Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)
Кафедра _____Інформаційних управляючих систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Дослідження моделей і методів аналізу конфігурації ІТ-продукту
(тема)

Виконала:
студентка _2_ курсу, групи __УПГІТм-20-1_
_____Кузьма Є.А._____
(прізвище, ініціали)

Спеціальність ____122 Комп'ютерні_____
науки_____
(код і повна назва спеціальності)
Тип програми __освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма Управління проектами в галузі інформаційних технологій_____
(повна назва освітньої програми)

Керівник _____проф. Євланов М.В._____
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ ___Петров К.Е.__
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерних наук_____
Кафедра_____Інформаційних управляючих систем_____
Рівень вищої освіти_____другий (магістерський)_____
Спеціальність_____122 Комп'ютерні науки_____
(код і повна назва)
Тип програми _____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма_Управління проектами в галузі інформаційних технологій_
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці_____Кузьмі Єлизаветі Анатоліївні_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження моделей і методів аналізу конфігурації ІТ-продукту_____
затверджена наказом університету від 05 _листопада_ 2021 р. № __1646_Ст_
2. Термін подання студентом роботи до екзаменаційної комісії 11 грудня 2021 р.
3. Вихідні дані до роботи: науково-технічні публікації; джерела інтернету; науково-технічна література, що стосуються теми кваліфікаційної роботи_____
4. Перелік питань, що потрібно опрацювати в роботі: дослідження управління конфігурацією ІТ-продукту, основного понятійного апарату; дослідження задач управління конфігурацією; дослідження методів вирішення задачі аналізу конфігурації ІТ-продукту; дослідження інструментів управління конфігурацією; дослідження формальної постановки задачі кластеризації; дослідження алгоритмів вирішення задачі кластеризації придатних для вирішення задачі аналізу конфігурації; розробка моделей аналізу конфігурації ІТ-продукту; розробка плану проекту, в рамках якого проходитиме практична апробація методу; практичне використання методу аналізу конфігурації ІТ-продукту.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Дослідження управління конфігурацією ІТ-продукту, основного понятійного апарату та задач	08.11.21-11.11.21	
2	Дослідження методів вирішення задачі аналізу конфігурації ІТ-продукту та інструментів управління конфігурацією	12.11.21-16.11.21	
3	Дослідження формальної постановки задачі кластеризації	17.11.21-18.11.21	
4	Дослідження алгоритмів вирішення задачі кластеризації придатних для вирішення задачі аналізу конфігурації	19.11.21-20.11.21	
5	Розробка моделей аналізу конфігурації ІТ-продукту	21.11.21-25.11.21	
6	Розробка плану проекту, в рамках якого проходитиме практична апробація методу	26.11.21-30.11.21	
7	Практичне використання методу аналізу конфігурації ІТ-продукту	01.12.21-04.12.21	
8	Оформлення пояснювальної записки та графічного матеріалу	05.12.21-06.12.21	
9	Підготовка презентації	07.12.21-08.12.21	
10	Подання студентом роботи для перевірки на плагіат	09.12.2021	
11	Надання роботи на підпис науковому керівнику	11.12.2021	
12	Попередній захист роботи	11.12.2021	
13	Надання роботи на рецензію	11.12.2021	
14	Надання роботи на підпис завідувачу кафедри	11.12.2021	
15	Захист кваліфікаційної роботи	13.12.2021	

Дата видачі завдання _08_ __листопада___2021 р.

Студент _____
(підпис)

Керівник роботи _____ ___проф. Євланов М. В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 107 сторінок, 32 рисунка, 9 таблиць, 2 додатка, 33 джерела.

АНАЛІЗ КОНФІГУРАЦІЇ, ДЕНДРОГРАМА, КЛАСТЕРИЗАЦІЯ, КОНФІГУРАЦІЯ, УПРАВЛІННЯ КОНФІГУРАЦІЄЮ

Об’єкт дослідження кваліфікаційної роботи – конфігурації ІТ-продукту.

Мета дослідження – побудова моделей і методів, що описують результати вирішення задачі кластеризації для розподілу опису архітектури ІТ-продукту як окремих завдань між членами команди виконавців. Подібне рішення дозволить скоротити час на аналіз опису архітектури ІТ-продукту та автоматизувати процес його поділу на відповідні завдання та аналітично обґрунтувати обраний варіант конфігурації продукту.

Проведено аналіз процесу управління конфігураціями, виділено процес, який потребує автоматизації – процес ідентифікації конфігурації ІТ-продукту. Розроблена постановка задачі дослідження.

Проведена розробка моделей аналізу конфігурації ІТ-продукту, в тому числі формальна постановка задачі кластеризації, дослідження алгоритмів вирішення задачі кластеризації придатних для вирішення задачі дослідження, визначено міри подібності.

Проведено розробку методу аналізу конфігурації ІТ-продукту, в рамках якого виділено підготовчий та основний етапи, описано особливості застосування розробленого методу.

Проведено й описано практичне використання методу аналізу конфігурації ІТ-продукту, в тому числі розроблено відповідний плану проекту, описано вхідні дані методу, виконано підготовчий та основний етапи методу та проаналізовано отримані дані.

ABSTRACT

The explanatory note to the qualification work: 107 pages, 32 images, 9 tables, 2 attachmenti, 33 sources.

CLUSTERIZATION, CONFIGURATION, CONFIGURATION ANALYSIS, CONFIGURATION MANAGEMENT, DENDROGRAM

The object of study of the qualification work - the configuration of the IT product. The purpose of the study is to build models and methods that describe the results of solving the clustering problem for the distribution of the description of the architecture of the IT product as separate tasks among the members of the team of performers. Such a solution will reduce the time to analyze the description of the architecture of the IT product and automate the process of its division into appropriate tasks and analytically justify the chosen product configuration.

The analysis of the configuration management process is carried out, the process that needs automation is highlighted - the process of identifying the configuration of the IT product. The formulation of the research problem is developed.

The development of models of IT product configuration analysis, including the formal formulation of the clustering problem, the study of algorithms for solving the clustering problem suitable for solving the research problem, the degree of similarity is determined. The method of analysis of the IT product configuration was developed, within the framework of which the preparatory and main stages were singled out, the peculiarities of application of the developed method were described.

The practical use of the IT product configuration analysis method is carried out and described, including the corresponding project plan is developed, the input data of the method are described, the preparatory and main stages of the method are performed and the obtained data are analyzed.

ЗМІСТ

Вступ 9
1.1 Управління конфігурацією ІТ-продукту 10
1.1.1 Дослідження процесу управління конфігурацією 10
1.1.2 Дослідження основного понятнійного апарату 12
1.1.3 Дослідження основних задач управління конфігурацією 15
1.1.4 Дослідження окремих особливостей перебігу задачі ідентифікації конфігурації 18
1.2 Дослідження методів вирішення задачі аналізу конфігурації ІТ-продукту 19
1.3 Дослідження інструментів управління конфігурацією 21
1.4 Постановка задачі дослідження 25
2 Розробка моделей аналізу конфігурації ІТ-продукту 26
2.1 Формальна постановка задачі кластеризації 26
2.1.1 Загальна формальна постановка задачі кластеризації 26
2.1.2 Міри подібності при порівнянні об’єктів, що мають кількісні ознаки 28
2.1.3 Міри подібності при порівнянні об’єктів, що мають якісні ознаки . 29
2.2 Дослідження алгоритмів вирішення задачі кластеризації придатних для вирішення задачі аналізу конфігурації ІТ-продукту 31
2.2.1 Дослідження та вибір класу алгоритмів кластеризації 31
2.2.2 Дослідження та вибір алгоритму ієрархічної кластеризації 33
2.3 Визначення міри подібності між описами елементів архітектури ІТ-продукту, що порівнюються 36
2.4 Висновки 38
3 Розробка методу аналізу конфігурації ІТ-продукту 39
3.1 Опис підготовчого етапу 39
3.2 Опис основного алгоритму аналізу конфігурації ІТ-продукту 40

3.3 Особливості застосування розробленого методу 41
3.4 Висновки 42
4 Практичне використання методу аналізу конфігурації ІТ-продукту 44
4.1 Розробка плану проекту, в рамках якого проходитиме практична апробація методу 44
4.2 Опис вхідних даних методу 48
4.3 Виконання підготовчого етапу методу аналізу конфігурації ІТ-продукту 51
4.4 Виконання основного алгоритму аналізу конфігурації ІТ-продукту 54
4.5 Аналіз отриманих даних 57
Висновки 65
Перелік джерел посилання 67
Додаток А Звіт «Data Usage Report» 71
Додаток Б Графічний матеріал кваліфікаційної роботи на тему «Дослідження моделей і методів аналізу конфігурації ІТ-продукту» 82

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних;

ДСТУ – Державний стандарт України;

ІС – інформаційна система;

ІТ − інформаційна технологія;

ХНУРЕ – Харківський національний університет радіоелектроніки;

ANSI – American national standards institute;

BPMN (Business Process Model and Notation) – система умовних позначень (нотація) для моделювання бізнес-процесів;

CI – Configuration item;

DFD (Data Flow Diagram) – діаграма потоків даних;

ERD (Entity-Relationship Diagrams) – діаграма «сутність-зв’язок»;

ICAM (Integrated Computer Aided Manufacturing) – комплексне автоматизоване виробництво;

IDEF (Integrated DEFinition or ICAM Definition) – методологія функціонального моделювання;

IEEE – Institute of Electrical and Electronics Engineers;

IP – Internet Protocol;

ISO – International Organization for Standardization;

PMBOK – Project Management Body Of Knowledge;

TCP – Transmission Control Protocol;

UPGA (unweighted pair-group average) – метод середнього зв’язку;

UPGC (unweighted pair-group centroid)– метод центроїда;

WPGA (weighted pair-group average) – метод середньозваженого парно-групового середнього;

WPGC (weighted pair-group centroid) – метод зважених центроїдів або медіан.

ВСТУП

Забезпечення якісного процесу розробки ІТ-продукту вимагає правильної постановки задачі команді розробників. Для чого одним з найбільш важливих процесів виступає аналіз конфігурації ІТ-продукту, оскільки базується на розподілі опису архітектури на елементи конфігурації (завдання), що будуть надаватися команді для розробки.

Актуальність роботи визначається тим, що хоча аналіз конфігурації ІТ-продукту й виділяється багатьма стандартами як важливий, практичних рекомендацій щодо його реалізації та впровадження немає.

Об’єктом дослідження кваліфікаційної роботи виступають конфігурації ІТ-продукту.

Метою дослідження є побудова моделей і методів, що описують результати вирішення задачі кластеризації для розподілу опису архітектури ІТ-продукту як окремих завдань між членами команди виконавців. Подібне рішення дозволить скоротити час на аналіз опису архітектури ІТ-продукту та автоматизувати процес його поділу на відповідні завдання та аналітично обґрунтувати обраний варіант конфігурації продукту.

Виконання кваліфікаційної роботи виконується відповідно до вимог методичних вказівок з організації та виконання кваліфікаційної роботи [1] та національного стандарту ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення» [2].

Оформлення переліку джерел посилання відбулося відповідно до національного стандарту ДСТУ 8302:2015 «Бібліографічне посилання. Загальні положення та правила складання» [3].

1 АНАЛІЗ КОНФІГУРАЦІЇ ІТ-ПРОДУКТУ

1.1 Управління конфігурацією ІТ-продукту

1.1.1 Дослідження процесу управління конфігурацією

Конфігурацією продукту зазвичай прийнято називати функціональні та/або фізичні характеристики апаратного чи програмного забезпечення, викладені в технічній документації та досягнуті в продукті [4]. Однак більш відповідним предметній області дослідження буде альтернативне визначення конфігурації як однієї з серій послідовно створених варіацій продукту, в тому числі продуктивності, функціональних та фізичних властивостей існуючого або запланованого продукту або їх поєднання [4].

Варіації продукту також називають версіями робочих продуктів, де власне робочими продуктами можуть бути: вимоги до кінцевого продукту, вихідний код, тести, різна документація та інші допоміжні інструменти. При цьому що саме буде робочим продуктом визначається окремо для кожного конкретного проекту. У ході розробки кожен ІТ-продукт зазнає змін, тому виникає потреба їх врахування, а саме: хто їхній автор, коли вони відбулися і який їхній сенс. Інакше кажучи, необхідно відслідковувати появу версій продуктів, тобто таких станів робочого продукту, які можуть бути відновленими в будь-який момент часу незалежно від історії зміни.

Контроль, облік та інші аспекти управління версіями продукту стають все більш та більш складними. Така тенденція пов’язана з багатьма факторами, серед яких одні з найбільш значимих пов’язані безпосередньо зі складом та характеристиками продукту: тенденція до постійного збільшення розмірів, ускладнення або модифікації архітектури, часті зміни складових ІТ-продукту. Іншими, не менш важливими факторами є ті, що пов’язані з командою проекту: зміни її розмірів, територіальна розподіленість проектної команди, забезпечення можливої передачі відповідальності тощо. Тому для мінімізації можливих проблем при управлінні версіями ІТ-продукту, підвищення рівня

автоматизації та уніфікації процесу декілька міжнародних організацій, включаючи Інститут інженерів з електротехніки та електроніки (IEEE), Американський національний інститут стандартів (ANSI), Міжнародна організація по стандартизації (ISO) та міністерство оборони Сполучених Штатів Америки почали вирішувати проблему управління конфігурацією у процесі розробки програмного забезпечення та висунули власні стандарти. Серед них найбільш широко використовуються стандарти ANSI/IEEE [4].

Керівництво до Зводу знань з управління проектами PMBOK виділяє та розподіляє 47 процесів управління проектом на 10 окремих областей знань, визначає важливі аспекти кожної області та показує їх взаємодію з п’ятьма групами процесів [5]. Серед цих процесів наявний процес інтегрованого контролю змін, що класифікується за областю знань до управління інтеграцією проекту, а за групами процесів управління проектом – до групи процесів моніторингу та контролю.

Інтегрований контроль змін визначається як процес аналізу всіх запитів на зміни, їх схвалення та управління змінами наданих результатів, процесів організацій, документів та планів управління проектом, а також надання інформації про їх стан [5]. Саме в рамках цього процесу виділяється управління конфігурацією, що концентрується на специфікаціях та деталізації наданих результатів і процесів, на відміну від самого контролю змін, що зосереджується на виявленні, документуванні та схваленні чи відхиленні змін документів, наданих результатах або базових планах проекту.

Власне, управління конфігурацією є дисципліною визначення конфігурації системи в окремі моменти часу з метою систематичного контролю за змінами конфігурації та підтримки цілісності та відстеження конфігурації протягом усього життєвого циклу системи [6].

Більш формальне визначення управлінню конфігурацією надає ISO 10007:2017: як діяльність управління, яка застосовує технічні та адміністративні вказівки протягом життєвого циклу продукту та послуги, їхню

ідентифікацію та статус конфігурації та відповідну інформацію про конфігурацію продукту та послуги [7].

Метою управління конфігурацією програмного забезпечення є планування, організація, контроль та координація ідентифікації, зберігання та зміни програмного забезпечення шляхом розробки, інтеграції та передачі[8].

Управління конфігурацією програмного забезпечення формує та підтримує наступні важливі аспекти:

− можливість визначити компоненти програмного забезпечення, в тому числі й ті, що не входять в поточну конфігурацію продукту;

− узгодженість набору компонент, з яких формується версія продукту;

− певний ступінь захищеності компонент продукту від можливих втрат, особливо пов’язаних зі збереженням;

− узгодженість та документальну підтвердженість кожної зміни;

− певний ступінь захищеності змін від втрат, в тому числі пов’язаних з одночасним оновленням;

− можливість повернення до попередньої версії та зміни конфігурації продукту в цілому;

− ведення та збереження історія змін з чітким визначення ролей та зон відповідальності за конкретну зміну.

1.1.2 Дослідження основного понятійного апарату

Основною одиницею усіх робіт з управління конфігурацією є «елемент конфігурації» (Configuration item, CI) [9].CI визначається як сукупність апаратних засобів, програмного забезпечення або того й іншого, яка виступає одиницею для управління конфігурацією та розглядається єдиною сутністю у процесі керування конфігурацією [9]. CI може розглядатися окрема, однак

може бути сформованою з певної сукупності інших СІ, організованих в ієрархію.

Кожен СІ має бути визначеним, зафіксованим та систематично позначеним якомога раніше. Офіційно розглянути елементи конфігурації узгоджуються та визначаються основою для подальшої розробки, після цього будь-які зміни мають вноситися лише з використанням формальних процедур контролю змін. Такі елементи конфігурації формують базову лінію – baseline. Будь-яка розробка ведеться від однієї базової лінії до іншої доти, поки її не можна буде випустити, а всі зміни контролюватися та обов’язково документувати наступні аспекти:

- проблеми чи завдання, що підлягають вирішенню;
- зміни, які пропонуються задля вирішення проблеми чи завдання;
- власне затверджені та внесені зміни.

