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 методів автоматизованого призначення пакетів робіт командам виконавців ІТ-проєкту
(тема)

Виконав:
студент 2 курсу, групи УПГІТм-22-1
Тіханов Олександр Іванович
(прізвище, власне ім'я, по батькові)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Управління проектами в галузі інформаційних технологій
(повна назва освітньої програми)
Керівник доцент каф. ІУС Борисенко Тетяна Іванівна
(посада, прізвище, власне ім'я, по батькові)

Допускається до захисту

Зав. кафедри _____ Костянтин ПЕТРОВ
(підпис) (власне ім'я, прізвище)

2024 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-наукова_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _Управління проектами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_01__»__квітня__ 20 24_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____Тіханову Олександру Івановичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів автоматизованого призначення пакетів робіт командам виконавців ІТ-проєкту_____
_____

затверджена наказом університету від 01 квітня 2024 р. № 258Ст

2. Термін подання студентом роботи до екзаменаційної комісії _____06.06.2024 р.

3. Вихідні дані до роботи _наукова література, публікації та інтернет-джерела з досліджуваної проблеми, матеріали переддипломної практики._____
_____
_____
_____
_____
_____
_____

4. Перелік питань, що потрібно опрацювати в роботі _сучасний стан проблеми автоматизації призначення пакетів робіт командам виконавців ІТ-проєкту; дослідження особливостей методу призначення пакетів робіт командам виконавців ІТ-проєкту; удосконалення методу призначення пакетів робіт командам виконавців ІТ-проєкту; експериментальна перевірка методу призначення пакетів робіт командам виконавців ІТ-проєкту.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Змістовна постановка задачі призначення пакетів робіт командам виконавців ІТ-проєкту	01.04-03.04	Виконано
2	Аналіз наявних методів та підходів до вирішення задачі призначення пакетів робіт командам виконавців ІТ-проєкту	04.04-09.04	Виконано
3	Постановка задач магістерської кваліфікаційної роботи	10.04-12.04	Виконано
4	Постановка задачі класифікації	13.04-15.04	виконано
5	Дослідження алгоритмів вирішення задачі класифікації	16.04-19.04	виконано
6	Дослідження та вибір класу алгоритмів наївного байєсівського класифікатора	20.04-25.04	Виконано
7	Модифікація алгоритму класифікації на основі мультиномінального наївного байєсівського класифікатору	26.04-29.04	Виконано
8	Адаптація мультиномінального наївного байєсівського класифікатора для вирішення завдань магістерської роботи	30.04-01.04	Виконано
9	Опис підготовчого етапу методу призначення пакетів робіт командам виконавців ІТ-проєкту	02.05-06.05	Виконано
10	Розробка загального алгоритму формування пакета завдань для команд виконавців ІТ-проєкту	07.05-10.05	Виконано
11	Опис особливостей застосування розробленого удосконаленого методу	11.05-13.05	Виконано
12	Опис особливостей організації, на базі робіт якої проходитиме практична апробація методу	14.05-16.05	Виконано
13	Опис вхідних даних методу	17.05-19.05	Виконано
14	Виконання підготовчого етапу методу	20.05-22.05	Виконано
15	Визначення виконавців для нових завдань	23.05-24.05	Виконано
16	Аналіз отриманих результатів	25.05-26.05	Виконано
17	Оформлення пояснювальної записки до кваліфікаційної роботи	27.05-03.06	Виконано
18	Захист кваліфікаційної роботи в екзаменаційній комісії	07.06.2024	Виконано

Дата видачі завдання __01__ ___квітня__2024_ р.

Студент _____
(підпис)

Керівник роботи _____ доц. каф. ІУС Тетяна БОРИСЕНКО
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи містить: 124 с., 11 рис., 16 табл., 1 додаток, 27 джерел.

БАЙЄСІВСЬКИЙ КЛАСИФІКАТОР, ВИКОНАВЕЦЬ, ЗАВДАННЯ, КЛАСИФІКАЦІЯ, КОМАНДА ВИКОНАВЦІВ, МЕТОД, РОЗПОДІЛ ЗАВДАНЬ.

Об’єктом дослідження в кваліфікаційній роботі є процес призначення пакетів робіт командам виконавців ІТ-проєкту.

Як методи дослідження в кваліфікаційній роботі застосовувалися спостереження, наукові факти, порівняння, експеримент та аналіз.

Метою кваліфікаційної роботи є дослідження та удосконалення процесу призначення пакетів завдань командам виконавців ІТ-проєкту.

В ході досліджень отримані такі теоретичні результати: запропоновано модифікацію алгоритму наївного баєсівського класифікатора для вирішення задач кваліфікаційної роботи; розроблено удосконалений метод формування пакетів завдань командам виконавців ІТ-проєкту з використанням мультиноміального алгоритму наївного баєсівського класифікатора.

Після проведення експериментальної перевірки удосконаленого методу на основі реальних даних було отримано результати, що демонструють працездатність методу.

Результати роботи можуть бути використані в системах управління проєктами.

Кваліфікаційну роботу виконано згідно методичних вказівок щодо розробки та оформлення кваліфікаційної роботи [1], ДСТУ 3008:2015 [2] та ДСТУ 8302:2015 [3].

ABSTRACT

The explanatory note to the master's qualification work contains: 124 pages, 11 figures, 16 tables, 1 appendix, and 27 references.

BAYESIAN CLASSIFIER, EXECUTOR, TASK, CLASSIFICATION, EXECUTION TEAM, METHOD, TASK DISTRIBUTION.

The object of research in the qualification work is the process of assigning work packages to IT project execution teams.

The research methods used in the qualification work include observation, scientific facts, comparison, experiment, and analysis.

The aim of the qualification work is to study and improve the process of assigning task packages to IT project execution teams.

The following theoretical results were obtained during the research: a modification of the naive Bayesian classifier algorithm for solving the tasks of the qualification thesis was proposed; an improved method for forming task packages for IT project execution teams using a multinomial naive Bayesian classifier algorithm was developed.

After conducting an experimental verification of the improved method based on real data, results were obtained that demonstrate the method's functionality.

The results of the work can be used in project management systems.

The qualification work was performed according to the methodological guidelines for the development and design of qualification work [1], DSTU 3008:2015 [2], and DSTU 8302:2015 [3].

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ДСТУ – Державний стандарт України

ІТ – інформаційні технології

ІУС – інформаційні управляючі системи

ISO – International Organization for Standartization

KPI – key performance indicator

PMBOK – Project Management Body Of Knowledge

SP – Story Points

WBS – Work Breakdown Structure

ВСТУП

Зростаюча кількість новітніх технологій та процесів в сучасному світі надає безліч можливостей для покращення управління проєктами та розробки програмного забезпечення. Однак, незважаючи на це, існують певні виклики, з якими стикаються організації у плануванні та ефективному використанні ресурсів.

Однім з таких викликів є процес призначення пакетів робіт командам виконавців. Цей процес може бути складним та затратним у часовому вимірі, особливо у великих та складних IT-проєктах. Важливою проблемою є правильне розподілення робіт між командами, забезпечення оптимізація ресурсів для виконання робіт та максимізація продуктивності. Процес розподілення завдань здебільшого відбувається вручну, що може призводити до суб’єктивних помилок, нераціонального використання ресурсів та перевантаження окремих команд. Часто це трапляється на великих проєктах з великою кількістю виконавців, де складність координації збільшується пропорційно кількості завдань та виконавців.

Саме тому розгляд проблеми призначення пакетів робіт, а саме автоматизації цього процесу, стає вкрай важливим завданням. Беззаперечно, ці зміни будуть сприяти оптимізації процесів планування, а в подальшому дозволять повністю уникнути зайвого адміністративного тягаря, забезпечити більш точне та ефективне розподілення робіт, та знизити ризик помилок чи навіть затримок у проєкті. Таким чином, розробка та впровадження програмного засобу для призначення пакетів робіт командам виконавців IT-проєкту може значно полегшити процес управління проєктами та покращити результативність роботи команд виконавців.

Сучасні технології, в першу чергу такі, як штучний інтелект, відкривають нові можливості для автоматизації процесів управління проєктами. Використання цих технологій дозволяє розробити алгоритми та

моделі, які здатні аналізувати великі обсяги даних та приймати ефективні рішення щодо призначення завдань командам виконавців. Такий підхід здатен забезпечити точність, швидкість у призначенні робіт, що є критично важливими в умовах сучасного над швидкого темпу розвитку технологій та конкуренції на ринку інформаційних послуг.

Метою даного дослідження є аналіз, вибір та удосконалення сучасних рішень, методів та підходів щодо можливості автоматизації процесу призначення пакетів робіт командам виконавців ІТ-проєктів.

Загалом сучасний стан автоматизації бізнес-процесів, в тому числі і процесу призначення пакетів робіт, має певні недоліки та проблемні аспекти. Однією з найбільш поширених проблем є неефективне розподілення завдань, що може призводити до затримок у виконанні проєкту та зниження якості робіт.

Отже, можна без перебільшення визначити що проблема з автоматизації призначення пакетів робіт в ІТ-проєктах на сучасному етапі є вкрай актуальною та потребує інноваційних рішень з використанням новітніх технологій, щоб забезпечити своєчасне виконання проєктів та досягнення поставлених цілей.

1 СУЧАСНИЙ СТАН ПРОБЛЕМИ АВТОМАТИЗАЦІЇ ПРИЗНАЧЕННЯ ПАКЕТІВ РОБІТ КОМАНДАМ ВИКОНАВЦІВ ІТ-ПРОЄКТУ

Розглядаючи проблему автоматизації призначення пакетів робіт командам виконавців у ІТ-проєктах, можна побачити, що насправді проблема полягає не тільки в межах пов’язаних з організаційними аспектами, а також стосується активного впровадження новітніх технологій у процес управління проєктами. Використання штучного інтелекту для аналізу та прийняття рішень щодо призначення завдань команді виконавців може значно підвищити ефективність та точність процесу, мінімізувати втручання в процес людини і тим самим дати змогу виконавцям проєкту сконцентруватися на інших процесах проєкту.

Таким чином, проведення аналізу, формування проблеми та пошуку можливих рішень щодо вдосконалення процесів управління проєктами з використанням сучасних методів та передових технологій стає надзвичайно важливим завданням. Результати досліджень будуть ініціювати нові підходи та ідеї, що сприятимуть подальшому розвитку галузі управління проектами в сфері інформаційних технологій.

Управління ІТ-проєктами в сучасних умовах вимагає системного та ефективного підходу до розподілу завдань між командами виконавців ІТ-проєкту. Формування та призначення пакетів робіт є ключовим етапом у цьому процесі, оскільки воно визначає чітку специфікацію завдань для кожної команди та забезпечує комплексний підхід до виконання проєкту.

1.1 Змістовна постановка задачі призначення пакетів робіт командам виконавців ІТ-проєкту

У сучасному світі продуктові та аутсорсингові компанії представляють собою дві різні моделі організації бізнесу з унікальними характеристиками та особливостями. Продуктова компанія спрямовує свою увагу на розробку та випуск власних продуктів або послуг, маючи внутрішні ресурси та команди, тоді як аутсорсингова компанія надає послуги з розробки або обслуговування іншим компаніям на замовлення, часто виконуючи завдання чи проекти, що не входять до основної сфери діяльності замовника [4].

Для проведення дослідження та аналізу стану автоматизації призначення пакетів робіт командам виконавців ІТ-проєкту, була обрана продуктова компанія, яка спеціалізується на випуску декількох продуктів.

Продукт у цьому контексті – це іменована сукупність бізнес-можливостей, цінних для певного сегмента клієнтів. Продукт може включати програмне забезпечення, дані, або комбінацію програмного забезпечення, обладнання та послуг, необхідних для повного досвіду використання продукту [5].

Кожен продукт розглянутої компанії є результатом реалізації декількох проектів, причому кожен проект має свою команду виконавців. Ці команди виконавців працюють над реалізацією вимог проекту та виконанням завдань, що відповідають їхнім професійним навичкам і компетенціям.

Проєкт - це серія структурованих завдань, дій та результатів, які ретельно виконуються для досягнення бажаного результату. Це тимчасові зусилля створення цінності за допомогою унікальних продуктів, послуг і процесів [6].

Під час проведення дослідження щодо методів автоматизованого призначення пакетів робіт командам виконавців ІТ-проєкту, у процесі розробки одного продукту компанії було залучено три команди.

Команди працюють над реалізацією проектів відповідно до принципів Agile методології.

Методологія Agile – це підхід до управління проектами, який передбачає розбиття проекту на етапи і наголошує на постійній співпраці та вдосконаленні. Команди слідують циклу планування, виконання та оцінки результатів виконаної роботи. Оцінка включає аналіз досягнень, виявлення проблем та пошук шляхів їх вирішення, що дозволяє командам постійно покращувати свої процеси та результати [7].

Команди виконавців слідують найкращим практикам Scrum, що передбачає ітеративний підхід до розробки та зосереджується на постійному вдосконаленні процесів розробки та управління проектом.

Scrum – це гнучка система управління проектами, яка допомагає командам структурувати свою роботу та керувати нею за допомогою набору цінностей, принципів та практик [8].

Ітеративний підхід (Iterative Approach) – це підхід до розробки програмних систем, зосереджений на початковій, спрощеній реалізації та поступовому додаванню нового функціонала до попереднього аж до отримання остаточного продукту [9].

Після визначення основ Agile методології та підходу команд виконавців до процесу управління проєктами, варто звернутися до конкретних інструментів та практик, які використовуються в їхній роботі. Наприклад, одним із важливих компонентів цього підходу є використання скрам-артефактів, таких як Product Backlog, Sprint Backlog, інкремент та інші, які допомагають команді ефективно керувати процесом розробки та досягати поставлених цілей.

Скрам-артефакти (Scrum Artifacts) – це інформація, яку Scrum-команда та зацікавлені сторони використовують для деталізації продукту, що розробляється, дій щодо його створення та дій, що виконуються в ході проекту [10].

На рисунку 1 відображені основні Scrum Artifacts.

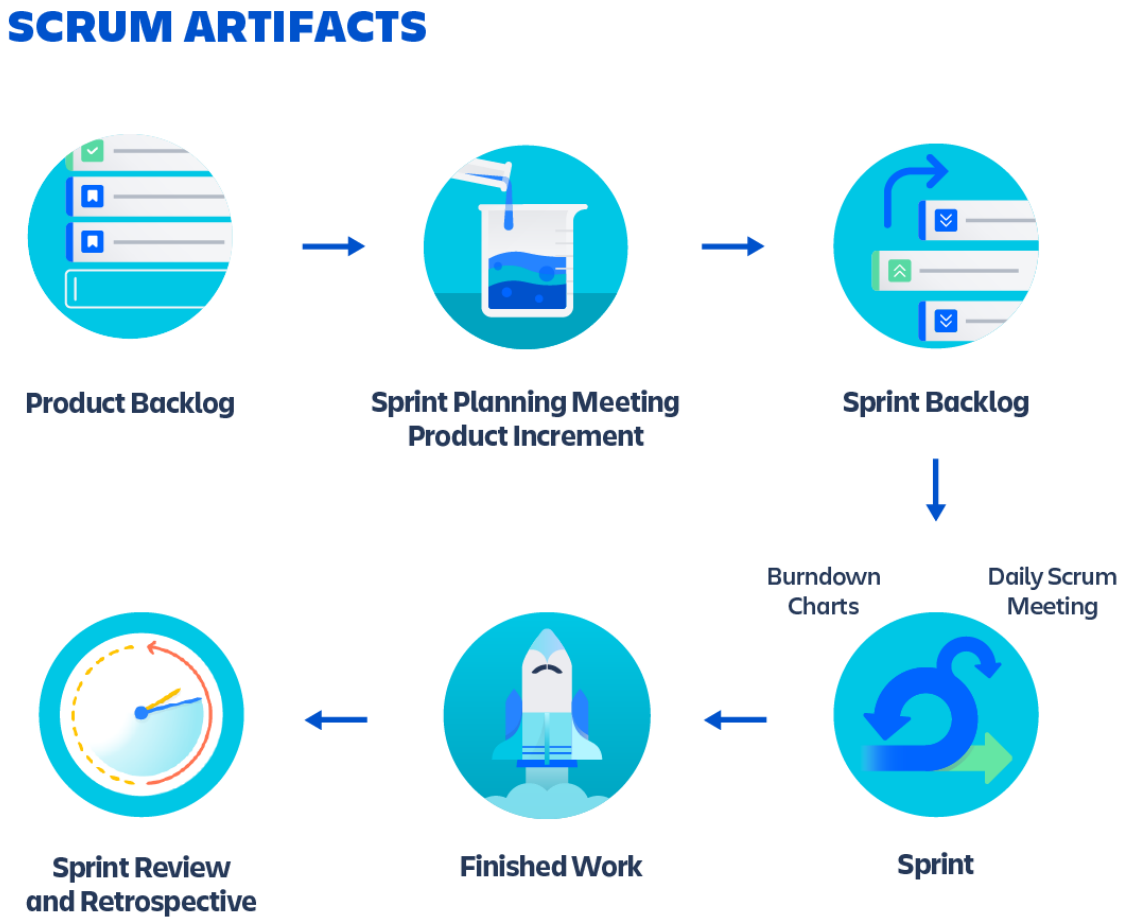

Рисунок 1 – Склад основних Scrum Artifacts

Далі ми звернемося до аналізу та опису того, як концепція проекту перетворюється на конкретні завдання, які потім збираються в пакети робіт, та як ці пакети робіт призначаються командам виконавців IT-проекту.

Почнемо зі стадії, що передує всьому процесу розробки – стадія створення ініціативи.

Згідно стандартів управління проєктами, термін ініціатива (Initiative) може мати кілька визначень залежно від контексту. Однак в загальному розумінні ініціативи - це початкова дія або напрямок, який вибирається з метою започаткування нового проєкту або реалізації певної діяльності. Ініціатива може виникнути з різних причин, від ідей та стратегічних цілей, до змін у бізнес-середовищі або потреб споживачів [11].

На етапі формування Initiative створюються високорівневі завдання компанії за участю топ-менеджменту, які узгоджуються з відповідними зацікавленими сторонами проєкту. Зазвичай за допомогою Initiative компанія формує стратегічні плани на наступний рік, що в свою чергу дає можливість залучення інвестицій. Створена ініціатива передбачає в подальшому залучення тієї чи іншої команди, або команд для досягнення запланованих компанією результатів. Щодо призначення ініціативи, то це відбувається за участі менеджерів продукту та власників продукту.

Власник продукту (Product Owner) – це роль управління в команді продукту, яка є відповідальною особою за беклог продукту для досягнення бажаного результату, до якого прагне команда розробки продукту [12].

Після визначення ініціативи, на етапі планування проекту, визначається відповідальна особа, власник продукту, для подальшого перетворення ініціативи на більш структуровані завдання – епікі.

Епік (Epics) – це великий, пов’язаний між собою обсяг робіт, покликаний ієрархічно організувати набір вимог та забезпечити конкретні бізнес результати [13].

