МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

АВТОМАТИЗОВАНІ СИСТЕМИ УПРАВЛІННЯ І ПРИЛАДИ АВТОМАТИКИ

**Всеукраїнський міжвідомчий
науково-технічний збірник**

Заснований у 1965 р.

Випуск 178

Харків
2022

УДК 62-50

У збірнику наведено результати досліджень, що стосуються моделей і методів штучного інтелекту, управління в організаційних і технічних системах, використання методів і засобів Data Mining для вирішення прикладних завдань. Запропоновано нові підходи, алгоритми та їх програмна реалізація в області автоматизованого управління обробкою великих даних, вирішення задач менеджменту для окремих аспектів діяльності IT-компаній.

Для викладачів університетів, науковців, фахівців, аспірантів.

The collection contains research results related to models and methods of artificial intelligence, management in organizational and technical systems, and the use of Data Mining methods and tools to solve applied problems. New approaches, algorithms and their software implementation in the field of automated management of big data processing, solving management problems for certain aspects of IT companies' activities are proposed.

For university professors, researchers, specialists, graduate students.

Свідоцтво про держреєстрацію
печатного засобу масової інформації КВ № 4619 від 18.10.2000 р.

Адреса редакційної колегії: Україна, 61166, Харків, просп. Науки, 14, Харківський національний університет радіоелектроніки, кімн. 254, тел. (057) 70-21-451

ЗМІСТ

УДК 004.94:004.78 DOI: 10.30837/0135-1710.2022.178.004

К.Е. ПЕТРОВ, Є.К. ВОРОБЙОВ, І.В. КОБЗЕВ

СИНТЕЗ МОДЕЛІ КЛАСИФІКАЦІЇ ДІАЛОГОВИХ АКТІВ НА ОСНОВІ ВИКОРИСТАННЯ РЕКУРЕНТНИХ НЕЙРОННИХ МЕРЕЖ

Запропоновано математичну модель класифікації діалогових актів, яка дозволяє враховувати контекст попередніх висловлювань в рамках взаємодії з користувачем. Модель побудовано на основі використання апарату рекурентних нейронних мереж та механізму уваги. Наведено архітектуру штучної нейронної мережі. Проведено її навчання та тестування з використанням набору даних Switchboard Dialogue Act Corpus. Наведено результати комп'ютерного моделювання, які демонструють працездатність та ефективність запропонованої моделі.

1. Вступ

Проблема взаємодії людини з комп'ютером існує з моменту появи обчислювальної техніки. На початку «спілкуватися» з електронною обчислювальною машиною могли лише програмісти. Згодом, по мірі розширення сфери використання комп'ютерної техніки та збільшення масштабів її застосування, користувачі почали втягуватись в процес безпосередньої взаємодії з комп'ютером, що призвело до появи масової категорії прямих кінцевих користувачів, що працюють в діалоговому режимі. Таким чином, сформувалась проблема обробки природної мови (Natural Language Processing, NLP) та її транслювання у машинне представлення.

На теперішній час обробка природної мови є однією з найбільш досліджуваних проблем в галузі штучного інтелекту. Напрями цих досліджень достатньо різноманітні, зокрема, сюди можна віднести такі завдання, як машинний переклад; інформаційний пошук; реферування та анотування текстів; класифікація текстів, в тому числі тематичне моделювання; створення чат-ботів; аналіз тональності текстів; видобування знань з текстів; автоматична генерація текстів тощо.

Ще одним з актуальних завдань цього напряму є автоматизація процесу ведення (генерації) діалогів типу «машина-людина». Прикладами такої автоматизації можуть слугувати персональні мобільні асистенти, системи типу «запитання-відповідь», чат-боти тощо. Подібні системи дозволяють користувачу вирішувати певні задачі через спілкування з комп'ютером природною мовою у форматі діалогу.

Однією з ключових задач генерації діалогів є класифікація діалогових актів (Dialog Act, DA). Під DA розуміється висловлювання в контексті розмовного діалогу, яке виконує певну функцію в ньому. Іншими словами, акт певної репліки - це мета, яку хоче досягти автор, висловлюючи її. Прикладами таких актів можуть слугувати запити інформації, твердження, згода тощо. Розпізнавання та класифікація DA суттєво підвищує якість та релевантність згенерованих відповідей, що, в свою чергу, підвищує якість усього діалогу.

2. Аналіз сучасних досліджень у галузі класифікації діалогових актів

Діалогові системи можуть бути умовно поділені на два основні класи.

