УДК 004.65; 004.91 DOI: 10.30837/0135-1710.2022.178.037

Н.В. ВАСИЛЬЦОВА. І.Ю. ПАНФЬОРОВА

ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ МЕТОДІВ ІЄРАРХІЧНОЇ КЛАСТЕРИЗАЦІЇ ПІД ЧАС ВИРІШЕННЯ ЗАДАЧІ АНАЛІЗУ КОНФІГУРАЦІЇ ІТ-ПРОДУКТУ

Розглянуто основні особливості існуючих способів рішення задачі аналізу конфігурації ІТ-продукту в рамках процесу управління конфігурацією. Виділено основні недоліки цих способів. Розглянуто рішення задачі аналізу конфігурації ІТ-продукту із застосуванням дивізимного та агломеративного алгоритмів. Проведено порівняльний аналіз особливостей застосування ієрархічних алгоритмів кластеризації для вирішення задачі аналізу конфігурації ІТ-продукту. Запропоновано модифікацію алгоритму найближчого сусіда, що дозволяє своєчасно виявляти конфігураційні елементи з описами, які повністю збігаються.

1. Вступ

Сучасна точка зору на процеси управління проєктами виділяє процес управління конфігурацією ІТ-продукту як один з процесів інтегрованого контролю змін. Процес управління конфігурацією спрямований на визначення конфігурації товару в окремі моменти часу. Метою даного процесу є систематичний контроль за змінами конфігурації, а також підтримка цілісності та відстеження конфігурації протягом усього життєвого циклу продукту [1]. Основними роботами процесу управління конфігурацією є: планування та управління процесом управління конфігурацією; ідентифікація конфігурації; контроль конфігурації; облік стану конфігурації; аудит конфігурації та управління випуском та доставкою продукту. Особливе місце серед цих робіт посідає робота "ідентифікація конфігурації". Дана робота визначає елементи, що підлягають контролю, встановлює схеми ідентифікації елементів та їх версій, а також інструменти та методи, які будуть використовуватися для отримання та управління виділеними елементами [1].

Проте слід визнати, що у теперішній час виділення процесу управління конфігурацією та специфікації його окремих робіт досить умовно. Рекомендації щодо впровадження процесу управління конфігурацією у життєвий цикл ІТ-продукту носять переважно теоретичний та

методичний характер [1]. Як одну з причин цієї ситуації слід зазначити невизначеність, яка виникає в ході виділення елементів ІТ-продукту, що підлягають контролю. Як такий елемент прийнято розглядати так званий елемент конфігурації (Configuration Item, CI). CI прийнято визначати як сукупність апаратних засобів, програмного забезпечення, або того та іншого разом, яка виступає одиницею для управління конфігурацією та розглядається як єдина сутність у процесі управління конфігурацією. У той самий час CI найчастіше розглядається в межах процесу управління конфігурацією ІТ-продукту як деяка сукупність інших CI, організованих деяким способом (найчастіше - ієрархічно). Наслідком цього є виникнення в ході виконання роботи "ідентифікація конфігурації" великої кількості теоретичних та прикладних проблем, пов'язаних із виділенням оптимальної кількості CI. Під оптимальною тут слід розуміти таку кількість CI, яка вимагатиме мінімальних витрат часу та ресурсів на виконання подальших робіт процесу управління конфігурацією ІТ-продукту

Головною особливістю виконання процесу управління конфігурацією в сучасних ІТ-проєктах є необхідність ідентифікації конфігурації розроблюваного ІТ-продукту якомога раніше. У [2] показано, що ранній поділ великої програмно-технічної системи на окремі елементи дозволяє усунути більшість помилок, що виникають при спробах інтеграції окремих локальних рішень у рамках загального дизайну системи, а також дозволяє підвищити якість даної системи. При цьому більшість помилок, які все ж таки виникали в ході проєктування подібних систем, були наслідками неправильного тлумачення вимог або упередженості особистого досвіду [2]. Тому використання людино-машинних чи машинних методів для автоматизації процесу управління конфігурацією слід визнати доцільним, а проведення досліджень у цій галузі - актуальним з теоретичної та прикладної точок зору.

2. Аналіз літературних даних і постановка проблеми дослідження

Головним напрямом досліджень у сфері управління конфігурацією ІТ-продуктів слід визнати вирішення задач, які виникають у ході декомпозиції опису архітектури продукту, що розробляється, на окремі послуги. Подібна проблема розглянута у [3]. При цьому опис архітектури в [3] запропоновано виконувати спеціальною предметно-орієнтованою мовою Silvera. Ця мова та її компілятор дозволяють під час проєктування програмної системи автоматизувати декомпозицію загальносистемного опису архітектури на описи окремих мікросервісів. Однак застосування цього рішення обмежено такими особливостями [3]:

- орієнтацією виключно на децентралізовану розробку мікросервісів;
- неможливістю визначення оптимальної кількості команд розробників виділених мікросервісів.

Задача виділення мікросервісів у ході рефакторингу вихідного коду монолітного програмного застосування розглянуто в [4]. Для рішення цієї задачі в [4] запропоновано використовувати алгоритми кластеризації. Вибір цього способу рішення задачі обумовлений тим, що він виділяє більш зв'язні мікросервіси, які мають менше проміжних взаємодій. Аналогічна ситуація спостерігається і для випадків рефакторингу складних багаторівневих монолітних програмних систем у ході їх декомпозиції на мікросервіси [5]. Однак розглянуті в [4, 5] рішення засновані на припущенні про відсутність залежності мікросервісів, що виділяються, від раніше виділених у цих ІТ-продуктах функцій. Передбачається, що функціональна декомпозиція опису архітектури вихідної програмної системи проводилася в ході її проєктування та не змінюється в ході рефакторингу.

