УДК 004.415.2 DOI https://doi.org/10.35546/kntu2078-4481.2025.3.2.33

Я. І. КОРНАГА
доктор технічних наук, професор
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID: 0000-0001-9768-2615

О. М. ГУБАРЄВ
аспірант кафедри інформаційних систем та технологій
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID: 0009-0001-1028-4604

МЕТОД АВТОМАТИЗОВАНОГО ФОРМУВАННЯ КЛАСТЕРІВ ПРОГРАМНОЇ СИСТЕМИ НА ОСНОВІ СТРУКТУРНО-СЕМАНТИЧНОГО АНАЛІЗУ КОМПОНЕНТІВ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

У статті розглянуто проблему трансформації монолітних архітектур програмних систем у мікросервісні, яка є одним із ключових завдань сучасної інженерії програмного забезпечення. Монолітні системи з часом набувають надмірної складності та високого рівня внутрішньої зв'язаності, що ускладнює їх масштабування, супровід і модифікацію. Мікросервісна архітектура, орієнтована на незалежні сервіси з чіткими інтерфейсами, значною мірою вирішує ці проблеми, однак постає нетривіальне завдання коректної декомпозиції, яке не має єдиного універсального розв'язку.

У роботі запропоновано метод декомпозиції, що поєднує структурний та семантичний аналіз вихідного коду з навчанням нейронних мереж для графів, спрямованих на автоматизоване формування кластерної структури системи. Особливістю підходу є багаторівнева побудова графа залежностей, яка враховує різні типи зв'язків (виклики методів, наслідування, використання ресурсів), а також запровадження вагових коефіцієнтів вузлів, розрахованих на основі показників центральності. Перед етапом кластеризації застосовано поетапну структурну оптимізацію графа, що включає виявлення сильно зв'язаних компонент, циклів, мостів і точок артикуляції, а також зниження впливу надлишкових і шумових зв'язків. Це дозволяє підвищити точність і надійність подальшого групування компонентів.

Додатково реалізовано адаптивний вибір алгоритму кластеризації на основі метрик графа (щільність, кількість вузлів, центральність, коефіцієнт спільного використання ресурсів), що дозволяє узгоджувати метод із поточними характеристиками структури. Навчання графової нейронної мережі здійснюється на векторах ознак, які інтегрують топологічні та семантичні характеристики компонентів, що забезпечує виявлення латентних закономірностей у складних залежностях.

Порівняльний аналіз із сучасними методами декомпозиції (CoGCN, DEEPLY, HyDEC, GDC-DVF, Mo2oM) підтвердив ефективність запропонованого підходу. Результати показали перевищення існуючих рішень за показником структурної модульності (SM), що відображає вищу когезію та чіткіші межі сформованих сервісів, при одночасному збереженні відносно низького рівня міжсервісних залежностей. Це свідчить про наукову новизну та практичну значущість розробленого методу, який може бути використаний як у дослідницькому середовищі, так і в реальних промислових системах.

Ключові слова: *монолітна архітектура, мікросервісна архітектура, декомпозиція, когезія, зв'язаність, графові моделі, семантичний аналіз, нейронні мережі.*

YA. I. KORNAHA
Doctor of Technical Sciences, Professor
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0000-0001-9768-2615

O. M. HUBARIEV
Postgraduate Student at the Department of Information Systems and Technologies
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0009-0001-1028-4604

A METHOD OF AUTOMATED FORMATION OF CLUSTERS OF THE SOFTWARE SYSTEM BASED ON STRUCTURAL-SEMANTIC ANALYSIS OF COMPONENTS USING NEURAL NETWORKS

The article examines the problem of transformation of monolithic architectures of software systems into microservice ones, which is one of the key tasks of modern software engineering. Monolithic systems eventually acquire excessive complexity and a high level of internal connectivity, making them difficult to scale, maintain and modify. Microservice architecture, oriented towards independent services with clear interfaces, largely solves these problems, however, the non-trivial task of correct decomposition arises, which does not have a single universal solution.

The paper proposes a decomposition method that combines structural and semantic analysis of source code with training of neural networks for graphs aimed at automated formation of the cluster structure of the system. A feature of the approach is the multi-level construction of the dependency graph, which takes into account different types of relationships (calls of methods, imitation, use of resources), as well as the introduction of node weights calculated on the basis of centrality indicators. Prior to the clustering step, a stepwise structural graph optimization is applied, involving the detection of strongly coupled components, cycles, bridges and articulation points, and the reduction of the influence of redundant and noise links. This makes it possible to increase the accuracy and reliability of further grouping of components.