Виділяють п’ять основних етапів, що виконуються в рамках управління конфігурацією:

- ідентифікація конфігурації та всіх конфігураційних елементів;
- зберігання елементів конфігурації;
- контроль змін конфігурації;
- облік стану конфігурації;
- випуск конфігурації.

Перш за все проводиться ідентифікація, або ж аналіз конфігурації, в рамках якого відбувається визначення СІ та формуються умовні познаки для їх подальшої ідентифікації.

Далі має забезпечуватися надійне та захищене зберігання СІ, щоб дані, програмне забезпечення ні у якому випадку не втратилися, а його цілісність не порушилася. Мінімально для кожного проект визначаються та використовуються наступні бібліотеки програмного забезпечення для зберігання конфігураційних елементів:

- бібліотеки для розробки (динамічні) – для СІ, що на даний момент пишуться або піддаються тестуванню;

− головні бібліотеки (керовані) – для СІ, що формують поточну базову лінію;

− архівні бібліотеки (статичні) – для СІ з попередніх випусків, для цього типу бібліотек СІ не підлягають модифікації.

Оскільки при будь-якій розробці зміни є абсолютно нормальною частиною процесу, то необхідно здійснювати їх контроль задля забезпечення оцінки запропонованих зміни до існуючих елементів конфігурації та подальшої координації та впровадження затверджених змін. Процес керування змінами має аналізувати описи проблем, вирішувати, чи є контрольовані СІ, що необхідно змінити, витягувати їх із керованої бібліотеки, передавати для зміни в динамічну бібліотеку та повертати вже змінені СІ до назад до керованої на зберігання. В рамках процесу кожна окрема зміна має бути перевіреною та задокументованою, навіть якщо надходить група змін, то кожна з них має розглядатися окремо.

З попереднім процесом тісно пов’язаний наступний – облік стану конфігурації програмного забезпечення, що відповідає за запис і звітування інформації, необхідної для ефективного керування конфігурацією. До такої інформації відносять перелік позначень ідентифікації затвердженої конфігурації, статус запропонованих змін до конфігурації та статус впровадження запропонованих змін. Всі статуси обов’язково документуються, а їхній облік триває протягом усього життєвого циклу проекту. Облік статусів конфігураційних елементів є важливим інструментом розробки, оскільки дозволяє завжди отримувати актуально інформацію про стан СІ та всі зміни, що було зроблені.

Завершальний етап наступає коли СІ поширюється за межами проекту, тобто реалізується продукт чи випускається його нова версія. Основою для процесу є базові лінії, які були успішно протестовані, й облікові записи їх статусів, що свідчать про готовність до випуску та беруться до уваги під час прийняття рішення щодо випуску. Результатами процесу є власне сам випуск (реліз) та його опис.

1.1.3 Дослідження основних задач управління конфігурацією

Концепції управління конфігурацією застосовуються до всіх елементів, які підлягають контролю, хоча існують деякі відмінності у реалізації між апаратного та програмного управління.

Основними задачами управління конфігурацією є планування та управлінням процесом управління конфігурацією, ідентифікація конфігурації ІТ-продукту, контроль конфігурації ІТ-продукту, облік стану конфігурації ІТ-продукту, аудит конфігурації ІТ-продукту та керування випуском та доставкою ІТ-продукту. Схематичне зображення задач та їхніх основних процесів наведено на рисунку 1.1 [6].

Управління конфігурацією здійснює:

− контроль над еволюцією та цілісністю продукту, ідентифікуючи його елементи;

− управління та контроль змін; а також перевірку, запис та звітування про конфігураційну інформацію.

Успішне впровадження управління конфігурацією ІТ-продукту вимагає ретельного планування та управління, що в свою чергу, вимагає розуміння організаційного контексту та обмежень, що накладаються на проектування та впровадження цього процесу.

Ідентифікація конфігурації ІТ-продукту здійснює:

− визначення елементів, які підлягають контролю;

− встановлення схем ідентифікації для елементів та їх версій;

− встановлення інструментів та методів, які будуть використовуватися для отримання та керування контрольованими елементами.

Саме результати діяльності цих процесів стають основою для інших видів діяльності управління конфігурацією ІТ-продукту.

Контроль конфігурації ІТ-продукту здійснює:

− визначення змін, які повинні бути внесеними;

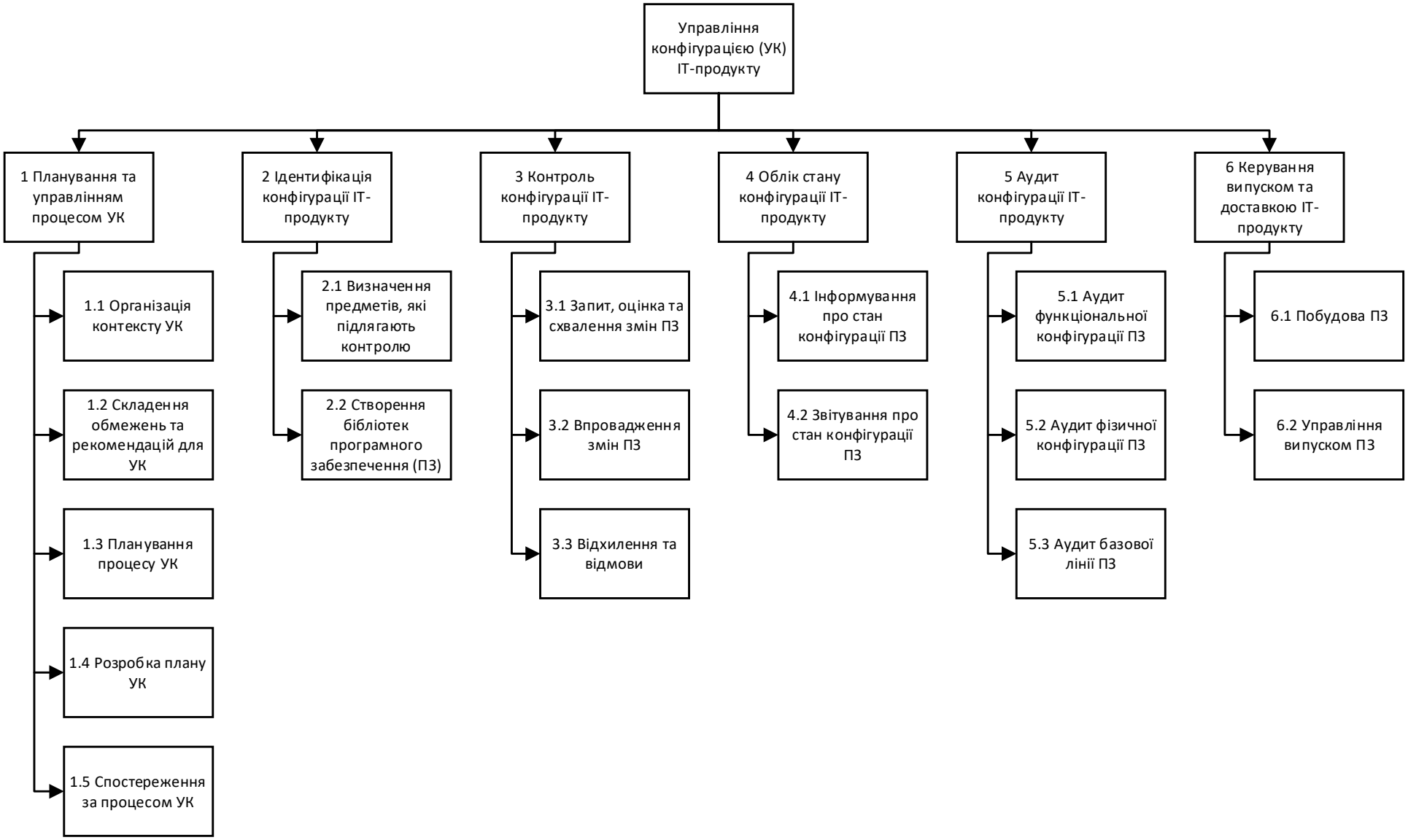

Рисунок 1.1 – Схема основних задач управління конфігурацією та їх головних процесів

− визначення повноважень для схвалення певних змін;

− підтримку впровадження схвалених змін;

− підтримку концепції формальних відхилень чи відмов від вимог проекту.

Ця задача стосується управлінь змінами протягом всього життєвого циклу ІТ-продукту. Тоді як результати діяльності її процесів є корисними для побудови та відслідковування трафіку змін, поломок, а також можливих переробок чи допрацювань елементів конфігурації.

Облік стану конфігурації ІТ-продукту здійснює безпосередньо запис, збереження та формування звітності по інформації, що є необхідною для ефективного керування конфігурацією.

Аудит конфігурації ІТ-продукту визначає ступінь, в якій об’єкт задовольняє необхідним функціональним і фізичним характеристикам. Неформальні аудити такого типу можуть проводитися на ключових етапах життєвого циклу, однак формальні аудити за керівним контрактом можуть вимагати двох типів офіційних аудитів:

− аудит функціональної конфігурації;

− аудит фізичної конфігурації.

Успішне проходження цих аудитів може бути передумовою для встановлення базової лінії ІТ-продукту.

Керування випуском та доставкою ІТ-продукту забезпечує поширення елемента конфігурації ІТ-продукту за межами діяльності розробки в рамках внутрішніх випусків, а також, зовнішніх – коли відбувається розповсюдження елементу серед клієнтів.

На даному етапі досить часто необхідно відтворити конкретні версії та запакувати правильні матеріали для доставки версії (особливо якщо існують різні версії продукту, що є доступними для доставки, як от для різних платформ, або з різною функціональною наповненістю). В такому випадку саме бібліотека програмного забезпечення є ключовим елементом коректного перебігу виконання завдань випуску та доставки ІТ-продукту.

1.1.4 Дослідження окремих особливостей перебігу задачі ідентифікації конфігурації

Ідентифікація конфігурації є основою для подальшого контролю конфігурації програмного забезпечення, оскільки визначає елементи, які підлягають контролю, встановлює схеми ідентифікації для елементів та їх версій, а також, інструменти та методи, які будуть використовуватися для отримання та керування контрольованими елементами [4].

Процес ідентифікації конфігурації обов’язково включає вибір, позначення та опис програмних елементів конфігурації. Вибір передбачає розподіл та групування програмного забезпечення в конфігураційні елементи, які будуть піддаватися управлінню конфігурацією. Позначення відповідає за розробку схеми нумерації або іменування, яка співвідносить компоненти програмного забезпечення та пов’язану з ними документацію. Опис передбачає документування функціональних, продуктивних та фізичних характеристик кожного з компонентів програмного забезпечення.

Попри те, що процес ідентифікації конфігурації є виділеним та формально описаним, рекомендації щодо його впровадження носять більшою мірою методичний та теоретичний характери, а практичних напрацювань надзвичайно мало.

Розглядаючи найпершу вказівку посібника з керування конфігурацією програмного забезпечення Alexis Leon, що звучить як «..Розробіть критерії вибору елементів, які підлягають контролю конфігурації», відразу постає питання категоризації критеріїв, їх ранжування та впливу один на одного. Окрім того виникає питання, що далі робити з виділеними елементами конфігурації: хто їх буде розробляти, як формувати завдання до розробки, скільки і яких елементів конфігурації доцільно включити в окреме завдання, кому видати завдання для розробки тощо.

Саме тому доцільно виділити ці процеси в окремий, який можна назвати аналізом конфігурації ІТ-продукту, адже вже при першому етапі управління конфігурацією виникає потреба в аналітичному вирішенні ряду питань, що виникають на ньому, та в подальшому практичному застосуванні вже при видачі проектній команді завдань (оскільки наступні етапи будуть базуватися на результатах ідентифікації).

1.2 Дослідження методів вирішення задачі аналізу конфігурації ІТ-продукту

Для вирішення задачі аналізу конфігурації ІТ-продукту має застосовуватися максимально простий та швидкий метод, оскільки в сьогоденних умовах час стає дуже важливим чинником. Замовники все більше прагнуть швидкого результату та прагнуть від самої ініціації проекту бути включеними в процес. Виходячи з цього методи, що мають багато підготовчих етапів, або які вимагають попереднього проведення розрахунків іншими методами є недоцільними та надлишковими в умовах даної задачі. До таких можна віднести групу методів Data Mining.

Найбільш застосовним методом з точки зору призначення методів для задачі аналізу конфігурації ІТ-продукту є використання методів, заснованих на вирішенні задачі кластеризації.

Кластерний аналіз, перш за все, застосовується для розбиття певних об’єктів, або груп об’єктів, по набору критеріїв, а не по одному з них. Оскільки аналіз конфігурації та наступна видача конфігураційних елементів до розробки не може базуватися лише на одному критерії, то ряд можливих до застосування методів значно звужується. Базові алгоритми кластеризації поділяють на ієрархічні та неієрархічні.

Ієрархічні алгоритми відштовхуються не від визначення числа кластерів, а орієнтуються на побудову повного дерева вкладених кластерів, яке ще називають дендрограмою. В такому випадку число кластерів визначається з припущень, що незалежні від роботи алгоритмів (наприклад, по динаміці зміни порога злиття кластерів). Такі алгоритми мають ряд добре вивчених проблем, серед яких: проблема вибору мір близькості кластерів, проблема інверсій індексації в дендрограмах, негнучкість ієрархічних класифікацій тощо. Не зважаючи на можливі недоліки, представлення результатів кластеризації у вигляді дендрограми дозволяє отримати найповніше представлення про структуру кластерів. В свою чергу ієрархічні алгоритми діляться на [10]:

− агломеративні алгоритми, в яких побудова кластерів проводиться методом знизу-вверх за рахунок послідовного об’єднання вихідних елементів та відповідним зменшенням числа кластерів;

− дивизимні, або подільні, в яких побудова кластерів проводиться методом зверху-вниз за рахунок послідовного розщеплення груп на кластери.

Неієрархічні алгоритми при вирішенні завдань кластеризації базуються на пошуку оптимального, в умовах конкретних задач, розбиття вихідної множини даних на кластери. Великого застосування знайшли алгоритми побудови розбиття, оскільки вони націлені на групування даних у кластери так, щоб цільова функція алгоритму розбиття досягала екстремуму. Основними алгоритмами, заснованими на методах розбиття, вважаються: алгоритм k-means (hard-c-means), алгоритм Fuzzy C-means та алгоритм кластеризації по Гюстафсону-Кесселю [10].

Окрім того виділяють ще й адаптивні методи кластеризації, котрі націлені на вирішення проблеми вибору найкращого розбиття вихідної множини даних на кластери, що вважається нетривіальною задачею. Для цього в алгоритм кластеризації додають певний адаптивний механізм вибору оптимального рішення серед можливих. Вибір такого рішення основується на якості кластеризації, де останнє означає ступінь наближеності результату

кластеризації до деякого ідеального рішення. Оскільки це ідеальне рішення задачі кластеризації невідоме, то оцінку якості результату можна дослідити або експертним способом, або формальним. Причому перевага віддається саме формальним способам, які оцінюють якість кластеризації за деяким показником, обрахованим з результатів кластеризації. Цей показник визначає оптимальним рішенням те, при якому значення показника досягатиме екстремального значення[10].

1.3 Дослідження інструментів управління конфігурацією

Все більш значимою стає роль інструментів управління конфігурацією в сьогоденних складних та масштабних IT-проектах, а особливо, під час підтримки цілісності та контрольованості середовища розробки програмного забезпечення. До того ж в умовах частої дистанційної роботи, а відповідно, й територіальної роздрібненості, гостро постає питання підтримки злагодженості проектної команди, продуктивності роботи та загального розуміння задач, що вирішуються.

Саме тому використання готових рішень, що пропонують інструменти управління конфігурацією, або управління проектами, що містять подібний функціонал, стають все більше корисними. Однак, просте використання у роботі інструментів, що не підходять для вирішення поставлених задач, або невміле використання інструментів можуть ще більше ускладнити процес управління конфігурацією [4].

Зазвичай інструменти управління конфігурацією програмного забезпечення узагальнено розділяють на три великі класи за масштабом їх застосування:

- інструменти індивідуальної підтримки;
- інструменти підтримки проектів;

− інструменти підтримки корпоративних процесів.

Інструменти індивідуальної підтримки найбільше підходять для невеликих організацій або груп розробників. До таких інструментів відносять[6]:

− інструменти контролю версій (version control tools), що дозволяють відстеження, документування та зберігання окремих елементів конфігурації (наприклад, вихідний код та зовнішню документацію);

− інструменти обробки збірок (build handling tools), що дозволяють збирати та компілювати виконувану версію програмного забезпечення, а до більш просунутих інструментів відносяться ті, що підтримують можливість витягу останньої версії програмного забезпечення для контролю версій, запуск регресійних тестів та створення різних форми звітності;

− інструменти контролю змін (change control tools), що підтримують контроль запитів на зміни та сповіщень про події (наприклад, про зміни статусу запиту на зміни).