До першого з них відносяться системи, орієнтовані на завдання, які мають на меті допомогти користувачеві виконати певні задачі (наприклад, знайти продукти, забронювати помешкання або столик у ресторані тощо). Основний підхід при реалізації таких діалогових систем полягає у тому, щоб розглядати відповідь в діалозі як конвеєр. Система спочатку намагається зрозуміти повідомлення, що передає людина, представляє його як внутрішній стан, а потім виконує деякі дії відповідно до правил, враховуючи поточний стан діалогу, і, нарешті, дія перетворюється на свою поверхневу форму - природну мову.

До другого класу можна віднести системи, не орієнтовані на виконання задачі (також відомі як чат-боти), які взаємодіють з людиною, щоб забезпечити розумні відповіді та розваги. Зазвичай вони зосереджені на спілкуванні з людиною у відкритих доменах. Незважаючи на те, що такі системи, на перший погляд, виконують розважальні функції, вони домінують у багатьох реальних застосунках. Для реалізації таких систем застосовуються два основні підходи - генеративні методи, такі як Seq2Seq [1], які генерують належні відповіді під час розмови, і методи на основі пошуку, які навчаються вибирати відповіді для поточної розмови з певної бази знань.

Обидва класи систем мають вміти генерувати коректні та змістовні відповіді на репліки співрозмовника. Першим та одним з головних кроків цього процесу є розуміння та опрацювання попередньої репліки або декількох реплік. Під розумінням репліки мається на увазі її лексичний та синтаксичний розбір, знаходження її сенсу та віднесення до попередніх висловлювань у діалозі. Для покращення розуміння реплік також може бути застосована їх класифікація - тобто віднесення реплік до певного класу в залежності від їх мети та сенсу [2]. Групування висловлювань у класи дозволяє застосовувати певні шаблони генерації відповідей в залежності від цих класів.

Класифікація загальної мети висловлювань користувача в розмові (таких, наприклад, як відкрите запитання, висловлювання думки або запит думки), також відомих як DA, є ключовим кроком у розумінні природної мови для розмовних агентів. Хоча класифікація DA була детально вивчена в розмовах між людьми, вона недостатньо вивчена для нових автоматизованих розмовних агентів з довільними темами. Більше того, незважаючи на значний прогрес у класифікації DA на рівні висловлювань, повне розуміння висловлювань діалогу вимагає знання контексту розмови.

При вирішені задачі класифікації DA необхідно розглянути два попередніх етапи, перш ніж можна буде застосувати певну техніку класифікації. Перший етап полягає у виборі набору тегів, а другий - у визначенні ознак, які будуть використовуватися класифікаторами.

Визначення набору тегів DA є важливим та одночасно достатньо складним етапом, оскільки цей процес є результатом компромісу між трьома суперечливими вимогами. По-перше, теги DA повинні бути достатньо конкретними, щоб кодувати детальні характеристики цільового завдання; по-друге, бути достатньо загальними, щоб бути корисними для різних завдань, або, принаймні, стійкими до мінливості та розвитку цільового застосування; по-третє, бути чіткими та легко відокремлюваними, щоб максимізувати узгодження між людьми, що анотують репліки.

Дослідження DA призвело до кількох схем анотації. Серед них - розмітка в кількох шарах (DAMSL) [3], яка слугує основою для анотації двох довідкових корпусів, SwDA [4] та MRDA [5]. DAMSL розроблявся як універсальний набір. Теги в DAMSL мають декілька рівнів класифікації. На першому рівні всі теги поділяються на чотири основні класи: комунікативний статус (Communicative Status) - фіксує, чи є висловлювання зрозумілим і чи було його успішно завершено; інформаційний рівень (Information Level) - характеристика сенсового змісту висловлювання; функції, орієнтовані на перспективу (Forward Looking Function) - визначають, як поточне висловлювання обмежує майбутні переконання та дії учасників і впливає на дискурс та функції зворотного огляду (Backward Looking Function), тобто визначає, як поточне висловлювання пов'язане з попереднім дискурсом. Загалом, ці класи вважаються ортогональними, і можна побудувати приклади для будь-якої можливої їх комбінації.