In addition, an adaptive selection of the clustering algorithm based on graph metrics (density, number of nodes, centrality, resource sharing coefficient) is implemented, which allows you to match the method with the current characteristics of the structure. Graph neural network training is carried out on feature vectors that integrate topological and semantic characteristics of components, which ensures the detection of latent regularities in complex dependencies.

Comparative analysis with current decomposition methods (CoGCN, DEEPLY, HyDEC, GDC-DVF, Mo2oM) confirmed the effectiveness of the proposed approach. The results showed an excess of existing solutions according to the structural modularity indicator (SM), which reflects higher cohesion and clearer boundaries of the formed services, while maintaining a relatively low level of interservice dependencies. This shows the scientific novelty and practical significance of the developed method, which can be used both in the research environment and in real industrial systems.

Key words: *monolithic architecture, microservice architecture, decomposition, cohesion, connectivity, graph models, semantic analysis, neural networks.*

Постановка проблеми

Проблема переходу від монолітних систем на мікросервіси є однією з ключових у сучасній програмній інженерії. Моноліти, історично розроблені як єдині інтегровані системи, з часом набувають надмірної складності та високого рівня внутрішнього зв'язку, що ускладнює масштабування, модифікацію та підтримку. Мікросервісна архітектура, яка базується на незалежних компонентах з чіткими межами, вирішує ці проблеми, але вимагає правильного визначення меж між сервісами.

Задача декомпозиції моноліту на мікросервіси не має єдиного рішення: надто дрібна сегментація створює надмірний міжсервісний трафік і збільшує складність координації, тоді як надто грубе розбиття зберігає проблеми моноліту. Тому в наукових дослідженнях запропоновано ряд методів, від евристичних підходів на основі статичного аналізу до методів глибинного навчання нейронних мереж для графових структур. Серед сучасних рішень варто згадати CoGCN, DEEPLY, HyDec, GDC-DVF, і Mo2oM. Вони по-різному інтегрують структурний і семантичний аналіз, демонструючи різні компроміси між когезією та каплінгом.

Метою даної роботи є розробка та дослідження вдосконаленого методу декомпозиції, який поєднує статичний аналіз вихідного коду, оптимізацію графів залежностей, динамічну вибір алгоритму кластеризації, оптимізацію результатів кластеризації з урахуванням когезії, каплінгу та семантичної однорідності, а також навчання нейронної мережі на основі отриманих оптимізованих даних. Запропонований підхід спрямований на досягнення високої когезії сервісів з контрольованим рівнем міжсервісної зв'язності та семантичної однорідності, що є необхідною умовою ефективної експлуатації програмних систем.

Аналіз останніх досліджень і публікацій

CO-GCN (Clustering and Outlier-aware Graph Convolutional Network) – метод декомпозиції монолітних застосунків, який представляє систему у вигляді графа, де вузлами є класи, а ребрами – залежності викликів та спільна участь у виконанні точок входу. На основі цього графа формується матриця ознак, що поєднує структурні характеристики та семантичні властивості.

Особливістю підходу є урахування вузлів-викидів, які спотворюють структуру системи. Для цього у CO-GCN використовується функція втрат із трьох компонентів: втрати від структурних викидів, втрати від атрибутивних викидів та втрати від кластеризації. Навчання здійснюється за схемою почергової мінімізації, коли послідовно оновлюються параметри мережі, оцінки викидів і центри кластерів [1].

Метод було протестовано на типових Java-системах, зокрема DayTrader, де він продемонстрував вищу якість декомпозиції порівняно з базовими методами. Водночас автори відзначають, що масштабованість підходу обмежена через високу обчислювальну складність процедури навчання та залежність результатів від повноти й якості даних про транзакції [6].

DEEPLY – метод декомпозиції монолітних застосунків, що ґрунтується на глибинному аналізі вихідного коду та транзакцій користувачів із метою автоматизованого формування мікросервісів. Система подається у вигляді матриці, яка враховує як статичні залежності між класами, так і динамічні зв'язки, що виявляються під час виконання. На основі цих даних нейронна мережа навчається виявляти приховані закономірності та групувати компоненти у функціонально узгоджені сервіси.