Інструменти підтримки проекту забезпечують підтримку управління робочим простором для команд розробників та інтеграторів; добре підходять для підтримки розподілених середовищ розробки. Такі інструменти найбільше підходять для середніх і великих організацій, що мають різні варіанти своїх IT-продуктів і займаються паралельною розробкою [6].

Інструменти підтримки корпоративних процесів зазвичай автоматизують окремі частини корпоративного процесу, забезпечуючи підтримку управління, моніторингу, аудиту та контролю за робочими процесами, ролями та відповідальностями [6].

Було розглянуто та проаналізовано найбільш широко використовувані інструменти управління конфігурацією, до яких відносяться:

− AccuRev від компанії AccuRev, Inc. забезпечує здійснення зв’язку через TCP/IP з використанням власного протоколу, функціонування серверів як командних, відповідальних за безперервну інтеграцію та підтримуючих збірки. Має потокову архітектура, в якій потоки утворюють ієрархічну

структуру змін коду й батьківські потоки передають окремі властивості дочірнім потокам[11];

– Ansible Configuration Tool від компанії Red Hat, Inc.. пропонує модель ресурсів, що орієнтується на бажаний стан комп’ютерних систем і служб, а не на шляхи їх доведення до нього. Дозволяє розробити конфігурацію IT-інфраструктури, уникаючи потенційних збоїв сценаріїв та окремих незворотніх дій [12];

– BitKeeper від компанії BitMover, Inc. є розподіленою масштабованою системою управління конфігурацією IT-продуктів, яка використовує повністю відтворену однорангову модель роботи, що дозволяє повністю відмовитися від використання головного репозиторію[13];

– CFEngine Configuration Tool від компанії Northern.tech AS є інструментом управління конфігурацією, що забезпечує автоматизацію конфігурації для величезних комп’ютерних систем, в тому числі забезпечує уніфіковане керування серверами, системами, користувачами, вбудованими мережевими пристроями, мобільними пристроями та системами[14];

– CHEF Configuration Tool від компанії Progress Software Corporation є системою управління конфігураціями, що застосовує предметно-орієнтовану мову для опису конфігурації. Використовується для спрощення завдань налаштування і підтримки множини серверів, може інтегруватися в хмарні платформи для автоматизації управління процесами нових серверів [15];

– ClearCase (IBM Rational ClearCase) від компанії IBM забезпечує контрольований доступ до програмних активів (коду, вимог, документації, моделей, планів та результатів тестування). Підтримує паралельну розробку, автоматизоване управління робочим простором, базові елементи управління, безпечний контроль версій, аудит збірок тощо [16];

– ConfigHub від компанії ConfigHub, LLC здатний централізувати конфігурацію для всього стека розробки через приватні, безпечні, версійні репозиторії конфігурації. Орієнтується на управлінні великими та різноманітними наборами конфігурацій для всього коду. За допомогою

визначень контексту дозволяє абстрагувати кожен сегмент програми так, щоб повністю виключати можливе дублювання конфігурації [17];

– JUJU Configuration Tool від компанії Canonical Ltd. орієнтується на зменшення операційних витрат на розробку ІТ-продуктів, надає засоби швидкого розгортання, налаштування, масштабування, інтеграції та виконання операційних завдань в публічних і приватних хмарних сервісах[18];

– IBM Rational Synergy від компанії IBM забезпечує управління конфігурацією ІТ-продукту для всіх артефактів, пов’язаних з розробкою (вихідний код, документи, зображення, остаточний ІТ-продукт та бібліотеки). Надає репозиторій для інструменту контролю змін – Rational Change [19];

– Puppet Configuration Tool від компанії Puppet забезпечує централізацію та автоматизацію управління конфігурацією, а також, для налаштування, керування, розгортання різноманітних програм і служб [20];

– SolarWinds Server Configuration Monitor від компанії SolarWinds Worldwide, LLC. забезпечує моніторинг конфігурації сервера для виявлення несанкціонованих змін конфігурації серверів і програм [21];

– Surround SCM for Helix ALM від компанії Perforce Software, Inc. використовується для розробки та запуску програм, включно з інструментами контролю версій, управління веб-сховищами, співпраці з розробниками, керування життєвим циклом додатків, серверами веб-додатків, інструментами налагодження та програмним забезпеченням Agile для планування [22];

– ZENworks Configuration Management від компанії Micro Focus допомагає автоматизувати та підтримувати процеси управління конфігурацією протягом життєвого циклу всіх пристроїв у фізичному, віртуальному та хмарному середовищах [23].

Дані засоби надають широкий функціонал можливостей з управління конфігурацією, однак окремих вбудованих алгоритмів, що дозволяли б аналітично підходити до розподілу опису архітектури ІТ-продукту на окремі СІ як завдання до розробки – немає.

1.4 Постановка задачі дослідження

Об’єктом дослідження в рамках магістерської кваліфікаційної роботи є конфігурації ІТ-продукту.

Предметом дослідження виступають моделі та методи аналізу конфігурації ІТ-продукту.

Метою дослідження є побудова моделі і методів, що описує результати рішення задачі кластеризації для розподілу завдань, як окремих частин ІТ-продукту, між членами команди виконавців. Це дає можливість скоротити час на аналіз опису архітектури ІТ-продукту та автоматизувати процес його поділу на відповідні завдання.

Для досягнення поставленої мети пропонується вирішити такі задачі дослідження:

- провести аналіз предметної галузі та сучасного стану проблеми;
- провести пошук та виділення критерію (чи критеріїв) за яким (якими) буде відбуватися розподіл завдань, як окремих частин ІТ-продукту, між членами команди виконавців;
- дослідити особливості використання математичного апарату задачі кластеризації для вирішення задачі аналізу конфігурації ІТ-продукту;
- провести апробацію результатів та порівняти ефективність автоматизованого розподілу завдань з ручним.

2 РОЗРОБКА МОДЕЛЕЙ АНАЛІЗУ КОНФІГУРАЦІЇ ІТ-ПРОДУКТУ

2.1 Формальна постановка задачі кластеризації

2.1.1 Загальна формальна постановка задачі кластеризації

Зазвичай при вирішенні задачі кластерного аналізу передбачається, що набір з m об'єктів множини $X = \{ x_1, x_2, \ldots, x_m \}$ задано.

Для різних цілей аналізу набір об'єктів може характеризуватися вбудованим або реляційним представленням.

При вбудованому представленні кожен об'єкт описується n-вимірним вектором $y_i = (y_{i1}, y_{i2}, \ldots, y_{in})^T$, де y_{ij} позначає значення j-ої ознаки об'єкта x_i. Вектор y_i називають вектором ознак або зображенням.

Таким чином, предмет кластерного аналізу складається не стільки з вихідної множини об'єктів X, скільки з її представлення, заданого через матрицю $Y = (y_1, y_2, \ldots y_m)^T$, і-й рядок якої є вектором ознаки, що описує і-й об'єкт. З огляду на те, що об'єкту x_i відповідає і-ий рядок матриці Y, поняття «об'єкт» використовується для позначення як елемента $x_i \in X$, так і вектора значень ознак y_i, що характеризують цей об'єкт. У статистиці вектор y_i називається (n-вимірним) спостереженням [24].

Незважаючи на те, що значення окремих вимірів значень ознак можуть бути виражені в різних шкалах (номінальних, порядкових чи часткових), більшість практичних і теоретичних результатів отримуються виходячи з припущення, що компоненти векторів y_i є дійсними числами ($y_i \in R$).

При реляційному представленні, як альтернативи вбудованому, передбачається, що інформація про набір даних надається у вигляді матриці S розмірності m×m, елементи якої s_{ij} представляють схожість (або відмінність) для пар об'єктів x_i, x_j. Використовуючи таке представлення можна відмовитися від вимоги обов'язково мати ознаки, що описують об'єкти, виміряні на кількісних шкалах [24].

Роль класичного кластерного аналізу полягає в тому, щоб розбити набір об'єктів (спостережень) на $k \leq m$ груп $C = \{C_1, C_2, \ldots, C_k\}$, де кожна i-та група C_i називається кластером. Такий поділ має відповідати трьом вимогам [24, 25]:

− кожен виділений кластер повинен містити принаймні один об'єкт, тобто $C_j \neq \emptyset$, де $j = \overline{1..k}$;

− кожен об'єкт повинен належати до певного виділеного кластера, тобто $\bigcup_{j=1}^{k} C_j = X$;

− кожен об'єкт повинен належати рівно одному виділеному кластеру, тобто $C_{j_1} \cap C_{j_2} = \emptyset$, де $j_1 \neq j_2$.

Зокрема, коли $k = m$, то кожен кластер містить рівно один елемент із множини X, це розбиття вважається тривіальним, тому, зазвичай, розглядаються випадки, коли k набагато менше за m.

Щоб мати можливість кількісно визначити зв'язки між парами об'єктів, вводиться міра подібності $s : X \times X \rightarrow R$ або несхожості. Ці дві міри є подвійними, тобто чим нижче значення несхожості, тим більше схожі два порівнюваних об'єкта. Конкретним прикладом несхожості є відстань (метрика), тобто функція $d : X \times X \rightarrow R_+ \cup \{0\}$, що задовольняє наступним умовам [24, 25]:

− $d(x, y) \geq 0$, для всіх x, y;

− $d(x, y) = 0$ тоді і тільки тоді, коли $x \equiv y$;

− $d(x, y) = d(y, x)$ (симетрія);

− $d(x, y) \leq d(x, z) + d(z, y)$, тобто нерівність трикутника, для довільних x, $y, z \in X$.

Наприклад, якщо d_{max} позначає максимальне значення відстані між парами об'єктів із множини X, то відстань можна перетворити на міру подібності (близькості) $s(y_i, y_j) = d_{max} - d(x_i, x_j)$. Отримана таким чином міра близькості досягає максимальних значень, коли $i = j$ (об'єкт x_i тотожний самому собі), а чим менше значення цього показника, тим менше взаємно подібні (більш несхожі) порівнювані об'єкти.

2.1.2 Міри подібності при порівнянні об’єктів, що мають кількісні ознаки

Коли всі ознаки, які використовуються для опису об’єктів із множини X, є кількісними, то кожен об’єкт $x_i \in X$ ототожнюється з n-вимірним вектором $y_i = (y_{i1}, y_{i2}, \ldots, y_{in})^T$. Для об’єктів з кількісними ознаками найпопулярнішими мірами подібності (близькості) є [24]:

– Евклідова відстань, що визначається за наступною формулою:

$$d(x_i, x_j) = \| y_i - y_j \| = \sqrt{\sum_{l=1}^{n}(y_{il} - y_{jl})^2}, \quad (2.1)$$

представляє собою геометричну відстань в багатовимірному просторі;

– квадрат Евклідової відстані, що визначається за наступною формулою:

$$d(x_i, x_j) = \| y_i - y_j \| = \sum_{l=1}^{n}(y_{il} - y_{jl})^2, \quad (2.2)$$

дозволяє додати більше ваги більш віддаленим один від одного об’єктам;

– відстань Мінковського, що визначається за наступною формулою:

$$d_p(x_i, x_j) = \| y_i - y_j \|_p = \left[\sum_{l=1}^{n} |y_{il} - y_{jl}|^p\right]^{\frac{1}{p}}, \text{ де } p \geq 1, p \in R. \quad (2.3)$$

Також, існують окремі випадки значень p при яких відстань Мінковського прирівнюється до інших, так при $p = 1$ можна отримати відстань Манхеттена, при $p = 2$ – Евклідову відстань, а при $p = \infty$ – відстань Чебишева;

– відстань Манхеттена або відстань міського кварталу, що визначається за наступною формулою:

$$d_1(x_i, x_j) = \| y_i - y_j \|_1 = \sum_{l=1}^{n} |y_{il} - y_{jl}|, \quad (2.4)$$

представляє собою середнє різниць по координатам;

– відстань Чебишева, що визначається за наступною формулою:

$$d_\infty(x_i, x_j) = \| y_i - y_j \|_\infty = \max_{l=\overline{1..n}} |y_{il} - y_{jl}|, \quad (2.5)$$

дозволяє розрізнити два об’єкти, якщо вони відрізняються по котрійсь з координат;

− відстань Махаланобіса, що визначається за наступною формулою:

$$d_s(x_i, x_j) = \sqrt{(y_i - y_j)^T S^{-1} (y_i - y_j)}, \quad (2.6)$$

де коваріаційна матриця S^{-1} розраховується наступним чином:

$$S^{-1} = \frac{1}{m} \sum_{i=1}^{m} (y_i - \bar{\mu})\,(y_i - \bar{\mu})^T, \quad (2.7)$$

а $\bar{\mu}$ – є вектором середніх значень та визначається як:

$$\bar{\mu} = \frac{1}{m} \sum_{i=1}^{m} y_i. \quad (2.8)$$

2.1.3 Міри подібності при порівнянні об’єктів, що мають якісні ознаки

З окремих причин, пов’язаних із можливостями обробки, кожен об’єкт з множини X може бути представлений вектором y, але з наступною відмінністю: його компоненти інтерпретуються більше як мітки. Наприклад, якщо такою ознакою є колір, припустимо, книги, то ця ознака може набувати таких значень як: 1–блакитний, 2–зелений тощо.

Важливо, що для таких міток може не існувати природного порядку значень. Тому можна використовувати як міру подібності узагальнену відстань Хеммінга [24]:

$$d_H(x_i, x_j) = \sum_{l=1}^{n} |x_{il} - x_{jl}|, \quad (2.9)$$

що дорівнює кількості цих ознак, значення яких для порівнюваних об’єктів різні.

У випадку змішаних даних, коли частина ознак має якісний характер, а частина – кількісний характер, можна застосувати так званий коефіцієнт Гауера, який є зваженою сумою часткових коефіцієнтів дивергенцій δ(i, j, l), що визначається для кожної ознаки об’єктів y_i і y_j [24].

Для якісної (або номінальної) ознаки береться $\delta(i, j, l) = 1$ тоді, коли значення цієї ознаки для обох об'єктів є різним, а у протилежному випадку приймається $\delta(i, j, l) = 0$.

Тоді як для кількісної ознаки з інтервалом значень $[x_l^{min}, x_l^{max}]$ значення суми часткових коефіцієнтів дивергенцій $\delta(i, j, l)$ розраховується за наступною формулою [24]:

$$\delta(i, j, l) = \frac{|x_{il} - x_{jl}|}{x_l^{max} - x_l^{min}}. \quad (2.10)$$

Тоді відстань між двома об'єктами визначається наступним чином [24]:

$$d_m(y_i, y_j) = \frac{\sum_{l=1}^{n} w(i,j,l)\ \delta(i,j,l)}{\sum_{l=1}^{n} w(i,j,l)}, \quad (2.11)$$

де $w(i, j, l)$ – вага, що дорівнює нулю, коли одне зі значень y_{il}, y_{jl} не спостерігалося, або у випадку асиметричної бінарної ознаки, коли для одного з порівнюваних об'єктів значення цієї ознака дорівнює нулю. В усіх інших випадках $w(i, j, l) = 1$.

У випадку рекомендаційних систем, коли метою є запропонувати набір найбільш зрозумілого та відповідного товару (об'єкту) використовують міри подібності $s(y_i, y_j)$ між уподобаннями конкретного користувача y_i й уподобань інших користувачів y_j, де $j = \overline{1..m}$, $j \neq i$. Однією з найбільш часто застосовуваних мір подібності в даному випадку є модифікований коефіцієнт кореляції Пірсона [24]:

$$r(y_i, y_j) = \frac{(y_i - \bar{y}_i) * (y_j - \bar{y}_j)}{||y_i - \bar{y}_i|| * ||y_j - \bar{y}_j||}, \quad (2.12)$$

де $\bar{y}_i$ позначає середнє значення вектора y_j. Модифікація стосується чисельника наведеного вище виразу: при підсумовуванні відповідних продуктів враховуються лише ті компоненти векторів, які представляють загальні оцінки порівнюваних користувачів.

Замість кореляції Пірсона можна застосувати й інші міри кореляції, адаптовані до характеру ознак, що використовуються при описі об'єктів. Найпопулярнішими варіантами, які застосовуються для якісних ознак, є кореляції Спірмена або Кендалла.

2.2 Дослідження алгоритмів вирішення задачі кластеризації придатних для вирішення задачі аналізу конфігурації ІТ-продукту

2.2.1 Дослідження та вибір класу алгоритмів кластеризації

Вихідними даними для вирішення задачі аналізу конфігурації є операційний опис архітектури ІТ-продукту у вигляді візуальної моделі з потоками робіт та даних, а також, структурний опис у вигляді сутностей та зв'язків між ними. Опис у вигляді візуальної моделі може бути виконаним, наприклад, з використанням:

− графічного представлення для визначення бізнес-процесів у моделі бізнес-процесу Business Process Model and Notation (BPMN);

− сімейства стандартів Integrated Definition або ICAM Definition (IDEF) для вирішення задач моделювання складних систем;

− методології графічного структурного аналізу й проектування інформаційних систем Data Flow Diagrams (DFD);

− діаграм різновидів використання, які ще називають Use Case діаграмами тощо.