Ще одна схема DIT++, яка призвела до створення стандарту ISO 24617-2 [6], пропонує організацію, що заснована на різних вимірах (наприклад, управління діалоговими ходами, управління соціальними зобов'язаннями тощо), кожен з яких містить різні DA. Загалом DIT++ пропонує виділити більше 100 DA, а DAMSL - 226 DA, які зазвичай групуються в 42 мітки. Але така велика кількість класів не призводить до підвищення ефективності вирішення задачі автоматичної класифікації. У значній кількості досліджень пропонують обмежити кількість класів, використовуючи або загальні метакласи, або найчастіше використовувані. Наприклад, у [7, 8], які базуються на використанні DAMSL, застосовується набір тегів, скорочений до 5 найпоширеніших класів (твердження, зворотна відповідь, думка, залишення, згода).

Зменшення набору тегів також можна зробити для пристосування класифікації до потреб конкретної доменної діалогової системи. Це, наприклад, пропонується у [9], де описується комунікація в контексті реагування на катастрофи за допомогою роботів. В ній пропонується використовувати конкретний набір метакласів ISO, адаптованих до потреб системи, об'єднавши 20 найкорисніших DA у 8 метакласів (зв'язок, повідомлення, твердження, запит, запитання, підтвердження, скасування, заперечення).

Розглянемо детальніше зміст другого етапу - виділення ознак для класифікації DA, тобто властивостей, які використовуються класифікаторами. Такі інформаційні властивості можна умовно поділити на декілька типів, описаних нижче.

Лексична інформація. Кожне висловлювання складається з послідовності слів. Як правило, DA висловлювання можна частково вивести зі списків слів, які утворюють це висловлю-

вання. Наприклад, запитання часто містять питальне слово, яке рідко зустрічається в інших класах DA. Лексичну інформацію зазвичай фіксують уніграми слів.

Синтаксична інформація, пов'язана з порядком слів у висловлюванні. Наприклад, у французькій та чеській мовах відносний порядок появи підмета і присудка може використовуватися для розрізнення декларацій та питань. N-грами слів часто використовуються при розпізнаванні DA для моделювання деякої локальної синтаксичної інформації.

Семантична інформація. DA також залежить від значень висловлювання та слів, які його складають. Прикладами можуть слугувати як широкі тематичні категорії, такі як «погода», «спорт», так і точні інтерпретації на основі фреймів, наприклад, «показати рейси з Лондона до Парижа 12 березня».

Просодія, і зокрема, мелодика висловлювання. Зазвичай запитання мають зростаючу мелодику в кінці висловлювання, тоді як висловлювання часто характеризуються дещо спадною мелодикою.

Контекст DA, тобто будь-який DA залежить від попередніх (і наступних). Найважливішим контекстом є попередні вислови. Наприклад, відповіді «так» або «ні», найімовірніше, будуть слідувати за запитанням. Послідовність DA також називається історією діалогу.

Тепер розглянемо детальніше методи, що використовуються для класифікації DA.

Основні типи підходів автоматичного розпізнавання DA, що розглядаються в літературі, можна широко класифікувати на баєсівські та небаєсівські підходи.

Баєсівські підходи ґрунтуються на баєсовому висновуванні [10]. Вони відносяться до найбільш розповсюджених та досліджених підходів. Розглянемо основні моделі, що використовують цей підхід.

Лексичні n-грам моделі [11] та n-грам моделі на основі послідовності реплік [12], в яких історія діалогу зазвичай моделюється за допомогою статистичної граматики дискурсу, яка представляє попередню ймовірність послідовності актів .

Приховані марковські моделі (ПММ) [13], де використовуються моделі n-го порядку, що означає, що кожен DA залежить від n попередніх DA (так само, як і для n-грам). Потім кожен стан ПММ моделює один DA, а спостереження відповідають ознакам рівня висловлювання. Імовірності переходу навчаються на проанотованому наборі даних. Розпізнавання DA здійснюється за допомогою деякого алгоритму динамічного програмування. Для моделювання історії діалогів успішно використовуються ПММ зі словами та просодичними ознаками. Так у [13] використовуються інтонаційні події та функції нахилу, такі як F0 (падіння/підйом), енергія, тривалість тощо. Запропонована модель досягає 64 % точності на корпусі DCIEM [14] з 12 класами DA. Існують також приклади поєднання ПММ з нейронними мережами [15], в якому був отриманий результат близько 76 % точності на іспанському корпусі CallHome.

Баєсові мережі. Приклад їх застосування для вирішення задачі розпізнавання DA наведено в [16]. Використовуються два типи ознак: ознаки висловлювання (слова у висловлюванні w_i) та ознаки контексту (попередній DA $C_{\tau-1}$). Автори порівнюють дві різні баєсові мережі, щоб розпізнати DA. Точність розпізнавання складає приблизно 64 % на підмножині корпусу MRDA і зі зменшеним набором класів.