Для підвищення якості рефакторингу монолітної системи шляхом поділу її архітектури на значну кількість мікросервісів [6] запропоновано використовувати в ході декомпозиції динамічний аналіз фактичної поведінки програмної системи. Однак при цьому не враховується, що загальна поведінка системи визначається насамперед сукупністю сценаріїв виконання множини окремих функціональних вимог до цієї системи.

Слід визнати, що розглянуті дослідження лише підтверджують міркування, викладені у [7]. Тут стверджується, що універсальних підходів до декомпозиції монолітної архітектури на окремі мікросервіси, скоріш за все, не існує. Однак подібне твердження потребує додаткової перевірки у разі перенесення вирішення задачі на більш високий рівень абстракції - у ході виділення CI, що реалізують окремі функції системи.

Варіант формального вирішення проблеми дослідження шляхом виділення окремих програмних артефактів, що забезпечують надійну роботу проєктованої програмної системи, з

опису мета-архітектури такої системи, запропоновано у [8]. Однак виділення з опису архітектури системи СІ за функціональною ознакою у [8] не розглядається.

У [9] наголошується, що вирішення більшості задач вибору ІТ-сервісів вимагає визначення множини функціонально еквівалентних ІТ-сервісів до початку вирішення подібних задач. Це означає, що задача декомпозиції опису архітектури системи, створеної на основі множини функціональних системних вимог, на окремі ІТ-сервіси має бути вирішена окремо для абстрактних описів окремих сервісів. Такі абстрактні описи не повинні залежати від особливостей реалізації цих сервісів та нефункціональних вимог, що висуваються до сервісів. Аналогічний підхід до здійснення функціональної декомпозиції опису архітектури системи на окремі елементи показаний у [10]. При цьому поставлена задача аналізу зміни конфігурації вирішується у два етапи:

- на першому етапі виконується функціональна декомпозиція опису архітектури на окремі елементи з урахуванням потрібних еволюційних змін;
- на другому етапі здійснюється вибір тих варіантів декомпозиції, які задовольняють обмеженням на вартість потрібних змін.

Аналіз розглянутих публікацій дозволяє сформулювати висновок про необхідність поділу задачі формування множини варіантів декомпозиції опису архітектури системи на окремі СІ на дві послідовно розв'язувані підзадачі:

- підзадачу формування множини варіантів декомпозиції опису архітектури системи на окремі функціональні СІ;
- підзадачу вибору з множини окремих функціональних СІ такої підмножини, яка задовольнятиме умовам відбору найкращим чином.

При цьому слід пам'ятати, що особливості вирішення задачі формування множини варіантів декомпозиції опису архітектури системи на окремі функціональні СІ досліджені дуже слабко.

Наведені результати аналізу дозволяють зробити висновок про необхідність проведення досліджень у сфері пошуку таких методів вирішення задачі аналізу конфігурації ІТ-продукту, які можуть сформувати множину усіх можливих варіантів декомпозиції опису архітектури ІТ-продукту на окремі СІ.

3. Мета і задачі дослідження

Метою даного дослідження є виявлення переваг та недоліків вирішення задачі аналізу конфігурації ІТ-продукту, отриманих в результаті застосування методів ієрархічної кластеризації. Ці методи дозволяють сформувати множину можливих варіантів декомпозиції опису архітектури ІТ-продукту у вигляді дендрограми кластерів окремих СІ. Тому досягнення цієї мети дозволить обґрунтовано вибрати найменш витратний спосіб розв'язання задачі аналізу конфігурації ІТ-продукту.

Для досягнення цієї мети у статті вирішуються такі задачі:

- короткий опис особливостей вирішення задачі аналізу конфігурації ІТ-продукту із застосуванням дивізимного алгоритму кластеризації;
- вирішення задачі аналізу конфігурації ІТ-продукту із застосуванням алгоритму найближчого сусіда;
- порівняльний аналіз отриманих рішень.

4. Моделі і методи, що використовуються для формального опису функціональних вимог до інформаційної системи

У ході дослідження пропонується використовувати методи ієрархічної кластеризації. Вибір даних методів обумовлений тим, що вони дозволяють отримати в результаті вирішення задачі кластеризації найповніше уявлення про структуру кластерів. Найчастіше таке уявлення дозволяє наочно сприймати результат вирішення підзадачі формування множини варіантів декомпозиції опису архітектури системи на окремі функціональні СІ. У свою чергу, візуальне представлення вирішення даної підзадачі дозволяє надалі використовувати для вирішення підзадачі вибору з множини окремих функціональних СІ підмножини, що задовольняє умовам відбору якнайкраще, не тільки формальні методи аналізу, але і методи візуального аналізу даних.

Методи ієрархічної кластеризації зазвичай поділяють на два великі класи: агломеративні та дивізимні алгоритми. Агломеративні алгоритми характеризуються послідовним поєднанням вихідних елементів та відповідним зменшенням числа кластерів. Дивізимні алгоритми характеризуються зростанням числа кластерів, починаючи з одного, в результаті чого утворюється послідовність груп, що розщеплюються.

Як приклад агломеративних алгоритмів у дослідженні запропоновано використати алгоритм найближчого сусіда. Даний алгоритм стосовно вирішуваної задачі може бути представлений у вигляді наступних кроків.

