Маркіян Шестакович[1], Юрій Шабатура[2]*
[1] НЛТУ України, Львів, Україна, ORCID: 0009-0005-6714-1582,
shestakovych.m@nltu.lviv.ua
[2] НАСВ України, Львів, Україна. ORCID: 0000-0002-9961-1244,
shabaturayuriy@gmail.com

Метод структурної нормалізації мікросервісної архітектури інформаційних систем

Анотація. *Запропоновано метод структурної нормалізації мікросервісної архітектури після автоматизованої декомпозиції моноліту. Підхід виявляє й ізолює міжкластерні залежності: будує граф викликів/ресурсів, обчислює метрику впливу для кожної залежності та ітеративно застосовує дві стратегії – винесення спільних компонентів у сервіс з формалізованим API або контрольоване дублювання логіки в доменах найбільшого використання. Експеримент на застосунку з 86 модулів: зовнішні залежності − 41%, когезія кластерів з 0,64 до 0,79, порівняно з вихідним станом.*

Ключові слова *– програмна архітектура, мікросервіси, декомпозиція моноліту, графові нейронні мережі, кластеризація.*

Вступ. Після декомпозиції монолітної архітектури інформаційної системи (ІС), зокрема з використанням моделей на основі Graph Neural Networks, у структурі ІС часто залишаються спільні модулі, бібліотеки або компоненти, що одночасно обслуговують декілька доменів. Такі елементи формують своєрідні «вузли-мости», через які зберігаються приховані залежності між кластерами. Наявність цих зв’язків знижує автономність сервісів, ускладнює окреме оновлення та масштабування компонентів, підвищує ризики помилок під час змін конфігурації системи. Аналіз сучасних досліджень і практичних підходів до переходу від монолітної архітектури до архітектури мікросервісів показує, що проблема надлишкових залежностей між доменами є одним із головних чинників деградації продуктивності й надійності після міграції [1–3]. Актуальним є розроблення формалізованого етапу структурного впорядкування архітектури, який дозволяє виявити й ізолювати залишкові міжкластерні взаємозв’язки, забезпечуючи стабільність та узгодженість програмної структури.

* науковий керівник

Мета – розробити метод виявлення й ізоляції міжкластерних залежностей після первинної GNN-кластеризації, що мінімізує міжсервісну зв'язність і підвищує когезію доменних кластерів.

Методика дослідження. Вхідними даними методу є множина кластерів, отриманих після застосування GraphSAGE у поєднанні з алгоритмом k-means, а також множина зв'язків між ними, що відображають виклики, обмін даними або використання спільних ресурсів. На основі цих даних формується міжкластерний граф залежностей $G_m = (K, E_m)$, де множина K описує сервіси або доменні кластери, а множина E_m містить ребра, які представляють взаємодії між ними. Для кожного зв'язку в графі обчислюється інтегральна метрика впливу, яка узагальнює три основні характеристики: частоту викликів, обсяг і чутливість спільних даних, а також показник центральності, що відображає роль компонента у маршрутах викликів. З метою збереження компактності обчислень метрика визначається як зважена комбінація нормованих складників і відображає ступінь критичності залежності для автономності кластерів. Зв'язки, для яких значення показника перевищує порогове значення τ, вважаються критичними і підлягають ізоляції.

Ізоляція таких елементів може здійснюватися двома основними способами. У першому випадку компонент переноситься до спільного утилітарного сервісу зі стабільним і чітко визначеним API, що спрощує контроль версій та моніторинг. У другому випадку застосовується контрольоване дублювання логіки – функціональність копіюється лише до тих кластерів, у яких вона використовується найчастіше, при цьому консистентність реалізацій забезпечується засобами безперервної інтеграції або автоматизованої генерації артефактів. Після виконання ізоляції проводиться повторне обчислення основних структурних показників архітектури: середньої когезії всередині кластерів, частки міжкластерних викликів та зміни в центральності вузлів, які залишаються спільними. Процес виконується ітераційно доти, доки інтегральний критерій стабільності ΔQ, що характеризує зміну співвідношення когезії й зв'язності між ітера-

* науковий керівник

ціями, наближається до нуля. Це свідчить про досягнення структурної рівноваги архітектури та завершення процесу нормалізації.

Експериментальні результати та обговорення. Апробація методу проведена на демонстраційному застосунку, який після первинної кластеризації GraphSAGE + k-means містив 86 модулів. Аналіз міжкластерного графа виявив 14 критичних залежностей, що становили понад 35% усіх міжсервісних викликів. Після застосування запропонованих стратегій ізоляції кількість зовнішніх залежностей зменшилася на 41%, а середнє значення когезії кластерів зросло з 0,64 до 0,79. Крім того, зафіксовано скорочення середньої довжини маршрутів викликів між доменами та зменшення кількості централізованих «вузлів-посередників». Отримані результати свідчать, що метод забезпечує відчутне зниження зв'язності архітектури та підвищення автономності сервісів, що позитивно впливає на стабільність процесів розгортання й оновлення.

Запропонований підхід не замінює етап початкової декомпозиції, а є її логічним продовженням у вигляді структурного впорядкування архітектури. Його ефективність залежить від якості даних телеметрії викликів і від повноти карти взаємодій між сервісами. Серед можливих обмежень слід відзначити ризик надмірного дублювання функцій у разі некоректного вибору стратегії ізоляції, а також необхідність емпіричного підбору порогового значення τ залежно від рівня навантаження та чутливості даних. У подальших дослідженнях планується автоматизувати вибір між стратегіями ізоляції на основі багатокритеріальної оптимізації, що враховуватиме вагомість залежностей, частоту використання та витрати на супровід.

Висновок. У роботі запропоновано метод структурної нормалізації як формалізований етап упорядкування мікросервісної архітектури після її декомпозиції. Запропонований підхід може бути інтегрований у процеси GNN-декомпозиції монолітних систем як етап постобробки архітектури, спрямований на усунення залишкових міжкластерних залежностей. Застосування методу забезпечує підвищення автономності сервісів, зменшення взаємозв'язків між кластерами та, як наслідок, підвищення стійкості архітектури до змін і розши-

* науковий керівник

рень. Розроблений підхід може використовуватися як практичний інструмент оцінювання структурної готовності систем до розподіленого розгортання, а також для контролю узгодженості життєвих циклів мікросервісів під час модернізації або масштабування програмних комплексів.

Список використаних літературних джерел

[1] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov. Microservices: The Journey So Far and Challenges Ahead. IEEE Software, vol. 35, no. 3, pp. 24–35, May 2018. DOI: https://doi.org/10.1109/MS.2018.2141039

[2] W. Hamilton, R. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS), 2017. DOI: https://doi.org/10.48550/arXiv.1706.02216

[3] A. Mathai, S. Bandyopadhyay, U. Desai, and S. Tamilselvam. Monolith to Microservices: Representing Application Software through Heterogeneous Graph Neural Network. Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI), pp. 3905–3911, Jul. 2022. DOI: https://doi.org/10.24963/ijcai.2022/542

Method for structural normalization of microservice architecture of information systems

Markiyan Shestakovych, Yuriy Shabatura

The paper presents a method for structural normalization of microservice architecture that increases the autonomy and stability of information systems after their decomposition from monolithic structures. The approach builds an inter-cluster dependency graph, computes an integrated influence metric, and applies isolation strategies to reduce coupling between services. Experimental results demonstrate a decrease in external dependencies and an improvement in cluster cohesion, which proves the effectiveness of the proposed approach for enhancing structural consistency and service autonomy.

* науковий керівник