Небаєсовські підходи також успішно використовуються в області розпізнавання DA, але вони не настільки популярні, як баєсовські. Прикладами таких підходів є дерева рішень, навчання на основі пам'яті та на основі трансформації, нейронні мережі, такі як багатошаровий перцептрон або мережі Кохонена тощо.

Моделі на основі дерев рішень (або дерев класифікації та регресії). У випадку розпізнавання DA рішення зазвичай стосуються особливостей висловлювання. Під час ухвалення кожного рішення щодо віднесення DA до відповідного класу реалізується процес порівняння значення деякої ознаки з пороговим.

Навчання на основі пам'яті (MBL) є застосуванням теорії міркувань на основі пам'яті в галузі машинного навчання. Ця теорія базується на припущенні, що можна обробляти новий зразок, порівнюючи його із збереженими представленнями попередніх зразків. Так у [17] розглядається використання MBL для автоматичної розмітки DA на корпусі SwDA [4], який складається зі спонтанних телефонних розмов між людьми. Автор експериментує з різною кількістю сусідів. Найкращий результат - близько 72 % точності з трьома сусідами.

Основна ідея навчання на основі трансформацій (TBL) полягає в тому, щоб почати з деякого простого рішення проблеми і застосувати перетворення для отримання кінцевого результату.

TBL можна застосувати до більшості задач класифікації, в тому числі для автоматичного розпізнавання DA. Наприклад, у [18] використовується TBL зі стратегією Монте-Карло на корпусі VERBMOBIL. Отримана точність класифікації DA становить близько 71 %.

Однією з найчастіше використовуваних моделей нейронної мережі в області розпізнавання DA є багатошаровий перцептрон (MLP). Так у [19] описується можливість використання набору бінарних функцій для навчання MLP. Ці функції обчислюються автоматично шляхом поєднання фразового аналізу на основі граматики та методів машинного навчання. Отримана точність розпізнавання DA близько 71 % для англійської та 69 % для німецької мов при використанні датасету NESPOLE.

Приклад використання мережі Кохонена для розпізнавання DA розглянуто в [20]. Автори використовують сім поверхневих ознак висловлювання: мовець, режим речення, наявність чи відсутність слів-маркерів відкритих запитань, наявність чи відсутність знаку питання тощо. Кожне висловлювання представлено шаблоном цих ознак, який кодується у двійковому форматі. Спочатку точна кількість класів DA невідома апріорі. Процес кластеризації переривається після того, як буде знайдено задану кількість кластерів.

Окремо слід розглянути підходи, що базуються на використанні рекурентних нейронних мереж (RNN).

Рекурентні нейронні мережі є дуже важливим варіантом нейронних мереж, які активно використовуються в обробці природної мови. Концептуально їх використання відрізняється від використання стандартної нейронної мережі, оскільки стандартним введенням в RNN є слово, а не вся вибірка, як у випадку стандартної нейронної мережі. Це дає мережі можливість працювати з реченнями різної довжини, чого неможливо досягти в стандартній нейронній мережі через її фіксовану структуру. Це також надає додаткову перевагу спільного використання функцій, засвоєних у різних позиціях тексту, які неможливо отримати в стандартній нейронній мережі.

Незважаючи на усі переваги рекурентних нейронних мереж, вони мають певні недоліки.

По-перше, RNN з класичною архітектурою здатні фіксувати залежності лише в одному напрямку мови. Так у випадку обробки природної мови передбачається, що слово, яке йде після, не впливає на значення слова, яке йде перед. Проте, враховуючи структури мов, можна стверджувати, що це твердження не завжди правильне.

По-друге, RNN також не дуже добре фіксує довгострокові залежності та проблему зникаючого градієнта [21].

Обидва ці обмеження породжують нові типи архітектур RNN, зокрема:

- вентильний рекурентний вузол (GRU) - це модифікація основного рекурентного блоку, яка допомагає фіксувати залежності на великій відстані, а також дуже допомагає у вирішенні проблеми зникаючого градієнта [21];

- двонаправлені рекурентні нейронні мережі (BRNN), здібні враховувати наслідки слова, написаного не тільки перед поточним словом, що дуже важливо при обробці природної мови.