Після визначення ініціатив, наступними кроками щодо формування епіків, є проведення квартального планування, де саме і обговорюються та узгоджуються питання щодо дроблення ініціатив на менші завдання. Власник продукту спільно з командами зазвичай проводить мозковий штурм, щоб визначити кращі критерії для формування епіків, це може відбуватися з урахуванням технічних вимог та бізнес-потреб. Власник продукту разом з командою проводить оцінювання сформованого епіку, в нашому випадку використовується система оцінювання T-shirt size.

T-shirt size - це інструмент оцінки проекту та планування потужності, який допомагає відстежувати, скільки часу та зусиль потрібно на реалізацію історії користувача без вказівки конкретних одиниць часу, таких як година або день. Замість цього використовуються розміри футболок (XS, S, M, L, XL, XXL і т. д.), щоб показати складність або зусилля, необхідні для кожного

предмета. Такий підхід допомагає команді швидко оцінити співвідносний розмір робочих елементів і окремо розставити їх пріоритети [14].

В подальшому процес створення епіків повторюється в залежності від необхідності створення нових або додаткових.

Важливо зауважити, що вже на етапі формування епіків власник продукту вже має інформацію про команди виконавців через попередні обговорення та планування. Це може бути результатом попередніх проєктів, де команди вже були залучені до роботи над подібними завданнями, або через попередні розмови та узгодження з керівниками цих команд.

Сформовані епіки піддаються процесу пріоритезації, і на наступних зустрічах на рівні компанії відбувається презентація дорожніх карт відповідних продуктів, яку проводять власники продуктів.

Дорожня карта продукту (Product Roadmap) — це загальнодоступний живий документ, у якому описується вид і напрямок розвитку продукту протягом усього його життєвого циклу. Дорожня карта на самому базовому рівні описує, що потрібно створити і чому, вона служить планом реалізації загальної стратегії продукту. За допомогою дорожньої карти відбувається візуалізація стратегії команди щодо створення цінностей у вигляді продукту, встановлюється хронологічний порядок реалізації різних компонентів продукту [15].

Сформовані та пріорітезовані епікі, за умови початку роботи над ним найближчими спринтами, додаються до беклогу продукту для подальшої роботи.

Беклог продукту (Product Backlog) – це пріоритетний список робіт для команди розробників, складений на основі дорожньої карти та її вимог. Найважливіші елементи відображаються у верхній частині списку невиконаних робіт, щоб команда знала, що потрібно виконати насамперед [16].

На рисунку 2 відображено продуктовий беклог разом з епіками, розташованими згідно пріоритетів у системі управління проєктами Jira.

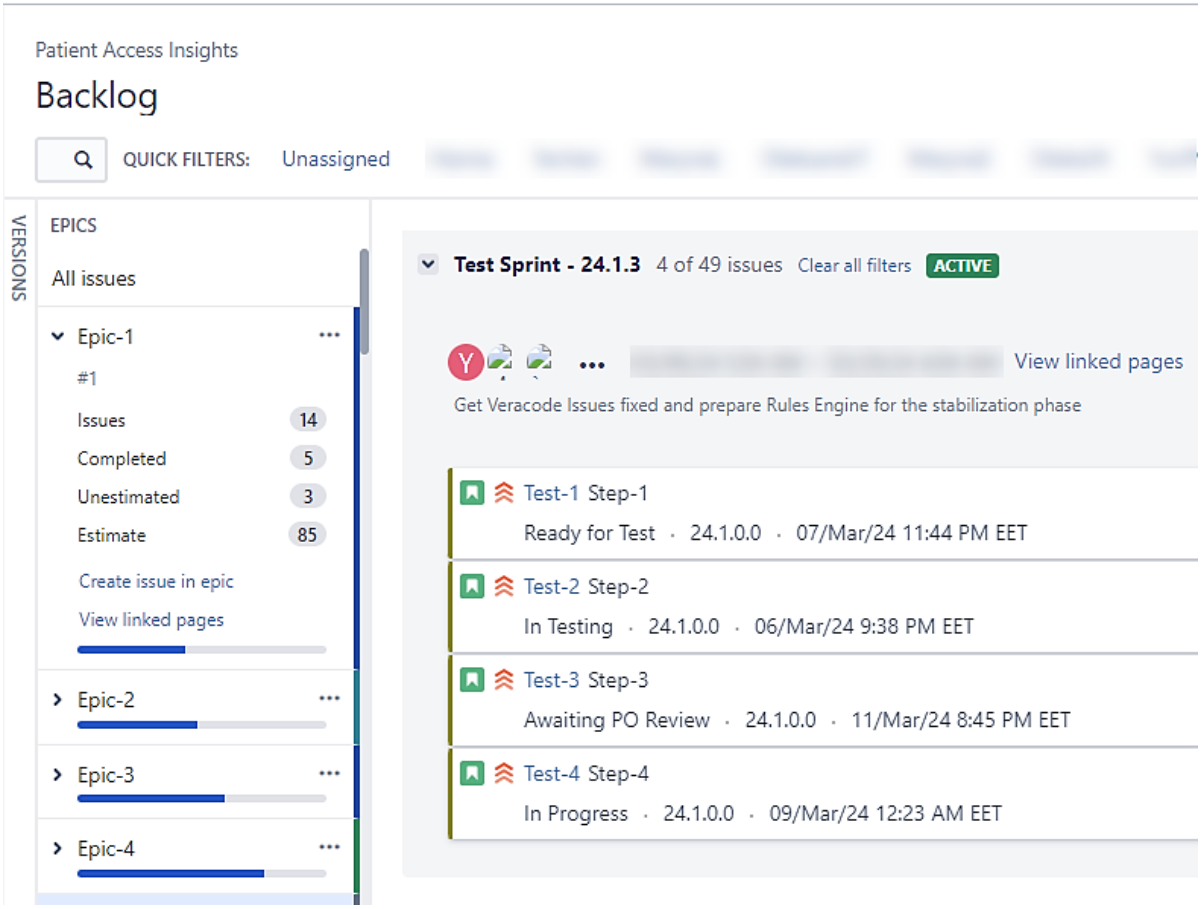

Рисунок 2 – Відображення пріоритезованих епіків в продуктовому беклозі у системі управління проєктами Jira

Стадія формування пакетів робіт є наступною частиною процесу управління проєктом після сформованих епіків.

Пакет робіт (Work Packages) – роботи, визначені на найнижчому рівні ієрархічної структури робіт, для яких здійснюють оцінку та управління вартістю та тривалістю [17].

Під час формування пакетів робіт, епіки проходять стадію декомпозиції на окремі завдання, або робочі одиниці, що дозволяє деталізувати вимоги та визначити конкретні кроки для досягнення бажаних результатів.

Декомпозиція (Decomposition) - це процес поділу проекту на дрібніші частини, які часто називають результатами. Це особливо корисно при

управлінні великими або мегапроектами, але цей метод застосовується до більшості проектів у різних галузях. Така декомпозиція проекту спрощує оцінку і призначення окремих завдань, тим самим роблячи робочий процес більш керованим від ініціації проекту, планування, виконання, моніторингу та контролю до закриття [18].

Декомпозиція обє’му робіт відбувається за допомогою опису проекту, щоб сформулювати основні завдання, пов'язані з проєктом, та встановити критерії прийняття для кожного етапу виконання. Обсяг також можна деталізувати шляхом декомпозиції до нижчих рівнів з використанням ієрархічної структури робіт (WBS). WBS – це ієрархічна декомпозиція всього обсягу робіт, який команда проєкту повинна виконати для досягнення цілей проєкту та створення необхідних результатів виконання. Кожен нижчий рівень в ієрархії представляє вищий рівень деталізації доробку та роботи, необхідної для його створення.

Інший спосіб деталізації обсягу робіт полягає у визначенні тем проєкту у Agile статуті, дорожній мапі або як частини ієрархії продукту. Теми представляють великі групи цінності для замовників, відображені у історіях користувачів, пов’язаних загальним фактором, таким як функціональність, джерело даних або рівень безпеки.

Для виконання тем команда проекту розробляє епіки, які є логічними контейнерами для великої історії користувача, яка є занадто великою для завершення в межах однієї ітерації. Епіки можуть бути розділені на елементи функціоналу – набір пов’язаних вимог, які зазвичай описують у вигляді короткої фрази або функції, які відображають конкретну поведінку продукту. Кожен елемент функціоналу міститиме кілька історій користувачів. Історія користувача – короткий опис кінцевого результату для конкретного користувача, що є запрошенням до обговорення деталей.

Команда проекту визначає деталі історії в останній момент ухвалення рішення, щоб уникнути марнотратного планування у разі зміни обсягу.

Історія – це чітке та стисле представлення вимоги, викладене з точки зору кінцевого користувача [19].

Процес декомпозиції відбувається під час проведення скрам-процесів у вигляді зустрічей: планування наступного спринта, беклог грумінг або рефайнмент. За потреби, як з боку команди, так і з боку власника продукту, можуть бути ініційовані додаткові зустрічі, для обговорення або вирішення питань щодо вимог. Як результат процесу декомпозиції епіків, ми маємо створенні історії користувача, які визначають функціональні вимоги або потреби користувачів щодо продукту чи сервісу.

Історія користувача (User Story) - це найменша одиниця роботи в гнучкому середовищі. Це кінцева мета, а не функція, виражена з погляду користувача програмного забезпечення. Історія користувача — це неформальне загальне пояснення функції програмного забезпечення, написане з точки зору кінцевого користувача або клієнта [20].

Планування спринту (Sprint Planning) - це подія у Scrum, яка запускає спринт. Мета планування спринту - визначити, що можна виконати за спринт і як ця робота буде виконана. Планування спринту здійснюється у співпраці з усією scrum-командою [21].

Прибирання беклогу (Backlog Grouming) - також називається «уточненням беклогу» або «часом історії», це регулярний захід для гнучкої розробки продуктів. Основна мета сеансу обробки беклогу є переконатися, що користувацькі історії в бэклог продукту на наступні кілька спринтів підготовлені для планування спринту. Регулярні сеанси очищення беклогу також допомагають гарантувати, що правильні історії розставлені за пріоритетами і що беклог продукту не перетвориться на чорну дірку [22].

Тобто на даному етапі епікі розбиваються на менші завдання – керовані одиниці для кращого контролю та виконання, і відповідно пакети робіт є набором завдань і заходів. В процесі створення історії користувача, є певні вимоги щодо заповнення відповідних розділів, а також вимоги щодо формулювання безпосередньо завдання яке має бути реалізовано для

досягнення певних очікувань з боку користувачів. Щодо вимог та рекомендацій заповнення та поновлення історії користувача – це регламентується на рівні компанії, продукту а де інколи навіть на рівні окремого проекту, і може бути формалізовано у вигляді документа для внутрішнього використання.

Також рівень деталізації вимог, наданих в пакетах робіт, дозволяє команді в цілому визначати необхідні ресурси для виконання завдання, тим самим даючи змогу одночасно працювати над різними компонентами проєкту. Ця паралельна операція підвищує продуктивність, оскільки всі члени команди дотримуються списку завдань/заходів для даного пакета робіт і виконують їх до зазначеного терміну.

В рамках розробки даного продукту, команди використовують програмний засіб Jira для керування процесами розробки, відстеження завдань та співпраці в рамках проекту.

Jira - популярний інструмент управління проектами, розроблений Atlassian. Він переважно використовується для відстеження проблем, відстеження помилок та гнучкого управління проектами.

Jira дозволяє командам ефективно планувати, відстежувати та керувати своєю роботою, забезпечуючи спільну роботу та прозорість протягом усього процесу розробки. Цей інструмент пропонує такі функції, як робочі процеси, що налаштовуються, дошки Kanban і Scrum, звіти в реальному часі та інтеграцію з іншими інструментами, які зазвичай використовуються при розробці програмного забезпечення [23].

На рисунку 3 наведено приклад відображення історії користувача в середовищі Jira з урахуванням вимог щодо заповнення.

Під час проведення зустрічей безпосередньо за участю потенційних виконавців робіт, у вигляді планування або грумінгу, відбувається формування завдань з попереднім оцінюванням цих завдань командою. У разі неможливості оцінити завдання на етапі формування з будь яких причин, до

оцінювання даного завдання повертаються при вирішені питань які саме унеможливлювали оцінювання.

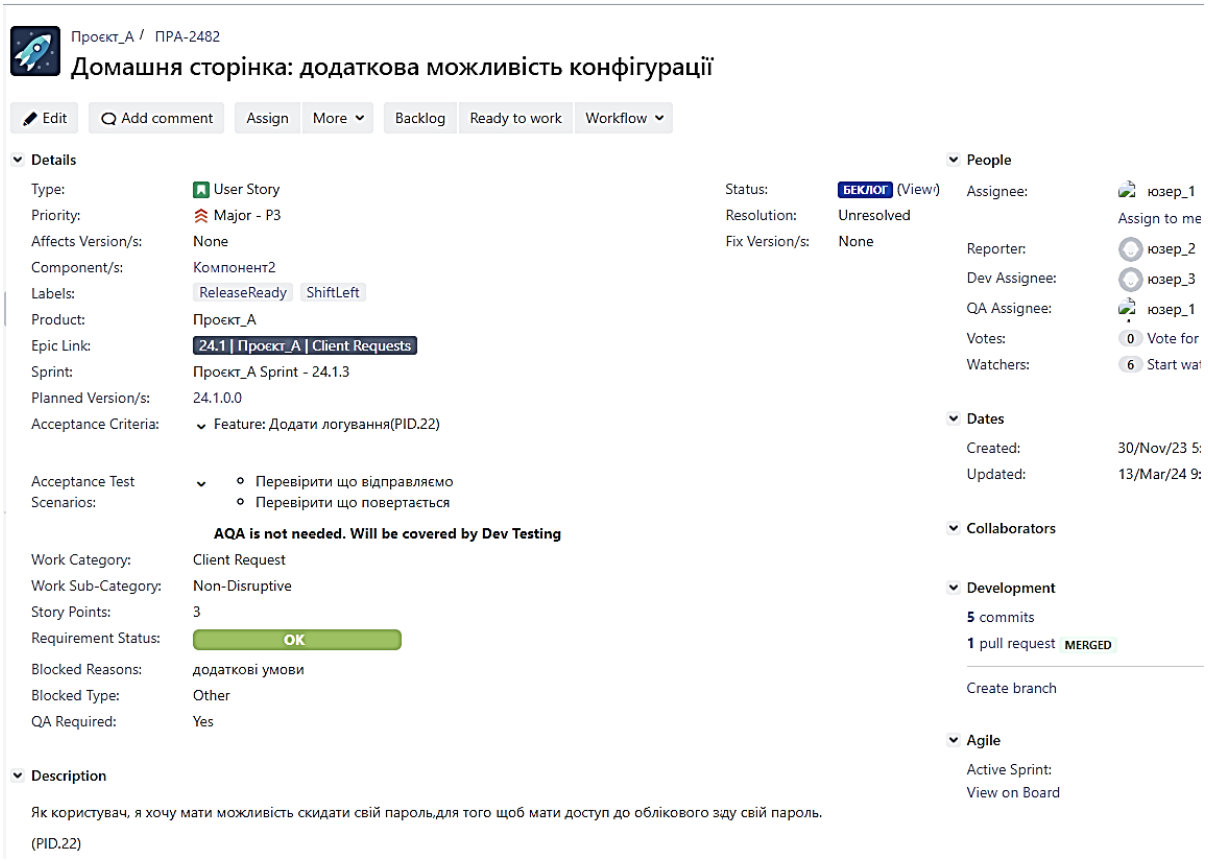

Рисунок 3 - Приклад заповненої історії користувача у середовищі Jira

У подальшому, відбувається наповнення продуктового беклогу створеними пакетами робіт, в нашому випадку за рахунок користувацьких історій, технічних завдань або помилок створених як користувачами так і командою.

Саме оцінені завдання в продуктовому беклозі допомагають власнику продукту та командам виконавців, вимірювати, керувати та контролювати проєкт.

Процес роботи над створенням або доопрацюванням користувацької історії носить ітеративний характер і відбувається в залежності від необхідності наповнення продуктового беклогу команди.

Здібності команди щодо надання ступеня деталізації вимог під час створення історії користувача, оцінювання складності завдання та можливість визначити потенційні ризики, відбувається за рахунок досвіду та певної експертизи команди виконавців.

На рисунку 4 схематично відображено процес дрібнення ініціатив до рівня окремих робіт у вигляді User Story (історій користувачів) та Task (завдань технічної спрямованості), з яких потім формуються пакети робіт командам виконавців.