Крок 1. Всю множину *CI* представити як множину кластерів *C*, кожен з яких містить один елемент CI_i, *i=1,...,n,* де *n* - кількість елементів у множині *CI*. Розрахувати матрицю відстаней *D* між елементами множини *C*.

Крок 2. Вибрати два кластери C_p та C_q, відстань між якими буде мінімальною, і об'єднати їх у новий кластер C_r, ввівши його замість кластерів C_p та C_q у множину кластерів *C*.

Крок 3. Перерахувати значення матриці відстаней *D*, використовуючи правило

$$d_{rs} = 0,5 \times d_{ps} + 0,5 \times d_{qs} + 0 \times d_{pq} - 0,5 \times \left| d_{ps} - d_{qs} \right|, r \neq s, r \neq p, r \neq q \,, \qquad (1)$$

де d_{ps} - відстань між центрами кластерів C_p та C_s; d_{qs} - відстань між центрами кластерів C_q та C_s; d_{pq} - відстань між центрами кластерів C_p та C_q.

Крок 4. Повторювати Крок 2 і Крок 3 доти, поки не буде сформовано один кластер, що включає всі елементи множини *CI*.

Як приклад дивізимних алгоритмів у дослідженні запропоновано використовувати алгоритм, запропонований Смітом Макнаотоном [11]. Даний алгоритм стосовно розв'язуваної задачі може бути представлений у вигляді наступних кроків [11, 12].

Крок 1. Сформувати один кластер C_1, що складається з усіх *m* елементів вихідної множини об'єктів кластеризації *CI*.

Крок 2. Вибрати елемент кластера C_1 з найбільшим середнім значенням відстані від інших елементів кластера. Середнє значення відстані для елемента CI_p кластера C_1 можна розрахувати таким чином

$$D_{C_1}(CI_p) = \frac{1}{m} \times \sum_{q=1}^{m} d(CI_p, CI_q) \;\; \forall CI_p, CI_q \in C_1, p \neq q \,, \qquad (2)$$

де *m* - кількість елементів в кластері C_1; CI_p - *p*-й елемент кластера C_1; CI_q - *q*-й елемент кластера C_1; $d(CI_p, CI_q)$ - відстань між елементами CI_p та CI_q.

Крок 3. Вибраний на Кроці 2 елемент видалити з кластера C_1 і включити в новий кластер C_2.

Крок 4. Серед елементів кластера C_1, що залишилися, знайти такий, для якого різниця між середньою відстанню до елементів кластера C_1, що залишилися, і середньою відстанню до елементів, включених в кластер C_2, позитивна і максимальна.

Крок 5. Вибраний на Кроці 4 елемент видалити із кластера C_1 і включити в новий кластер C_2.

Крок 6. Продовжувати виконувати Кроки 4 і 5, поки різниці середніх відстаней не стануть негативними, після чого завершити виконання алгоритму.

Результатом виконання даного алгоритму є поділ вихідного кластера C_1 на два дочірні - C_1 і C_2. Далі вибирається один із дочірніх кластерів і за допомогою цього алгоритму поділяється ще на два дочірні кластери. Процедура поділу зупиняється в одному з таких випадків [11, 12]:

а) у дочірньому кластері залишається лише один елемент;

б) усі елементи дочірнього кластера мають нульову відмінність один від одного.

Для визначення відстані в алгоритмах, що розглядаються, авторами в [12] запропоновано використовувати модифіковану відстань Чебишева, яка визначається таким чином

$$d_{\infty}(CI_p, CI_q) = \max_{1 \leq l \leq m} \sum_{k=1}^{m} (CI_{plk} \oplus CI_{qlk}) \,, \qquad (3)$$

де CI_{plk} - значення ідентифікатора *k*-ї сутності або класу, що входять в опис функції, вхідного чи вихідного потоку CI_{pl} СІ CI_p ; CI_{qlk} - значення ідентифікатора *k*-ї сутності або класу, що входять в опис функції, вхідного або вихідного потоку CI_{ql} СІ CI_q ; *m* - максимальне значення ідентифікатора, що бере участь в описі порівнюваних функцій, вхідних або вихідних потоків СІ CI_p та CI_q ; $\oplus$ - операція "сума за модулем 2".

5. Вирішення задачі аналізу конфігурації ІТ-продукту з використанням дивізимного алгоритму кластеризації

5.1. Опис задачі аналізу конфігурації ІТ-продукту

Вихідними даними для вирішення задачі аналізу конфігурації ІТ-продукту є опис СІ функціональної задачі "Формування та ведення індивідуального плану науково-педагогічного працівника кафедри". Цю задачу реалізовано у вигляді окремого ІТ-продукту для розвитку можливостей інформаційно-аналітичної системи "Університет" Харківського національного університету радіоелектроніки. Детальний опис найменувань та позначень функцій та потоків даних, що формують окремі СІ, наведено у табл. 1-3 [12]. У таблиці 1 прийняті скорочення: ІП - індивідуальний план; KPI - ключові показниках ефективності.