Механізм уваги (Attention Mechanism) пропонується як рішення обмеження моделі кодер-декодер, яка кодує вхідну послідовність до одного вектору фіксованої довжини, з якого декодується вихідний результат в кожний момент часу [22]. Вважається, що ця проблема з'являється під час декодування довгих послідовностей, оскільки нейронній мережі важко справлятися з довгими реченнями, особливо тими, які довші за речення в навчальному корпусі.

Ще одним видом уваги є самоувага. Вона визначає увагу тієї ж послідовності. Замість того, щоб шукати асоціацію/вирівнювання послідовності введення-виведення, відшукуються оцінки між елементами послідовності.

Використання уваги може значно покращити результати моделі завдяки усуненню проблеми зникаючого градієнта, оскільки цей механізм забезпечує прямі зв'язки між станами кодера та декодера. Концептуально він діє подібно до пропуску зв'язків у згорткових нейронних мережах.

Іншою перевагою є зрозумілість. Перевіряючи розподіл ваг уваги, ми можемо отримати уявлення про поведінку моделі, а також зрозуміти її обмеження.

BERT (Bidirectional Encoder Representations from Transformers) - це модель представлення мови, розроблена дослідниками Google AI Language [23]. Ключовою технічною інновацією BERT є застосування популярної моделі двонаправленого навчання Transformer до мовного моделювання. Результати роботи показують, що мовна модель, яка навчається двосторонньо, може мати більш глибоке відчуття мовного контексту та потоку, ніж одно-

спрямовані мовні моделі. У [23] докладно описується нова техніка під назвою Masked LM (MLM), яка дозволяє проводити двонаправлене навчання в моделях, у яких раніше це було неможливо. Існує також ще одна популярна стратегія навчання - прогноз наступного речення (NSP).

Під час навчання моделі BERT, MLM і NSP навчаються разом з метою мінімізації комбінованої функції втрат двох стратегій.

BERT можна використовувати для різноманітних мовних завдань, додаючи лише невеликий шар до основної моделі.

Таким чином, задача класифікації DA в системах обробки природної мови є актуальною, про що також свідчить і велика кількість публікацій на цю тему.

Метою даного дослідження є розробка моделі класифікації DA, що може бути використана для автоматизації процесів в рамках розробки сучасних інтелектуальних діалогових систем.

Для досягнення поставленої мети необхідно вирішити такі основні задачі:

- розробити математичну модель класифікації DA на основі використання апарату рекурентних нейронних мереж;
- провести експериментальну перевірку працездатності та ефективності запропонованої моделі класифікації.

3. Постановка задачі дослідження

Формально, при вирішенні задачі класифікації DA розмову C можна представити як вхідну інформацію, яка являє собою послідовність висловлювань різної довжини $(u_1, u_2, ..., u_L)$. Кожне висловлювання u_i, $i = \overline{1, L}$, у свою чергу, є послідовністю слів різної довжини $(w_i^1, w_i^2, ..., w_i^{N_i})$ і має відповідну цільову мітку y_i, $i = \overline{1, L}$. Таким чином, кожна розмова (послідовність висловлювань) відображається на відповідну послідовність цільових міток $(y_1, y_2, ..., y_L)$, яка представляє DA, пов'язані з відповідними висловлюваннями. Тобто для розпізнання намірів користувача в ході розмови необхідно створити навчений на векторному представленні висловлювання класифікатор, який є одним із важливих складових реалізації процесу автоматичної генерації діалогів.