Особливістю підходу є поєднання структурної інформації (виклики, спільні ресурси) та поведінкової інформації (шаблони виконання транзакцій). Це дозволяє моделі виділяти сервіси, які не лише мають внутрішню когезію, а й відповідають реальним сценаріям використання системи [2, 3].

Експерименти показали, що метод здатен перевершувати класичні евристичні підходи та конкурувати з експертними системами. Водночас обмеженням DEEPLY є висока потреба у даних трасування виконання, а також значні обчислювальні витрати при навчанні на великих системах [6].

HyDec– метод декомпозиції монолітних застосунків, що базується на ієрархічному застосуванні алгоритму DBSCAN для виявлення меж потенційних мікросервісів. Система представляється у вигляді графа залежностей між класами та транзакціями, після чого відбувається багаторівневе групування елементів із врахуванням щільності їхніх зв'язків. На відміну від класичного DBSCAN, який працює на одному рівні щільності, HyDec використовує ієрархічний підхід: спочатку формуються великі кластери, а потім вони поступово розділяються на менші підкласи доти, доки не досягається оптимальний баланс когезії та зв'язаності.

Ключовою особливістю підходу є здатність уникати проблем надмірного злиття або, навпаки, надмірної фрагментації сервісів. Алгоритм динамічно підлаштовує поріг щільності для різних ділянок графа, що дозволяє більш точно виявляти функціонально пов'язані компоненти [4].

Експерименти, проведені на типовому наборі Java-застосунків, показали, що HyDec здатен перевершувати як евристичні методи, так і окремі гібридні підходи, забезпечуючи кращий баланс між когезією та мінімізацією міжсервісних залежностей. Водночас метод чутливий до вибору параметрів початкового DBSCAN і може потребувати додаткового налаштування для систем із різною структурою [6].

GDC-DVF – метод виділення мікросервісів, що ґрунтується на глибокій кластеризації графів із використанням механізму «подвійного злиття уявлень» (dual view fusion). У даному підході програмна система моделюється як граф залежностей, який розглядається одночасно з двох точок зору: структурної (статичні зв'язки між класами, спільні ресурси, виклики) та атрибутивної (семантичні й поведінкові характеристики).

Ключова ідея методу полягає у поєднанні цих двох уявлень за допомогою механізму злиття, що дозволяє уникнути перекосів, властивих підходам, орієнтованим лише на один тип даних. Для цього застосовується глибока нейронна мережа кластеризації, яка навчається формувати узгоджені векторні подання вузлів і групувати їх у кластери, що відповідають потенційним мікросервісам [5].

Експерименти продемонстрували, що GDC-DVF забезпечує вищу якість декомпозиції порівняно з традиційними методами та низкою сучасних моделей, зокрема завдяки здатності адаптивно враховувати як структурні, так і семантичні ознаки. Водночас метод має підвищені вимоги до обсягів навчальних даних і є обчислювально затратним, що може ускладнити його застосування у великих промислових системах [6].

Mo2oM – метод декомпозиції монолітних систем у мікросервіси, який зосереджений на підтримці перекривних сервісів (overlapping microservices), тобто ситуацій, коли окремі компоненти можуть належати до кількох функціональних підсистем одночасно. Така особливість відповідає реальним вимогам великих програмних систем, де однакові класи чи модулі можуть обслуговувати різні бізнес-процеси.

У цьому підході використовується поєднання глибоких семантичних векторних подань (deep semantic embeddings), що моделюють зміст і функціональне призначення класів, та графових нейронних мереж, які враховують структурні залежності між компонентами. Кластеризація виконується у «м'якій» формі, тобто з неповним закріпленням вузлів за одним кластером, що дозволяє формувати перекривні множини мікросервісів.

Результати досліджень показали, що Mo2oM забезпечує вищі значення когезії сервісів і водночас дозволяє зберегти гнучкість при моделюванні складних архітектурних залежностей. Метод особливо ефективний на системах із великою кількістю повторно використовуваних класів. Водночас його обчислювальна складність є підвищеною, а для якісного результату потрібні багаті семантичні ознаки вихідного коду [6].

Аналіз показує, що поєднання графових моделей та методів машинного навчання має значний потенціал для підвищення якості декомпозиції, однак існуючі рішення залишаються обмеженими щодо точності та здатності до узагальнення.