Таблиця 1

Опис елементів конфігурації функціональної задачі "Формування і ведення індивідуального плану науково-педагогічного працівника кафедри"
(на основі схеми потоків даних)

Робота		Вхідний потік		Вихідний потік	
№	Найменування	№	Найменування	№	Найменування
СІ1	Конвертація розділу «Навчальна робота	1	Навчальне навантаження викладача на навчальний рік	2	Інформація з розділу ІП «Навчальна робота»
СІ2	Формування розділу «Наукова робота»	2	Інформація про викладача	3	Інформація з розділу ІП «Наукова робота»
		3	Інформація про заплановані для виконання роботи		
		5	Інформація про рекомендовані до виконання роботи		
		8	Інформація з розділу ІП «Наукова робота»		
		12	Залишок годин		
СІ3	Формування розділу «Методична робота»	2	Інформація про викладача	4	Інформація з розділу ІП «Методична робота»
		3	Інформація про заплановані для виконання роботи		
		5	Інформація про рекомендовані до виконання роботи		
		9	Інформація з розділу ІП «Методична робота»		
		12	Залишок годин		

Продовження таблиці 1

Робота		Вхідний потік		Вихідний потік	
№	Найменування	№	Найменування	№	Найменування
CI4	Формування розділу «Організаційна робота»	2	Інформація про викладача	5	Інформація з розділу ІП «Організаційна робота»
		3	Інформація про заплановані для виконання роботи		
		5	Інформація про рекомендовані до виконання роботи		
		10	Інформація з розділу ІП «Організаційна робота»		
		12	Залишок годин		
CI5	Формування переліку посад та довгострокових доручень	4	Інформація про посади та довгострокові доручення	6	Інформація з розділу ІП «Перелік посад та довгострокових доручень»
		11	Інформація з розділу ІП «Перелік посад та довгострокових доручень»		
CI6	Формування переліку рекомендованих до виконання робіт	5	Інформація про рекомендовані до виконання роботи	1	Інформація про рекомендовані до виконання роботи
CI7	Формування і ведення нормативно-довідкової інформації про KPI	6	Інформація про ключові KPI кафедри	7	Інформація про ключові KPI кафедри
CI8	Формування KPI викладача і частини від KPI кафедри	8	Інформація з розділу ІП «Наукова робота»	9	Інформація про KPI викладача і частини від KPI кафедри
CI9	Формування зведеної таблиці на навчальний рік	9	Інформація з розділу ІП «Методична робота»	8	Інформація про кількість годин по розділах ІП
		7	Інформація з розділу ІП «Навчальна робота»		
		8	Інформація з розділу ІП «Наукова робота»		
		10	Інформація з розділу ІП «Організаційна робота»		

Кінець таблиці 1

Робота		Вхідний потік		Вихідний потік	
№	Найменування	№	Найменування	№	Найменування
СІ10	Формування вихідного документа «ІП»	9	Інформація з розділу ІП «Методична робота»	10	ІП
		7	Інформація з розділу ІП «Навчальна робота»		
		8	Інформація з розділу ІП «Наукова робота»		
		10	Інформація з розділу ІП «Організаційна робота»		
		11	Інформація з розділу ІП «Перелік посад та довгострокових доручень»		

Таблиця 2

Множина описів сутностей функціональної задачі

ID	Найменування
1	Academic_load
2	Academic
3	Department
4	Individual_plan
5	Academic_section
6	Academic_year
7	Section
8	Recommended_works
9	Type_of_work
10	Section_Pos_Assign_Dept
11	PositionsAssignments
12	KPI

5.2. Результати вирішення задачі аналізу конфігурації ІТ-продукту з використанням дивізимного алгоритму кластеризації

Детально хід вирішення задачі аналізу конфігурації ІТ-продукту для описаних вище вихідних даних з використанням дивізимного алгоритму і модифікованої відстані Чебишева розглянуто в [12]. Результатом вирішення задачі є дендрограмма, яка наведена на рис. 1 [12].

6. Рішення задачі аналізу конфігурації ІТ-продукту з використанням алгоритму найближчого сусіда

Під час виконання Кроку 1 було сформовано 10 кластерів, в кожному з яких знаходилося по одному СІ задачі. Для цих кластерів було розраховано матрицю відстаней, яка наведена у табл. 4. Розрахунок відстаней здійснювався за формулою (3).

Під час виконання першої ітерації Кроку 2 було виділено пару найближчих один до одного кластерів *C2* и *C3*. Відстань між цими кластерами дорівнює 0. Вибір здійснювався під час розглядання матриці відстаней (табл. 4) зліва направо і зверху донизу. Було сформовано новий кластер *C11={C2, C3}*.

Таблиця 3

Векторні описи елементів конфігурації функціональної задачі "Формування та ведення індивідуального плану науково-педагогічного працівника кафедри"