Опис архітектури ІТ-продукту має бути попередньо підготовленим, в тому числі позбавленим дублюючих один одного функцій. Особливості моделей, методів та інформаційної технології розробки архітектури складних інформаційних систем на основі функціональних вимог, а також окремо, вибору раціональної архітектури інформаційної системи було досліджено та розроблено М. В. Євлановим у роботах [26-30].

Попередньо підготовлений операційний опис в поєднанні зі структурним описом архітектури ІТ-продукту необхідно розподілити на завдання, які можна давати проектній команді на розробку, до того ж так, щоб їх можна було реалізовувати паралельно та не залежати від кількості проектних команд (оскільки їх значення може змінюватися та не завжди є

відомим наперед). Отримуємо досить нетривіальну задачу, яку потрібно вирішувати ще на етапі ініціації проекту.

В таких умовах завдання, що будуть виділятися з опису архітектури ІТ-продукту, розглядаються як СІ, що й будуть формувати конфігурацію ІТ-продукту. Однак, що доцільніше приймати за СІ: окремий клас, функцію, модуль, підсистему тощо.

Обирати окремий клас не є хорошим підходом, оскільки значно підвищується складність тестування: спочатку потрібно протестувати окремий клас у відриві від інших, а також, забезпечити інтеграцію з іншими класами та знову протестувати їх роботу.

Вибір модуля чи підсистеми, також, є спірним рішенням, оскільки такий СІ є занадто укрупненим та все рівно буде поділятися на більш менші СІ, отже проблема аналізу конфігурації буде вирішена не повністю (адже питання комбінації СІ всередині модуля чи підсистеми буде залишатися відкритим).

Золотою серединою виступає саме функцію, оскільки вона виступає як цілісний СІ, який може безпосередньо стати частиною базової лінії, не потребуючи додаткового поділу на СІ. При цьому тестування конкретної функції є досить тривіальним завданням, що тільки полегшить контроль якості ІТ-продукту.

Виходячи з основних особливостей вирішуваної задачі необхідно дослідити та обрати клас методів кластеризації, що буде найбільш повно підходити. Базуючись на результатах дослідження методів вирішення задачі аналізу конфігурації ІТ-продукту та на дослідженні особливостей самої задачі можна стверджувати, що раціональним стане вибір саме алгоритмів ієрархічної кластеризації. Алгоритми ієрархічної кластеризації найбільш повно задовольняють вимогам, оскільки:

− є чіткими алгоритмами (кожен СІ буде належати тільки одному кластеру);

− не накладають обмежень на форму вихідних кластерів (форма кластерів довільна);

− формують повне дерево об’єднань кластерів (можна отримати розбиття на довільну кількість кластерів, а для вибору раціональної кількості кластерів в умовах конкретних проектів немає потреби запускати алгоритм додаткові рази).

2.2.2 Дослідження та вибір алгоритму ієрархічної кластеризації

Оскільки алгоритми ієрархічної кластеризації поділяються на агломеративні та дивизимні, то необхідно дослідити найпопулярніші з існуючих алгоритмів та обрати той, що буде найбільш повно підходити для вирішення задачі аналізу конфігурації ІТ-продукту.

Агломеративні методи починаються з виділення набору об’єктів, кожен з яких розглядається як окремий кластер. Кластери об’єднуються відповідно до зменшення ступеня подібності (або збільшення ступеня несхожості) до тих пір, доки не буде створено єдиний кластер.

В загальному вигляді агломеративні методи мають наступний алгоритм кластеризації [24, 25]:

а) сформувати множину з m окремих одноелементних кластерів зі всієї множини об’єктів вирішуваної задачі наступного виду:

$$c_1 = \{i_1\}, c_2 = \{i_2\}, \ldots, c_p = \{i_p\}, c_q = \{i_q\}, \ldots, c_m = \{i_m\}; \qquad (2.13)$$

б) обрати пару кластерів, найближчих один до одного та об’єднати в один кластер. Скласти нову множину з m-1 кластерів наступного виду (наприклад, коли c_p і c_q є найближчими кластерами):

$$c_1 = \{i_1\}, c_2 = \{i_2\}, \ldots, c_{pq} = \{i_p, i_q\}, \ldots, c_m = \{i_m\}; \qquad (2.14)$$

в) повторювати крок (2) поки не буде сформовано один єдиний кластер з усіх m кластерів.

Для виконання об’єднання пари найближчих один до одного кластерів існує декілька методів розрахунку відстаней, серед виділяються наступні [25]:

− метод найближчого сусіда (single linkage): відстань між двома кластерами дорівнює відстані між двома найближчими елементами з різних кластерів. Отримані кластери утворюють довгі «ланцюжки»;

− метод найдальшого сусіда (complete linkage): відстань між двома кластерами дорівнює відстані між двома найвіддаленішими об’єктами, що належать до різних кластерів. Цей метод найбільш доцільний, коли реальні об’єкти утворюють добре розділені та компактні кластери;

− метод середнього зв’язку (unweighted pair-group average, UPGA): відстань між двома кластерами дорівнює середній відстані між усіма парами об’єктів, що належать до обох розглянутих кластерів;

− метод середньозваженого парно-групового середнього (weighted pair-group average, WPGA): подібний до попереднього методу, але обчислення проводяться з ваговими коефіцієнтами, рівними кількості об’єктів у двох розглянутих кластерах. Цей метод рекомендується до використання, коли кластери мають чітко різну кількість об’єктів;

− метод центроїда (unweighted pair-group centroid, UPGC): відстань між двома кластерами дорівнює відстані між їх центроїдами (центрами тяжіння);

− метод зважених центроїдів або медіан (weighted pair-group centroid, WPGC): відстань між двома кластерами розраховується, як в попередньому методі, але з введенням ваг, які дорівнюють кількості об’єктів у розглянутих кластерах;

− метод мінімальної дисперсії Уорда: відстань між двома кластерами дорівнює приросту суми квадратів відстаней між об’єктами та центром кластера, що вийде в результаті їх об’єднання.

Дивизимні методи кластеризації починаються з формування єдиного кластеру з усієї множини елементів m. Кожен крок алгоритму рекурсивно поділяє один з існуючих кластерів на два дочірніх. Такий розподіл проходить

до тих пір, поки всі кластери не стануть складатися з одного об’єкту, або поки не буде досягнуто нульової відмінності кластерів один від одного.

В загальному вигляді дивизимні методи мають наступний алгоритм кластеризації [24, 25]:

а) сформувати один кластер C_1, що складається з усіх m елементів вихідного множини;

б) обрати елемент вихідної множини з найбільшим середнім значенням відстані від інших елементів в кластері. Середнє значення може розраховуватися наступним чином:

$$D_{C1} = \frac{1}{N_{C_1}} \times \sum_{p=1}^{m} \sum_{q=1}^{m} d(i_p, i_q) \forall i_p, i_q \in C_1, p \neq q. \quad (2.15)$$

Видалити елемент з вихідного кластеру C_1 та додати до нового кластеру C_2;

в) обрати елемент кластеру C_1 такий, щоб різниця середньої відстані до елементів кластеру C_2 й середньою відстанню до елементів кластеру C_1 була максимальною. Видалити елемент з вихідного кластеру C_1 та додати до нового кластеру C_2;

г) Повторювати крок (в) поки відповідні різниці середніх не стануть від’ємними. Зупинити алгоритм.

Обираючи поміж цих двох алгоритмів перевагу слід надати саме дивизимним алгоритмам, адже для застосування агломеративних алгоритмів необхідно попередньо виконати задачу декомпозиції опису архітектури IT-продукту на окремі частини, до того ж так, щоб кожен окремий кластер міг призначатися окремій команді виконавців.

За діючими стандартами, описаними в тому числі в керівництві до Зводу знань з управління проектами PMBOK, процес управління людськими ресурсами проекту проходить після процесу управління інтеграцією проекту, в рамках якого вирішується задача аналізу конфігурації IT-продукту.

Оскільки під час вирішення задачі аналізу конфігурації конкретна кількість команд-виконавців може бути невідомою, то найбільш підходящими алгоритмами є ті, що йдуть від одного кластеру та врешті дійдуть до

необхідного поділу опису архітектури ІТ-продукту. До того ж, за умов коли кількість команд відома та обмежена, отримати вирішення задачі дивизимним методом вийде швидше, бо досягнути розбиття єдиного кластеру на n частин вийде за меншу кількість кроків, ніж при об'єднанні однооб'єктних кластерів. Таке рішення буде більш застосовним для малих та середніх проектів, де немає великої кількості команд-виконавців.

2.3 Визначення міри подібності між описами елементів архітектури ІТ-продукту, що порівнюються

Розрахунок відстані в умовах даної задачі краще приймати як змішаний, де порівняння підмножин відбувається із застосування відстані Чебишева, а порівняння по кожній підмножині окремо – з відстанню Хеммінга.

Кожен елемент архітектури ІТ-продукту подається у вигляді множини, що складається з трьох підмножин атрибутів:

- підмножини атрибутів, що описують сутності конкретної роботи (операцію, функцію);
- підмножини атрибутів, що описують сутності вхідних потоків даних;
- підмножини атрибутів, що описують сутності вихідних потоків даних.

Для вхідних та вихідних потоків даних існують відповідні описи, а для опису робіт доцільно брати сукупність сутностей зі структурного опису архітектури ІТ-продукту, з якими робота взаємодіє.

Описи та найменування атрибутів доцільно перевести в цифровий код для того, щоб відійти від роботи з категорійними даними. При цьому атрибуту з унікальним найменуванням присвоюється унікальний номер (наприклад, атрибут «прізвище» – 1). Таким чином можна отримати множини, які складаються з цілих чисел та які піддаються кластеризації.

Оскільки кожна робота описується множиною з трьох підмножин, то постає питання як розраховувати відстані між кластерами. За міру подібності двох кластерів варто брати кількість спільних для обох кластерів атрибутів.

Пропонується застосовувати відстань Хеммінга, однак модифікувати цю відстань та брати суму по модулю 2. Для цієї булевої функції існує правило: при двох рівних операндах результат дорівнює нулю, в інших випадка – одиниці. Фізичний зміст такої модифікації полягає у тому, що при наявності атрибутів в обох кластерах сума по модулю 2 буде рівна нулю, а якщо один атрибут є присутнім в одній підмножині й відсутнім в іншій – то одиниці. Таким чином отримує модифіковану відстань Хеммінга:

$$d_H(x_i, x_j) = \sum_{l=1}^{n} x_{il} \oplus x_{jl}. \quad (2.16)$$

Тоді для кожної пари порівнюваних елементів архітектури (тобто функцій) отримаємо по три числових значення відстаней: сумарної відстані Хеммінга для роботи, сумарної відстані Хеммінга для вхідних потоків цієї роботи та сумарної відстані Хеммінга для вихідних потоків цієї роботи.

Серед отриманих значень відстаней Хеммінга беремо максимальне значення й отримуємо таким чином відстань Чебишева, тобто знаходимо відстань між парою порівнюваних елементів архітектури.

Таке багатоетапне визначення міри подібності двох кластерів є вимушеним оскільки об’єкти для порівняння є складними. До того ж міра подібності визначається з атрибутивного опису з урахуванням того, що кожен вхідний/вихідний потік даних реалізується як програмний інтерфейс ІТ-продукту, а сама задача є бізнес-логікою взаємодії з відповідними інтерфейсами та шаром взаємодії з відповідними таблицями бази даних (БД). В такому випадку робота багато в чому буде визначатися тими сутностями, що будуть взаємодіяти з БД, саме тому для опису роботи й обираємо тільки ті сутності, з якими вона безпосередньо взаємодіє.

2.4 Висновки

В другому розділі кваліфікаційної роботи було досліджено формальну постановку задачі кластеризації, в тому числі розглянуто загальну формальну постановку задачі кластеризації, досліджено зміст та розрахунок мір подібності при порівнянні об’єктів, що мають кількісні та якісні ознаки.

Було проведено дослідження алгоритмів вирішення задачі кластеризації придатних для вирішення задачі аналізу конфігурації ІТ-продукту:

− виходячи з особливостей вирішуваної задачі вирішено обрати ієрархічні алгоритми кластеризації;

− досліджено агломеративні та дивизимні алгоритми кластеризації та встановлено, що для вирішення задачі дослідження раціонально застосовувати саме дивизимні алгоритми;

− розроблено модель багатоетапного визначення міри подібності двох кластерів як пошук сумарних відстаней Хеммінга (із заміною модуля різниці на суму по модулю 2) для кожної підмножини об’єкта окремо та наступне поєднання значень для знаходження відстані Чебишева.

3 РОЗРОБКА МЕТОДУ АНАЛІЗУ КОНФІГУРАЦІЇ ІТ-ПРОДУКТУ

3.1 Опис підготовчого етапу

Нехай існують операційний опис архітектури ІТ-продукту у вигляді візуальної моделі $Arch_{Op}$ та структурний опис – у вигляді схеми БД ІТ-продукту $Arch_{DB}$. Тоді, базовий опис архітектури у загальному вигляді можна подати наступним чином:

$$Arch_{base} = Arch_{Op} + Arch_{DB}. \quad (3.1)$$

Розглядаючи опис архітектури ІТ-продукту як об'єднання CI – CI_i, то його можна описати так:

$$Arch_{config} = \bigcup_{i=0}^{m} CI_i, \quad (3.2)$$

де m – кількість CI.

CI складається з множини описів робіт та пов'язаних з ними описів потоків вхідних та вихідних даних та формується згідно з виразом:

$$CI_i = \{W_i, X_i, Y_i\}, \quad (3.3)$$

де W_i – підмножина описів робіт як наборів сутностей зі структурного опису архітектури ІТ-продукту, з якими безпосередньо взаємодіють конкретні роботи;

X_i – підмножина описів потоків вхідних даних роботи W_i;

Y_i – підмножина описів потоків вихідних даних роботи W_i.

Тоді, для застосування розробленого методу аналізу конфігурації ІТ-продукту на основі дивизимної кластеризації необхідно провести попередню підготовку вихідних даних.

Крок 1. Сформувати множини описів всіх робіт, вхідних та вихідних потоків W, X, Y з $Arch_{Op}$.

Крок 2. Сформувати множину описів всіх сутностей S з $Arch_{DB}$.

Крок 3. Закодувати атрибути множин W, X, Y, S використовуючи простий цифровий код.

Крок 4. Поєднати кожен опис роботи з $Arch_{Op}$ з описами сутностей з $Arch_{DB}$, з якими взаємодіє кожна робота.

Крок 5. Сформувати опис $Arch_{config}$, для цього представити описи робіт та пов'язаних з ними описи потоків вхідних та вихідних даних у вигляді CI.

Після виконання описаних вище кроків підготовчий етап можна вважати завершеним та доцільно переходити безпосередньо до розв'язування задачі кластеризації.

3.2 Опис основного алгоритму аналізу конфігурації IT-продукту

Крок 1. Сформувати кластер C_1 з усіх елементів множини $Arch_{config}$.

Крок 2. Розрахувати відстані між кожним елементом кластеру C_1 за кожною з трьох підмножин окремо використовуючи модифіковану відстань Хеммінга:

$$d_{Hz}(x_i, x_j) = \sum_{l=1}^{n} x_{il} \oplus x_{jl}, \quad (3.4)$$

де z – ідентифікатор підмножини (W, X, Y).

Крок 3. Розрахувати загальну відстань між кожним елементом кластеру C_1 за розрахованими відстанями по кожній з трьох підмножин використовуючи відстань Чебишева:

$$D\ (d_{HW}, d_{HX}, d_{HY}) = \max\ (d_{HW}, d_{HX}, d_{HY}). \quad (3.5)$$

Крок 4. Обрати елемент кластера C_1, у якого найбільша відстань Чебишева D, видалити його з кластера та сформувати новий кластер C_2 з цим елементом. Для двох чи більше елементів з однаковими відстанями обрати той, у котрого сума відстаней Чебишева буде більшою.

Крок 5. Обрати з елементів, що залишилися та не розглядалися, перший за порядком.

Крок 6. Визначити середню відстань Чебишева від обраного елемента до всіх елементів, що залишилися в кластері C_1. Середня відстань Чебишева визначається за наступною формулою:

$$D_{C1} = \frac{1}{N_{C_1}-1} \times \sum_{l=1}^{N_{C_1}-1}(\max(\left|w_{C_1} - w_{C_2 l}\right|,\ \left|x_{C_1} - x_{C_2 l}\right|,\ \left|y_{C_1} - y_{C_2 l}\right|). \quad (3.6)$$

Крок 7. Для обраного елемента визначити середню відстань Чебишева від нього до всіх елементів, що перейшли в кластер C_2.