Формулювання мети дослідження

Метою дослідження є розробка методу декомпозиції монолітних архітектур, що забезпечує підвищення ефективності мікросервісних систем шляхом поєднання структурного й семантичного аналізу та використання нейронних мереж для формування висококогерентних і слабо зв'язаних сервісів.

Викладення основного матеріалу дослідження

Запропонований підхід спрямований на підвищення точності та надійності процесу декомпозиції монолітних застосунків шляхом поєднання статичного аналізу коду, формальної оптимізації графа залежностей, динамічного вибору алгоритму кластеризації та використання нейронних мереж для моделювання структурних і семантичних властивостей компонентів.

У запропонованому методі вихідний код системи перетворюється на орієнтований зважений граф, де вузли відображають різні типи компонентів – від центральних сервісів до допоміжних утиліт. Оскільки їхня роль у функціонуванні системи є нерівнозначною, просте використання лише зв'язків між ними може призвести до хибних результатів кластеризації: наприклад, до об'єднання ключових компонентів у один кластер або, навпаки, до відокремлення другорядних модулів у незалежні мікросервіси.

Щоб уникнути таких викривлень, у методі запроваджено ваговий коефіцієнт вузла W(v), який кількісно відображає архітектурну значущість кожного компонента. Розрахунок цього коефіцієнта ґрунтується на трьох аспектах:

1. кількості його вхідних і вихідних зв'язків;
2. ролі у передачі даних між іншими модулями;
3. частоті використання як залежності в інших компонентах.

Формула розрахунку ваги вузла:

$$W(v) = \alpha \cdot DC_{in}(v) + \beta \cdot DC_{out}(v) + \gamma \cdot BC(v),$$

де: $DC_{in}(v)$ – вхідна центральність вузла v; $DC_{out}(v)$ – вихідна центральність вузла v; $BC(v)$ – посередницька центральність вузла v; $\alpha, \beta, \gamma \in [0; 1]$ – вагові коефіцієнти, визначаються емпірично на основі типу системи.

Ребра відображають різні типи залежностей: виклики методів, використання спільних ресурсів, наслідування, використання інтерфейсу, використання класу. Проте для коректної декомпозиції недостатньо враховувати лише сам факт існування такого зв'язку. У реальних системах різні типи залежностей мають нерівнозначний вплив на архітектуру. Тому під час побудови орієнтованого графа залежностей важливо не лише відображати наявність ребер, але й кількісно оцінювати їхню значущість. Це досягається шляхом призначення вагових коефіцієнтів, які відображають силу взаємозв'язку та його вплив на узгодженість майбутніх кластерів. Таким чином формується більш реалістична топологія графа, що дозволяє уникнути спотворень під час подальшої кластеризації.

Формула для розрахунку ваги зв'язку:

$$W(e_{ij}) = \lambda_k \cdot w_k,$$

де: λ_k – коефіцієнт важливості конкретного типу зв'язку; w_k – вага відповідного типу зв'язку.

Вага зв'язку w_k визначає загальну силу зв'язку між модулями, але не враховує конкретний контекст. Коефіцієнт важливості λ_k дозволяє адаптувати модель під специфіку системи (змінювати значущість різних зв'язків). Це додає гнучкість для налаштування алгоритму аналізу та кластеризації, особливо при масштабуванні системи.

Це дає змогу кількісно оцінити інтегрованість компонентів і створити більш реалістичну модель системи.

На відміну від підходів, де граф залежностей використовується без попередньої обробки, у запропонованому методі передбачено його поетапну оптимізацію. На першому етапі за допомогою алгоритму обходу в глибину визначаються критичні точки графа, зокрема мости та точки артикуляції, що впливають на глобальну зв'язність системи. Далі виконується пошук сильно зв'язаних компонент, які характеризуються високою внутрішньою когезією та відображають циклічні залежності у коді. Такі компоненти розглядаються як єдині логічні блоки, оскільки їхнє штучне розділення призводить до втрати цілісності та появи структурних антипатернів.

Виявлення мостів, точок артикуляції та сильно зв'язаних компонент дає змогу виділити критичні області графа та зменшити надмірні циклічні залежності, завдяки чому граф набуває більш збалансованої структури й стає придатним для подальшої кластеризації.