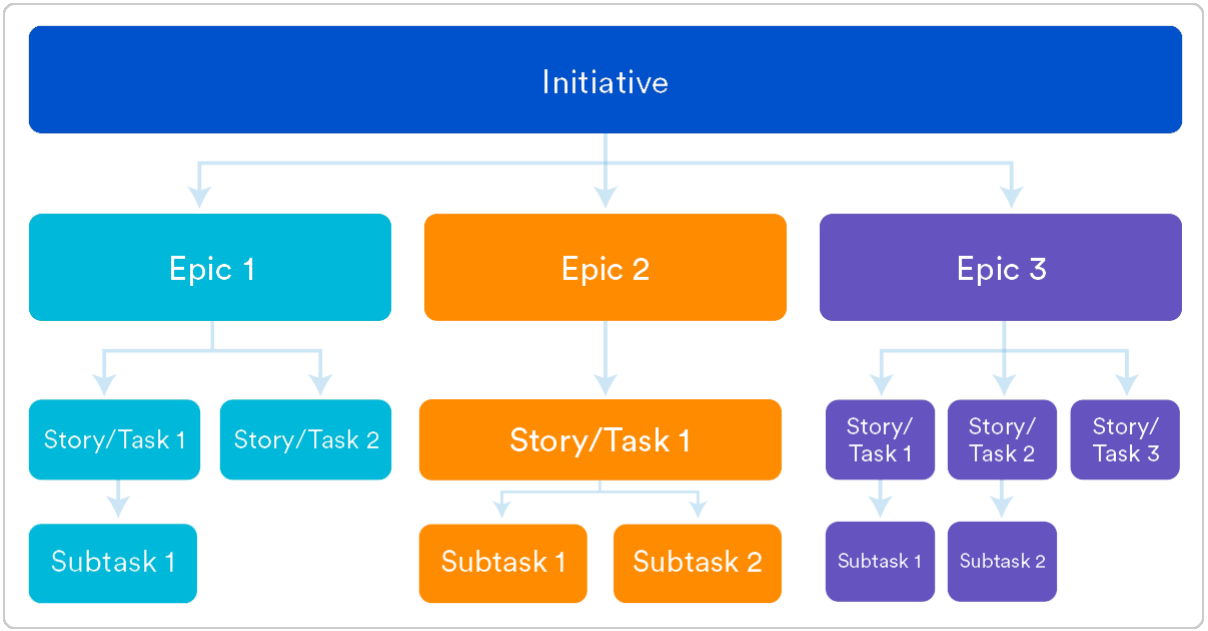

Рисунок 4 – Процес дрібнення ініціатив до рівня User Story та Task

На рисунку 5 відображено процес декомпозиції робіт та формування пакетів робіт командам виконавців

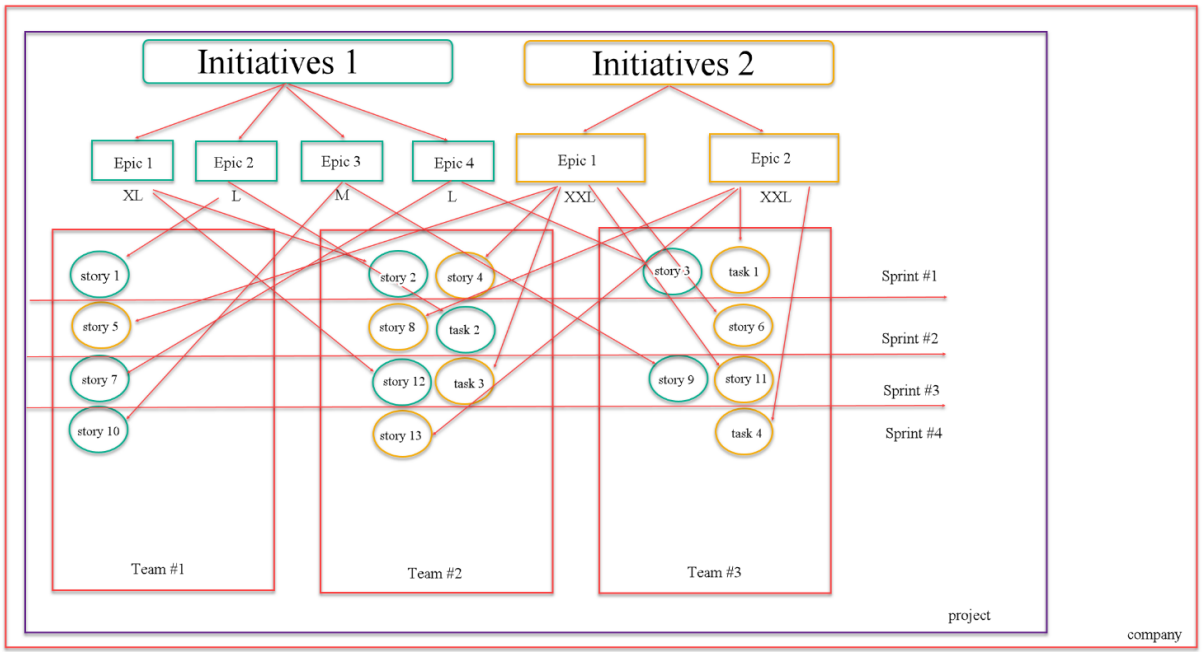

Рисунок 5 – Процес декомпозиції робіт та формування пакетів робіт командам виконавців

1.2 Аналіз наявних методів та підходів до вирішення задачі призначення пакетів робіт командам виконавців IT-проєкту

Під час аналізу існуючих методів вирішення проблеми автоматизації призначення пакетів робіт командам виконавців IT-проєкту, було виявлено декілька різноманітних підходів:

- ручне призначення;
- використання штучного інтелекту в системах управління;
- використання платформ для управління проєктами.

Розберемо більш детально кожний з наведених вище підходів.

Ручне призначення задач - традиційний підхід, при якому керівники проєктів вручну призначають пакети робіт командам виконавців IT-проєкту.

Метод ручного призначення пакетів робіт командам виконавців IT-проєкту має таки переваги:

− гнучкість – саме ручне призначення задач дозволяє менеджерам проєкту робити цей процес гнучким при розподіленні пакетів робіт серед команд виконавців відповідно до умов;

− легкість в керуванні – ручне призначення є простим у виконанні, особливо в рамках невеликих проєктів, де командна комунікація знаходиться на високому рівні та є можливість безпосередньої взаємодії між членами команди;

− врахування особистих переваг – менеджер має можливість призначати завдання командам виконавців, враховуючи особисті переваги та інтереси потенційних виконавців;

− процес призначення пакетів робіт виконавцям відбувається майже завжди на етапі створення пакетів робіт у вигляді історій користувача або технічних завдань.

Слабкі сторони методу ручного призначення пакетів робіт командам виконавців ІТ проєкту:

− в рамках великого проекту з розділеними командами в різних тайм зонах, можливо обмежена прогнозованість – ручне призначення є менш прогнозованим, оскільки воно базується на особистих оцінках та аналітичних висновках менеджера проєкту щодо досвіду виконавців. Беззаперечно це може призвести до недооцінювання або переоцінювання часу на виконання або ресурсів, які потрібно залучити;

− ризик помилок – при ручному призначення існує ризик помилок та непорозумінь, особливо під час роботи зі складними або технічно вимогливими завданнями, що в свою чергу зумовлює затримки виконання задач та може бути неефективним у великих проєктах.

Використання штучного інтелекту (ШІ) в системах управління – це новітній підхід, який полягає у використанні алгоритмів штучного інтелекту для аналізу даних та прийняття рішень щодо призначення пакетів робіт. Цей метод здатен враховувати індивідуальні особливості проєкту та команд виконавців ІТ-проєктів, що дозволяє оптимізувати процес. Але, використання

алгоритмів штучного інтелекту потребує додаткових даних для навчання та створення необхідної моделі.

Сильні сторони використання штучного інтелекту (ШІ) в системах управління для призначення пакетів робіт командам виконавців ІТ проєкту:

− швидкість – ШІ здатен доволі ефективно аналізувати великі обсяги даних за короткий проміжок часу, що в свою чергу дає можливість ефективніше та швидше виконувати призначення задач, ніж при ручному призначенні;

− автоматизація – автоматизація процесу звільняє ресурси та час який може бути спрямований на вирішення інших пріоритетних задач.

Слабкі сторони використання штучного інтелекту (ШІ) в системах управління для призначення пакетів робіт командам виконавців ІТ-проєкту:

− нехтування особистих преференцій – з великою долею вірогідністю, ШІ не буде враховувати особисті преференції, що в свою чергу може сприяти неоптимальному розподілу задач;

− потреба в налаштуваннях та навчанні – залучення ШІ до використання в рамках проєкту потребує постійного навчання моделей в залежності від змін вхідних даних та підтримки щодо актуальності, а це вимагає додаткових часо- та ресурсозатрат;

− ризики помилок в алгоритмах – алгоритми дуже чутливі щодо вхідних даних, що в свою чергу може бути наслідком неоптимального призначення, і як слідство затримка виконання задач.

На ринку програмних продуктів існують сучасні програмні платформи для управління проєктами, які в якоїсь мірі автоматизують процеси формування та призначення пакетів робіт. До них можна віднести таки інструменти управління проєктами, як Jira та ClickUp.

Після проведеної роботи щодо ознайомлення та вивчення можливостей налаштування та використання вищезгаданих платформ для автоматизованого процесу призначення пакетів робіт, зроблені певні висновки.

ClickUp – сервіс який надає певний спектр послуг для таск-менеджмента (див. рисунок 6). Говорячи про можливість автоматизації призначення пакетів робіт виконавцям, можна знайти реалізацію у вигляді можливості налаштування певних правил та фільтрів, які наприклад дають змогу автоматично призначати виконавця, наприклад у разі, коли статус завдання змінено.

Функціонал створення правил та фільтрів у сервісі ClickUp є обмеженим, що в свою чергу не дає можливості автоматизувати призначення пакетів робіт за умов наприклад використання ключових слів, або за умови зміни пріоритету завдання.

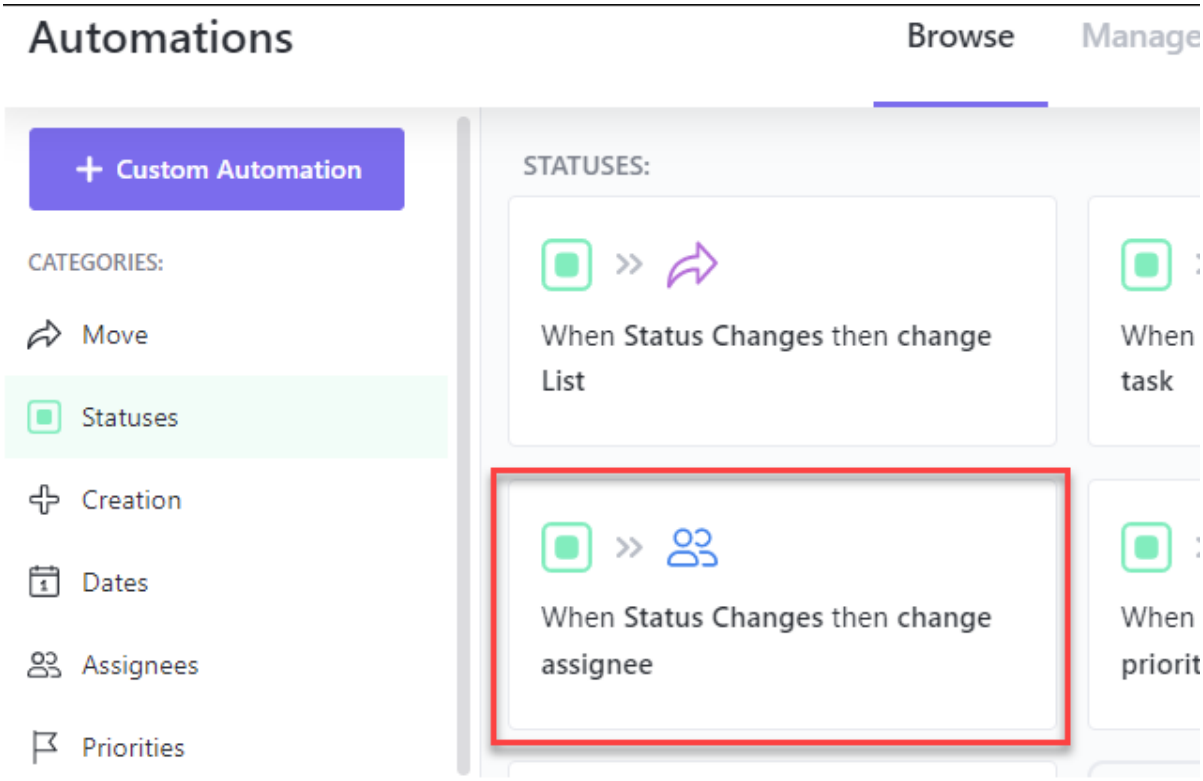

Рисунок 6 – Демонстрація можливостей автоматизації розподілу завдань в сервісі ClickUp

Jira – це доволі відомий програмний продукт який має дуже потужні можливості та велику кількість додатків та налаштувань (див. рисунок 7). З

точки зору можливостей автоматизації процеса призначення пакетів робіт командам так само має певні обмеження. Реалізація доволі схожа з попередньо розглянутим сервісом, тобто маємо можливість створити певні правила та фільтри, за якими автоматизовано виконується призначення пакету робіт у разі, коли виконавець закінчив роботу над попереднім завданням, і наразі немає на собі ніяких пакетів робіт. Тоді перше завдання з продуктового беклогу буде призначено цьому виконавцю. Але знову, немає можливості більш гнучкого налаштування правил, скажімо, з урахуванням ключових слів або з урахуванням пріоритету завдання.

На рисунку 7 відображено можливості сервісу Jira з автоматизації призначення завдань виконавцям.

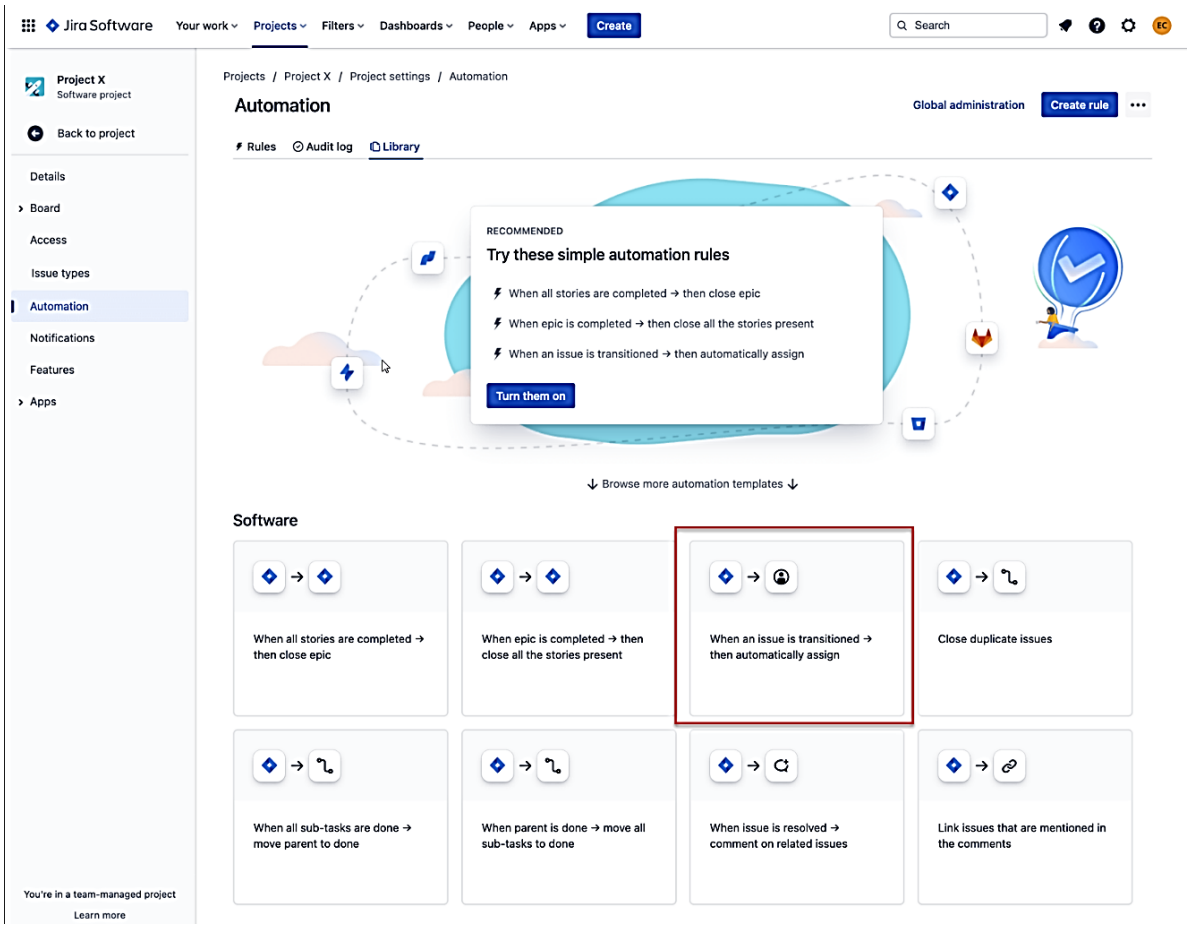

Рисунок 7 – Можливості з автоматизації призначення завдань виконавцям в інструменті управління прєктами Jira

Роблячи висновки, можна стверджувати, що на даний момент всі вище згадані інструменти мають обмежені можливості щодо автоматизації процесу призначення пакетів робіт. Вони не дозволяють автоматично призначати завдання виконавцям та формувати пакети завдань командам виконавців.

1.3 Постановка задач магістерської кваліфікаційної роботи

Об’єктом дослідження в рамках магістерської кваліфікаційної роботи є процес призначення пакетів завдань командам виконавців ІТ-проєкту.

Предметом дослідження є моделі та методи, які можна використовувати при формуванні та призначенні пакетів завдань командам виконавців.

Проблемою дослідження є недостатній ступінь автоматизації та відсутність оптимальності розподілу робіт, яка б забезпечувала високу ефективність процесу виконання ІТ-проєкту.

Метою дослідження є застосування методів класифікації до автоматизації процесу призначення пакетів робіт командам виконавців ІТ проєкту.

При виконанні магістерської роботи для досягнення поставленої мети потрібно вирішити такі задачі дослідження:

− провести аналіз об’єкта дослідження та сучасного стану вирішення проблеми;

− сформувати критерії та вимоги до розподілу робіт з виконання ІТ-проєкту між командами виконавців;

− дослідити можливість та особливості використання математичного апарату методів класифікації для вирішення задачі розподілу робіт між командами виконавців;

− розробити алгоритм розподілу робіт між командами виконавців з врахуванням використання методів класифікації;

− виконати експериментальну перевірку результатів роботи та порівняти ефективність автоматизованого призначення пакетів завдань командам виконавців ІТ-проєкту з ручним.

2 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ МЕТОДУ ПРИЗНАЧЕННЯ ПАКЕТІВ РОБІТ КОМАНДАМ ВИКОНАВЦІВ ІТ-ПРОЄКТУ

Процес розподілу завдань виконавцям напередодні нового спринту обмежений кількома критеріями:

- пріоритетом виконання завдання;
- можливою належністю завдання епіку, який закріплений за однією чи декількома командами виконавців;
- сумарною трудомісткістю на один спринт кожного виконавця (вона залежить від посади), яка не повинна бути перевищена під час розподілу завдань;
- можливою наявністю спеціалізації виконавців усередині команди.

Необхідність врахування таких критеріїв при розподілі завдань виконавцям унеможливлює вирішення проблеми лише застосуванням одного якогось відомого математичного методу. Але проблему можна вирішити, якщо розробити алгоритм розподілу завдань виконавцям, частиною етапів якого буде використання методу класифікації завдань за їх найменуваннями.

При вирішенні схожих завдань класифікації добре зарекомендував себе наївний байєсівський метод. Тому розглянемо методи вирішення проблеми класифікації за допомогою цього метода.

2.1 Задачі класифікації

Класифікація є процесом, що спрямований на розподіл об'єктів або подій на кілька визначених класів або категорій на основі їхніх характеристик або властивостей. Цей процес полягає у навчанні моделі на основі доступних даних, після чого вона може класифікувати нові об'єкти або події відповідно до вивчених патернів.

Загальну концептуальну постановку задачі класифікації можна сформулювати таким чином.

Мається множина об’єктів A, які розподілені якимось чином на класи. Задана кінцева множина об’єктів B, для яких відомо, до яких класів вони відносяться (ця множина називається навчаючою вибіркою). Класова належність інших об’єктів множини A не відома. Необхідно побудувати алгоритм, який спроможний класифікувати будь-який об’єкт з вихідної множини A. Під класифікацією будемо розуміти процедуру визначення номера або найменування класу, до якого відноситься об’єкт, що класифікується.

Множина класів має бути:

− скінченною;

− такою, що рахується;

− апріорно визначена – усі класи мають бути відомі до початку вирішення задачі.

Нехай X – множина описів об'єктів класифікації, а Y – множина класів, тоді існує невідома цільова залежність $X^m = \{(X_1, Y_1), \dots, (X_m, Y_m)\}$, значення якої відомі тільки на об'єктах кінцевої навчальної вибірки $X^m = \{(X_1, Y_1), \dots, (X_m, Y_m)\}$. Необхідно побудувати алгоритм $\lambda: X \to Y$, здатний класифікувати довільний об'єкт $x \in X$.