Крок 8. Відняти від середньої відстані Чебишева, отриманої на Кроці 9, середню відстань Чебишева, отриману на Кроці 9.

Крок 9. Якщо результат виконання Кроку 8 негативний, то залишити обраний елемент в кластері C_1, інакше перенести елемент в новий кластер C_2.

Крок 10. Повторювати виконання Кроків 5-9 до тих пір, поки всі елементи кластера C_1 не будуть розглянуті.

Крок 11. Виконувати кроки 1-10 для кожної пари отриманих кластерів до тих пір, поки кожен кластер не буде складатися лише з одного елемента множини $Arch_{config}$, після чого завершити виконання алгоритму.

3.3 Особливості застосування розробленого методу

Застосування методу проходить в рамках одного з важливих процесів будь-якого IT-проект – ідентифікації конфігурації. Проектний менеджер є тією уповноваженою особою, що буде безпосередньо застосовувати розроблений метод у своїй робочій діяльності.

Отримані результати дозволяють впровадити аналітичний підхід до аналізу конфігурації IT-продукту та подальшого управління нею. Завдяки використанню саме ієрархічного класу алгоритмів вирішення задачі буде подано у вигляді повного дерева розбиття об’єктів на кластери, що дозволяє

обрати базову лінію конфігурації з усіх можливих варіантів так, щоб задовільнити поточним обмеженням в рамках проекту.

Також, отримані результати можна візуалізувати за допомогою дендрограми, що дозволить наочно представити результати так, щоб вони були легкими для візуального сприйняття та зрозумілими для осіб, що не знайомі з безпосереднім процесом розбиття. Окрім того, подібна візуалізація даних дозволяє легко відслідкувати порядок розбиття.

За умов зміни кількості проектних ресурсів (неважливо зменшення чи збільшення) проектному менеджеру не доведеться заново проводити розбиття опису архітектури ІТ-продукту на СІ, достатньо буде проаналізувати існуючі варіанти конфігурацій. При цьому аналіз варіантів буде набагато легшим, аніж без попереднього застосування алгоритму, адже не маючи усі можливі варіанти дуже складно раціонально приймати рішення.

Більш того, проведення такого аналізу значно спрощує процес видачі завдань до розробки виконавцям, адже виділені в рамках базової лінії СІ можна реалізовувати паралельно.

3.4 Висновки

В третьому розділі кваліфікаційної роботи було розроблено метод аналізу конфігурації ІТ-продукту: описано підготовчий етап, основний алгоритм виконання аналізу конфігурації ІТ-продукту та особливості застосування розробленого методу.

В рамках підготовчого етапу необхідно сформувати множини всіх робіт, вхідних та вихідних потоків W, X, Y з $Arch_{Op}$ та множину всіх сутностей S з $Arch_{DB}$, закодувати їхні атрибути в цифровий код, поєднати кожну роботи з множини W з сутностями з множини S, з якими вона взаємодіє.

Основний алгоритм виконання аналізу конфігурації ІТ-продукту описує послідовність кроків, що дозволяє вирішити задачу кластеризації СІ.

В ході алгоритму для кожного CI_i, описаного трьома підмножинами W_i, X_i, Y_i, застосовувалося дві відстані:

− для кожної пари порівнюваних елементів по кожній підмножині розраховувалася модифікована відстань Хеммінга;

− з трьох знайдених відстаней знаходилася відстань Чебишева.

Після виділення першого елемента в інший кластер застосовується середня відстань Чебишева, що дозволяє визначити для кожного з елементів вихідного кластеру до якого з двох кластерів він ближче розташований.

Отримані результати представляють повне дерево розбиття кластерів, можуть бути представленими у вигляді дендрограми та знайдуть практичне застосування як аналітичне обґрунтування доцільності вибору конкретного варіанту конфігурації при роботі проектного менеджера.

4 ПРАКТИЧНЕ ВИКОРИСТАННЯ МЕТОДУ АНАЛІЗУ КОНФІГУРАЦІЇ ІТ-ПРОДУКТУ

4.1 Розробка плану проекту, в рамках якого проходитиме практична апробація методу

Проектом, на прикладі якого буде проводитися практична апробація методу, виступає ІТ-проект розробки задачі «Формування та ведення індивідуального плану викладача» ІС «Кафедра». Об’єктом дослідження й автоматизації, в рамках даного проекту, виступає типова кафедра закладу вищої освіти.

Переважна більшість проектів виконується за моделлю життєвого циклу проекту, що виступає поєднанням двох базових моделей: каскадної (водоспадної) та спіральної. Основними ознаками таких поєднань є часткове чи повне повторення окремих робіт, етапів чи фаз життєвого циклу.

Для даного ІТ-проекту було обрано каскадну модель життєвого циклу та проведено розподіл всіх робіт на чотири фази: ініціацію, планування, виконання та контроль, завершення.

У випадку зміни кількості команд-виконавців модель зазнає мінімальних змін, оскільки між будь якими роботами можна додати часову затримку, або змінити власну тривалість окремих робіт. Однією з альтернатив є винесення робіт фази «Виконання та контроль» в окремий проект та, власне, виконати комбінування каскадної моделі зі спіральною. Окрім того, таке рішення є раціональним ще й у тому випадку, коли правовласник виступає за часті та невеликі релізи ІТ-продукту, щоб мати можливість користуватися мінімальним функціоналом та виявляти потенційні помилки чи неточності.

Результатами виконання побудови плану проекту розробки задачі «Формування та ведення індивідуального плану викладача» є робоча структура, діаграма Ґантта робіт та діаграма Ґантта ресурсів проекту, що наведені на рис. 4.1 – 4.3 відповідно та є отриманими засобами Spider Project.

Уровень	Код	Код WBS	Название	Начало	Окончание	Длительность, Дни	Длительность, Часы
1	ІП	0	**Розробка задачі "Формування та ведення індивідуального плану ро**	**01.09.21 08:00**	**15.10.21 16:00**	**33.00**	**264.00**
2	1	1	**Ініціація**	**01.09.21 08:00**	**13.09.21 16:00**	**9.00**	**72.00**
	1	1.1	Аналіз предметної області	01.09.21 08:00	02.09.21 16:00	2.00	16.00
	2	1.2	Збір та формування вимог правовласників	03.09.21 08:00	07.09.21 16:00	3.00	24.00
	3	1.3	Постановка задачі	08.09.21 08:00	08.09.21 16:00	1.00	8.00
	4	1.4	Розробка та затвердження технічного завдання	09.09.21 08:00	10.09.21 16:00	2.00	16.00
	5	1.5	Оформлення договору про розробку ІТ-продукту	13.09.21 08:00	13.09.21 16:00	1.00	8.00
2	2	2	**Планування**	**14.09.21 08:00**	**24.09.21 16:00**	**9.00**	**72.00**
3	6	2.1	**Розробка варіантів архітектури ІТ-продукту**	**14.09.21 08:00**	**15.09.21 16:00**	**2.00**	**16.00**
	6	2.1.1	Розробка варіантів архітектури ІТ-продукту	14.09.21 08:00	15.09.21 16:00	2.00	16.00
3	7	2.2	**Вибір та опис архітектури ІТ-продукту**	**16.09.21 08:00**	**22.09.21 16:00**	**5.00**	**40.00**
	7	2.2.1	Вибір та узгодження обраного варіанту архітектури ІТ-продукту	16.09.21 08:00	16.09.21 16:00	1.00	8.00
	8	2.2.2	Розробка операційного опису архітектури ІТ-продукту	17.09.21 08:00	20.09.21 16:00	2.00	16.00
	9	2.2.3	Розробка структурного опису архітектури ІТ-продукту	21.09.21 08:00	22.09.21 16:00	2.00	16.00
3	8	2.3	**Ідентифікація конфігурації ІТ-продукту**	**23.09.21 08:00**	**24.09.21 16:00**	**2.00**	**16.00**
	10	2.3.1	Аналіз конфігурації ІТ-продукту	23.09.21 08:00	23.09.21 16:00	1.00	8.00
	11	2.3.2	Розробка плану управління конфігурацією	24.09.21 08:00	24.09.21 16:00	1.00	8.00
2	3	3	**Виконання та контроль**	**27.09.21 08:00**	**14.10.21 16:00**	**14.00**	**112.00**
	12	3.1	Створення бази даних ІТ-продукту	27.09.21 08:00	01.10.21 16:00	5.00	40.00
	13	3.2	Формування базової лінії ІТ-продукту	04.10.21 08:00	04.10.21 16:00	1.00	8.00
	14	3.3	Реалізація завдань згідно з базовою лінією ІТ-продукту (кодування)	05.10.21 08:00	11.10.21 16:00	5.00	40.00
	15	3.4	Створення/доповнення робочої документації	05.10.21 08:00	05.10.21 16:00	1.00	8.00
	16	3.5	Контроль якості виконаних завдань (тестування)	12.10.21 08:00	13.10.21 16:00	2.00	16.00
	17	3.6	Випуск базової лінії ІТ-продукту	14.10.21 08:00	14.10.21 16:00	1.00	8.00
2	4	4	**Завершення**	**15.10.21 08:00**	**15.10.21 16:00**	**1.00**	**8.00**
	18	4.1	Оформлення договору про завершення	15.10.21 08:00	15.10.21 16:00	1.00	8.00

Рисунок 4.1 – Робоча структура проекту

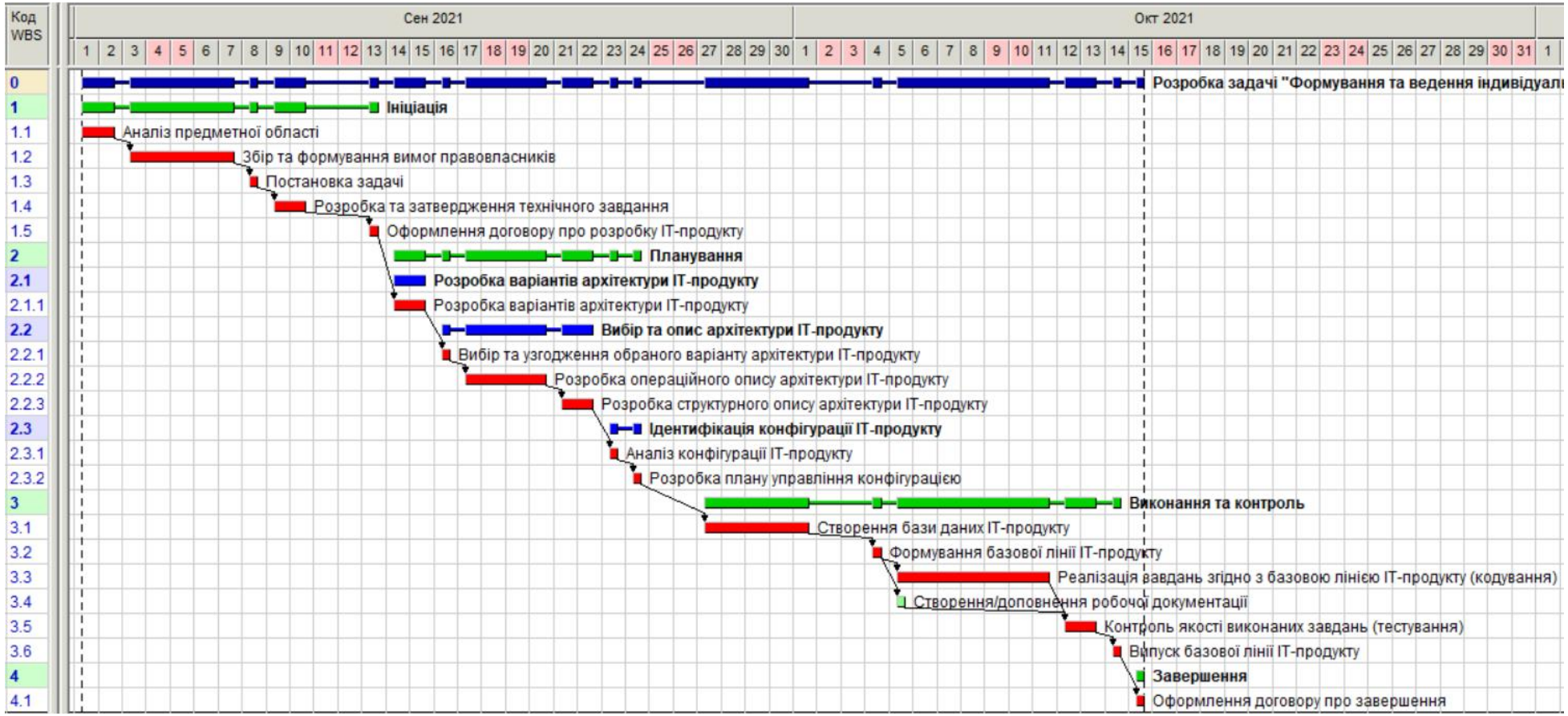

Рисунок 4.2 – Діаграма Ґантта робіт проекту

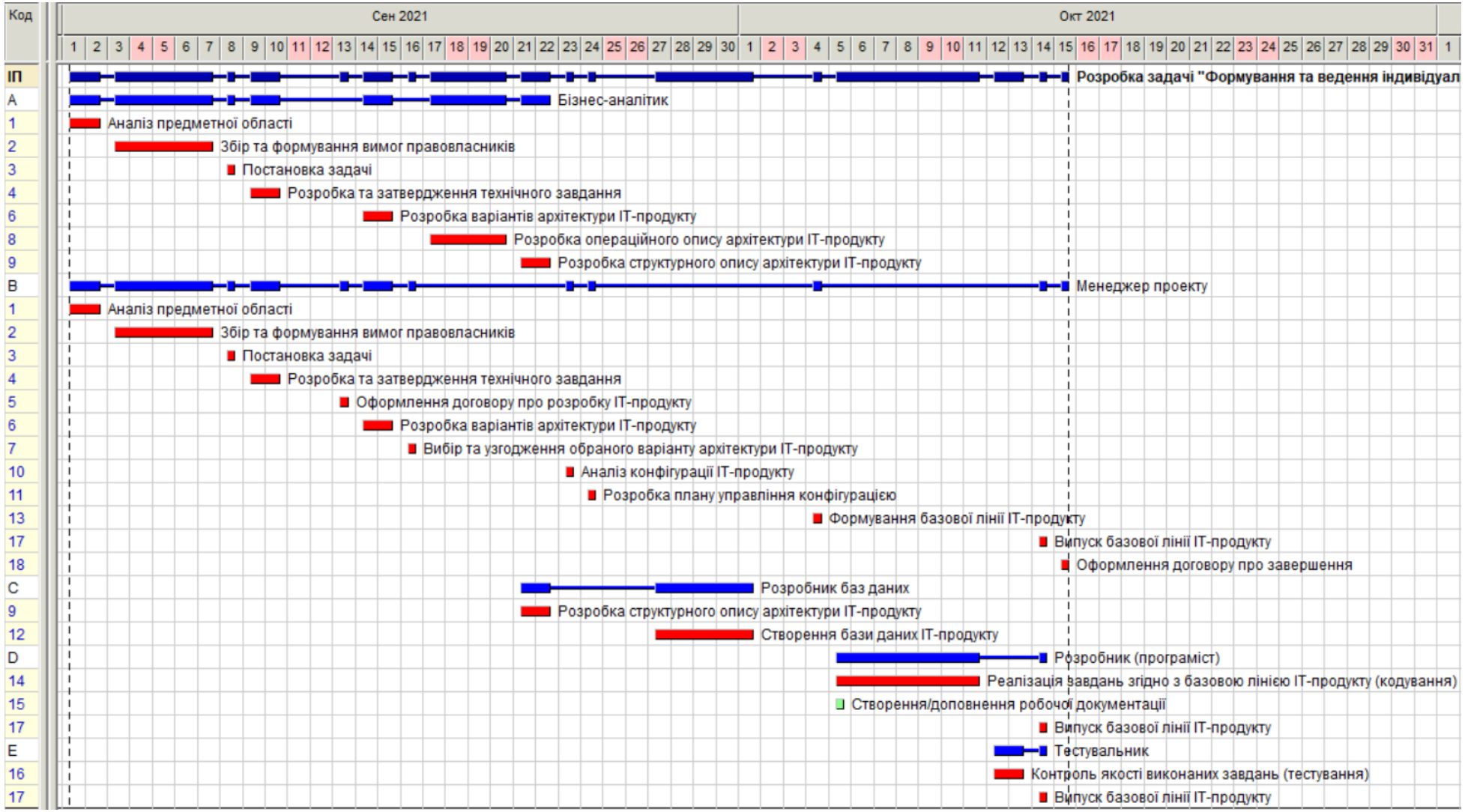

Рисунок 4.3 – Діаграма Ґантта ресурсів проекту

Досліджуючи робочу структуру проекту можна побачити один з важливих етапів фази «Планування», а саме ідентифікацію конфігурації ІТ-продукту. В рамках цього етапу виконується робота «Аналіз конфігурації ІТ-продукту». Саме протягом цієї роботи проектному менеджеру необхідно поєднати операційний опис зі структурним та, найголовніше, провести аналіз конфігурації ІТ-продукту, що стане основою для подальшої розробки плану управління конфігурацією.