Опис функції CI												
CI	ID сутностей											
	1	2	3	4	5	6	7	8	9	10	11	12
CI1	1	1	1	1	1	1	0	0	0	0	0	0
CI2	1	1	1	1	1	1	1	1	1	0	0	0
CI3	1	1	1	1	1	1	1	1	1	0	0	0
CI4	1	1	1	1	1	1	1	1	1	0	0	0
CI5	0	1	0	1	0	1	1	0	0	1	1	0
CI6	0	0	0	0	0	0	0	1	1	0	0	0
CI7	0	0	0	0	0	0	0	0	0	0	0	1
CI8	1	1	0	1	1	1	1	1	1	0	0	1
CI9	1	1	0	1	1	1	1	1	1	0	0	0
CI10	1	1	0	1	1	1	1	1	1	1	1	0
Опис вхідних потоків даних CI												
CI	ID сутностей											
	1	2	3	4	5	6	7	8	9	10	11	12
CI1	1	1	1	0	0	0	0	0	0	0	0	0
CI2	1	1	1	1	1	1	1	1	1	0	0	0
CI3	1	1	1	1	1	1	1	1	1	0	0	0
CI4	1	1	1	1	1	1	1	1	1	0	0	0
CI5	0	1	0	1	0	1	1	0	0	1	1	0
CI6	0	0	0	0	0	0	0	1	1	0	0	0
CI7	0	0	0	0	0	0	0	0	0	0	0	1
CI8	1	1	0	1	0	1	1	1	1	0	0	0
CI9	1	1	0	1	1	1	1	1	1	0	0	0
CI10	1	1	0	1	1	1	1	1	1	1	1	0
Опис вихідних потоків даних CI												
CI	ID сутностей											
	1	2	3	4	5	6	7	8	9	10	11	12
CI1	1	1	0	1	1	1	0	0	0	0	0	0
CI2	0	1	0	1	0	1	1	1	1	0	0	0
CI3	0	1	0	1	0	1	1	1	1	0	0	0
CI4	0	1	0	1	0	1	1	1	1	0	0	0
CI5	0	1	0	1	0	1	1	0	0	1	1	0
CI6	0	0	0	0	0	0	0	1	1	0	0	0
CI7	0	0	0	0	0	0	0	0	0	0	0	1
CI8	1	1	0	1	1	1	1	0	0	0	0	1
CI9	1	1	0	1	1	1	1	1	0	0	0	0
CI10	1	1	0	1	1	1	1	1	1	1	1	0

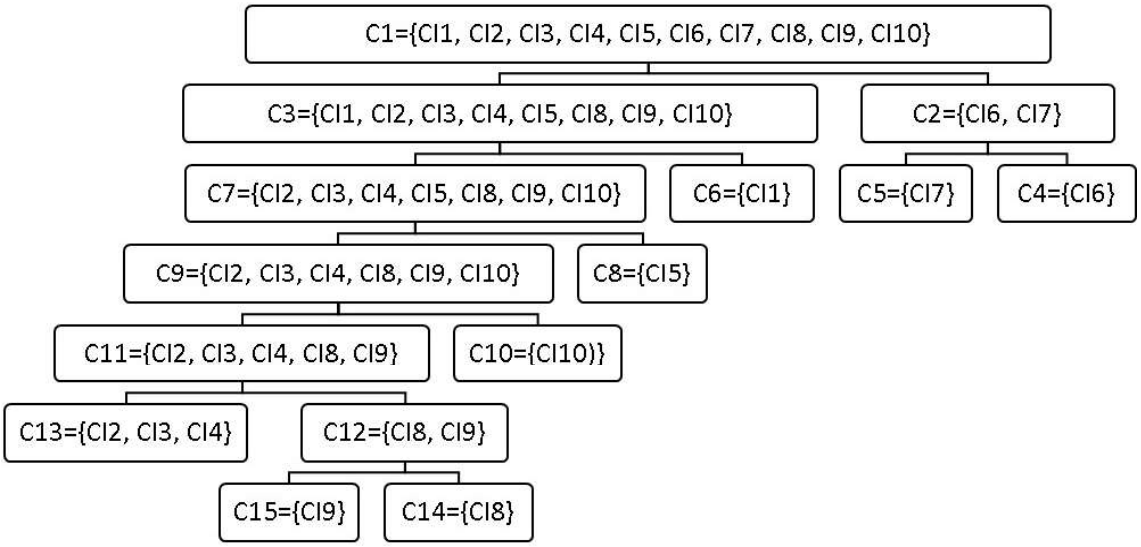

Рис. 1. Дендрограма кластерів конфігураційних елементів, сформована в результаті застосування дивізимного алгоритму

У ході виконання першої ітерації Кроку 3 було проведено перерахунок матриці відстаней *D* з урахуванням наявності нового кластера *C11*. Результат перерахунку показано в табл. 5.

У ході виконання другої ітерації Кроку 2 було виділено пару найближчих один до одного кластерів *C11* і *C4*. Відстань між цими кластерами дорівнює 0. Було сформовано новий кластер *C12={C11, C4}*.

У ході виконання першої ітерації Кроку 3 було проведено перерахунок матриці відстаней *D* з урахуванням наявності нового кластера *C12*. Результат перерахунку показано в табл. 6.

У ході виконання третьої ітерації Кроку 2 було виділено пару найближчих один до одного кластерів *C8* і *C9*. Відстань між цими кластерами дорівнює 2. Було сформовано новий кластер *C13={C8, C9}*.

У ході виконання третьої ітерації Кроку 3 було проведено перерахунок матриці відстаней *D* з урахуванням наявності нового кластера *C13*. Результат перерахунку показано в табл. 7.

У ході виконання четвертої ітерації Кроку 2 було виділено пару найближчих один до одного клас-

Таблиця 4

Вихідна матриця відстаней
(алгоритм найближчого сусіда)

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	0	6	6	6	7	8	7	6	7	9
C2	6	0	0	0	7	7	10	4	3	4
C3	6	0	0	0	7	7	10	4	3	4
C4	6	0	0	0	7	7	10	4	3	4
C5	7	7	7	7	0	8	7	7	6	4
C6	8	7	7	7	8	0	3	7	7	9
C7	7	10	10	10	7	3	0	8	9	11
C8	6	4	4	4	7	7	8	0	2	4
C9	7	3	3	3	6	7	9	2	0	3
C10	9	4	4	4	4	9	11	4	3	0