Класифікація вирішує різноманітні завдання у багатьох галузях. Наприклад, вона допомагає класифікувати об'єкти, розподіляючи їх (або події, явища) до певних категорій або класів на основі їхніх характеристик. Вона також може використовуватися для прогнозування, визначаючи клас або категорію, до якої належить новий об'єкт, на основі вивчення зразків даних. Виявлення аномалій є ще однією важливою задачею класифікації, оскільки воно дозволяє визначати відхилення або аномалії у даних, що можуть вказувати на потенційні проблеми або виняткові ситуації.

Крім того, класифікація застосовується для фільтрації даних, де вона сортує дані за критеріями, щоб виділити ті, які відповідають певним умовам

або категоріям. І нарешті, класифікація використовується для ранжування, визначаючи порядок або пріоритет об'єктів на основі їхніх характеристик або значень.

Цей широкий спектр задач робить класифікацію незамінним інструментом у багатьох сферах, включаючи ІТ-проєкти, де вона може допомогти автоматизувати процеси, підвищуючи ефективність та якість виконання завдань.

2.2 Дослідження алгоритмів вирішення задачі класифікації

Аналіз існуючих алгоритмів вирішення задачі класифікації дозволяє розглянути різні підходи до вирішення цієї задачі та порівняти їх ефективність у різних умовах. Під час такого аналізу розглядаються переваги та недоліки кожного алгоритму, його швидкодія, точність, стійкість до перенавчання та інші важливі аспекти.

Одним із найпростіших та широко використовуваних алгоритмів класифікації є наївний байєсівський класифікатор [24]. Цей метод ґрунтується на теоремі Байєса та використовує ймовірнісний підхід до класифікації об'єктів.

Теорема Байєса – це математичний принцип, який дозволяє обчислити умовну ймовірність події A, знаючи умовну ймовірність події B, що настала.

Формально теорему Байєса можна записати так:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}, \tag{2.1}$$

де P(A|B) – умовна ймовірність події A при умові, що подія B відбулась;

P(B|A) – умовна ймовірність події B при умові, що подія A відбулась;

P(A) – апріорна (початкова) ймовірність події A;

P(B) – апріорна (початкова) ймовірність події B.

Теорема Байєса застосовується в багатьох галузях, включаючи машинне навчання, статистику, медицину, теорію прийняття рішень і т.д. У контексті машинного навчання, теорема Байєса використовується в методі байєсівської класифікації для прийняття рішень щодо класифікації об'єктів на основі їх властивостей.

Основні переваги наївного байєсівського класифікатора такі:

- простота та легка реалізація;
- ефективність при роботі з великими обсягами даних;
- висока швидкодія, особливо на текстових даних;
- добрі результати в умовах обмежених обчислювальних ресурсів.

Проте наївний байєсівський класифікатор також має свої обмеження:

- він передбачає незалежність між ознаками, що може бути неправдивим у реальних даних;
- цей метод може бути чутливим до неправильного форматування даних.

Методи наївного байєсівського класифікатора представлені різними алгоритмами, які використовуються для вирішення задач класифікації. Ці алгоритми різняться за своєю придатністю до вхідних даних та математичними моделями, які вони використовують для прогнозування класів об'єктів [25].

2.3 Дослідження та вибір класу алгоритмів наївного байєсівського класифікатора

Для ефективного вирішення задач автоматизованого призначення пакетів робіт командам виконавців IT-проєкту необхідно обрати відповідний клас алгоритмів класифікації, який зможе ефективно вирішити поставлену задачу.

Метою класифікації є оптимальний розподіл об’єктів між класами з урахуванням їх характеристик. Класифікатор повинен мати можливість працювати з великою кількістю ознак, як числових, так і категоріальних, а також враховувати можливу кореляцію між ними.

Розглянемо декілька підходів та алгоритмів, які можуть бути застосовані для нашого завдання.

2.3.1 Гаусівський наївний байєсівський класифікатор

Гаусівський наївний байєсівський класифікатор — це статичний алгоритм машинного навчання, який базується на теоремі Байєса та припущенні, що значення кожної ознаки у вхідних даних має гаусівський (нормальний) розподіл. Під час класифікації нового випадку, алгоритм спочатку визначає ймовірність належності цього випадку до кожного класу, використовуючи густину ймовірності гаусівського розподілу для кожної ознаки, а потім застосовує формулу Байєса для розрахунку остаточних ймовірностей.

Для гаусівського наївного байєсівського класифікатора, ймовірність $P(x_l|C_k)$ кожної ознаки x_l за умови класу C_k обчислюється за формулою густини ймовірності нормального розподілу:

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x_i - \mu_k)^2}{2\sigma_k^2}\right), \quad (2.2)$$

де μ_k – середнє значення ознаки для класу C_k;

σ_k^2 - стандартне відхилення ознаки для класу C_k;

x_i – значення ознаки.

Вхідні дані для гаусівського наївного байєсівського класифікатора складаються з набору ознак, які характеризують об'єкти, що класифікуються, та міток класів, які визначають категорії, до яких належать ці об'єкти.

Переваги використання цього алгоритму включають ефективність в обробці даних з багатьма ознаками, відносну простоту реалізації та невимогливість до великої кількості даних для навчання. Він також здатний працювати з числовими та категоріальними ознаками.

Однак, є деякі обмеження. Наприклад, потреба у припущенні про гаусівський розподіл ознак у кожному класі, а також необхідність вважати ознаки незалежними, навіть якщо вони насправді корелюють між собою. Крім того, алгоритм чутливий до викидів у даних.

Цей алгоритм базується на припущенні, що значення кожної ознаки у вхідних даних мають гаусівський (нормальний) розподіл. Гаусівський наївний Байєсівський класифікатор є ефективним у випадку, коли дані є неперервними і підпорядковуються нормальному розподілу. Проте, у нашому завданні дані можуть мати інший розподіл, а також містити категоріальні ознаки, що знижує доцільність використання цього методу.

2.3.2 Мультиноміальний наївний байєсівський класифікатор

Мультиноміальний наївний байєсівський класифікатор — це статистичний алгоритм машинного навчання, який використовується для класифікації об'єктів з дискретними ознаками. Основна ідея цього методу полягає у використанні теореми Байєса для визначення ймовірності того, що об'єкт належить до певного класу, враховуючи ймовірності ознак цього класу.

Цей метод має кілька переваг. Він добре працює з категоріальними ознаками, які часто зустрічаються у специфікаціях пакетів робіт. Мультиноміальний наївний байєсівський класифікатор є відносно простим в реалізації і не потребує складних обчислень, що робить його доступним для

широкого використання. Крім того, цей метод добре працює навіть у випадку кореляції між ознаками.

Однак, мультиноміальний наївний байєсівський класифікатор має і деякі обмеження. Деякі числові дані можуть потребувати категоризації або перетворення у категоріальні для використання цього методу. Метод передбачає мультиноміальний розподіл для кожного класу, що може бути не завжди точним при моделюванні деяких даних. Вхідними даними для цього алгоритму є набір об'єктів, для яких відомі значення дискретних ознак, та мітки класів, до яких належать ці об'єкти.

Процес класифікації починається з побудови моделі, під час якої обчислюються ймовірності появи кожної ознаки для кожного класу, а також апріорні ймовірності кожного класу. Після побудови моделі можна класифікувати нові об'єкти, обчислюючи ймовірність належності об'єкта до кожного класу, використовуючи відомі ознаки об'єкта та параметри моделі.

Для застосування цього методу необхідно спочатку підготувати дані, тобто перевести всі ознаки пакетів робіт у дискретний формат. Далі, використовуючи навчальний набір даних, побудувати модель. Після навчання модель можна використовувати для класифікації нових пакетів робіт, визначаючи ймовірність призначення кожного пакету робіт до певного виконавця.

Формула для ймовірності належності нової точки даних до класу y_k виглядає так:

$$P(y_k|X) = \frac{P(X|y_k) \cdot P(y_k)}{P(X)}, \qquad (2.3)$$

де $P(y_k|X)$ – апостеріорна ймовірність класу y_k за умови даних X;

$P(X|y_k)$ – ймовірність даних X за умови класу y_k;

$P(y_k)$ – апріорна ймовірність класу C_k;

$P(X)$ – ймовірність даних X .

Ця формула дозволяє розрахувати ймовірність того, що новий об’єкт належить до певного класу, враховуючи відомі ознаки об’єкта та параметри моделі.

Якщо об'єкт описується не однією, а декількома ознаками $X_1, X_2, ..., X_n$, то формула набуває вигляду:

$$P(y_k \mid X_1, X_2, ..., X_n) = \frac{P(y_k)\prod_{i=1}^{n} P(X_i \mid y_k)}{P(X_1, X_2, ..., X_n)} \quad (2.4)$$

Тобто, це апостеріорна ймовірність приналежності об'єкта до класу y_k з урахуванням всіх його ознак.

В цієї формулі лише чисельник становить інтерес. Знаменник може бути опущений при порівнянні можливостей різних класів, оскільки він залежить тільки від ознак об'єкта. У остаточному підсумку правило класифікації буде пропорційно вибору класу з максимальною апостеріорною ймовірністю:

$$y_k \propto \arg\max P(y_k)\prod_{i=1}^{n} P(X_i \mid y_k) \ , \quad (2.5)$$

де $\propto$ – знак пропорційності.

Для оцінки параметрів моделі, тобто ймовірностей $P(y_k)$ и $P(X_i \mid y_k)$, зазвичай застосовується метод максимальної правдоподібності, який в даному випадку заснований на частотах класів і ознак у навчальній вибірці.

Апріорна ймовірність при заданому класі y належності об'єкта до класу розраховується за формулою:

$$P(y) = \frac{O_y}{O} \ , \quad (2.6)$$

де O_y – число об'єктів в навчальної виборці, що входять до класу y;

O – загальне число об'єктів в навчальної виборці.

Ймовірність ознаки при заданому класі y обчислюється за формулою:

$$P(x_i \mid y) = \frac{N_{yi} + \alpha}{N_y + \alpha n}, \quad (2.7)$$

де N_{yi} – це кількість разів, коли ознака i зустрічається в об'єктах, що входять до класу y;

N_y – загальна кількість всіх ознак об'єктів навчальної вибірки, що входять до класу y;

n – кількість різних ознак в навчальної виборці;

α – параметр згладжування, що запобігає виникненню нульових ймовірностей.

Цей алгоритм добре працює з категоріальними ознаками, які часто зустрічаються у специфікаціях пакетів робіт. Він є відносно простим в реалізації і не потребує складних обчислень. Мультиноміальний наївний байєсівський класифікатор може ефективно використовуватися для класифікації завдань, де ознаки мають дискретний характер. Це робить його придатним для нашого завдання.

2.3.3 Наївний байєсівський класифікатор Бернуллі

Наївний байєсівський класифікатор Бернуллі — це статистичний алгоритм машинного навчання, який використовується для бінарної класифікації, тобто розділення об'єктів на два класи. Вхідні дані для цього алгоритму складаються з набору об'єктів та їх ознак, представлених у вигляді бінарних значень (1 або 0), де 1 вказує на наявність ознаки, а 0 — на її відсутність. Кожен об'єкт також має мітку класу (1 або 0), яка вказує на приналежність до одного з двох класів.

Алгоритм будує модель, в якій обчислюється ймовірність наявності кожної ознаки для кожного класу, а також апріорна ймовірність кожного класу. Для класифікації нових об'єктів обчислюється ймовірність належності кожного об'єкта до кожного класу, використовуючи відомі ознаки об'єкта та параметри моделі.

Цей метод має свої переваги та обмеження [26]. Зокрема, алгоритм добре працює з бінарними даними і є відносно простим в реалізації. Однак, він базується на "наївних" припущеннях про незалежність між ознаками, що може бути неправильним для деяких типів даних, де ознаки можуть бути взаємозалежними. Якщо в навчальному наборі даних є дуже рідкісні ознаки або ознаки, які ніколи не з'являлися разом, це може призвести до низької точності класифікації. Крім того, алгоритм не враховує контекст аналізу, що може призвести до неправильних прогнозів у складних ситуаціях.

Цей алгоритм використовується для бінарної класифікації і добре працює з ознаками, представленими у вигляді бінарних значень (1 або 0), що вказують на наявність або відсутність ознаки. Наївний байєсівський класифікатор Бернуллі може бути корисним, якщо ознаки пакетів робіт можна представити у бінарному форматі. Однак, цей підхід має обмеження, оскільки не всі ознаки можна легко перетворити на бінарні значення.

2.3.4 Комплементарний наївний Байєсівський класифікатор

Комплементарний наївний байєсівський класифікатор – це варіація наївного байєсівського класифікатора, яка використовується для рішення задач класифікації. У цьому методі використовується те саме базове припущення, що всі ознаки незалежні одна від одної, але відповіді на класи розглядаються як випадкові змінні, які мають різний розподіл, ніж у звичайному наївному байєсівському класифікаторі.

Вхідні дані для комплементарного наївного байєсівського класифікатора повинні містити набір ознак, які мають категоріальний або числовий формат. Кожна ознака може бути представлена числовим значенням або категоріальною змінною. Також потрібно мати відповідні класи або мітки, за якими буде проводитися класифікація.

2.3.5 Категоріальний наївний Байєсівський класифікатор

Варіант наївного байєсівського класифікатора, який працює з категоріальними ознаками. У цьому методі вважається, що кожна ознака має дискретний набір можливих значень, а розподіл ймовірностей відносно класів обчислюється за формулою наївного байєсівського класифікатора, але враховуючи тільки категоріальні ознаки.

Вхідні дані для категоріального наївного байєсівського класифікатора повинні містити категоріальні ознаки, тобто ознаки, що можуть приймати обмежений набір значень або категорій. Формат даних може бути у вигляді таблиці, де кожен стовпець відповідає окремій ознаці, а кожен рядок - окремому спостереженню або прикладу. Для класифікації потрібно мати також відповідні мітки класів для навчальних даних.

2.3.6 Результат аналізу та вибір алгоритму

Після проведеного аналізу наявних алгоритмів було визначено, що більш прийнятним для нашого завдання є мультиноміальний наївний байєсівський класифікатор (MultinominalNB). Вагомим аргументом того що саме цей класифікатор найкраще підходить для наших досліджень є те, що дані містять дискретні ознаки, такі як кількість входжень певних слів у тексті, що відповідає мультиноміальному розподілу. По друге, мультиноміальний

наївний байєсівський класифікатор є простим в реалізації та швидким, що дозволяє обробляти великі обсяги даних.

Саме цей алгоритм забезпечить надійність і точність класифікації завдань, враховуючи їх ознаки (слова, з яких складаються найменування завдань) та історичні дані. Таким чином, мультиноміальний наївний байєсівський класифікатор є оптимальним вибором для нашої задачі з класифікації.

2.4 Модифікація алгоритму класифікації на основі мультиноміального наївного Байєсівського методу

Алгоритм класифікації на основі класичного мультиноміального наївного баєсовського методу складається з семи кроків (див. рисунок 9).

На першому кроці алгоритму виконується формування навчальної вибірки об'єктів, що передбачає збір та підготовку даних для навчання класифікатора. Навчальна вибірка має включати перелік об'єктів, які вже були класифіковані раніше, та перелік ознак кожного такого об'єкта.

Другий крок включає перевірку гіпотези про незалежність окремих ознак об'єктів в навчальній вибірці одна від одної. Це важливий крок, оскільки наївний баєсівський класифікатор базується на припущенні, що всі ознаки незалежні між собою.

На третьому кроці відбувається виділення множини класів, до яких можуть належати об'єкти. Це передбачає визначення всіх можливих категорій, які будуть використовуватися для класифікації нових об'єктів.

На четвертому кроці обчислюється апріорна ймовірність належності випадково обраного об’єкту до класу. Апріорна ймовірність обчислюється як відношення кількості об’єктів, що належать до класу, до загальної кількості об’єктів у навчальній вибірці. Розрахунок відбувається за формулою (2.6).

П'ятий крок передбачає обчислення ймовірності кожної ознаки об'єкта при кожному заданому класі. Визначення умовних ймовірностей ознак для кожного класу здійснюється на основі частот ознак у таблиці частоти входжень ознак. Розрахунок відбувається за формулою (2.7).

На шостому кроці виконується обчислення апостеріорної ймовірності приналежності об'єкта до кожного класу з урахуванням всіх його ознак. На основі попередніх обчислень визначається ймовірність того, що об'єкт належить до кожного класу, враховуючи всі його ознаки. Розрахунок відбувається за формулою (2.4).

На сьомому кроці виконується пошук класу з максимальною апостеріорною ймовірністю приналежності об'єкта до нього та віднесення об'єкта до цього класу.

Після виконання кроку 7 здійснюється перевірка, чи є ще черговий об'єкт для класифікації. Алгоритм закінчує роботу у випадку, якщо всі об'єкти класифіковано.

Таким чином, наведений алгоритм дозволяє ефективно виконувати класифікацію об’єктів на основі мультиноміального наївного байєсівського методу, враховуючи різні ознаки об’єктів та їх ймовірності приналежності до певних класів.

Цей алгоритм забезпечує ефективну класифікацію об’єктів на основі їхніх ознак, використовуючи ймовірнісний підхід на основі теореми Байєса.

Для вирішення задач магістерської кваліфікаційної роботи класичний алгоритм класифікації на основі мультиноміального наївного байєсівського методу не зовсім підходить у зв'язку з тим, що в ньому процес класифікації продовжується доки є об'єкти, які ще не класифіковані та кількість об'єктів, що можуть належати до класу ніяк не обмежується.