4.2 Опис вхідних даних методу

Вхідними даними методу виступають операційний та структурний опис архітектури задачі «Формування та ведення індивідуального плану викладача» (далі архітектури ІТ-продукту).

В умовах даного проекту операційний опис архітектури ІТ-продукту представлений у вигляді функціональної структури ІТ-продукту, що наведена на рисунку 4.4. Побудова якої була проведена з використанням методології графічного структурного аналізу, а саме: діаграм потоків даних (Data Flow Diagram, DFD) засобами інтегрованого програмного пакету AllFusion Process Modeler.

Структурний опис архітектури ІТ-продукту представлений у вигляді схеми логічної структури бази даних, що наведена на рисунку 4.5. Побудова була проведена з використанням методології діаграм «сутність-зв’язок» ERD (Entity-Relationship Diagrams) засобами інтегрованого програмного пакету AllFusion ERwin Data Modeler.

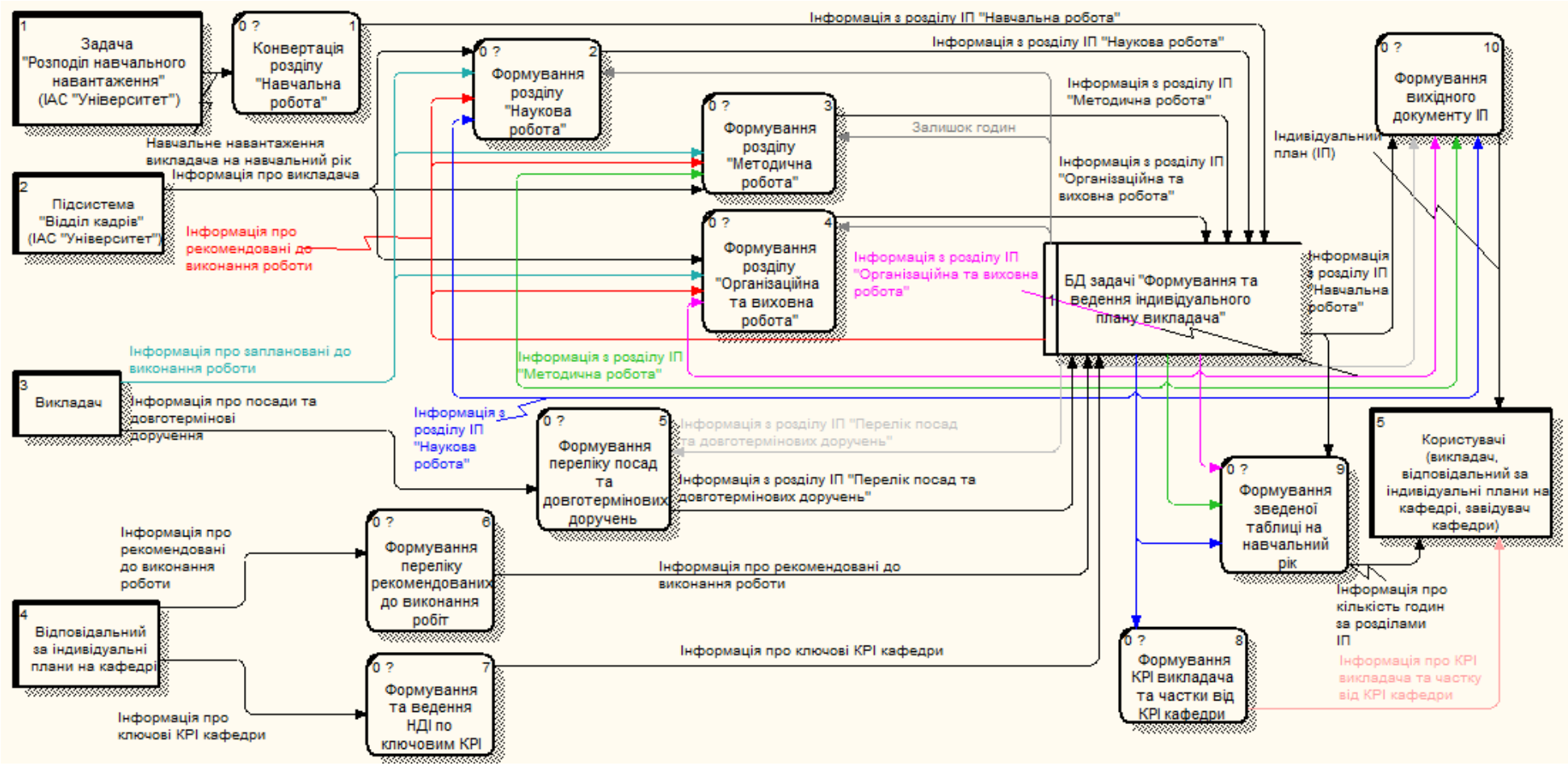

Рисунок 4.4 – Операційний опис архітектури ІТ-продукту у вигляді функціональної структури ІТ-продукту

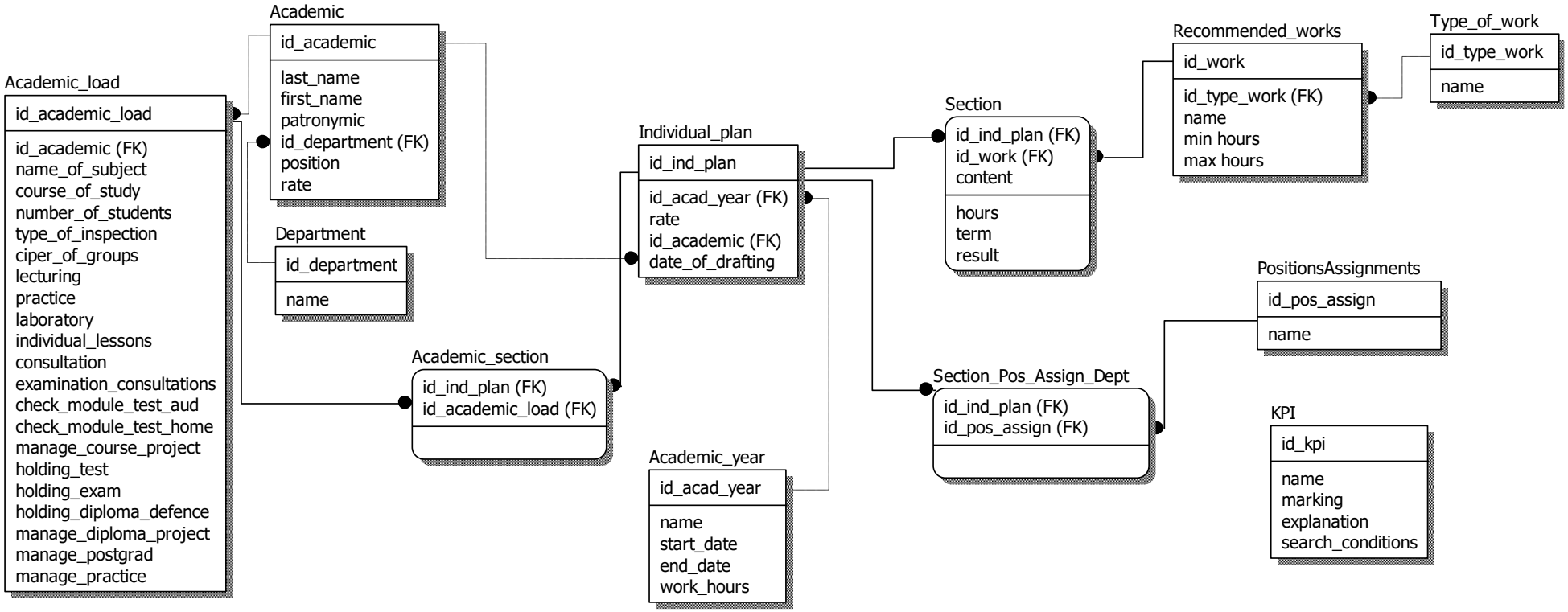

Рисунок 4.5 – Структурний опис архітектури IT-продукту у вигляді схеми логічної структури бази даних

4.3 Виконання підготовчого етапу методу аналізу конфігурації ІТ-продукту

Сформуємо множини всіх робіт, вхідних та вихідних потоків даних W, X, Y з операційного опису архітектури ІТ-продукту $Arch_{Op}$, всіх сутностей S з структурного опису $Arch_{DB}$ та закодуємо їх елементи цифровим кодом. Отримані результати наведено в таблицях 4.1 – 4.4 відповідно.

Таблиця 4.1 – Множина всіх робіт W

Код	Елемент множини
1	Конвертація розділу «Навчальна робота»
2	Формування розділу «Наукова робота»
3	Формування розділу «Методична робота»
4	Формування розділу «Організаційна та виховна робота»
5	Формування переліку посад та довготермінових доручень
6	Формування переліку рекомендованих до виконання робіт
7	Формування та ведення НДІ по ключовим КРІ
8	Формування КРІ викладача та частки від КРІ кафедри
9	Формування зведеної таблиці на навчальний рік
10	Формування вихідного документу ІП

Таблиця 4.2 – Множина вхідних потоків даних X

Код	Елемент множини
1	Навчальне навантаження викладача на навчальний рік
2	Інформація про викладача
3	Інформація про заплановані до виконання роботи
4	Інформація про посади та довготермінові доручення
5	Інформація про рекомендовані до виконання роботи
6	Інформація про ключові КРІ кафедри

Кінець таблиці 4.2

7	Інформація з розділу ІП «Навчальна робота»
8	Інформація з розділу ІП «Наукова робота»
9	Інформація з розділу ІП «Методична робота»
10	Інформація з розділу ІП «Організаційна та виховна робота»
11	Інформація з розділу ІП «Переліку посад та довготермінових доручень»
12	Залишок годин

Таблиця 4.3 – Множина вихідних потоків даних Y

Код	Елемент множини
1	Інформація про рекомендовані до виконання роботи
2	Інформація з розділу ІП «Навчальна робота»
3	Інформація з розділу ІП «Наукова робота»
4	Інформація з розділу ІП «Методична робота»
5	Інформація з розділу ІП «Організаційна та виховна робота»
6	Інформація з розділу ІП «Переліку посад та довготермінових доручень»
7	Інформація про ключові KPI кафедри
8	Інформація про кількість годин за розділами ІП
9	Інформація про KPI викладача та частку від KPI кафедри
10	Індивідуальний план (ІП)

Таблиця 4.4 – Множина всіх сутностей S

Код	Елемент множини
1	Academic_load
2	Academic
3	Department
4	Individual_plan

Кінець таблиці 4.4

5	Academic_section
6	Academic_year
7	Section
8	Recommended_works
9	Type_of_work
10	Section_Pos_Assign_Dept
11	PositionsAssignments
12	KPI

Далі поєднаємо кожну роботу з $Arch_{Op}$ з сутностями з $Arch_{DB}$, з якими вона взаємодіє. Поєднання виконувалося завдяки імпорту сутностей зі структурного опису засобами пакету AllFusion ERwin Data Modeler, подальшого їхнього експорту до операційного опису засобами пакету AllFusion Process Modeler та безпосереднього призначення завдяки вбудованому функціоналу Data Usage Editor. За результатами виконання поєднання було сформовано спеціальний звіт (Data Usage Report), який наведено у додатку А.

Останнім кроком підготовчого етапу є формування опису $Arch_{config}$, результати чого наведено на рисунку 4.6.

Arch config: {{Si}, {Xi}, {Yi}}	
1	{ {1, 2, 3, 5, 6, 4}, {1}, {2} }
2	{ {1, 2, 5, 6, 7, 4, 8, 9, 3}, {12, 8, 2, 3, 5}, {3} }
3	{ {1, 2, 5, 6, 4, 7, 8, 9, 3}, {12, 9, 2, 3, 5}, {4} }
4	{ {1, 2, 5, 6, 4, 7, 3, 8, 9}, {12, 10, 2, 3, 5}, {5} }
5	{ {2, 6, 4, 11, 7, 10}, {11, 4}, {6} }
6	{ {8, 9}, {5}, {1} }
7	{ {12}, {6}, {7} }
8	{ {1, 2, 6, 4, 8, 7, 9, 5, 12}, {8}, {9} }
9	{ {2, 6, 4, 8, 7, 9, 1, 5}, {9, 7, 8, 10}, {8} }
10	{ {2, 6, 4, 8, 7, 9, 1, 5, 11, 10}, {9, 7, 8, 10, 11}, {10} }

Рисунок 4.6 – Елементи опису архітектури IT-продукту як об’єднання CI

4.4 Виконання основного алгоритму аналізу конфігурації IT-продукту

Сформуємо кластер C_1 з усіх елементів множини $Arch_{config}$. Для зручного візуального сприйняття відсортуємо елементи підмножин в порядку зростання. Результати наведено на рисунку 4.7.

C1	
1	{ {1, 2, 3, 4, 5, 6}, {1}, {2} }
2	{ {1, 2, 3, 4, 5, 6, 7, 8, 9}, {2, 3, 5, 8, 12}, {3} }
3	{ {1, 2, 3, 4, 5, 6, 7, 8, 9}, {2, 3, 5, 9, 12}, {4} }
4	{ {1, 2, 3, 4, 5, 6, 7, 8, 9}, {2, 3, 5, 10, 12}, {5} }
5	{ {2, 4, 6, 7, 10, 11}, {4, 11}, {6} }
6	{ {8, 9}, {5}, {1} }
7	{ {12}, {6}, {7} }
8	{ {1, 2, 4, 5, 6, 7, 8, 9, 12}, {8}, {9} }
9	{ {1, 2, 4, 5, 6, 7, 8, 9}, {7, 8, 9, 10}, {8} }
10	{ {1, 2, 4, 5, 6, 7, 8, 9, 10, 11}, {7, 8, 9, 10, 11}, {10} }

Рисунок 4.7 – Елементи кластеру C_1

Далі розрахуємо відстані попарно між кожним елементом кластеру C_1 за кожною з трьох підмножин з використанням модифікованої відстані Хеммінга. Результати розрахунків наведено на рисунку 4.8. Оскільки результати формують матрицю, то матриця є симетричною відносно діагоналі (адже порівняння йде попарно), тому з питань економії часу заповнення нижньої половини матриці не має практичної цінності.

	1	2			3			4			5			6			7			8			9			10		
1		3	6	2	3	6	2	3	6	2	6	3	2	8	2	2	7	2	2	5	2	2	4	5	2	6	6	2
2					0	2	2	0	2	2	6	7	2	6	4	2	10	6	2	2	4	2	1	7	2	3	8	2
3								0	2	2	7	7	2	8	4	2	10	6	2	2	6	2	1	7	2	3	8	2
4											7	7	2	7	4	2	10	6	2	2	6	2	1	7	2	3	8	2
5														8	3	2	7	3	2	7	3	2	6	6	2	4	5	2
6																	3	2	2	8	2	2	7	5	2	9	6	2
7																				8	2	2	9	5	2	11	6	2
8																							1	3	2	3	4	2
9																										2	1	2
10																												

Рисунок 4.8 – Результати розрахунку модифікованої відстані Хеммінга по кожній підмножині

Використовуючи отримані відстані за кожною підмножиною розрахуємо загальну відстань між елементами кластеру C_1 користуючись відстанню Чебишева. Візуально результати розрахунків помічено зеленим кольором та наведено на рисунку 4.9.

	1	2			3			4			5			6			7			8			9			10		
1		3	6	2	3	6	2	3	6	2	6	3	2	8	2	2	7	2	2	5	2	2	4	5	2	6	6	2
2					0	2	2	0	2	2	6	7	2	6	4	2	10	6	2	2	4	2	1	7	2	3	8	2
3								0	2	2	7	7	2	8	4	2	10	6	2	2	6	2	1	7	2	3	8	2
4											7	7	2	7	4	2	10	6	2	2	6	2	1	7	2	3	8	2
5														8	3	2	7	3	2	7	3	2	6	6	2	4	5	2
6																	3	2	2	8	2	2	7	5	2	9	6	2
7																				8	2	2	9	5	2	11	6	2
8																							1	3	2	3	4	2
9																										2	1	2
10																												

Рисунок 4.9 – Результати розрахунку відстані Чебишева

З отриманих відстаней Чебишева обираємо максимальну (11) та вилучаємо відповідний елемент (7, адже для нього сума відстаней Чебишева більша ніж для елемента 11) з кластера C_1 та формуємо новий кластер C_2 з цим елементом. Отримані кластери наведено на рисунку 4.10.