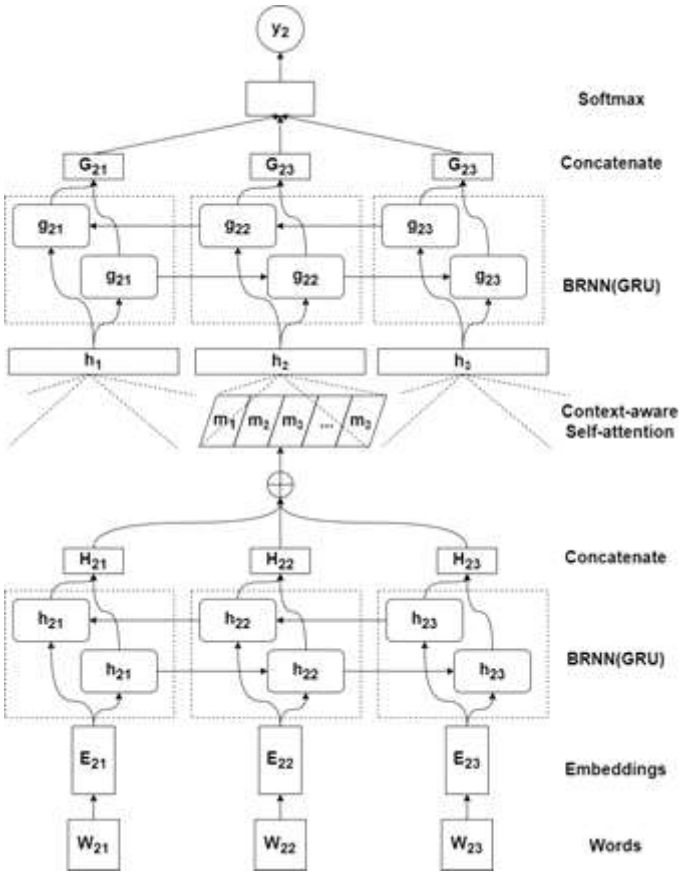

Рис. 1. Архітектура штучної нейронної мережі, що реалізує модель класифікації діалогових актів

4. Модель класифікації діалогових актів

Як показано вище, рекурентні нейроні мережі та їх модифікації дуже добре зарекомендували себе у вирішенні різноманітних задач обробки природної мови. Саме тому буде доцільно обрати саме їх як основу для синтезу моделі. Архітектуру моделі представлено на рис. 1.

Модель складається з трьох основних компонентів: RNN на рівні висловлювання, яка кодує інформацію у висловлюваннях на рівні слів та символів; контекстно-свідомий механізм самоуваги, який об'єднує репрезентації слів у репрезентації висловлювань; RNN рівня розмови - шар

класифікатора для визначення класу DA висловлювання, який об'єднує попередні та координує їх обчислення.

Розглянемо ці компоненти детальніше.

RNN рівня висловлювання. На цьому шарі слова вхідної репліки кодуються в так звані ембедінги - векторні представлення. Для цього використовується модель RoBERTa [24]. Ця модель є вдосконаленням моделі BERT командою Facebook AI Research. До основних відмінностей цієї моделі від BERT можна віднести спрощену систему навчання та значно збільшений корпус даних для навчання. RoBERTa демонструє від 2 до 20 % покращення результату порівняно з BERT в залежності від завдання та застосування.

Таким чином, кожне слово репліки кодується в ембедінг. Потім йде двонаправлений шар GRU (BGRU). Об'єднання прямих і зворотних вихідних даних BGRU генерує ембедінги висловлювання, які служать вхідним сигналом для контекстно-свідомого механізму самоуваги на рівні висловлювання, який вивчає остаточне представлення висловлювання.

Далі йде контекстно-свідомий шар з самоувагою. Самоуважні представлення кодують послідовність змінної довжини у фіксований розмір, використовуючи механізм уваги, який враховує різні позиції в послідовності. Тут використовується попередній прихований стан з RNN рівня висловлювання, який надає контекст розмови на даний момент і потім об'єднує його з прихованими станами всіх складових слів у висловлюванні в самоуважний кодер, який обчислює 2D-представлення кожного вхідного висловлювання.

Репліка u_i, яку можна представити як послідовність слів $(w_i^1, w_i^2, ..., w_i^n)$, за допомогою шару емебдінгів кодується в вектор розмірності d для кожного слова. Потім отримані вектори передаються до шару BGRU, приховані виходи якого об'єднуються для кожного кроку часу. Формально ці кроки можна представити формулами

$$\overrightarrow{h_l^j} = \overrightarrow{GRU}(w_i^j \cdot \overrightarrow{h_l^{j-1}});\ \overleftarrow{h_l^j} = \overleftarrow{GRU}(w_i^j \cdot \overleftarrow{h_l^{j+1}});\ h_i^j = concat(\overrightarrow{h_l^j}, \overleftarrow{h_l^j});\ H_i = (h_i^1, h_i^2, h_i^3, ..., h_i^n), \quad (1)$$

де H_i - n-й вихід BGRU.

Далі контекстні оцінки самоуваги S_i обчислюються за формулою

$$S_i = W_{s2} \cdot (W_{s1} \cdot H_i^T + W_{s3} \cdot \overrightarrow{g_{l-1}} + b), \quad (2)$$

де W_{s1}, W_{s2}, W_{s3} - матриці ваг відповідних розмірів; b - зміщення; $\overrightarrow{g_{l-1}}$ - прихований стан RNN рівня розмови.