На основі аналізу ключових графових метрик – кількості вузлів, щільності, середньої нормалізованої посередницької центральності, середня кількість вихідних залежностей та коефіцієнта спільного використання ресурсів – було сформовано адаптивний алгоритм вибору методу кластеризації. Такий підхід дозволяє динамічно узгоджувати обраний алгоритм зі структурними особливостями графа, що забезпечує вищу ефективність обчислень і покращує якість кластерного розбиття.

Після первинного розбиття графа на кластери виконується додатковий етап перевірки та удосконалення отриманої структури. На цьому етапі враховуються три ключові фактори:

- когезія (щільність внутрішніх зв'язків);
- зв'язаність (мінімізація міжкластерних викликів);
- семантична однорідність (узгодженість назв класів, пакетів, доменних понять).

Відповідно, оптимізаційна задача полягає у знаходженні такого розбиття $C_1, C_2, \ldots, C_k$, яке максимізує узагальнену функцію:

$$\max_{C_1,\ldots,C_K}\left[\lambda_1\cdot\frac{1}{K}\sum_{i=1}^{K}Cohesion\left(C_i\right)-\lambda_2\cdot\frac{1}{|E|}\sum_{i\neq j}Coupling\left(C_i,C_j\right)\right],\ \text{за умови:}\forall i,\ Sem\left(C_i\right)\geq\delta,$$

де: λ_1, λ_2 – вагові коефіцієнти, які відображають важливість кожного критерію; δ – фіксований пороговий рівень.

Розв'язання вищезазначеної задачі дозволить досягти балансованого архітектурного зонування програмної системи з чітко вираженими межами відповідальності підсистем.

Запропонована стратегія оптимізації є адаптивною та багатокритеріальною, оскільки враховує як топологічні, так і семантичні властивості системи. Вона забезпечує поступове вдосконалення кластерної структури шляхом локальних змін, усуває надлишкові або семантично нечіткі зв'язки та створює підґрунтя для переходу до архітектури з високою когезією та контрольованою зв'язаністю.

У межах дослідження було використано графову нейронну мережу для автоматизованого уточнення кластеризації, сформованої на основі попередньої оптимізації графа та евристичного структурування. На відміну від підходів із жорстко заданими правилами, запропонована модель здатна відображати як структурні, так і семантичні закономірності у системі залежностей.

Для кожного компонента сформовано вектор ознак, що поєднує топологічні характеристики та семантичні індикатори. Це дозволило врахувати як локальну, так і глобальну роль компонентів.

Отримані ознаки були інтегровані у нормалізовану матрицю суміжності з додаванням самопетель, що забезпечило стабільність агрегації та збереження власної інформації вузлів. Навчання нейронної мережі виконувалося на основі кластерних міток, отриманих під час оптимізації графа, із використанням функції втрат, яка поєднувала три складники: максимізацію внутрішньої когезії, мінімізацію міжкластерних залежностей та підвищення семантичної однорідності.

У результаті сформовано кластерну структуру, яка не лише відтворила початкову функціональну логіку, але й покращила її за рахунок узгодження структурних і семантичних властивостей. Додатковий етап оптимізації результатів дозволив знизити рівень міжкластерних зв'язків шляхом локального перенесення вузлів, що мали надмірні залежності, у суміжні кластери за умови збереження їх когезії.

Завершальний етап дослідження полягав у порівняльному аналізі запропонованого підходу до декомпозиції монолітної архітектури з сучасними методами CoGCN, DEEPLY, HyDEC, GDC-DVF та Mo2oM. Числові значення для цих методів узято з узагальненої таблиці результатів у роботі [6], що забезпечує коректність зіставлення завдяки використанню єдиних метрик і тестових застосунків.

Якість декомпозиції оцінювалась за двома взаємодоповнюючими метриками: структурна модульність (SM), яка характеризує рівень когезії всередині сервісів, та частка міжсервісних викликів (ICP), що відображає ступінь зовнішньої зв'язаності. Високі значення SM свідчать про ефективне інкапсулювання бізнес-логіки, тоді як низькі значення ICP означають мінімізацію міжсервісних залежностей. Для системи, де враховано всі типи зв'язків, виконується співвідношення SM+ICP=1 що підкреслює їх взаємодоповнюваність.

З огляду на різнорідність тестових систем і чутливість методів до предметної області, для зіставлення враховано найкращі зафіксовані показники SM та ICP для кожного з підходів. Це дозволяє оцінити граничний потенціал алгоритмів за сприятливих умов, а не усереднені результати, що можуть згладжувати сильні сторони окремих методів.