C1	
1	{ {1, 2, 3, 4, 5, 6}, {1}, {2} }
2	{ {1, 2, 3, 4, 5, 6, 7, 8, 9}, {2, 3, 5, 8, 12}, {3} }
3	{ {1, 2, 3, 4, 5, 6, 7, 8, 9}, {2, 3, 5, 9, 12}, {4} }
4	{ {1, 2, 3, 4, 5, 6, 7, 8, 9}, {2, 3, 5, 10, 12}, {5} }
5	{ {2, 4, 6, 7, 10, 11}, {4, 11}, {6} }
6	{ {8, 9}, {5}, {1} }
8	{ {1, 2, 4, 5, 6, 7, 8, 9, 12}, {8}, {9} }
9	{ {1, 2, 4, 5, 6, 7, 8, 9}, {7, 8, 9, 10}, {8} }
10	{ {1, 2, 4, 5, 6, 7, 8, 9, 10, 11}, {7, 8, 9, 10, 11}, {10} }

C2	
7	{ {12}, {6}, {7} }

Рисунок 4.10 – Елементи кластеру C_1 та нового кластеру C_2

Далі, послідовно для кожного елемента кластера C_1 визначаємо середню відстань Чебишева до всіх елементів, що залишилися в кластері C_1. Після чого визначаємо середню відстань Чебишева до всіх елементів, які перейшли в кластер C_2. Знак різниці розрахованих середніх відстаней Чебишева дозволить

дізнатися чи ближчий елемент до вихідного кластера (якщо отримуємо від'ємне значення) чи до нового (якщо отримуємо додатнє).

Отже, першим елементом, що переноситься до кластеру C_2 стає елемент з позначкою 6, бо саме в нього додатна різниця відстаней, що наведена на рисунку 4.11.

Ел. C1	Відстань Чебишева		Різниця
	до ел-тів C1	до ел-тів C2	
1	6	7	-1
2	5,25	10	-4,75
3	5,75	10	-4,25
4	5,625	10	-4,375
5	6,625	7	-0,375
6	7,625	3	4,625

Рисунок 4.11 – Результати розрахунків відстаней Чебишева та їхньої різниці до першої додатної різниці

В результаті переносу даного елемента (6) до кластеру C_2 маємо наступний вигляд розподілу елементів за кластерами, що наведений на рисунку 4.12.

C1	
1	{ {1, 2, 3, 4, 5, 6}, {1}, {2} }
2	{ {1, 2, 3, 4, 5, 6, 7, 8, 9}, {2, 3, 5, 8, 12}, {3} }
3	{ {1, 2, 3, 4, 5, 6, 7, 8, 9}, {2, 3, 5, 9, 12}, {4} }
4	{ {1, 2, 3, 4, 5, 6, 7, 8, 9}, {2, 3, 5, 10, 12}, {5} }
5	{ {2, 4, 6, 7, 10, 11}, {4, 11}, {6} }
8	{ {1, 2, 4, 5, 6, 7, 8, 9, 12}, {8}, {9} }
9	{ {1, 2, 4, 5, 6, 7, 8, 9}, {7, 8, 9, 10}, {8} }
10	{ {1, 2, 4, 5, 6, 7, 8, 9, 10, 11}, {7, 8, 9, 10, 11}, {10} }

C2	
7	{ {12}, {6}, {7} }
6	{ {8, 9}, {5}, {1} }

Рисунок 4.12 – Вигляд розподілу елементів кластеру C_1 та нового кластеру C_2

Знову повторюємо розрахунки середніх відстаней Чебишева та їхньої різниці для нового розподілу елементів за кластерами. За результатами,

наведеними на рисунку 4.12, видно, що більше немає елементів, ближчих до нового кластеру C_2, тому перший розподіл між кластерами завершено.

Ел.	Відстань Чебишева		Різниця
C1	до ел-тів C1	до ел-тів C2	
1	5,714285714	7,5	-1,78571429
2	5,142857143	8	-2,85714286
3	5,428571429	9	-3,57142857
4	5,428571429	8,5	-3,07142857
5	6,428571429	7,5	-1,07142857
8	5	8	-3
9	5,285714286	8	-2,71428571
10	5,857142857	10	-4,14285714

Рисунок 4.13 – Результати розрахунків відстаней Чебишева та їхньої різниці після винесення елементу 6 до кластеру C_2

Продовжуючи розрахунки за алгоритмом далі було проведено ряд розподілів кластерів до тих пір, поки кожен кластер почав складатися лише з одного елементу.

Було отримано наступні результати, представлені у вигляді дендрограми та наведені на рисунку 4.14. Отримана дендрограма дозволяє не лише переглянути порядок розбиття на кластери, але й сам склад кожного з кластерів, що є зручним для подальшого аналізу та застосування отриманих результатів.

4.5 Аналіз отриманих даних

Результатом виконання алгоритму є розгорнута дендрограма, що стає одним з основних вихідних документів, на рівні з описом архітектури IT-продукту, що буде використовуватися проектним менеджером у своїй професійній діяльності.

Отже, наступним кроком після виконання алгоритму стає аналіз отриманих даних в рамках процесу аналізу конфігурації IT-продукту.

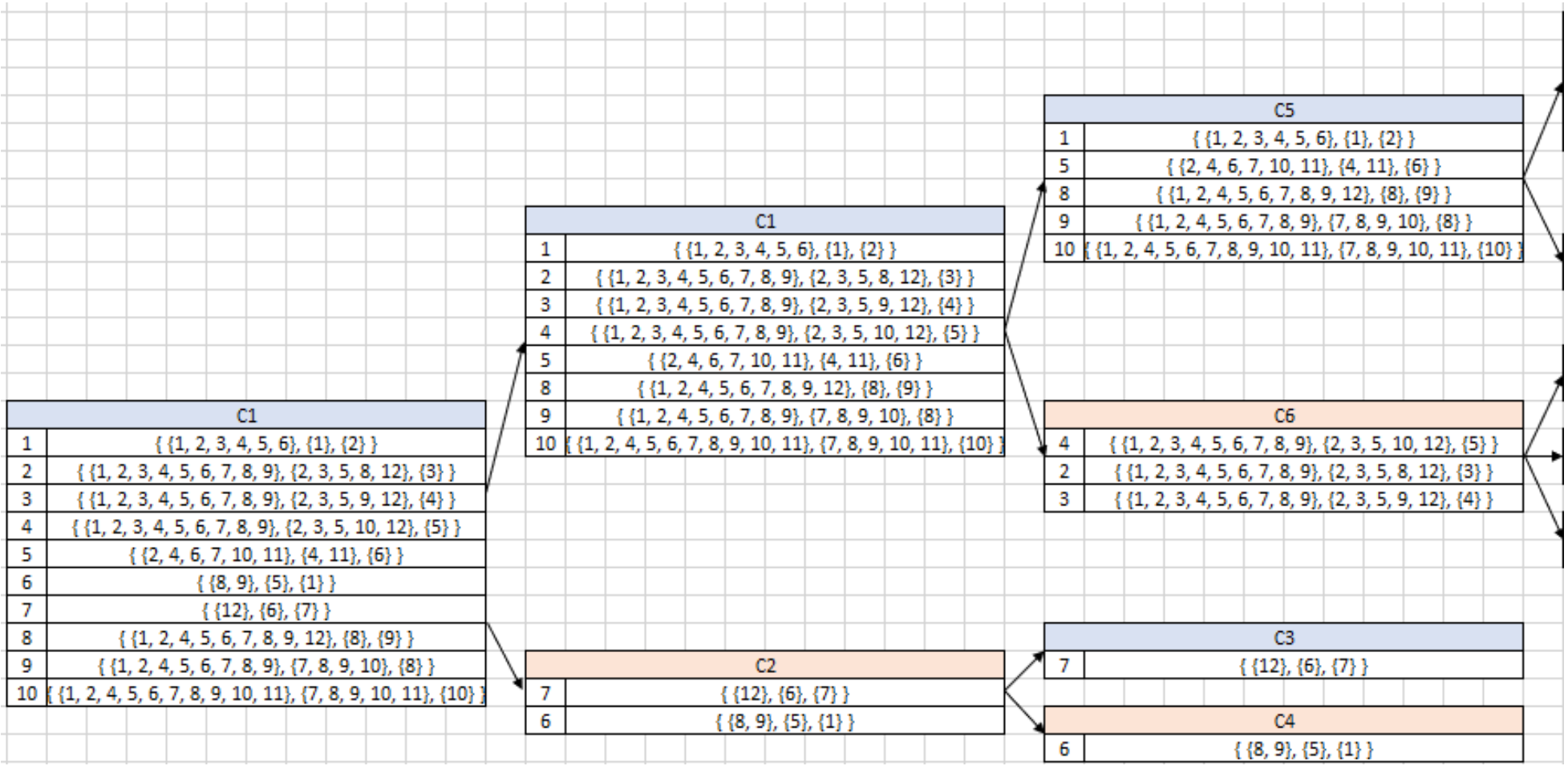

Рисунок 4.14 – Дендрограма повного розбиття кластерів

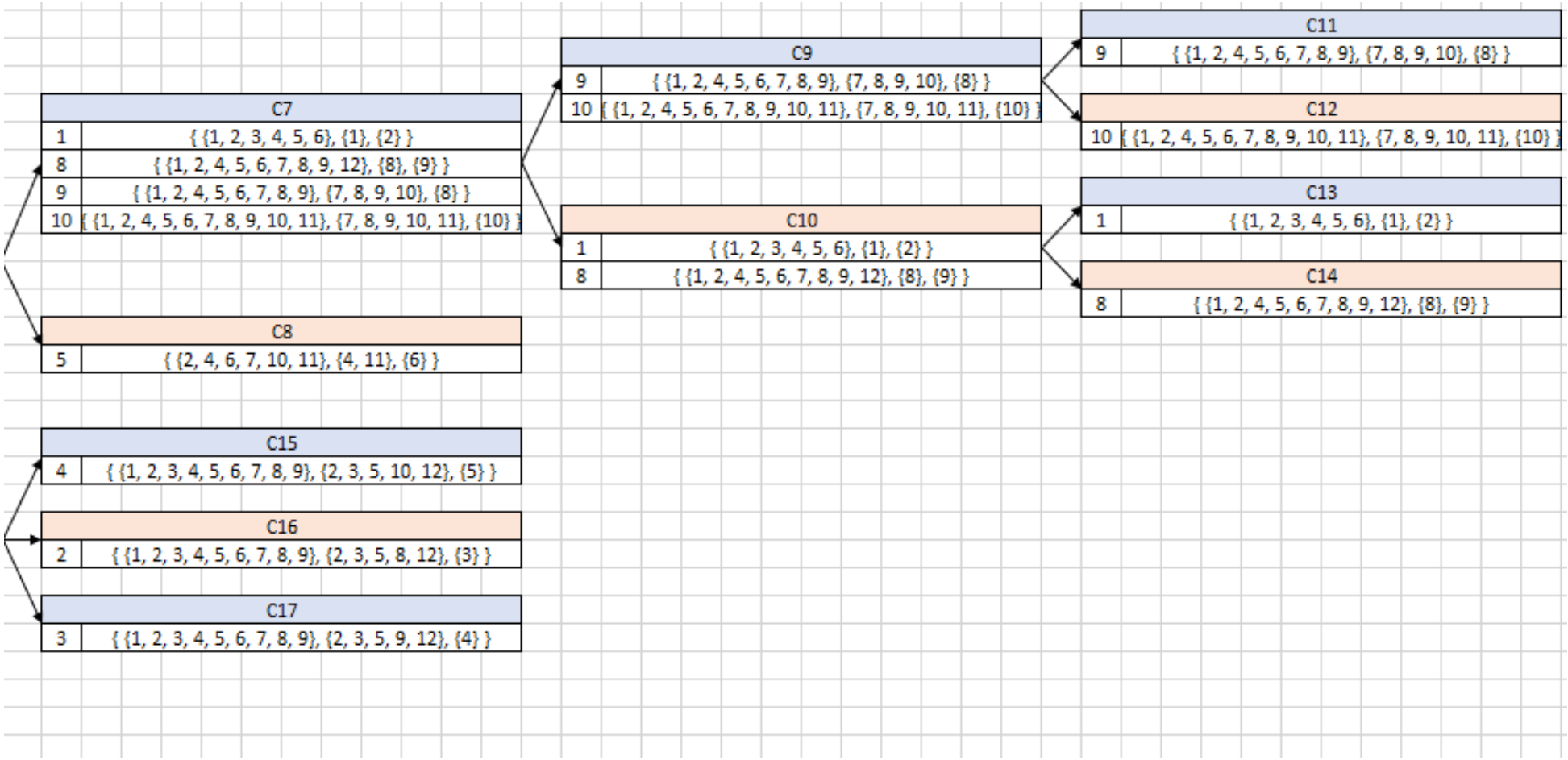

Рисунок 4.14, аркуш 2

Для цього, також, потрібно виконати декодування отриманих даних, а для зручного візуального сприйняття його було проведено безпосередньо на отриманій дендрограмі, а результати наведено на рисунку 4.15. Подібна маніпуляція забезпечує зручність при порівнянні змісту кластерів та дозволяє уникнути постійного звернення до переліку вхідних даних.

Підходи до того як саме формувати конфігурацію ІТ-продукту будуть різнитися від проекту до проекту, однак можна виділити найбільш значимі серед них – це кількість команд-виконавців (або безпосередньо виконавців, якщо в проектній команді багато розробників), між якими потрібно розподілити СІ для розробки та взаємозв'язок самих СІ, їх суть.

Оперуючи останнім критерієм, суттю та взаємопов'язаністю СІ, раціональною конфігурацією ІТ-продукту можна назвати ту, що складається з кластерів C_3, C_4, C_6, C_9 та C_{10}.

Виділення кластерів C_3 та C_4 як окремих СІ обґрунтовується тим, що елементи, формуючі ці кластери, значною мірою відрізняються один від одного, тому при розробці їхня реалізація буде виконуватися окремо один від одного, а за умов наявності вільних виконавців може проводитися паралельно.

На відміну від попередньої ситуації кластери C_9 та C_{10} доцільно більше не розділяти, оскільки елементи в кластерах більш близькі один до одного, тому їхня реалізація може бути тісно пов'язаною. Лише за умови великої кількості виконавців, або коротких строків реалізації, доцільно їх поділити.

Кластер C_6 також недоцільно поділяти, оскільки елементи відрізняються між собою лише потоком вихідних даних, тому взагалі може бути програмно реалізованим у вигляді єдиної функції (власне, гіпотеза про подібну реалізацію була розглянута та висунута в [31, 32]).

Отже, виділені кластери доцільно розглядати як СІ та саме у такому вигляді подавати як завдання до розробки проектній команді. В порівнянні з ручним аналізом конфігурації використання методу забезпечує можливість повторного використання дерева розбиття та полегшує пошук можливих комбінацій для формування конфігурації.