Таблиця 5

	C1	C11	C4	C5	C6	C7	C8	C9	C10
C1	0	6	6	7	8	7	6	7	9
C11	6	0	0	7	7	10	4	3	4
C4	6	0	0	7	7	10	4	3	4
C5	7	7	7	0	8	7	7	6	4
C6	8	7	7	8	0	3	7	7	9
C7	7	10	10	7	3	0	8	9	11
C8	6	4	4	7	7	8	0	2	4
C9	7	3	3	6	7	9	2	0	3
C10	9	4	4	4	9	11	4	3	0

Таблиця 6

	C1	C12	C5	C6	C7	C8	C9	C10
C1	0	6	7	8	7	6	7	9
C12	6	0	7	7	10	4	3	4
C5	7	7	0	8	7	7	6	4
C6	8	7	8	0	3	7	7	9
C7	7	10	7	3	0	8	9	11
C8	6	4	7	7	8	0	2	4
C9	7	3	6	7	9	2	0	3
C10	9	4	4	9	11	4	3	0

Таблиця 7

	C1	C12	C5	C6	C7	C13	C10
C1	0	6	7	8	7	6	9
C12	6	0	7	7	10	3	4
C5	7	7	0	8	7	6	4
C6	8	7	8	0	3	7	9
C7	7	10	7	3	0	8	11
C13	6	3	6	7	8	0	3
C10	9	4	4	9	11	3	0

Таблиця 8

	C1	C14	C5	C6	C7	C10
C1	0	6	7	8	7	9
C14	6	0	6	7	8	3
C5	7	6	0	8	7	4
C6	8	7	8	0	3	9
C7	7	8	7	3	0	11
C10	9	3	4	9	11	0

Таблиця 9

	C1	C15	C5	C6	C7
C1	0	6	7	8	7
C15	6	0	4	7	8
C5	7	4	0	8	7
C6	8	7	8	0	3
C7	7	8	7	3	0

терів *C12* і *C13*. Відстань між цими кластерами дорівнює 3. Було сформовано новий кластер *C14={C12, C13}*.

У ході виконання четвертої ітерації Кроку 3 було проведено перерахунок матриці відстаней *D* з урахуванням наявності нового кластера *C14*. Результат перерахунку показано в табл. 8.

У ході виконання п'ятої ітерації Кроку 2 було виділено пару найближчих один до одного кластерів *C14* і *C10*. Відстань між цими кластерами дорівнює 3. Було сформовано новий кластер *C15={C14, C10}*.

У ході виконання п'ятої ітерації Кроку 3 було проведено перерахунок матриці відстаней *D* з урахуванням наявності нового кластера *C15*. Результат перерахунку показано в табл. 9.

У ході виконання шостої ітерації Кроку 2 було виділено пару найближчих один до одного кластерів *C6* і *C7*. Відстань між цими кластерами дорівнює 3. Було сформовано новий кластер *C16={C6, C7}*.

У ході виконання шостої ітерації Кроку 3 було проведено перерахунок матриці відстаней *D* з урахуванням наявності нового кластера *C16*. Результат перерахунку показано в табл. 10.

У ході виконання сьомої ітерації Кроку 2 було виділено пару найближчих один до одного кластерів *C15* і *C5*. Відстань між цими кластерами дорівнює 4. Було сформовано новий кластер *C17={C15, C5}*.

У ході виконання сьомої ітерації Кроку 3 було проведено перерахунок матриці відстаней *D* з урахуванням наявності нового кластера *C17*. Результат перерахунку показано в табл. 11.

У ході виконання восьмої ітерації Кроку 2 було виділено пару найближчих один до одного кластерів *C1* і *C17*. Відстань між цими кластерами дорівнює 7. Було сформовано новий кластер *C18={C1, C17}*.

У ході виконання восьмої ітерації Кроку 3 було проведено перерахунок матриці відстаней *D* з урахуванням наявності нового кластера *C18*. Результат перерахунку показано в табл. 12.

Під час виконання дев'ятої ітерації Кроку 2 на основі кластерів *C1* і *C17* було сформовано кластер *C19*, який включає в себе всі вихідні кластери. На цьому виконання алгоритму найближчого сусіда завершується.

Результатом виконання алгоритму найближчого сусіда є дендрограма, що має вигляд, показаний на рис. 2.

Таблиця 10

	C1	C15	C5	C16
C1	0	6	7	7
C15	6	0	4	7
C5	7	4	0	7
C16	7	7	7	0

Таблиця 11

	C1	C17	C16
C1	0	7	7
C17	7	0	7
C16	7	7	0

Таблиця 12

	C18	C16
C18	0	7
C16	7	0

7. Порівняльний аналіз отриманих рішень

Наведені на рис. 1 та рис. 2 дендрограми, отримані в результаті вирішення задачі із застосуванням дивізимного алгоритму та алгоритму найближчого сусіда відповідно, збігаються майже повністю. Винятком є кластери *C2*, *C3*, *C4* и *C11*, виділені під час побудови дендрограми, показаної на рис. 2. Поява цих кластерів обумовлена тим, що алгоритм найближчого сусіда не здатний розпізнавати CI, описи яких повністю збігаються. Цей недолік є причиною того, що перші дві ітерації виконання Кроків 2 і 3 алгоритму найближчого сусіда були спрямовані на злиття кластерів, в яких знаходилися CI з повністю ідентичними описами.