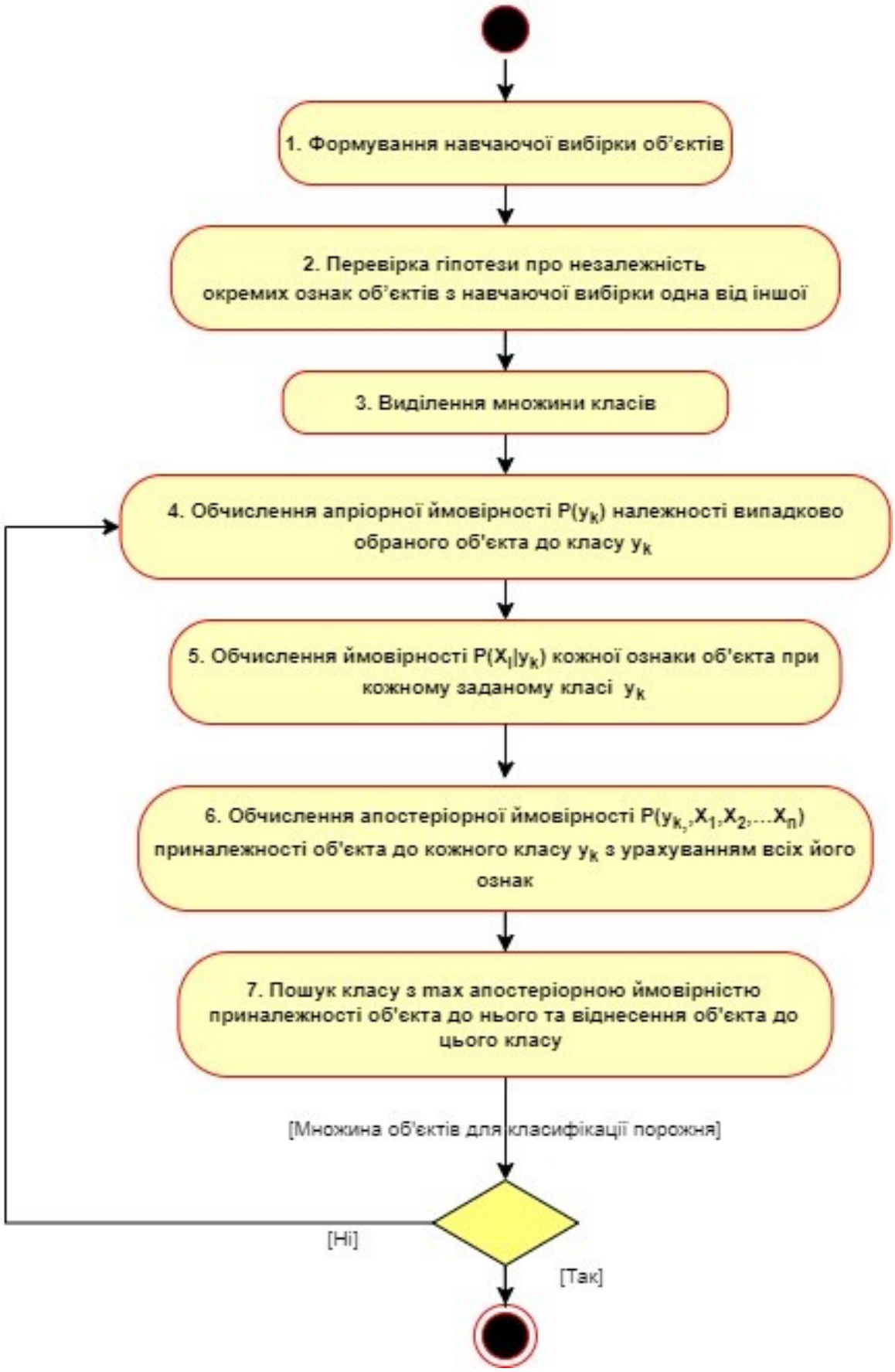

Рисунок 9 – Алгоритм класифікації на основі класичного мультиноміального наївного байєсівського методу

При розподілу завдань виконавцям процес розподілу має продовжуватися доки кожний виконавець не набере необхідну для його посади сумарну трудомісткість. Тобто в класичному алгоритмі необхідно змінити (обмежети) умови додавання об'єкта до класу.

Тому до класичного алгоритму були внесені зміни. Схема модифікованого алгоритму класифікації на основі мультиноміального наївного байєсівського методу наведено на рисунку 10.

Відповідно до модифікації алгоритму, кожному класу має бути приписана максимально допустима вага $v_{max\,k} > 0$. Кожному новому об'єкту класифікації має бути також приписана вага $v_1 > 0$.

Як тільки в результате розрахунків апостеріорної ймовірності належності об'єкта до кожного класу визначається клас до якого має належати черговий об'єкт, тоді необхідно виконати перевірку загальної накопиченої ваги класу s_{vk} разом з вагою нового об'єкта на те, що $S_{vk} < v_{max\,k}$. Якщо ця умова порушена, тоді апостеріорна ймовірність належності об'єкта до цього класу має бути обнулена, а пошук класу з максимальною апостеріорною ймовірністю приналежності об’єкта до нього має бути продовжено.

У зв'зку з описаною модифікацією класичного алгоритму до нього було додано п’ять нових кроків та один (з номером 7) змінено.

Четвертим кроком було додано визначення максимально допустимої сумарної ваги наповнення кожного класу $v_{max\,k}$. Це здійснюється за допомогою встановлення обмежень на загальну вагу об’єктів, які можуть бути призначені до кожного класу. Цей крок допомогає уникнути перевантаження окремих класів та забезпечує більш збалансований розподіл.

П'ятий крок передбачає визначення ваги кожного об'єкта класифікації. Вага об’єкта визначається на основі характеристик об’єктів, що впливають на їх вагу. Вага може бути визначена на основі історичних даних або експертних оцінок.

На дев’ятому кроці (це змінений крок) виконується пошук класу з максимальною апостеріорною ймовірністю приналежності об'єкта до нього та віднесення об'єкта до цього класу.

Після виконання дев'ятого кроку виконується перевірка загальної ваги класу разом з вагою нового об'єкта на перевищення допустимого максимуму. У разі перевищення ваги класу разом з вагою нового об'єкта допустимого максимуму виконується крок десять: апостеріорна ймовірність для цього класу обнуляється, і після цього знов виконується крок дев'ять – процес повторюється для наступного класу з найбільшою ймовірністю.

Якщо загальна вага класу не перевищує допустиму, виконується одинадцятий крок: віднесення об'єкта до класу з найбільшою апостеріорною ймовірністю.

Умова завершення модифікованого алгоритму залишилась такою ж, як і в класичному алгоритму.

Ці п’ять кроків модифікують алгоритм класифікації таким чином, що він дозволяє враховувати вагу об’єктів та максимально допустиму вагу для кожного класу, тим самим забезпечуючи ефективний та збалансований розподіл об’єктів між класами.

На рисунку 10 наведено схема модифікованого алгоритму класифікації на основі мультиномінального наївного байєсівського методу.

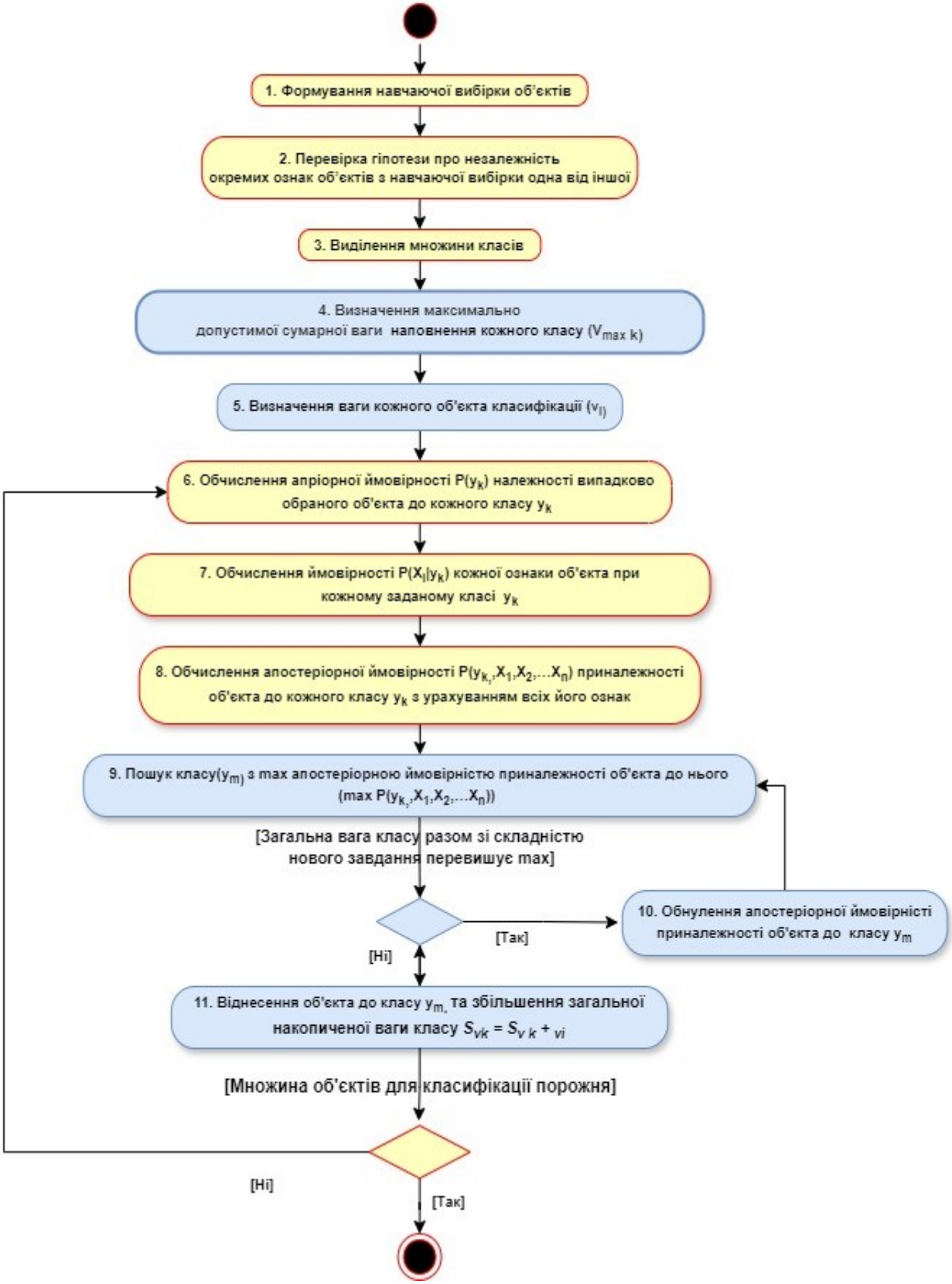

Рисунок 10 – Схема модифікованого алгоритму класифікації на основі мультиноміального наївного байєсівського методу

3 УДОСКОНАЛЕННЯ МЕТОДУ ПРИЗНАЧЕННЯ ПАКЕТІВ РОБІТ КОМАНДАМ ВИКОНАВЦІВ ІТ-ПРОЄКТУ

3.1 Адаптація мультиномінального наївного байєсівського класифікатора для вирішення завдань магістерської роботи

Для використання мультиномінального наївного байєсівського класифікатора в методі призначення пакетів робіт командам виконавців ІТ-проекту потрібно виконати його адаптацію до задач магістерської роботи.

Метою використання методу класифікації в роботі є призначення завдань до початку чергового спринта окремим виконавцям та забезпечення рівномірного розподілу завдань з урахуванням трудомісткості, компетенцій виконавців та команд.

Об'єктами класифікації наївного байєсівського класифікатора мають бути завдання, які представлені у вигляді User Story та Task. Тоді ознаками об'єкта класифікації (завдання), що враховуються в процесі класифікації мають бути слова, з яких складається найменування завдання. Найменування завдань мають відображати характерні особливості проблемної області або частини проєкту.

Кожному завданню потрібно присвоїти вагу у вигляді трудомісткості виконання, яка вимірюється в сторі поїнтах.

Класами, за якими виконується розподіл об'єктів (завдань), в нашому випадку мають бути окремі виконавці, які входять до якоїсь команди виконавців. Кожен виконавець має свою максимальну трудомісткість для одного спринту, яка відповідає поняттю максимально допустимої ваги класу.

Навчальна вибірка для класифікації має містити дані про вже виконані завдання (їх найменування) та про виконавців цих завдань.

Апріорна ймовірність належності завдання заданому виконавцю e розраховується за формулою:

$$P(e) = \frac{T_e}{T}\ , \tag{3.1}$$

де T_e – число завдань у навчальній вибірці, які виконав заданий виконавець e;

T – загальне число завдань у навчальній вибірці.

Ймовірність кожного слова завдання, що розподіляється, при заданому виконавці e обчислюється за формулою:

$$P(\omega_i|e) = \frac{N_{\omega_i e} + \alpha}{Ne + \alpha V}, \tag{3.2}$$

де $N_{\omega_i e}$ – кількість поява слова ω_i у завданнях в навчальній вибірці, які були виконані виконавцем e.

N_e – загальна кількість слів у завданнях в навчальній вибірці, які були виконані виконавцем e.

V – розмір словника (загальна кількість слів в навчальній вибірці).

α – коефіцієнт згладжування.

3.2 Опис підготовчого етапу методу призначення пакетів робіт командам виконавців IT-проєкту

Створення ефективного методу призначення пакетів робіт командам виконавців IT-проєкту потребує ретельної підготовки, щоб забезпечити точність і дієвість запропонованого підходу. Підготовчий етап є ключовим, адже саме на цьому етапі закладаються основи для подальшої роботи алгоритму.

Першим кроком підготовчого етапу є збір даних. Це включає в себе акумулювання інформації про всі поточні та минулі завдання.

Важливо зібрати історичні дані про виконані завдання, зокрема інформацію про виконавців, які їх виконували, та результати їхньої роботи. Ці дані формують базу для аналізу та навчання моделі. Вона готується перед першим використанням системи, але потім може поповнюватися, якщо автоматизований розподіл завдань буде вдалим. Ці дані слугують основою для навчання моделі, що дозволяє класифікувати нові завдання.

Після збіру історичних даних необхідно перевірити гіпотезу про незалежність окремих слів у найменуваннях завдань шляхом попарного кореляційного аналізу кожного поєднання із двох слів. Це можна зробити, наприклад, порахувавши для кожної пари слів, скільки разів зустрічається таке поєднання у найменуваннях завдань навчальної вибірки та співвіднести цю кількість до загальної кількості пар слів у навчальній вибірці. Тоді, якщо процент кореляції буде менше або дорівнює 5% (визначено експертним шляхом), можна вважати, що кореляція між цими словами відсутня.

За наявності кореляції між двома словами їх можна вважати однією ознакою, тобто одним словом.

Інформація про поточні завдання, які потрібно буде розподіляти між виконавцями, містить їх найменування, пріоритетність, трудомісткість, епік, до якого належить завдання (необов'язкова для заповнення ознака). Пріоритетність завдань може бути різною і визначається в залежності від важливості та терміновості виконання завдань. В рамках нашої роботи ми використовуємо наступні рівні пріоритетності:

− високий пріоритет може бути у завдань, які є найбільш критичними для проєкту та повинні бути виконані в першу чергу. Він визначається числовим значенням 1. Це може бути завдання, які впливають на ключовий функціонал або терміни проєкту;

− середній пріоритет – це коли завдання мають важливе значення, але не є критичними для негайного виконання. Він визначається числовим значенням 2. Вони можуть бути виконані після завдань з максимальним пріоритетом;

− низький пріоритет – це коли завдання є менш важливими і можуть бути виконані пізніше або коли є вільний час. Він визначається числовим значенням 3. Це можуть бути завдання, які не впливають на ключові аспекти проекту і можуть бути відкладені без значного ризику.

Важливо також для кожного виконавця встановити нормативи трудомісткості на один спринт та визначити команду, до якої він належить. Ця інформація готується перед першим використанням системи, а потім може тільки коригуватися у випадку необхідності.

Однією з важливих складових підготовчого етапу є розробка тезаурусу. Тезаурус або словник ключових слів допомагає у класифікації завдань на основі їх найменувань. Встановлення відповідностей між ключовими словами та класами (виконавцями) дозволяє автоматизувати процес класифікації завдань. Це також сприяє стандартизації та узагальненню термінології, що використовується в описах завдань.

Таким чином, підготовчий етап є критично важливим для успішного виконання методу призначення пакетів робіт командам виконавців ІТ-проєкту. Він забезпечує основу для автоматизованого розподілу завдань, що дозволяє підвищити ефективність роботи команд та зменшити вплив людського фактору.

3.3 Загальний алгоритм формування пакетів робіт для команд виконавців ІТ-проєкту

В роботі розроблений алгоритм формування пакетів робіт для команд виконавців ІТ-проєкту, який базується на автоматичному розподілі завдань між виконавцями за допомогою мультиномінального наївного байєсівського класифікатора та містить додаткові кроки для забезпечення використання всіх критеріїв розподілення завдань, які були визначені в розділі 2.

Схема алгоритму формування пакетів робіт для команд виконавців ІТ-проєкту наведена на рисунку 11. В опису алгоритму номери підпунктів відповідають номерам кроків алгоритму.

3.3.1 Першим кроком алгоритму є формування навчальної вибірки. Особливості виконання цього кроку вже були викладені в підрозділу 3.2.

3.3.2 На другому кроці алгоритму здійснюється визначення максимально допустимої трудомісткісті кожного виконавця на один спринт ($L^s_{\max i}$) . Розраховується загальна кількість сторі поінтів або часу, витраченого кожним виконавцем на виконання завдань у минулому. В нашому випадку, розрахунок виконується скрам-мастером, який веде підрахунок для кожного виконавця безпосередньо перед початком нової ітерації у вигляді спринта.

При цьому під час розрахунку повинно бути враховані таки вхідні дані, як загальна кількість днів спринту, кількість виконавців, кількість святкових днів, кількість сторі поінтів, які були опрацьовані під час попередніх спринтів. Ця інформація допомагає спланувати та зарезервувати кількість сторі поінтів для кожного виконавця та тим самим спланувати навантаження команд завданнями із беклога таким шляхом, щоб досягнути кінцевої мети спринта, тобто максимально виконати завдання, які були взяті в роботу.

3.3.3 Третій крок алгоритму передбачає розбиття нових ініціатив на великі логічні блоки – епіки. Це полегшує управління великими проектами, розділяючи їх на більш керовані частини. Розбиття ініціатив на епіки зазвичай здійснюється проджект-менеджерами за участю власників продукту та залученням скрам-майстра, які відповідають за планування та управління проектом. Цей процес відбувається на початку кожного проекту.