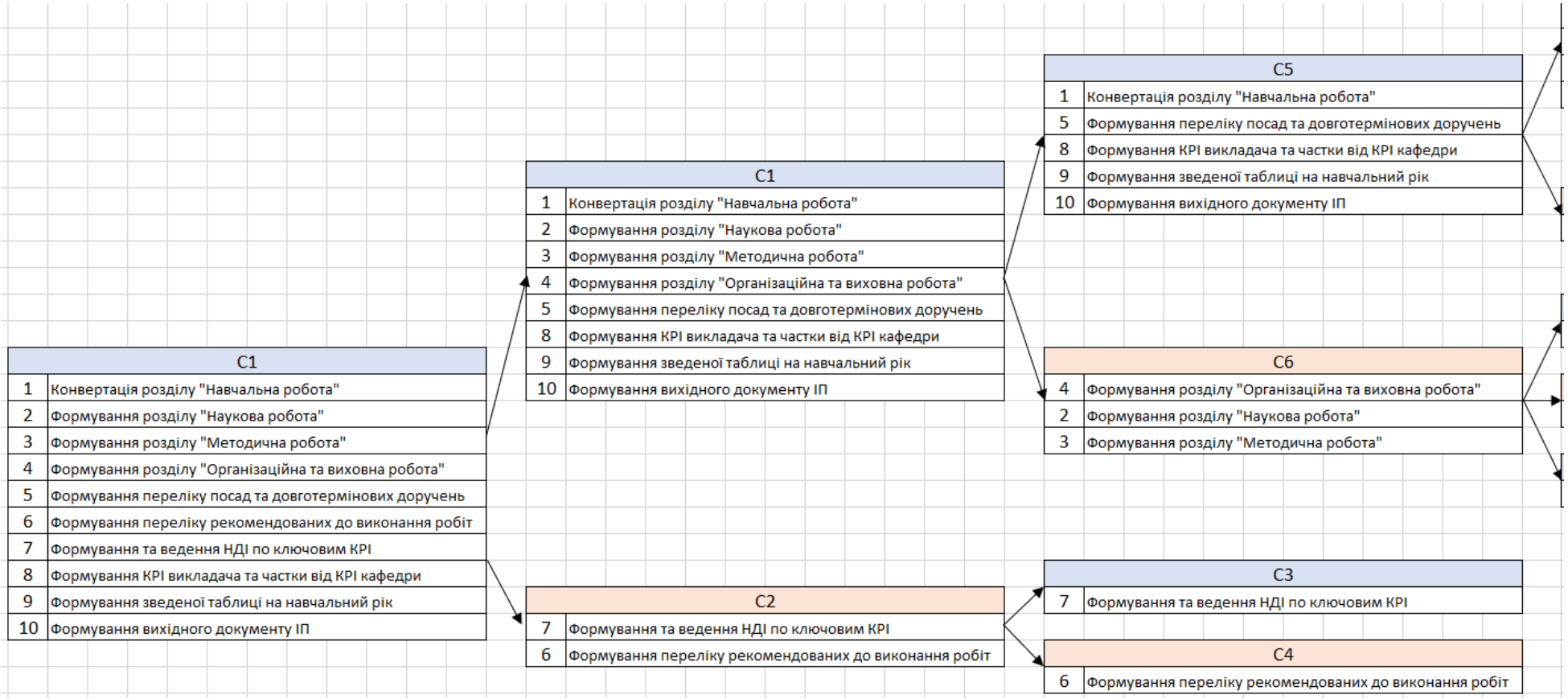

Рисунок 4.15 – Декодована дендрограма повного розбиття кластерів

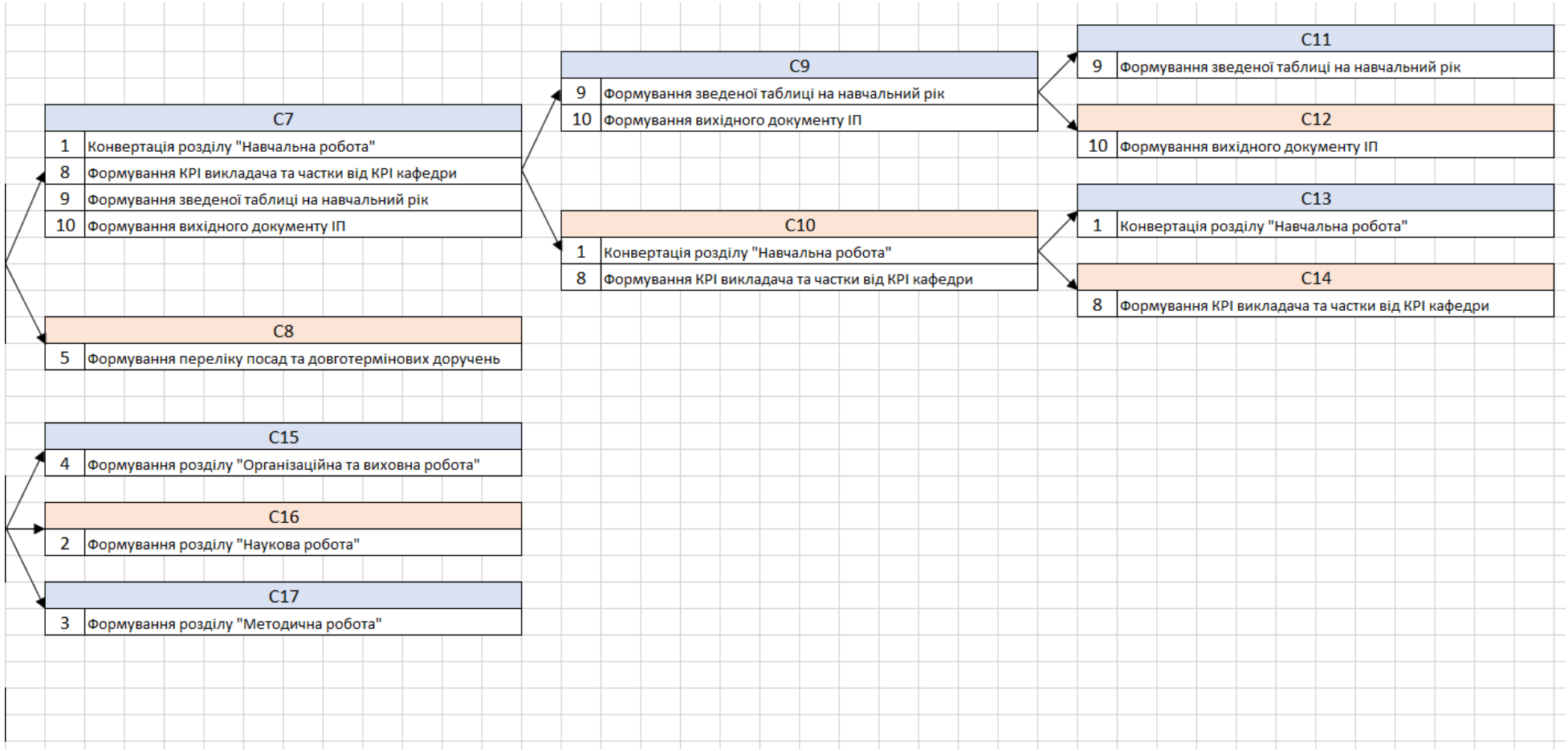

Рисунок 4.15, аркуш 2

4.6 Висновки

В четвертому розділі кваліфікаційної роботи було здійснено розробку плану проекту, в рамках якого відбувалася практична апробація методу, проведено опис вхідних даних методу, виконано підготовчий та основний етапи методу аналізу конфігурації ІТ-продукту та проведено аналіз отриманих даних.

В рамках розробки плану проекту, на основі якого проходила практична апробація розробленого методу, було побудовано та наведено:

- робочу структуру проекту;
- діаграму Ґантта робіт проекту;
- діаграму Ґантта ресурсів проекту.

Також, було виділено роботу, протягом якої проектному менеджеру доводиться вручну виконувати поєднання описів архітектури ІТ-продукту (операційного та структурного) та аналізувати конфігурацію ІТ-продукту.

Було встановлено, що вхідними даними методу під час його практичної апробації виступають операційний та структурний опис архітектури задачі «Формування та ведення індивідуального плану викладача».

При чому операційний опис архітектури представлений у вигляді функціональної структури ІТ-продукту, а структурний опис – у вигляді схеми логічної структури бази даних.

В рамках виконання підготовчого етапу було сформовано множини робіт W, відповідних вхідних та вихідних потоків даних – X та Y з операційного опису архітектури $Arch_{Op}$, всіх сутностей S з структурного опису $Arch_{DB}$. Проведено їхнє кодування простим цифровим кодом, наступне поєднання робіт з сутностями, які вони безпосередньо використовують, та сформовано опис $Arch_{config}$.

В ході виконання основного алгоритму аналізу конфігурації ІТ-продукту було детально розглянуто, описано та наведено послідовність виконання всіх

кроків до першого розподілу кластеру надвоє. За результатами повного виконання алгоритму було отримано розгорнуту дендрограму, що показує розбиття кластерів та наповненість кластерів на кожному етапі.

Задля проведення зручного аналізу отриманих результатів дендрограму було декодовано та наведено назви виконуваних робіт. Аналіз доцільної конфігурації в умовах проекту з реалізації поставленої задачі дозволив визначити раціональну конфігурацію, виходячи з двох основних критеріїв:

- кількості виконавців (команд-виконавців, проектних команд), що є в наявності;
- взаємозв’язок СІ (суть СІ, що знаходяться в одному кластері).

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було досліджено управління конфігурацією ІТ-продукту, в тому числі сам процес управління, понятійний апарат, основні задачі управління конфігурацією та окремі особливості перебігу ідентифікації конфігурації. Було встановлено важливість процесу, вплив на якість майбутнього продукту та інші переваги успішного управління процесом.

Дослідження методів вирішення задачі аналізу конфігурації продукту дозволило визначити основні вимоги до необхідного алгоритму, а розгляд існуючих інструментів управління конфігурацією показав, що мають місце практичні напрацювання щодо автоматизації процесу, однак в ньому відсутня реалізація саме аналізу конфігурації. Власне, що й підтверджує потребу в дослідженні на дану тематику та робить його актуальним.

Результати цих досліджень стали основою для постановки задачі розробки моделей та методів аналізу конфігурації ІТ-продукту.

В ході виконання кваліфікаційної роботи було розроблено модель вирішення задачі кластеризації задля розподілу вихідного опису архітектури ІТ-продукту на елементи конфігурації, що виступають завданнями до розробки. Також, було розроблено метод аналізу конфігурації ІТ-продукту для вирішення задачі дивизимної кластеризації (виходячи з потреби розподілити опис на окремі елементи). В рамках методу пропонується до виконання підготовчий та основний етап, де в підготовчому формуються та кодуються вхідні дані, а в основному – відбувається безпосередня кластеризація. Оскільки вхідні дані представлені досить складно у вигляді множини, що складається з трьох підмножин, то для визначення мір подібності елементів кластерів один між одним використовувалися три міри. Першою є модифікована відстань Хеммінга, де для порівняння елементів використовується сума за модулем 2. Наступною є відстань Чебишева, що

береться з отриманих відстаней Хеммінга та розраховується до виділення першого елемента в окремий кластер. Останньою виступає середня відстань Чебишева, що розраховується окремо для елемента й інших елементів цього ж кластеру та для елемента й інших елементів нововиділеного кластеру. Різниця цих середніх відстаней дозволяє визначити до якого з кластерів елемент, що розглядається, розташований ближче.

Для практичної апробації запропонованого методу аналізу конфігурації ІТ-продукту було обрано ІТ-проект розробки задачі «Формування та ведення індивідуального плану викладача» ІС «Кафедра». Для даного проекту було розроблено план проекту, а також, наведено робочу структуру, діаграму Ґанта робіт та діаграму Ґанта ресурсів проекту. На основі вхідних даних, а саме операційного та структурного опис архітектури задачі «Формування та ведення індивідуального плану викладача», що формують загальний опис архітектури, було практично застосовано розроблений метод. Результатом виконання є дендрограма розбиття опису архітектури на окремі елементи конфігурації, тобто завдання, що можуть бути виділеними до розробки. Завдяки використанню ієрархічної дивизимної кластеризації дендрограма відображає повне дерево розбиття на кластери, тому потенційна зміна конфігурації ІТ-продукту не потребує нового запуску методу. Виходячи з отриманих результатів було виділено раціональну конфігурацію для проекту.

Отже, розроблений метод є ефективним та полегшує процес аналізу конфігурації ІТ-продукту за рахунок аналітичного підходу до розбиття опису архітектури ІТ-продукту. Подальші модифікації методу дозволять покращити отриманий результат.

За тематикою атестаційної роботи було проведено:

− публікування тез доповіді «Дослідження моделі та методів аналізу конфігурації ІТ-продукту» у 25-му Міжнародному молодіжному форумі «Радіоелектроніка і молодь у XXI столітті» (ХНУРЕ, 2021 р.) [33].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи (для студентів усіх форм навчання другого (магістерського) рівня вищої освіти спеціальності 122 Комп'ютерні науки освітньо-професійної програми «Управління проектами в галузі інформаційних технологій») / Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Саєнко В.І., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2021. – 30 с.

2. ДСТУ 3008:2015. Державний стандарт України. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення. – К.: ДП «УкрНДНЦ», 2016. – 31 с.

3. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. / Видання офіційне. – К.: ДП «УкрНДНЦ», 2016 – 20 с.

4. Alexis Leon. Software configuration management handbook, third edition. 685 Canton Street Norwood, MA: Artech House, 2015. – 414 с.

5. A guide to the project management body of knowledge (PMBOK guide). Sixth edition. Newtown Square, PA: Project Management Institute, 2017. – 756 с.

6. P. Bourque and R. E. Fairley, eds. Guide to the Software Engineering Body of Knowledge. Version 3.0. IEEE Computer Society, 2014. – 335 с.

7. ISO 10007:2017 Quality management – Guidelines for configuration management. URL: https://www.iso.org/obp/ui/#iso:std:iso:10007:ed-3:v1:en (дата звернення: 20.05.2021).

8. ESA Board for Software Standardisation and Control (BSSC). Guide tosoftware configuration management. Noordwijk, The Netherlands: European Space Agency Publications Division, 1995. – 74 с.

9. 5026-1-2019 – ISO/IEC/IEEE International Standard – Systems and software engineering--Systems and software assurance --Part 1:Concepts and

vocabulary. URL: https://ieeexplore.ieee.org/document/8657410 (дата звернення: 20.05.2021).

10. Barsehian, A. A., Kupriianov M. S., Kholod I. I., Tess M. D., Yelizarov S. I. Analiz dannykh y protsessov. SPb.: BKhV-Peterburh, 2009. – 512 s.

11. AccuRev. Simplifying the Process to Build and Maintain Complex Software. URL: https://www.microfocus.com/en-us/products/accurev/overview (дата звернення: 30.08.2021).

12. Use case. Configuration management. URL: https://www.ansible.com/use-cases/configuration-management (дата звернення: 30.08.2021).

13. BitKeeper. Scalable Version Control. URL: https://www.bitkeeper.org/ (дата звернення: 30.08.2021).

14. CFEngine. Know more, React Faster.URL: https://cfengine.com/ (дата звернення: 30.08.2021).

15. Chef INFRA. Powerful Policy-Based Configuration Management System Software URL: https://www.chef.io/products/chef-infra (дата звернення: 30.08.2021).

16. IBM Rational ClearCase. URL: https://www.ibm.com/products/rational-clearcase (дата звернення: 30.08.2021).

17. What is ConfigHub? URL: https://www.confighub.com/what-is-confighub (дата звернення: 30.08.2021).

18. JAAS. How it works. URL: https://jaas.ai/how-it-works (дата звернення: 30.08.2021).

19. IBM Rational Synergy. URL: https://www.ibm.com/products/rational-synergy (дата звернення: 30.08.2021).

20. Continuous Configuration Automation. URL: https://puppet.com/use-cases/continuous-configuration-automation (дата звернення: 30.08.2021).

21. Server Configuration Monitor. URL: https://www.solarwinds.com/server-configuration-monitor (дата звернення: 30.08.2021).

22. Surround SCM. SCM Software for Your Development Lifecycle. URL: https://www.perforce.com/products/surround-scm (дата звернення: 30.08.2021).

23. ZENworks Suite Endpoint Management Tools. URL: https://www.microfocus.com/en-us/portfolio/zenworks-suite/overview (дата звернення: 30.08.2021).

24. Sławomir T. Wierzchoń, Mieczysław A. Kłopotek. Modern Algorithms of Cluster Analysis. Springer International Publishing, 2018. – 421 с.

25. Christian Hennig, Marina Meila, Fionn Murtagh, Roberto Rocci. Handbook of Cluster Analysis. CRC Press, 2016. – 730 с.

26. Євланов, М. В., Васильцова, Н. В., Панфьорова, І. Ю. Моделі і методи синтезу опису раціональної архітектури інформаційної системи. Вісник наукового університету «Львівська політехніка». Серія «Інформаційні системи та мережі», 2015. № 829. – С. 135-152.

27. Ievlanov, M. Development of the model and method of selecting the description of rational architecture of information system. Eastern-European Journal of Enterprise Technologies, 2016. 1(2(79). – С. 4-12.

28. Levykin, V., Ievlanov, M., Neumyvakina, O. Developing the models of patterns in the design of requirements to an information system at the knowledge level. Eastern-European Journal of Enterprise Technologies, 2017. 5(2 (89). – С 19-26.

29. Mohammed, M. Q., Muhamed, S. Q., Ievlanov, M., Gazetdinova, Z. Improvement of the method of scenario analysis of functional requirements to an information system. Eastern-European Journal of Enterprise Technologies, 2019. 3(2 (99), – С 25-35.

30. Ievlanov, M., Vasiltcova, N., & Panforova, I. Development of methods for the analysis of functional requirements to an information system for consistency and illogicality. Eastern-European Journal of Enterprise Technologies, 2018. 1(2 (91), – С 4-11.

31. Кузьма Є. А. Концепція побудови функціональної задачі «Формування та ведення індивідуального плану викладача. 24-й Міжнародний

молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 6. – Харків: ХНУРЕ. 2020. – 306 с.

32. Vasyltsova N.V., Panforova I.Y., Kuzma Y.A. Formation of Function Use Cases Based on Its Mathematical Model. 4-а Міжнародна науково-технічна конференція «Комп’ютерні та інформаційні системи і технології». Зб. матеріалів конференції. – Харків: ХНУРЕ. 2020. – С 59-60.

33. Кузьма Є. А. Дослідження моделі та методів аналізу конфігурації ІТ-продукту. 25-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 6. – Харків: ХНУРЕ. 2021. – С 76-77.