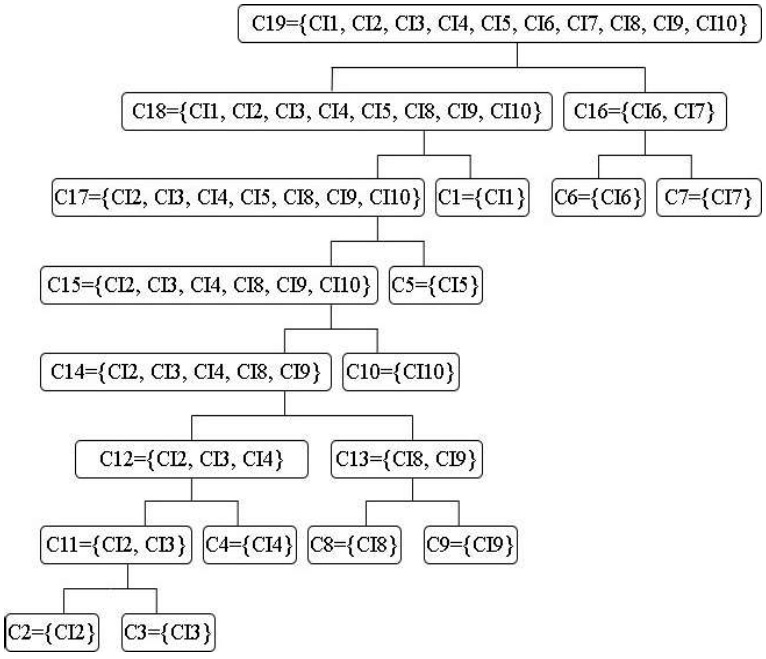

Рис. 2. Дендрограма кластерів конфігураційних елементів, сформована в результаті застосування агломеративного алгоритму найближчого сусіда

На практиці цей недолік алгоритму найближчого сусіда може призвести до того, що виконавцям IT-проєкту в ході планування їх робіт для реалізації CI з описами, що збігаються (в даному випадку *CI2*, *CI3* и *CI4*), будуть виділятися окремі спринти. Це призведе до невиправданого завищення оцінок трудовитрат і витрат часу виконання робіт з реалізації цих CI.

Слід зазначити, що агломеративний алгоритм найближчого сусіда з погляду обчислювальної складності простіше за дивізимний алгоритм Сміта Макнаотона. Зокрема, для отримання аналогічного результату алгоритм найближчого сусіда не вимагає проведення досить складних обчислень щодо розрахунку середніх відстаней та пошуку елементів, що переводяться з батьківського кластера до новоствореного дочірнього кластера. Тому для автоматизації вирішення задачі аналізу конфігурації IT-продукту, а саме підзадачі формування множини варіантів декомпозиції опису архітектури системи на окремі функціональні CI, доцільніше вибирати агломеративні алгоритми (у даному випадку - алгоритм найближчого сусіда) за умови усунення зазначеного вище недоліку. Для усунення зазначеного недоліку пропонується скоригувати алгоритм найближчого сусіда, доповнивши його операціями з коригування множини вихідних кластерів, сформованих на Кроці 1. У результаті цього доповнення модифікований алгоритм найближчого сусіда буде представлений послідовністю таких кроків.

Крок 1. Всю множину *CI* представити як множину вихідних кластерів *C*, кожен з яких містить один елемент CI_i, $i=1,...,n$, де n - кількість елементів у множині *CI*.

Крок 2. Розрахувати матрицю відстаней *D* між елементами множини *C*.

Крок 3. Скоригувати множину вихідних кластерів C шляхом поєднання кожної пари вихідних кластерів, що задовольняє умові

$$\forall d_{pq} = 0 \quad \exists C_r = C_p \cup C_q, p, q = 1, m, p \neq q, \tag{4}$$

після чого виключити кластери CI_p і CI_q з подальшого розгляду та не відображати їх на підсумковій дендрограмі.

Крок 4. Вибрати два кластери CI_p і CI_q, відстань між якими буде мінімальною, і об'єднати їх у новий кластер c_r, ввівши його замість кластерів CI_p і CI_q у множину кластерів C.

Крок 5. Перерахувати значення матриці відстаней D, використовуючи таке правило

$$d_{rs} = 0,5 \times d_{ps} + 0,5 \times d_{qs} + 0 \times d_{pq} - 0,5 \times \left| d_{ps} - d_{qs} \right|, r \neq s, r \neq p, r \neq q. \tag{5}$$

Крок 6. Повторювати Крок 2 і Крок 3 доти, поки не буде сформований один кластер, що включає всі елементи множини CI.

Крок 7. Сформувати підсумкову дендрограму та завершити роботу алгоритму.

Запропонована модифікація усуває зазначений вище недолік алгоритму найближчого сусіда та не призводить до серйозного збільшення обчислювальної складності даного алгоритму.

8. Висновки та перспективи подальших досліджень

У ході даного дослідження задачу аналізу конфігурації ІТ-продукту було вирішено із застосуванням агломеративного алгоритму найближчого сусіда. Як вихідні дані були використані описи СІ функціональної задачі "Формування та ведення індивідуального плану науково-педагогічного працівника кафедри".

Результатом рішення є дендрограма (рис. 2), яка відображає результати об'єднання кластерів, що містять опис окремих СІ функціональної задачі. Дана дендрограма може бути використана для подальшого вирішення підзадачі вибору з множини окремих функціональних СІ такої підмножини, яка задовольнятиме умовам відбору якнайкраще. Під умовами відбору слід розуміти сукупність проєктних обмежень, що враховуються при призначенні окремих СІ для реалізації виконавцям ІТ-проєкту.