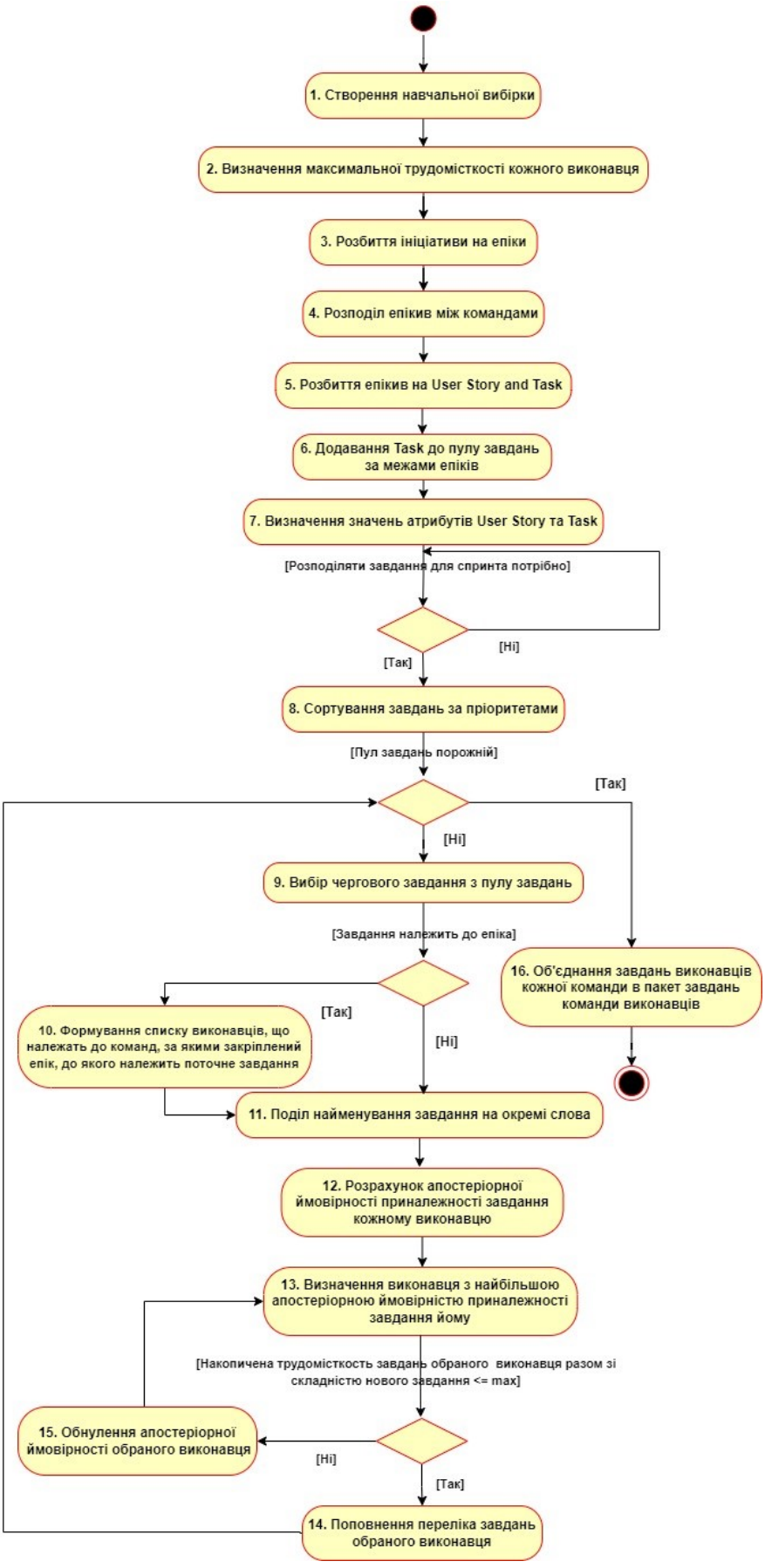

Рисунок 11 – Схема загального алгоритму формуванням пакетів завдань для команд виконавців ІТ-проєкту

При формуванні епіків проджект-менеджер керується принципами логічної зв’язаності, за розміром та складністю і взаємозв’язками між завданнями. Логічна зв’язаність має на меті об'єднувати завдання, що належать до однієї функціональної області або мають схожі цілі. Це допомагає уникнути розпорошення зусиль і забезпечує більш цілісне виконання проекту. Щодо розміру та складності, то епіки повинні бути достатньо великими, щоб їх розділення мало сенс, але не настільки великими, щоб вони стали некерованими.

Часто епіки розділяються на основі оцінок трудомісткості та складності завдань. Щодо взаємозв’язків між завданнями, то при розбитті ініціатив необхідно враховувати залежності між завданнями, щоб уникнути ситуацій, коли одне завдання не може бути виконане до завершення іншого.

3.3.4 Четвертим кроком відбувається розподіл епіків між командами. Кожен епік розподіляється між командами виконавців. Методика розподілу епіків між командами включає кілька кроків та враховує різні фактори для забезпечення ефективного та збалансованого розподілу завдань. Розподіл епіків зазвичай здійснюється власником продукту, за участю скрам-мастера, а подекуди в залежності від проєкту і з залученням в процес обговорення та призначення безпосередньо команди потенційних виконавців.

При розподілі епіків між командами враховується, по перше, спеціалізація команди та взаємозв’язки між компонентами, що закладені в епіках. Епіки розподіляються таким чином, щоб завдання відповідали спеціалізації та компетенції команди. Також при розподілі епіків, необхідно враховувати залежності між завданнями, щоб уникнути ситуацій, коли одна команда не може виконати своє завдання через залежність від завдання іншої команди. Епік може бути розподілений одночасно кільком командам, якщо він містить завдання, які потребують різних компетенцій. У такому випадку завдання епіку розбиваються на окремі підзадачі, які потім розподіляються між відповідними командами.

3.3.5 Наступним, п’ятим кроком є розбиття епіків на User Story та Task. Епіки розбиваються на конкретніші завдання – User Story та Task, що дозволяє детальніше планувати роботу та полегшує оцінку трудомісткості. Розбиттям епіків на User Story та Task зазвичай займаються власники продукту за участі скрам-майстра, в тісній співпраці з командами виконавців. Цей процес відбувається під час планування спринтів або перед початком нової фази проекту.

Процес розбиття епіків на User Story та Task здійснюється за принципом логічної послідовності, трудомісткості та складності. Епіки розбиваються на завдання, які логічно пов'язані між собою і можуть бути виконані незалежно один від одного. Це полегшує управління завданнями та їх подальшу інтеграцію. Епікі розбиваються на такі частини, щоб трудомісткість завдань відповідала запланованому об’єму, тобто кількість сторі поінтів завдання не може перебільшувати максимальну трудомісткість спринта. Це дозволяє точніше оцінити час та ресурси, необхідні для виконання кожного завдання.

Частота розбиття епіків на User Story та Task залежить від темпів і специфіки проекту. Це може відбуватися на регулярних планувальних зустрічах (наприклад, на початку кожного спринту) або коли з'являються нові епіки, які потребують розподілу. Такий підхід до розбиття епіків забезпечує детальне планування робіт, полегшує оцінку трудомісткості завдань і дозволяє ефективно керувати проектом.

3.2.6 Наступним кроком є додавання завдань до загального пулу завдань, які не входять до складу жодного епіку. Це дозволяє не втратити важливі, але менш глобальні задачі, які все ж потребують виконання. Додавання таких завдань до пулу забезпечує їх включення до загального розподілу та виконання в рамках проекту. Такі завдання можуть з’являтися як результат зворотнього зв’язку від клієнта, як результат тестування , аналізу ринку або як результат сформованої у вигляді завдання ініціативи команди.

Додавання завдань до пулу за межами епіків зазвичай здійснюється власником продукту, скрам-майстрами або іншими відповідальними особами,

залежно від структури команди та проекту, але здебільшого після обговорення та узгодження на рівні команд. Цей крок виконується регулярно, на планувальних зустрічах або спринт-плануваннях, коли з'являються нові завдання, які не належать до існуючих епіків. Це забезпечує актуальність і повноту пулу завдань для подальшого розподілу та виконання.

3.2.7 Наступним кроком є визначення атрибутів для Task та User Story. Кожне завдання отримує атрибути, такі як пріоритет, складність, тип тощо. Це важливо для подальшої класифікації та розподілу завдань. Процес розподілу завдань може починатися у випадку, якщо в цьому є потреба та напередодні спринта.

3.3.8 З восьмим кроком починається процес розподілу завдань. На цьому кроці виконується сортування завдань в беклозі за пріоритетами таким чином, щоб спочатку виконувати найважливіші з них, тобто за зростанням числа пріоритета. Це допомагає оптимізувати ресурсні витрати та підвищити ефективність. Пріоритети завдань можуть бути встановлені на основі їх впливу на проект, термінів виконання та важливості для клієнта. Зазвичай пріоритет визначає власник продукту, розміщуючи завдання в беклозі по пріоритетності згори вниз, від найвищого до найнижчого.

Надалі відбувається перевірка щодо наповненості пулу завдань. У разі, якщо пул завдань є порожним, керування передається на крок 16 – формування пакетів завдань для кожної команди. В іншому випадку переходим до наступного кроку (кроку 9).

3.3.9 Кроком 9 є вибір чергового завдання з пулу завдань за найвищим пріоритетом. Це гарантує, що найважливіші завдання отримують увагу в першу чергу.

Після виконання цього кроку здійснюється перевірка завдання щодо приналежності його до будь-якого епіку. У разі, якщо завдання має признак приналежності до епіку, виконується крок 10. В іншому випадку переходим до кроку 11.

3.3.10 На кроці 10 формуємо список виконавців що належать до команд (команди), за якими закріплений епік, до якого відноситься поточне завдання. Це дозволяє більш точно розподіляти завдання між членами команди з урахуванням їхніх навичок та можливостей. Список виконавців формується на основі інформації про членів команд та атрибутів епіку.

3.3.11 Наступник кроком є поділ найменування завдання на окремі слова. Це необхідно для застосування мультиноміального наївного баєсівського класифікатора, який працює з окремими словами найменувань завдань.

3.3.12 На кроці 12 проводиться розрахунок апостеріорної ймовірності належності завдання кожному виконавцю. На основі атрибутів завдання та історичних даних розраховується ймовірність того, що конкретний виконавець найкраще впорається з цим завданням. Апостеріорна ймовірність розраховується за допомогою формули (2.4), враховуючи частоти появи слів у завданнях, виконаних цим виконавцем раніше.

3.3.13 Наступним кроком є визначення виконавця з найбільшою апостеріорною ймовірністю належності завдання йому. Вибирається виконавець, для якого ймовірність успішного виконання завдання є найвищою. Це допомагає забезпечити якісне виконання роботи.

Після виконання цього кроку здійснюється перевірка накопиченої трудомісткості завдань обраного виконавця разом із складністю нового завдання, чи не перевищує вона максимально допустиму трудомісткість цього виконавця на один спринт ($L^s_{\max i}$). Якщо перевищує, керування передається на крок 15. Інакше виконується наступний крок (крок 14).

3.3.14 На цьому кроці відбувається поповнення переліку завдань обраного виконавця. Список завдань обраного виконавця оновлюється з урахуванням нового призначення. Це дозволяє контролювати навантаження на кожного виконавця та забезпечувати ефективне управління ресурсами.

Після виконання цього етапу керування передається на крок 9 – на вибір чергового завдання для розподілу з виконанням попередньої перевірки пулу завдань на присутність в ньому завдань.

3.3.15 На кроці 15 апостеріорна ймовірність поточного завдання для обраного класу (виконавця) обнуляється і керування передається на крок 13, тобто процес повторюється для наступного виконавця з найбільшою ймовірністю.

3.3.16 Завершальним кроком алгоритму є об’єднання завдань виконавців кожної команди в пакет завдань команди виконавців.

Ці кроки забезпечують структурований та ефективний підхід до розподілу завдань, враховуючи всі важливі аспекти управління проектом та особливості командної роботи.

3.4 Особливості застосування розробленого методу

Використання розробленого методу призначення пакетів робіт командам виконавців ІТ-проєкту має декілька унікальних особливостей, які роблять його ефективним інструментом для підвищення продуктивності та оптимізації процесу управління проектами.

Однією з головних переваг цього методу є автоматизація процесу розподілу завдань. Використання мультиноміального наївного байєсівського класифікатора дозволяє значно скоротити час, необхідний для розподілу завдань між виконавцями, та знизити залежність від людського фактору. Завдання розподіляються на основі об'єктивних критеріїв, що забезпечує більш справедливий та ефективний процес. Це особливо важливо в умовах великих і складних проектів, де кількість завдань може бути значною, а ручний розподіл займає багато часу і ресурсів.

Об'єктивність у процесі розподілу завдань є ще однією вагомою перевагою розробленого методу. Використання наївного байєсівського

алгоритму дозволяє приймати рішення на основі статистичних даних та історичної інформації, що мінімізує можливість суб'єктивних помилок та упередженості. Це забезпечує прозорість процесу розподілу завдань, яка в свою чергу підвищує довіру з боку виконавців і сприяє підвищенню їхньої мотивації. Виконавці можуть бути впевнені, що завдання розподіляються на основі об'єктивних показників, а не суб'єктивних рішень менеджерів.

Метод легко адаптується до різних проектів і команд завдяки можливості налаштування параметрів алгоритму відповідно до специфічних вимог. Це дозволяє використовувати метод у різних контекстах, від малих стартапів до великих корпоративних проектів. Гнучкість і масштабованість методу забезпечують його ефективність незалежно від обсягу завдань та кількості виконавців. Завдяки цьому можна налаштувати алгоритм під конкретні потреби проекту, що забезпечує максимальну ефективність його використання.

Для коректного функціонування алгоритму необхідні точні та актуальні дані про навички виконавців та трудомісткість завдань. Ці дані можна зібрати через аналіз попередніх проектів, оцінки виконання та зворотний зв'язок від команд. Регулярне оновлення та валідація даних є критично важливими для забезпечення точності та актуальності результатів. Цей підхід гарантує, що алгоритм завжди буде працювати з найсвіжішими даними, що дозволяє приймати найточніші рішення.

Постійне вдосконалення алгоритму є ще однією важливою особливістю розробленого методу. Регулярна валідація та корекція моделі на основі нових даних та зворотного зв'язку від виконавців та менеджерів проектів дозволяє постійно вдосконалювати алгоритм, адаптуючись до змін у проекті та специфіки завдань. Такий підхід забезпечує безперервне підвищення точності та ефективності методу, що є ключовим фактором успіху в динамічних умовах.

Автоматизація процесу розподілу завдань також дозволяє зменшити навантаження на менеджерів проектів, даючи їм можливість зосередитися на

стратегічних аспектах управління проектами. Це підвищує загальну ефективність управління та сприяє більш успішному виконанню проектів. Замість витрачання часу на рутинні операції з розподілу завдань, менеджери можуть зосередитися на плануванні, аналізі ризиків і оптимізації ресурсів.

Забезпечення рівномірного розподілу трудомісткості та об'єктивність у призначенні завдань сприяють підвищенню продуктивності команд. Виконавці отримують завдання, які відповідають їхнім навичкам та компетенціям, що підвищує якість виконання робіт та зменшує час на їх виконання. Це дозволяє командам працювати більш злагоджено і ефективно, що в кінцевому підсумку веде до успішного завершення проектів.

Таким чином, розроблений метод призначення пакетів робіт командам виконавців ІТ-проєкту має значні переваги, які забезпечують його ефективність та надійність у різних умовах. Його використання дозволяє автоматизувати та оптимізувати процес розподілу завдань, підвищуючи продуктивність та ефективність роботи команд.

4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА МЕТОДУ ПРИЗНАЧЕННЯ ПАКЕТІВ РОБІТ КОМАНДАМ ВИКОНАВЦІВ ІТ-ПРОЄКТУ

4.1 Опис особливостей організації, на базі робіт якої проходитиме практична апробація методу

Для практичної апробації розробленого методу призначення пакетів робіт командам виконавців ІТ-проєкту було обрано аутсорсингову ІТ-компанію. Цей вибір обумовлений декількома причинами, які забезпечують ідеальні умови для тестування та впровадження нового методу.

Аутсорсингова ІТ-компанія характеризується великим обсягом проектів та різноманітністю завдань, які варіюються від розробки програмного забезпечення до забезпечення кібербезпеки. Ця різноманітність дозволяє перевірити метод у різних сценаріях, забезпечуючи його гнучкість та адаптивність. Компанія має мультидисциплінарні команди, що складаються з фахівців різних напрямків: розробників (Frontend, Backend), DevOps інженерів, QA інженерів, BI інженерів тощо. Така структура дозволяє тестувати метод на різних типах завдань і з різними комбінаціями виконавців.

Важливою перевагою аутсорсингової компанії є її високий ступінь гнучкості в управлінні проектами та розподілі ресурсів, що дозволяє легко інтегрувати нові методи та підходи. Компанія також має накопичену базу даних про попередні проекти, завдання та виконання робіт, що є важливим для навчання та налаштування алгоритму. Орієнтація на продуктивність та ефективність роботи команд відповідає основним цілям розробленого методу.

Команди у компанії зазвичай складаються з різних спеціалістів, які можуть бути як постійними, так і формуватися на час виконання конкретних проектів.

Менеджери проектів відіграють ключову роль у впровадженні нового методу, забезпечуючи зв'язок між алгоритмом і реальними виконавцями.

Використання сучасних інструментів управління проектами, таких як Jira, Trello або Asana, дозволяє легко інтегрувати новий метод у вже існуючі процеси. Наявність платформ для спільної роботи (GitHub, GitLab) полегшує відстеження виконання завдань та управління репозиторіями коду.

Процес апробації методу включає декілька етапів. Спочатку проводиться підготовчий етап, під час якого здійснюється збір та аналіз даних про навички виконавців, оцінки трудомісткості завдань та історичних даних про виконані проекти. Отримані дані використовуються для налаштування алгоритму.

Далі метод тестується на кількох поточних проектах. Використовується алгоритм для автоматичного розподілу завдань між командами, результати якого моніторяться та порівнюються з результатами, отриманими традиційними методами.

Оцінка ефективності команд включає аналіз продуктивності команд, якості виконаних завдань та задоволеності клієнтів. Збирається зворотний зв'язок від менеджерів проектів та виконавців, що дозволяє коригувати алгоритм на основі отриманих результатів.

При успішній апробації методу відбувається поступове розширення його використання на інші проекти та команди в організації. Постійне вдосконалення алгоритму здійснюється на основі нових даних та зворотного зв'язку.

Таким чином, апробація методу в аутсорсинговій IT-компанії дозволяє перевірити його ефективність у реальних умовах, забезпечуючи цінний досвід та дані для подальшого вдосконалення та адаптації.

4.2 Опис вхідних даних методу

Для навчання моделі ми використовуємо історичні дані про завдання, які вже були виконані конкретними виконавцями в минулому. Ці дані включають назви завдань (User Story, Task) та інформацію про те, хто саме виконував ці завдання. Використання історичних даних дозволяє моделі навчитися розпізнавати патерни і закономірності у назвах завдань та відповідно призначати їх оптимальним виконавцям.

В таблиці 1 наведена інформація з навчаючої вибірки. Ця таблиця відображає певний перелік завдань спринта та відповідних виконавців цих завдань.

Таблиця 1 – Перелік виконаних завдань та їх виконавців

Завдання	Виконавець завдання
Frontend Design Update	Іван Петров (Junior)
UX/UI Improvement	Іван Петров (Junior)
Frontend Framework Update	Іван Петров (Junior)
Backend API Integration	Олексій Ковалек (Middle)
Web Page Load Optimization	Олексій Ковалек (Middle)
Business Intelligence Report Creation	Олексій Ковалек (Middle)
Backend Data Processing Optimization	Олексій Ковалек (Middle)
Real-time Data Analytics Dashboard	Олексій Ковалек (Middle)
Database Performance Tuning	Марія Іванова (Senior)
User Authentication Implementation	Марія Іванова (Senior)
Data Encryption Setup	Марія Іванова (Senior)

Кінець таблиці 1

Завдання	Виконавець завдання
Data Migration from Legacy Systems	Марія Іванова (Senior)
DevOps Pipeline Automation	Марія Іванова (Senior)
Security Vulnerability Assessment	Анна Сергієнко (Middle)
Cloud Resource Allocation	Анна Сергієнко (Middle)
Continuous Deployment Setup	Анна Сергієнко (Middle)
Network Configuration Management	Анна Сергієнко (Middle)
Mobile App Bug Fixing	Дмитро Орлов (Junior)
Automated Testing Script Development	Дмитро Орлов (Junior)
System Backup Configuration	Дмитро Орлов (Junior)

Наступним кроком проводимо токенізацію, тобто, розбиваємо назви завдань з навчаючої виборки на окремі слова (токени) за винятком союзів та прийменників

На цьому етапі аналізуємо кожну назву завдання, виділяючи окремі слова, які представляють ключові характеристики завдань.