Рівняння (2) можна розглядати як двошаровий MLP зі зміщенням b та W_{s1}, W_{s2}, W_{s3} як ваговими параметрами. Оцінки S_i відображаються в матрицю ймовірностей A_i за допомогою функції $A_i = soft\max(S_i)$.

Потім ця матриця ймовірностей A_i використовується для отримання двовимірного представлення M_i вхідного висловлювання шляхом використання прихованих станів GRU H_i відповідно до вагових коефіцієнтів уваги, заданих A_i, за формулою

$$M_i = A_i \cdot H_i. \quad (3)$$

Це двовимірне представлення потім проєктується на одновимірний ембедінг (позначається як h_i), використовуючи повнозв'язаний шар. Потім за допомогою RNN рівня розмови цей ембедінг перетворюється на g_i за формулами

$$\overrightarrow{g_l} = \overrightarrow{GRU}(h_i \cdot \overrightarrow{g_{l-1}});\ \overleftarrow{g_l} = \overleftarrow{GRU}(h_i \cdot \overleftarrow{g_{l+1}});\ g_i = concat(\overrightarrow{g_l}, \overleftarrow{g_l}). \quad (4)$$

Вектор g_i надає контекст розмови, який використовується для вивчення показників уваги та двовимірного представлення M_{i+1} для наступного висловлювання в розмові h_{i+1}.

Наприкінці використовуємо RNN рівня розмови.

Представлення висловлювання з попереднього кроку передається на рівень RNN рівня розмови, який є іншим двонаправленим шаром GRU, що використовується для кодування висловлювань у розмові. Приховані стани об'єднуються, щоб отримати остаточне представлення G_i кожного висловлювання, яке далі поширюється на шар класифікатора. Він, в свою чергу, складається з трьох повноз'єднаних шарів з функцією активації типу "нещільний випрямлений лінійний вузол". Перші два шари необхідні для зменшення розмірності мережі. Обраний датасет має 43 класи, тому вихід нейронної мережі має відповідну розмірність.

Розмірності ключових шарів нейронної мережі наведено в табл. 1.

Таблиця 1

Назва шару	Вхідна розмірність	Вихідна розмірність
RoBERTa Embeddings	256	768
BRNN	768	768*2
ContextAwareAttention	1536	1
BRNN	1	768*2
Linear	1536	256
Linear	256	128
Linear	128	43

5. Реалізація та навчання RNN для класифікації діалогових актів

Як набір даних для навчання та тестування моделі було обрано SwDA (Switchboard Dialogue Act Corpus) [4]. Цей набір складається з близько 2400 телефонних розмов між 543 абонентами (302 чоловіками та 241 жінкою) з усіх регіонів США. Автоматичний комп'ютеризований оператор обробляв дзвінки, надаючи абоненту відповідні записані підказки, вибираючи для участі в розмові іншу особу (абонента) та набираючи її номер, вводячи тему для обговорення та записуючи промову двох суб'єктів на окремі канали. Було представлено близько 70 тем, з яких близько 50 були найбільш розповсюджені. Вибір тем і тих, хто викликає, був обмежений таким чином, щоб, по-перше, два абоненти не говорили разом більше одного разу і, по-друге, ніхто не говорив більше одного разу на певну тему.

SwDA є модифікацією оригінального датасету, кожна розмова була проанотована з використанням набору тегів DAMSL. У кодуванні було використано 220 тегів; 130 з них зустрічалися менше ніж 10 разів кожен, тому 220 тегів було об'єднано у 42 більші класи.

Для реалізації моделі було використано наступний інструментарій: мова програмування Python, фреймворк машинного навчання з відкритим кодом PyTorch, оболонка для високопродуктивних обчислень PyTorch Lightning та бібліотека Pandas для попередньої обробки даних. Навчання моделі відбувалося з використанням платформ Kaggle та Google Colaboratory.

Графіки зміни точності та функції втрати представлено відповідно на рис. 2 та рис. 3. Для відображення загальної тенденції було застосовано згладжування.