Було проведено порівняльний аналіз ходу та результатів вирішення задачі аналізу конфігурації ІТ-продукту із застосуванням агломеративного алгоритму найближчого сусіда та дивізимного алгоритму Сміта Макнаотона.. Отримані рішення майже повністю збігаються. Однак з точки зору обчислювальної складності алгоритм найближчого сусіда кращий, ніж дивізимний алгоритм Сміта Макнаотона за умови усунення недоліку, зазначеного в ході аналізу. Цей недолік полягає в нездатності алгоритму найближчого сусіда розпізнавати описи окремих СІ, що повністю збігаються один з одним. Для усунення цього недоліку було здійснено модифікацію алгоритму найближчого сусіда. Слід зазначити, що запропонована модифікація не призводить до серйозного збільшення обчислювальної складності даного алгоритму.

Як перспективи подальших досліджень слід зазначити, перш за все, дослідження рішення задачі розподілу результатів аналізу конфігурації ІТ-продукту між командами виконавців ІТ-проєкту як окремого випадку підзадачі вибору з множини окремих функціональних СІ такої підмножини, яке задовольнятиме умовам відбору найкращим чином. Іншим напрямом подальших досліджень є вивчення можливості застосування для вирішення задачі аналізу конфігурації ІТ-продукту неієрархічних алгоритмів кластеризації (наприклад, різних модифікацій алгоритмів k-means та fuzzy c-means).

Список літератури: 1. *Bourque P., Fairley R.E. (eds).* Guide to the Software Engineering Body of Knowledge. Version 3.0. IEEE Computer Society, 2014.335 p. 2. *Cadavid H., Andrikopoulos V., Avgeriou P., Chris Broekema P.* System and software architecting harmonization practices in ultra-large-scale systems of systems: A confirmatory case study. Information and Software Technology. 2022. 150. № 106984. DOI: https://doi.org/10.1016/j.infsof.2022.106984. 3. *Suljkanovic A., Milosavljevic B., Indic V., Dejanovic I.* Developing Microservice-Based Applications Using the Silvera Domain-Specific Language. Applied Sciences (Switzerland). 2022. 13 (12). № 6679. DOI: https://doi.org/10.3390/app12136679 4. *Sellami Kh., Saied M.A.,*

Ouni A. A Hierarchical DBSCAN Method for Extracting Microservices from Monolithic Applications. 2022 ACM International Conference on Evaluation and Assessment in Software Engineering, EASE. 2022. P. 201-210. DOI: 10.1145/3530019.3530040. 5. *Krause A., Zirkelbach C., Hasselbring W., Lenga S., Kroger D.* Microservice Decomposition via Static and Dynamic Analysis of the Monolith. 2020 IEEE International Conference on Software Architecture Companion, ICSA-C 2020. 2020. P. 9-16. DOI: 10.1109/ICSA-C50368.2020.00011. 6. *Matias T., Correia F.F., Fritzsch J., Bogner J., Ferreira H.S., Restivo A.* Determining microservice boundaries: A case study using static and dynamic software analysis. 14th European Conference on Software Architecture, ECSA 2020. 2020. P. 315-332. DOI: 10.1007/978-3-030-58923-3_21. 7. *Fritzsch J., Bogner J., Zimmermann A., Wagner S.* From monolith to microservices: A classification of refactoring approaches. 1st International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment, DEVOPS 2018. 2019. P. 128-141. DOI: 10.1007/978-3-030-06019-0_10. 8. *Shahin R.* Towards Assurance-Driven Architectural Decomposition of Software Systems. 40th International Conference on Computer Safety, Reliability and Security, SAFECOMP 2021 held in conjunction with Workshops on DECSoS, MAPSOD, DepDevOps, USDAI and WAISE. 2021. P. 187-196. DOI: 10.1007/978-3-030-83906-2_15. 9. *Reiff-Marganiec S., Tilly M (Eds.).* Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions. Hershey: IGI Global. 2012. 21 p. DOI: 10.4018/978-1-61350-432-1. 10. *Faitelson D., Heinrich R., Tyszberowicz Sh.* From monolith to microservices: Supporting software architecture evolution by functional decomposition. 5th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2017. 2017. P. 435-442. DOI: 10.5220/0006206204350442 .11. *Wierzchon S., Klopotek M.* Modern Algorithms of Cluster Analysis. Springer Cham, 2018. 441 c. DOI: https://doi.org/10.1007/978-3-319-69308-8. 12. *Ievlanov M., Vasiltcova N., Neumyvakina O., Panforova I.* Development of a method for solving the problem of IT product configuration analysis. Eastern-European Journal of Enterprise Technologies. 2022. Vol. 6. № 2. P. 6-19. DOI: 10.15587/1729-4061.2022.269133.

Надійшла до редколегії 30.11.2022

Васильцова Наталія Володимирівна, канд. техн. наук, доцент, професор кафедри ІУС ХНУРЕ. Наукові інтереси: проєктування інформаційних систем управління підприємствами та складними техніко-економічними об'єктами; управління командами виконавців ІТ-проєктів. Адреса: 61166, Харьков, пр. Науки 14, ХНУРЕ, каф. ІУС, контактний телефон: +38(057)7021451.

Панфьорова Ірина Юріївна, канд. техн. наук, доцент, професор кафедри ІУС ХНУРЕ. Наукові інтереси: інформаційні технології управління базами даних, проєктування баз та сховищ даних складних інформаційних систем. Адреса: 61166, Харьков, пр. Науки 14, ХНУРЕ, каф. ІУС, контактний телефон: +38(057)7021451.