Далі створюємо матрицю частоти входжень токенів до найменувань завдань кожного виконавця, де кожний рядок представляє окреме слово (токен) з назв завдань, а кожна колонка представляє кількість входжень слова до найменувань завдань, виконаних окремим виконавцем. Якщо слово жодного разу не зустрілося в найменуваннях завдань, виконаних виконавцем, тоді значення в стовпці цього виконавця дорівнюватиме 0.

Треба зауважити, у разі наявності однакових слів у різних назвах, в таблицю додаємо тільки унікальні слова, тобто таблиця не має містити

дублікати. Таким чином, кожне слово (токен) представлено лише один раз, що спрощує аналіз і підвищує ефективність алгоритму.

Матриця частоти входжень токенів до найменувань завдань кожного виконавця наведена в таблиці 2.

Таблиця 2 – Матриця частоти входжень токенів до найменувань завдань кожного виконавця

Слово	Іван Петров (Junior)	Олексій Ковалек (Middle)	Марія Іванова (Senior)	Гана Сергієнко (Middle)	Дмитро Орлов (Junior)
Frontend	2	0	0	0	0
Design	1	0	0	0	0
Update	1	0	0	0	0
Backend	0	2	0	0	0
API	0	1	0	0	0
Integration	0	1	0	0	0
Database	0	0	1	0	0
Performance	0	0	1	0	0
Tuning	0	0	1	0	0
Mobile	0	0	0	0	1
App	0	0	0	0	1
Bug	0	0	0	0	1
Fixing	0	0	0	0	1
Security	0	0	0	1	0
Vulnerability	0	0	0	1	0
Assessment	0	0	0	1	0
User	0	0	1	0	0
Authentication	0	0	1	0	0
Implementation	0	0	1	0	0
Web	0	1	0	0	0

Продовження таблиці 2

Слово	Іван Петров (Junior)	Олексій Коваленко (Middle)	Марія Іванова (Senior)	Гана Сергієнко (Middle)	Дмитро Орлов (Junior)
Page	0	1	0	0	0
Load	0	1	0	0	0
Data	0	2	2	0	0
Encryption	0	0	1	0	0
Setup	0	0	1	1	1
Resource	0	1	0	1	0
Allocation	0	1	0	1	0
Automated	0	0	0	0	1
Testing	0	0	0	0	1
Script	0	0	0	0	1
Development	0	1	0	0	1
Business	0	1	0	0	0
Intelligence	0	1	0	0	0
Report	0	1	0	0	0
Creation	0	1	0	0	0
Continuous	0	0	0	1	0
Deployment	0	0	0	1	0
System	0	0	0	1	1
Backup	0	0	0	0	1
Configuration	0	0	0	1	1
UX/UI	1	0	0	0	0
Improvement	1	0	0	0	0
Migration	0	0	1	0	0
Legacy	0	0	1	0	0
Systems	0	0	1	0	0
Framework	1	0	0	0	0

Кінець таблиці 2

Слово	Іван Петров (Junior)	Олексій Коваленко (Middle)	Марія Іванова (Senior)	Гана Сергієнко (Middle)	Дмитро Орлов (Junior)
Processing	0	1	0	0	0
Optimization	0	2	0	0	0
Network	0	0	0	1	0
Management	0	0	0	1	0
DevOps	0	0	1	1	0
Pipeline	0	1	1	1	0
Automation	0	1	1	1	0
Real-time	0	1	0	0	0
Analytics	0	1	0	0	0
Dashboard	0	1	0	0	0

4.3 Виконання підготовчого етапу методу призначення пакетів робіт командам виконавців IT-проєкту

Епіки відіграють важливу роль у структурізції та організації завдань. Вони допомагають групувати взаємопов’язані завдання, що сприяє кращому управлінню проєктом і забезпечує більш зручний контроль за виконанням робіт. З основних атрибутів епіків можна визначити ідентифікатор епіка та назва епіка.

Атрібути епіків до яких можуть належати завдання для розроділу представлені в таблиці 3.

Таблиця 3 – Атрибути епіків до яких належать завдання у прикладі

Ідентифікатор епіка	Назва епіка
E1	Frontend improvement
E2	Backend optimization
E3	Mobile security
E4	Database optimization
E5	Web performance
E6	Cloud infrastructure
E7	Continuous integration
E8	Business intelligence
E9	Network automation
E10	System backup
E11	Data security
E12	Machine learning implementation

Під час підготовчого етапу особливу увагу було приділено врахуванню важливих атрибутів завдань, які необхідно розподілити, епіків, до яких можуть належить завдання, потенційних виконавців завдань, та команд виконавців. Ці атрибути необхідні для забезпечення досягнення точності та ефективності розподілу завдань між командами виконавців.

Трудомісткість завдань оцінюється в сторі поінтах (SP), що є стандартною метрикою в Agile-підході до управління проектами. Трудомісткість може варіюватися від мінімальної (1 SP) до максимальної (13 SP). Використовуються такі значення сторі поінтів: 1, 3, 5, 8 та 13. Значення 1 SP зазвичай означає дуже просте завдання, яке можна виконати швидко, тоді як 13 SP вказує на складне завдання, що потребує значних зусиль і часу. Ця шкала допомагає командам ефективно планувати і розподіляти робочі завдання.

Також дуже важливим є пріоритет завдань, який повинен бути врахований при розподілі. Завдання з високим пріоритетом мають бути виконані першочергово.

Значення атрибутів кожного завдання надано у вигляді пріоритету, трудомісткості та ідентифікатора епіка до якого належить завдання, наведені в таблиці 4. Завдання розташовані згідно визначеного пріоритету.

Для апробації методу ми використали дві команди, позначені як команда A, та команда B.

Таблиця 4 – Атрибути нових завдань (об'єктів класифікації) для розподілу

Назва завдання	Пріоритет	Трудомісткість(SP)	Ідентифікатор епіка
Backend API performance tuning	1	8	E2
Mobile app security audit	1	8	E3
Web page speed optimization	1	8	E5
Cloud infrastructure setup	1	13	E6
Network configuration automation	1	8	E9
Business intelligence dashboard creation	2	13	E8
Backend service integration	2	8	E2
Continuous integration pipeline setup	2	5	E7
System backup strategy implementation	2	5	E10
Automated testing suite development	2	8	
Frontend Feature Extension	2	8	E1
Frontend design enhancement	2	5	E1

Кінець таблиці 4

Назва завдання	Пріоритет	Трудомісткість(SP)	Ідентифікатор епіка
Database schema optimization	2	5	E4
Real-time data analysis implementation	2	5	
Mobile application testing	2	5	E3
Data encryption security setup	3	3	E11
User interface design update	3	3	E1
Cloud service migration	3	13	E6
Machine learning model training	3	13	E12

Виконавці розподілені між командами на основі їхнього досвіду та компетенцій. Цей розподіл забезпечує оптимальне використання навичок та ресурсів для ефективного виконання завдань. Належність виконавців до команд наведено в таблиці 5.

Таблиця 5 – Належність виконавців до команд

Команда	Виконавець
Команда A	Іван Петров (Junior)
	Олексій Ковалек (Middle)
Команда B	Марія Іванова (Senior)
	Гана Сергієнко (Middle)
	Дмитро Орлов (Junior)

Виконавці класифікуються на три рівні – Junior, Middle та Senior. Ця градація враховує досвід, навички та компетенції виконавців.

Охарактеризуємо виконавців які приймають участь у розподілі завдань.

Значення атрибутів виконавців наведені в таблиці 6.

Таблиця 6 – Атрибути виконавців (класів)

Виконавець	Градація	Максимальна трудомісткість (SP)
Іван Петров	Junior	20
Олексій Коваленко	Middle	35
Марія Іванова	Senior	50
Гана Сергієнко	Middle	35
Дмитро Орлов	Junior	20

4.4 Визначення виконавців для нових завдань

Використовуючи дані таблиці 2 «Матриця відповідності завдань виконавцям» провели розрахунок ймовірностей слів для виконавців згідно формули (2.3).

Загальна кількість появи слів у найменуваннях виконаних завдань виконавців відображена в таблицях 7-11 окремо для кожного виконавця.

Таблиця 7 – Кількість слів в найменуваннях завдань виконавця Івана Петрова з навчаючої вибірки

Слово	Кількість наявності слова в найменуваннях завдань
Frontend	2
Design	1
Update	1
UX/UI	1

Кінець таблиці 7

Слово	Кількість наявності слова в найменуваннях завдань
Integration	1
Improvement	1
Framework	1

Загальна кількість слів у завданнях, які виконані виконавцем Іваном Петровим, дорівнює 7 словам.

Таблиця 8 – Кількість слів в завданнях виконавця Олексія Коваленка з навчаючої вибірки

Слово	Кількість наявності слова в найменуваннях завдань
Backend	2
API	1
Integration	1
Web	1
Page	1
Load	1
Data	2
Resource	1
Allocation	1
Development	1
Business	1
Intelligence	1
Report	1
Creation	1
Processing	1
Optimization	2
Pipeline	1

Кінець таблиці 8

Слово	Кількість наявності слова в найменуваннях завдань
Automation	1
Real-time	1
Analytics	1
Dashboard	1

Загальна кількість слів у завданнях, які виконані виконавцем Олексіїм Коваленком, дорівнює 24 словам.

Таблиця 9 – Кількість слів в завданнях виконавця Марії Іванової з навчаючої вибірки

Слово	Кількість наявності слова в найменуваннях завдань
Database	1
Performance	1
Tuning	1
Data	2
User	1
Authentication	1
Implementation	1
Setup	1
Encryption	1
System	1
Migration	1
Legacy	1
Systems	1
DevOps	1
Pipeline	1

Загальна кількість слів у завданнях, які виконані виконавцем Марією Іванової, дорівнює 17 словам.

Таблиця 10 – Кількість слів в завданнях виконавця Гани Сергієнко з навчаючої вибірки

Слово	Кількість наявності слова в найменуваннях завдань
Security	1
Vulnerability	1
Assessment	1
Setup	1
Resource	1
Allocation	1
Continuous	1
Deployment	1
System	1
Network	1
Management	1
DevOps	1
Pipeline	1
Automation	1

Загальна кількість слів у завданнях, які виконані виконавцем Ганою Сергієнко, дорівнює 14 словам.

Таблиця 11 – Кількість слів в завданнях виконавця Дмитро Орлов з навчаючої вибірки

Слово	Кількість наявності слова в найменуваннях завдань
Mobile	1
App	1

Кінець таблиці 11

Слово	Кількість наявності слова в найменуваннях завдань
Bug	1
Fixing	1
Automated	1
Testing	1
Script	1
Development	1
Backup	1
Configuration	1
System	1

Загальна кількість слів у завданнях, які виконані виконавцем Дмитром Орловим, дорівнює 11 словам.

Розмір словника навчаючої вибірки:V = 56.

Коефіцієнт згладжування приймаємо такий: α=1.

Продемонструємо обчислювання ймовірностей на прикладі Івана Петрова для нового завдання з найменуванням «Frontend Design Enhancement».

Ймовірність слова «Frontend»:

$$P(Frontend|\text{Іван Петров}) = \frac{2+1}{7+1\times 56} = \frac{3}{63} \approx 0.0476.$$

Ймовірність слова «Design»:

$$P(Design|\text{Іван Петров}) = \frac{1+1}{7+1\times 56} = \frac{2}{63} \approx 0.0317.$$

Ймовірність слова «Enhancement»:

$$P(Enhancement|\text{Іван Петров}) = \frac{0+1}{7+1\times 56} = \frac{1}{63} \approx 0.0159.$$

Приклад обчислення апостеріорної ймовірності для нового завдання «Frontend Design Enhancement».

Слова «Frontend», «Design» присутні в навчальній вибірці, а слово «Enhancement» не зустрічалось раніше.

Нижче, наведена таблиця 12, де відображено, як кожне слово (токен) з назви нового завдання асоціюється з історичними даними про виконавців, відображаючи кількість завдань, в яких це слово зустрічається для кожного виконавця.

Апріорна ймовірність того, що завдання має належати цьому виконавцю:

$$P(\text{Іван Петров}) = \frac{1}{5} = 0.2.$$

Апостеріорна ймовірність завдання для цього виконавця:

$$P(\text{Іван Петров}|Frontend\ Design\ Enhancement) \infty P(\text{Іван Петров}) \times \\ \times P(\text{Іван Петров}|Frontend) \times P(\text{Іван Петров}|Design) \times \\ \times P(\text{Іван Петров}|Enhancement).$$

Розрахунок:

$$(\text{Іван Петров}|Frontend\ Design\ Enhancement) \infty 0.2 \times 0.0476 \times 0.0317 \times \\ \times 0.0159 \approx 0.0000048.$$

Таблиця 12 – Частота входжень токенів з нових завдань для кожного виконавця

Завдання	Слово	Іван Петров (Junior)	Олексій Коваленко (Middle)	Марія Іванова (Senior)	Гана Сергієнко (Middle)	Дмитро Орлов (Junior)
Backend API performance tuning	Backend	1	2	0	0	0
	Api	1	0	0	0	0
	Performance	0	0	1	0	0
	Tuning	0	0	1	0	0
Mobile APP security audit	Mobile	0	0	0	0	1
	App	0	0	0	0	1
	Security	0	0	0	1	0
	Audit	0	0	0	0	0
Web page speed optimization	Web	0	1	0	0	0
	Page	0	1	0	0	0
	Speed	0	0	0	0	0
	Optimization	0	2	0	0	0
Cloud infrastructure setup	Cloud	0	0	0	1	0
	Infrastructure	0	0	0	0	0
	Setup	0	0	1	1	0
Network configuration automation	Network	0	0	0	1	0
	configuration	0	0	0	1	1
	automation	0	0	1	0	0
Business intelligence dashboard creation	Business	0	1	0	0	0
	intelligence	0	1	0	0	0
	dashboard	0	1	0	0	0
	creation	0	1	0	0	0
Backend service integration	Backend	1	2	0	0	0
	service	0	0	0	0	0
	integration	1	0	0	0	0

Продовження таблиці 12

Завдання	Слово	Іван Петров (Junior)	Олексій Коваленко (Middle)	Марія Іванова (Senior)	Гана Сергієнко (Middle)	Дмитро Орлов (Junior)
Continuous integration pipeline setup	Continuous	0	0	0	1	0
	integration	1	0	0	0	0
	pipeline	0	0	1	0	0
	setup	0	0	1	1	0
System backup strategy implementation	System	0	0	1	0	1
	backup	0	0	0	0	1
	strategy	0	0	0	0	0
	implementation	0	0	1	0	0
Automated testing suite development	Automated	0	0	0	0	1
	testing	0	0	0	0	1
	suite	0	0	0	0	0
	development	0	0	0	0	1
Frontend feature extension	Frontend	2	0	0	0	0
	Feature	0	0	0	0	0
	Extension	0	0	0	0	0
User interface design update	User	0	0	1	0	
	interface	0	0	0	0	
	design	1	0	0	0	
	update	1	0	0	0	
Cloud service migration	Cloud	0	0	0	1	
	Service	0	0	0	0	
	migration	0	0	1	0	
Machine learning model training	Machine	0	0	0	0	
	Learning	0	0	0	0	
	Model	0	0	0	0	
	training	0	0	0	0	
Frontend design enhancement	Frontend	2	0	0	0	0
	design	1	0	0	0	0
	enhancement	0	0	0	0	0

Кінець таблиці 12

Завдання	Слово	Іван Петров (Junior)	Олексій Коваленко (Middle)	Марія Іванова (Senior)	Гана Сергієнко (Middle)	Дмитро Орлов (Junior)
Database schema optimization	Database	0	0	1	0	0
	schema	0	0	0	0	0
	optimization	0	2	0	0	0
Real-time data analysis implementation	Real-time	0	1	0	0	0
	data	0	2	2	0	0
	analysis	0	0	0	0	0
	implementation	0	0	1	0	0
Mobile app testing	Mobile	0	0	0	0	1
	app	0	0	0	0	1
	testing	0	0	0	0	1
Data encryption security setup	Data	0	2	2	0	
	encryption	0	0	1	0	
	security	0	0	0	1	
	setup	0	0	1	1	
User interface design update	User	0	0	1	0	
	interface	0	0	0	0	
	design	1	0	0	0	
	update	1	0	0	0	
Cloud service migration	Cloud	0	0	0	1	
	Service	0	0	0	0	
	migration	0	0	1	0	
Machine learning model training	Machine	0	0	0	0	
	Learning	0	0	0	0	
	Model	0	0	0	0	
	training	0	0	0	0	

Далі були виконані розрахунки апостеріорних ймовірностей призначення нових завдань виконавцям. Результати розрахунків відображені в таблиці 13. Ця таблиця демонструє обчислені апостеріорні ймовірності для кожного завдання та виконавця. Вона показує, з якою ймовірністю кожне конкретне завдання може належати кожному виконавцю, що дозволяє ефективно розподілити завдання на основі історичних даних.