Для запропонованого методу показники отримані під час експериментів на реальному промисловому проєкті, що перебуває в експлуатації. Це забезпечило перевірку підходу в умовах, коли архітектурні залежності безпосередньо відображають бізнес-процеси та робочі навантаження, і підтвердило його практичну ефективність. Результати представлені в таблиці 1.

Таблиця 1

Порівняння сучасних методів декомпозиції монолітних систем за метриками структурної модульності (SM) та частки міжсервісних викликів (ICP)

Метод	Найкращий SM	Найкращий ICP
CoGCN	0.086	0.300
DEEPLY	0.215	0.347
HyDEC	0.509	0.037
GDC-DVF	0.152	0.274
Mo2oM	0.692	0.100
Запропонований метод	0.726	0.274

Результати аналізу свідчать, що за показником структурної модульності (SM) запропонований підхід досягає найвищого значення (0.726), перевищуючи навіть сучасний метод Mo2oM (0.692). Це підтверджує більш високу когезію сформованих сервісів, їх внутрішню узгодженість та чіткість архітектурних меж.

За показником частки міжсервісних викликів (ICP) найнижче значення демонструє метод HyDEC (0.037), проте воно супроводжується істотним зниженням когезії. Найбільш збалансованим серед існуючих підходів є Mo2oM (SM = 0.692, ICP = 0.100). Водночас запропонований метод забезпечує ще вищу когезію при збереженні відносно низького рівня міжсервісної зв'язаності (ICP = 0.274). Таким чином, він демонструє оптимальний баланс: сформовані сервіси є внутрішньо стійкими й водночас достатньо ізольованими для мінімізації міжсервісних залежностей.

Висновки

Порівняльний аналіз сучасних підходів до декомпозиції засвідчує, що найвищі результати демонструють методи, які поєднують графові евристики з навчальними моделями. Запропонований у цьому дослідженні метод, що інтегрує оптимізацію графа залежностей, динамічний вибір алгоритму кластеризації та навчання графової нейронної мережі, перевищує всі відомі рішення за показником структурної модульності, що підтверджує його ефективність у формуванні когерентних і добре структурованих сервісів. Подальші напрями розвитку пов'язані з удосконаленням механізмів зниження міжсервісних викликів та впровадженням методів підкріплювального навчання для автоматизації вибору кластеризаційної стратегії.

References

1. Desai, U., Bandyopadhyay, S., & Tamilselvam, S. (2021). *Graph Neural Network to Dilute Outliers for Refactoring Monolith Application*. AAAI 2021, DOI: https://doi.org/10.48550/arXiv.2102.03827

2. Yedida, R., Krishna, R., Kalia, A., Menzies, T., Xiao, J., & Vukovic, M. (2021). *Partitioning cloud-based microservices (via deep learning)*. arXiv preprint arXiv:2109.14569. DOI: http://dx.doi.org/10.48550/arXiv.2109.14569

3. Yedida, R., Krishna, R., Kalia, A., Menzies, T., Xiao, J., & Vukovic, M. (2023). An expert system for redesigning software for cloud applications. Expert Systems with Applications, 219, 119673. DOI: https://doi.org/10.1016/j.eswa.2023.119673

4. Sellami, K., Saied, M. A., & Ouni, A. (2022). *A hierarchical DBSCAN method for extracting microservices from monolithic applications*. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*. (EASE '22), pp. 201–210. DOI: https://doi.org/10.1145/3530019.3530040

5. Qian, L., Li, J., He, X., Gu, R., Shao, J., & Lu, Y. (2023). *Microservice extraction using graph deep clustering based on dual view fusion*. Information and Software Technology, 158, 107171. DOI: https://doi.org/10.1016/j.infsof.2023.107171

6. Ziabakhsh, A., Rezaee, M., Eskandari, M., & Goudarzi, M. (2025). *Extracting Overlapping Microservices from Monolithic Code via Deep Semantic Embeddings and Graph Neural Network–Based Soft Clustering*. arXiv preprint arXiv:2508.07486. DOI: https://doi.org/10.48550/arXiv.2508.07486

Дата першого надходження рукопису до видання: 27.09.2025
Дата прийнятого до друку рукопису після рецензування: 24.10.2025
Дата публікації: 28.11.2025