навколо проектної документації (у випадку проектування «з нуля»), або аналізу вихідного коду та пов'язаних з ним артефактів (міграція), для останнього випадку існують платформонезалежні методи та інструменти декомпозації, що працюють з абстрактною моделлю успадкованої монолітної системи, яка описується набором YAML-файлів. З таких методів декомпозиції варто розглянути Pangaea [16], та Cromlech [17]. У випадку Pangaea, на вхід подається модель системи, в яких описано сутності даних та операції, разом з їх характеристиками та взаємозв'язками, а також вхідні параметри декомпозиції, які обумовлюють роботу інструменту. Сам інструмент складається з парсера, вирішувача та візуалізатора. Парсер приймає вхідні дані та перетворює їх на оптимізаційну проблему, що побудована на трьох складових: мінімізації зв'язності, мінімізації накладних витрат на комунікацію та мінімізації накладних витрат реплікації. Вирішувач надає рішення оптимізаційної проблеми – можливе розбиття на мікросервіси. Візуалізатор продукує візуальну репрезентацію запропонованої декомпозиції та надає аналіз витрат, які вона за собою тягне [19]. Інструмент Cromlech побудовано на ідеях Pangaea, а тому робочий процес має ті самі складові. Але у Pangaea організаційні аспекти побудовано навколо сутностей баз даних, що не дає повної гарантії розгортання функціональності деякого бізнес-домену на відповідний йому мікросервіс. Натомість, основний фокус роботи Cromlech знаходиться переважно в області операцій, а не даних, а тому мікросервіси, що є результатом роботи даного інструменту мають покращену пов'язність [20].

Висновки. Проаналізувавши наявні на сьогоднішній день дослідження, присвячені декомпозиції предметної області та успадкованих монолітних систем, можна відзначити широкий спектр наявних принципів, підходів та інструментів. Частина з них закріпилася у якості відомих практик в індустрії, проте велика їх кількість все ще залишається відомою, в основному, академічній спільноті, а тому може виявитися

перспективною з точки зору їх широкого застосування у майбутньому. Таким чином, до перспектив подальших досліджень відносимо практичне застосування та отримання зворотного зв'язку від архітекторів та інженерів, які можуть використати розглянуті методи, інструменти та підходи на практиці, та надати об'єктивні прикладні результати, які допоможуть здійснити широке впровадження найбільш оптимальних інструментів на практиці.

Список літератури:

1. Richardson C. Microservices patterns: with examples in Java. Shelter Island, New York: Manning Publications, 2019. 490 p.

2. Newman S. Building microservices: designing fine-grained systems. Second Edition. Beijing: O'Reilly Media, 2021. 586 p.

3. Richards M., Ford N. Fundamentals of software architecture: an engineering approach. First edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2020. 400 p.

4. Rudrabhatla C.K. Impacts of Decomposition Techniques on Performance and Latency of Microservices // IJACSA. 2020. Vol. 11, № 8.

5. Vera-Rivera F.H. et al. Microservices Backlog – A Model of Granularity Specification and Microservice Identification // Services Computing – SCC 2020 / ed. Wang Q. et al. Cham: Springer International Publishing, 2020. Vol. 12409. P. 85–102.

6. Daoud M. et al. Towards an Automatic Identification of Microservices from Business Processes // 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). Bayonne, France: IEEE, 2020. P. 42–47.

7. Khoshnevis S. A search-based identification of variable microservices for enterprise SaaS // Front. Comput. Sci. 2023. Vol. 17, № 3. P. 173208.

8. Li S. et al. A dataflow-driven approach to identifying microservices from monolithic applications // Journal of Systems and Software. 2019. Vol. 157. P. 110380.

9. Bajaj D., Goel A., Gupta S.C. GreenMicro: Identifying Microservices From Use Cases in Greenfield Development // IEEE Access. 2022. Vol. 10. P. 67008–67018.

10. Bajaj D., Goel A., Gupta S. A Comprehensive Microservice Extraction Approach Integrating Business Functions and Database Entities // IAJIT. 2024. Vol. 21, № 1.

11. Kamimura M. et al. Extracting Candidates of Microservices from Monolithic Application Code // 2018 25th Asia-Pacific Software Engineering Conference (APSEC). Nara, Japan: IEEE, 2018. P. 571–580.

12. Brito M., Cunha J., Saraiva J. Identification of microservices from monolithic applications through topic modelling // Proceedings of the 36th Annual ACM Symposium on Applied Computing. Virtual Event Republic of Korea: ACM, 2021. P. 1409–1418.

13. Carvalho L. et al. Search-based many-criteria identification of microservices from legacy systems // Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion. Cancún Mexico: ACM, 2020. P. 305–306.

14. Al-Debagy O., Martinek P. Dependencies-based microservices decomposition method // International Journal of Computers and Applications. 2022. Vol. 44, № 9. P. 814–821.

15. Zhu L., Tamburri D.A., Casale G. RADF: Architecture decomposition for function as a service // Softw Pract Exp. 2023. P. spe.3276.

16. Staffa S. et al. Pangaea: Semi-automated Monolith Decomposition into Microservices // Service-Oriented Computing / ed. Hacid H. et al. Cham: Springer International Publishing, 2021. Vol. 13121. P. 830–838.

17. Quattrocchi G. et al. Cromlech: Semi-Automated Monolith Decomposition Into Microservices // IEEE Trans. Serv. Comput. 2024. P. 1–16.

18. Ward T., Gulin D. AWS Prescriptive Guidance – Decomposing monoliths into microservices. Amazon Web Services, Inc., 2024.

19. Waseem M. et al. Decision Models for Selecting Patterns and Strategies in Microservices Systems and their Evaluation by Practitioners: arXiv:2201.05825. arXiv, 2022.

20. Vera-Rivera F.H. et al. Microservices Backlog – A Genetic Programming Technique for Identification and Evaluation of Microservices From User Stories // IEEE Access. 2021. Vol. 9. P. 117178–117203.

Beirak D.Ya., Vakaliuk T.A. APPROACHES TO THE DOMAIN DECOMPOSITION OF MICROSERVICE SYSTEMS IN THE SCIENTIFIC LITERATURE

Currently, the issue of building microservice systems, which is complex and manifold, continues to actualize. Microservice architecture allows for designing systems with low coupling, which, owing to the possibility of horizontal scaling and because of smaller separate services, are easier to understand and make changes to

compared to big complex monoliths, improves such architecture parameters as scalability, maintainability, efficiency, etc. These and other reasons shape industry specialists' choice that are making it in favor of this kind of architecture more and more frequently and are moving towards migrating existing legacy monoliths to microservices. One important problem that engineers and architects who choose the microservice architecture face is the decomposition of the domain for greenfield projects or the existing monolith decomposition for brownfield cases. Being that the quality of decomposition has a great initial impact on further development of the system, the necessity of choosing the appropriate methods and tools for performing this task arises. In this article, the greenfield and brownfield decomposition methods and tools are examined. For each of these aspects, both approaches and patterns established in the industry, and methods and tools found primarily in scientific studies are covered. Aside from categorizing methods into green and brownfield, the approaches and instruments covered in this article can be nominally divided into manual, semi-automatic, and automatic. Furthermore, the methods and tools covered are based on various decomposition approaches, the adaptation of metrics, usage of user stories, use cases, and flow diagrams, adaptation of clusterization and graph analysis, data and related functionality analysis, etc.

Key words: *microservice architecture, decomposition, modeling, migration, patterns.*