Таблиця 13 – Результати обчислень апостеріорної ймовірності для кожного завдання та виконавця

Назва завдання	Іван Петров (Junior)	Олексій Ковалек (Middle)	Марія Іванова (Senior)	Гана Сергієнко (Middle)	Дмитро Орлов (Junior)
Backend API performance tuning	$5.08E^{-8}$	$1.46E^{-8}$	$2.82E^{-8}$	$8.36E^{-9}$	$9.86E^{-9}$
Mobile app security audit	$1.28E^{-8}$	$4.88E^{-9}$	$7.05E^{-9}$	$8.36E^{-9}$	$3.97E^{-8}$
Web page speed optimization	$1.28E^{-8}$	$5.86E^{-8}$	$7.05E^{-9}$	$8.36E^{-9}$	$9.86E^{-9}$
Cloud infrastructure setup	$8.04E^{-7}$	$3.91E^{-7}$	$1.03E^{-6}$	$2.34E^{-6}$	$1.33E^{-6}$
Network configuration automation	$8.04E^{-7}$	$3.91E^{-7}$	$1.03E^{-6}$	$2.34E^{-6}$	$1.33E^{-6}$
Business intelligence dashboard creation	$1.28E^{-8}$	$7.81E^{-8}$	$7.05E^{-9}$	$8.36E^{-9}$	$9.86E^{-9}$
Backend service integration	3.20E-6	$2.34E^{-6}$	$5.14E^{-7}$	$5.85E^{-7}$	$6.62E^{-7}$
Continuous integration pipeline setup	$2.55E^{-8}$	$1.95E^{-8}$	$1.41E^{-8}$	$6.69E^{-8}$	$1.98E^{-8}$
System backup strategy implementation	$1.28E^{-8}$	$4.88E^{-9}$	$2.82E^{-8}$	$1.67E^{-8}$	$3.97E^{-8}$
Automated testing suite development	$1.28E^{-8}$	$9.77E^{-9}$	$1.41E^{-8}$	$8.36E^{-9}$	$7.97E^{-8}$
Frontend Feature Extension	$2.41E^{-6}$	$3.91E^{-7}$	$5.14E^{-7}$	$5.85E^{-7}$	$6.62E^{-7}$
Frontend design enhancement	$4.80E^{-6}$	$3.91E^{-7}$	$5.14E^{-7}$	$5.85E^{-7}$	$6.62E^{-7}$
Database schema optimization	$8.04E^{-7}$	$1.17E^{-6}$	$1.03E^{-6}$	$5.85E^{-7}$	$6.62E^{-7}$

Кінець таблиці 13

Назва завдання	Іван Петров (Junior)	Олексій Ковалек (Middle)	Марія Іванова (Senior)	Гана Сергієнко (Middle)	Дмитро Орлов (Junior)
Real-time data analysis implementation	$1.28E^{-8}$	$2.93E^{-8}$	$4.23E^{-8}$	$8.36E^{-9}$	$9.86E^{-9}$
Mobile application testing	$8.04E^{-7}$	$3.91E^{-7}$	$5.14E^{-7}$	$5.85E^{-7}$	$2.66E^{-6}$
Data encryption security setup	$1.28E^{-8}$	$1.46E^{-8}$	$8.45E^{-8}$	$3.35E^{-8}$	$9.86E^{-9}$
User interface design update	$5.08E^{-8}$	$4.88E^{-9}$	$1.41E^{-8}$	$8.36E^{-9}$	$9.86E^{-9}$
Cloud service migration	$8.04E^{-7}$	$3.91E^{-7}$	$2.06E^{-6}$	$1.17E^{-6}$	$6.62E^{-7}$
Machine learning model training	$1.28E^{-8}$	$4.88E^{-9}$	$7.05E^{-9}$	$8.36E^{-9}$	$9.86E^{-9}$

На основі розрахованих апостеріорних ймовірностей належності кожного завдання кожному виконавцю була визначена черговість призначення кожного завдання кожному виконавцю, яка відображена в таблиці 14. Черговість виконання завдання ранжована від найімовірнішого виконавця до найменш ймовірного.

Таблиця 14 – Черговість призначення завдань виконавцям відповідно до їх апостеріорних ймовірностей

Завдання	Виконавці
Backend API performance tuning	Іван Петров (Junior);
	Марія Іванова (Senior);
	Олексій Ковалек (Middle);
	Дмитро Орлов (Junior);
	Гана Сергієнко (Middle).
Mobile app security audit	Дмитро Орлов (Junior);
	Іван Петров (Junior);
	Гана Сергієнко (Middle);
	Марія Іванова (Senior);
	Олексій Ковалек (Middle).
Web page speed optimization	Олексій Ковалек (Middle);
	Іван Петров (Junior);
	Дмитро Орлов (Junior);
	Гана Сергієнко (Middle);
	Марія Іванова (Senior).

Продовження таблиці 14

Завдання	Виконавці
Cloud infrastructure setup	Гана Сергієнко (Middle);
	Дмитро Орлов (Junior);
	Марія Іванова (Senior);
	Іван Петров (Junior);
	Олексій Ковалек (Middle).
Network configuration automation	Гана Сергієнко (Middle);
	Дмитро Орлов (Junior);
	Марія Іванова (Senior);
	Іван Петров (Junior);
	Олексій Ковалек (Middle).
Business intelligence dashboard creation	Олексій Ковалек (Middle);
	Іван Петров (Junior);
	Дмитро Орлов (Junior);
	Гана Сергієнко (Middle);
	Марія Іванова (Senior)
Backend service integration	Іван Петров (Junior);
	Олексій Ковалек (Middle);
	Дмитро Орлов (Junior);
	Гана Сергієнко (Middle);
	Марія Іванова (Senior).
Continuous integration pipeline setup	Гана Сергієнко (Middle);
	Дмитро Орлов (Junior);
	Іван Петров (Junior);
	Олексій Ковалек (Middle);
	Марія Іванова (Senior).
System backup strategy implementation	Дмитро Орлов (Junior);
	Марія Іванова (Senior);
	Гана Сергієнко (Middle);
	Іван Петров (Junior);
	Олексій Ковалек (Middle).
Automated testing suite development	Дмитро Орлов (Junior);
	Марія Іванова (Senior);
	Іван Петров (Junior);
	Гана Сергієнко (Middle);
	Олексій Ковалек (Middle)
Frontend Feature Extension	Іван Петров (Junior);
	Дмитро Орлов (Junior);
	Гана Сергієнко (Middle);
	Марія Іванова (Senior);
	Олексій Ковалек (Middle).

Кінець таблиці 14

Завдання	Виконавці
Frontend design enhancement	Іван Петров (Junior);
	Дмитро Орлов (Junior);
	Гана Сергієнко (Middle);
	Марія Іванова (Senior);
	Олексій Ковалек (Middle)
Database schema optimization	Олексій Ковалек (Middle);
	Марія Іванова (Senior);
	Іван Петров (Junior);
	Дмитро Орлов (Junior);
	Гана Сергієнко (Middle).
Real-time data analysis implementation	Марія Іванова (Senior);
	Олексій Ковалек (Middle);
	Іван Петров (Junior);
	Дмитро Орлов (Junior);
	Гана Сергієнко (Middle).
Mobile application testing	Дмитро Орлов (Junior);
	Іван Петров (Junior);
	Марія Іванова (Senior);
	Гана Сергієнко (Middle);
	Олексій Ковалек (Middle).
Data encryption security setup	Марія Іванова (Senior);
	Гана Сергієнко (Middle);
	Олексій Ковалек (Middle);
	Іван Петров (Junior);
	Дмитро Орлов (Junior).
User Interface design update	Іван Петров (Junior);
	Марія Іванова (Senior);
	Дмитро Орлов (Junior);
	Гана Сергієнко (Middle);
	Олексій Ковалек (Middle).
Cloud service migration	Марія Іванова (Senior);
	Гана Сергієнко (Middle);
	Іван Петров (Junior);
	Дмитро Орлов (Junior);
	Олексій Ковалек (Middle).
Machine learning model training	Іван Петров (Junior);
	Марія Іванова (Senior);
	Гана Сергієнко (Middle);
	Дмитро Орлов (Junior);
	Олексій Ковалек (Middle).

Далі на основі результатів розрахунків апостеріорних ймовірностей відбувся процес призначення завдань.

Результати призначення завдань відображені в таблиці 15. Ця таблиця демонструє розподіл завдань з урахуванням максимально можливої трудомісткості кожного виконавця. Згідно з адаптованим алгоритмом, при призначенні завдань було враховано, що кожен виконавець не може прийняти більше завдань за спринт, ніж вказано в його максимально можливій трудомісткості.

Таблиця 15 – Розподіл завдань з урахуванням максимальної трудомісткості виконавців

Завдання	Трудомісткість завдання (SP)	Виконавець	Max трудомісткість виконавця	Залишок SP
1. Backend API performance tuning	8	Іван Петров (Junior)	20	12
2. Backend service integration	8			4
1. Web page speed optimization	8	Олексій Ковалек (Middle)	35	27
2. Business intelligence dashboard creation	13			14
3. Database schema optimization	5			9
1. Automated testing suite development	8	Марія Іванова (Senior)	50	42
2. Real-time data analysis implementation	5			37
3. Mobile application testing	5			32
4. Data encryption security setup	3			29
5. User interface design update	5			24
6. Cloud service migration	13			11
1. Cloud infrastructure setup	13	Гана Сергієнко (Middle)	35	22
2. Network configuration automation	8			14
3. Continuous integration pipeline setup	5			9
4. Frontend Feature Extension	8			1

Кінець таблиці 15

Завдання	Трудомісткість завдання (SP)	Виконавець	Max трудомісткість виконавця	Залишок SP
1. Mobile app security audit	8	Дмитро Орлов (Junior)	20	12
2. System backup strategy implementation	5			7
3. Frontend design enhancement	5			2
Machine learning model training	13			

Якщо виконавець досяг максимальної трудомісткості, завдання призначалось наступному виконавцю з найвищою апостеріорною ймовірністю. Якщо всі виконавці перевищили свою максимальну трудомісткість, завдання має бути перенесено на наступний спринт.

Слід зауважити, що призначення завдань відбувається з урахуванням пріоритетності, тобто, найбільш пріоритетні завдання призначались в першу чергу, тому що завдання з більш високим пріоритетом в переліку завдань розташовані раніше, ніж відповідно.

Також згідно таблиці, можна побачити, що деякі з виконавців мають невикористану трудомісткість, яка не може бути використана, тому що вона менша ніж трудомісткість завдання. В такому разі, ці залишки виступають в ролі буфера, або як резерв у разі появи критично важливих завдань, які повинні бути виконанні якомога швидше і можуть бути додані в спринт в його середині, без урахування трудомісткості цієї активності під час планування спринта.

Після розподілу завдань між виконавцями були сформовані пакети завдань для кожної команди виконавців шляхом об'єднання завдань, які були розподілені виконавцям одної команди. Сформовані пакети завдань кожної команди наведені в таблиці 16.

Таблиця 16 – Пакети завдань команд виконавців IT-проєкту

Команда	Завдання
A	Backend API performance tuning
	Backend service integration
	Web page speed optimization
	Business intelligence dashboard creation
	Database schema optimization
B	Automated testing suite development
	Real-time data analysis implementation
	Mobile application testing
	Data encryption security setup
	User interface design update
	Cloud service migration
	Cloud infrastructure setup
	Network configuration automation
	Continuous integration pipeline setup
	Frontend Feature Extension
	Mobile app security audit
	System backup strategy implementation
	Frontend design enhancement

4.5 Аналіз отриманих результатів

Аналіз отриманих даних показав, що мультиноміальний наївний баєсовський класифікатор є ефективним інструментом для автоматизованого призначення пакетів робіт командам виконавців IT-проєкту. Метод дозволяє розподіляти завдання на основі історичних даних та найменувань завдань, забезпечуючи об'єктивність та прозорість процесу.

Додатково було виявлено, що алгоритм демонструє високу стабільність при обробці великих обсягів даних, що робить його придатним для застосування у великих IT-проєктах. Одним із ключових факторів успіху є використання словника ключових слів, який дозволяє класифікатору більш точно визначати категорії завдань.

Аналіз також виявив кілька недоліків алгоритму, які потребують подальшого доопрацювання. Наприклад, алгоритм може мати труднощі з класифікацією завдань, що мають схожі найменування, але відрізняються за змістом. Це може призводити до помилкового розподілу завдань між командами.

Щоб покращити точність класифікації, необхідно розширення навчальної вибірки. Використання більшої кількості історичних даних для навчання моделі може підвищити точність класифікації. Регулярне оновлення та розширення словника ключових слів допоможе алгоритму краще розуміти контекст завдань.

Проте розподіл завдань, що відбувся за допомогою метода, в більшості відповідає спеціалізації виконавців. Аналіз показав, що завдання були розподілені між командами з урахуванням їхніх компетенцій та попереднього досвіду виконання схожих завдань. Це дозволило забезпечити високу ефективність та продуктивність команд, зменшивши час на виконання завдань та покращивши загальну якість проєкту.

Однак деякі завдання все ще потребують ручної перевірки та коригування розподілу, особливо у випадках, коли завдання виходять за рамки стандартних параметрів класифікації. Це вказує на необхідність гнучкої системи, яка може комбінувати автоматизоване призначення з ручним контролем.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досліджено методи вирішення проблеми формування пакетів завдань командам виконавців ІТ-проєкту.

У першому розділі проведено огляд та аналіз сучасного стану проблеми автоматизації призначення пакетів робіт командам виконавців ІТ-проєкту: виконано дослідження етапів процесу призначення пакетів робіт, проведено аналіз існуючих програмних засобів, які автоматизують цей процес.

У другому розділі досліджено особливості методу автоматичного призначення завдань виконавцям ІТ-проєкту з використанням метода класифікації – наївного байєсівського класифікатора, визначена можливість його використання для автоматизації формування пакетів завдань командам виконавців. Також запропоновано модифікація наївного байєсівського класифікатора для вирішення завдань кваліфікаційної роботи.

У третьому розділі виконано удосконалення методу призначення пакетів робіт командам виконавців ІТ-проєкту: виконано адаптацію мультиномінального наївного байєсівського класифікатора до розподілу завдань, розроблено алгоритм формування пакетів робіт командам виконавців, який базується на використанні мультиномінального наївного байєсівського класифікатора.

У четвертому розділі проведено експериментальну перевірку удосконаленого методу призначення пакетів робіт командам виконавців ІТ-проєкту, яка підтвердила ефективність методу. Метод забезпечує рівномірний розподіл трудомісткості завдань між командами відповідно до навичок виконавців, що підвищує якість виконання робіт.

Новизна отриманих результатів полягає у запропонованій модифікації наївного баєсівського класифікатора для вирішення завдань кваліфікаційної роботи та в розробці удосконаленого методу формування пакетів завдань

командам виконавців ІТ-проєкту на основі мультиноміального наївного байєсівського класифікатора.

Таким чином, удосконалений метод має значні переваги, які забезпечують його ефективність та надійність у різних умовах. Його використання дозволяє автоматизувати та оптимізувати процес розподілу завдань, підвищуючи продуктивність та ефективність роботи команд, що є важливим кроком у розвитку сучасних систем управління ІТ-проєктами.

Даний напрямок має перспективи для подальшого дослідження та вимагає проведення експериментальних досліджень з більшою навчальною вибіркою і більшою кількістю завдань для розподілу. Метод потребує подальшого вдосконалення та тестування.

За тематикою кваліфікаційної роботи було опубліковано тези доповіді на тему «Особливості оптимізації призначення пакетів робіт на проекті в умовах дистанційної роботи команд» [27].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи другого (магістерського) рівня вищої освіти за освітньо-науковою програмою «Управління проектами в галузі інформаційних технології»/Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2024. – 24 с.

2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання, Чинний від 22.06.2015. Київ: ДП «УкрНДНЦ», 2016. 26 с.

3. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання / Нац. стандарт України. Вид. офіц. [Уведено вперше; чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 17 с.

4. Seeking Clarity in the World of Tech: Product – Based and Service – Based Companies URL: https://medium.com/@polowaliapinky12/seeking-clarity-in-the-world-of-tech-product-based-and-service-based-companies-287d963d7b42 (дата звернення: 05.03.2024).

5. Pruduct (digital Business) URL: https://www.gartner.com/en/information-technology/glossary/product-digital-business (дата звернення: 07.03.2024).

6. What is Project Management? https://www.pmi.org/about/learn-about-pmi/what-is-project-management (дата звернення: 07.03.2024).

7. What is the Agile methodology? URL: https://www.atlassian.com/agile#:~:text=The%20Agile%20methodology%20is%20a,planning%2C%20executing%2C%20and%20evaluating (дата звернення: 07.03.2024).

8. What is scrum and how to get started URL: https://www.atlassian.com/agile/scrum (дата звернення: 06.03.2024).

9. Project Management Institute. A guide to the project management body of knowledge (PMBOK guide) ISBN 978-1628256642 Seventh edition. | Newtown Square, PA: Project Management Institute, 2021.

10. Agile scrum artifacts URL: https://www.atlassian.com/agile/scrum/artifacts#:~:text=Summary%3A%20Agile%20scrum%20artifacts%20are,%2C%20sprint%20backlog%2C%20and%20increments (дата звернення: 06.03.2024).

11. Project Initiation Phase URL: https://www.umb.edu/it/about/project-management-office/project-initiation-phase/ (дата звернення: 06.03.2024).

12. What is Product Owner? URL: https://www.agilealliance.org/glossary/product-owner/ (дата звернення: 06.03.2024).

13. Project Management Institute. A guide to the project management body of knowledge (PMBOK guide) ISBN 978-1628256642 Seventh edition. | Newtown Square, PA: Project Management Institute, 2021.

14. T-Shirt Sizing Methodology(sizing&estimation methodologies) URL: https://ankur-javaarch.medium.com/t-shirt-sizing-methodology-sizing-estimation-methodologies-80ff986574e6 (дата звернення: 06.03.2024).

15. Guide to building a product roadmap (with template and examples) URL: https://blog.logrocket.com/product-management/product-roadmap-template-examples/ (дата звернення: 06.03.2024).

16. Product Backlog – What is it & How to create one URL: https://www.atlassian.com/agile/scrum/backlogs (дата звернення: 06.03.2024).

17. Work Package In Project Management [Activity Examples] URL: https://www.pmbypm.com/work-package-vs-activity/ (дата звернення: 11.03.2024).

18. Decomposition in Project Management | Tool, Benefit & Examples URL: https://study.com/academy/lesson/decomposition-in-project-management-definition-importance.html#:~:text=Decomposition%20of%20your%20project%20in,execute

%20your%20project%20deliverable%20successfully (дата звернення: 11.03.2024).

19. Project Management Institute. A guide to the project management body of knowledge (PMBOK guide) ISBN 978-1628256642 Seventh edition. | Newtown Square, PA: Project Management Institute, 2021.

20. User stories with examples and a template URL: https://www.atlassian.com/agile/project-management/user-stories#:~:text=Summary%3A%20A%20user%20story%20is,simply%20put%2C%20software%20system%20requirements (дата звернення: 11.03.2024).

21. Sprint planning URL: https://www.atlassian.com/agile/scrum/sprint-planning (дата звернення: 11.03.2024).

22. Backlog Grooming URL: https://www.productplan.com/glossary/backlog-grooming/ (дата звернення: 12.03.2024).

23. Jira Core means business: introducing boards and mobile apps – Work Life by Atlassian URL: https://www.atlassian.com/search?q=jira&f=Articles (дата звернення: 12.03.2024).

24. Bishop, C. M. Pattern Recognition and Machine Learning. ISBN 978-0387310732. Berlin: Springer, 2006.

25. Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. Data Mining: Practical Machine Learning Tools and Techniques. ISBN 978-0128042915. Fourth Edition. San Francisco, CA: Morgan Kaufmann, 2016.

26. Hand, D. J., & Yu, K. "Idiot's Bayes — Not So Stupid after All?". International Statistical Review, 69(3), 385-398. ISBN 978-0470022973. 2001.

27. Тіханов О. І. Особливості оптимізації призначення пакетів робіт на проекті в умовах дистанційної роботи команд / О. І. Тіханов // Радіоелектроніка та молодь у XXI столітті : матеріали 27-го Міжнар. молодіж. форуму, 10–12 травня 2023 р. – Харків : ХНУРЕ, 2023. – Т. 6, ч.1. – С. 140–141.