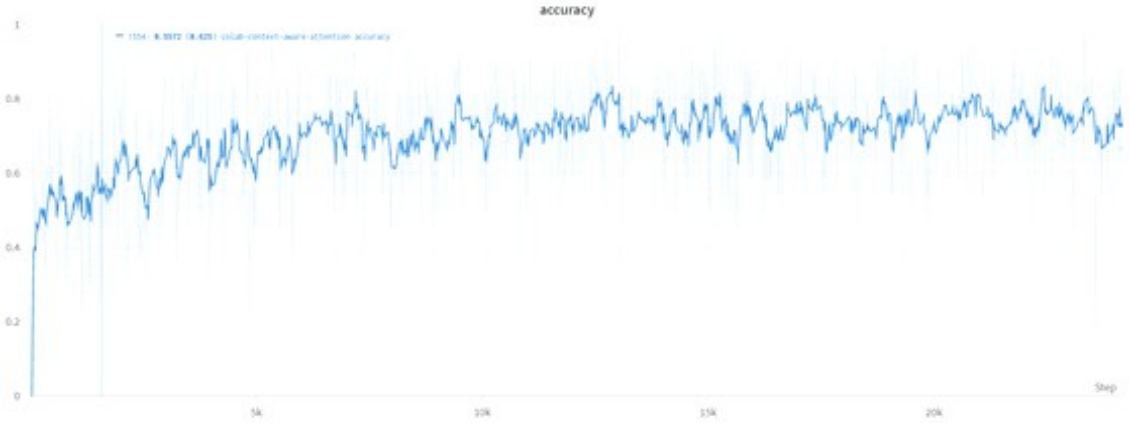

Рис. 2. Графік зміни точності моделі

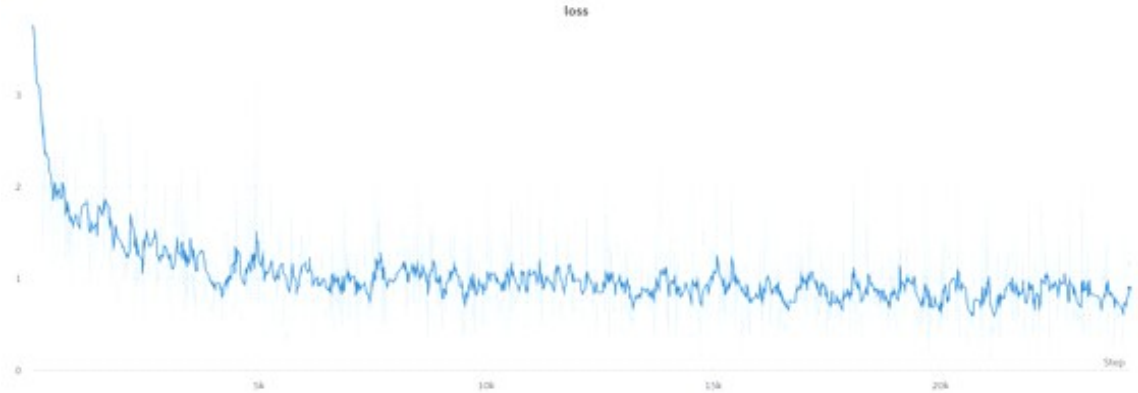

Рис. 3. Графік зміни функції втрат

Запропонована в роботі модель навчалася протягом 10 поколінь. Приблизний час навчання одного покоління з використанням прискорених графічних процесорів склав 2,5 години. В результаті було отримано максимальну точність 75,35 % для тестової вибірки та 76,62 % для валідаційної.

6. Обговорення результатів та висновки

Розпізнавання та класифікація DA дозволяє значно покращити якість генерації відповідей у діалогових системах.

В роботі розроблено математичну модель класифікації DA, яка базується на використанні апарату рекурентних нейронних мереж та механізму уваги. Її можна віднести до семантичних моделей, оскільки вона працює лише з текстовим представленням діалогу та не бере до уваги просодичні ознаки.

Було розроблено архітектуру відповідної RNN та проведено її навчання з використанням одного з найбільш розповсюджених для вирішення цієї задачі набору даних SwDA.

До основних недоліків запропонованої моделі можна віднести достатньо тривалий час навчання через її складність та велику кількість параметрів, а також відсутність можливості опрацювання просодичних ознак, що може вплинути на точність класифікації.

В результаті тестування моделі було отримано достатньо високі показники точності класифікації. Досягнута в роботі точність класифікації перевершує результати, представлені в [25] на 2,2 %, проте все ще менша, ніж в [26], де точність складає 81,3 %.

Таким чином, постає питання про подальше вдосконалення моделі для отримання кращих результатів. Серед перспективних шляхів подальших досліджень можна розглянути застосування моделей умовного випадкового поля та довгої короткочасної пам'яті (LSTM) [27] як класифікатора на останньому шарі запропонованої мережі.

Можна також дослідити інші моделі уваги, використання яких може вплинути на зміну ваги слів у кожному висловлюванні.

В перспективі було б доцільним спробувати об'єднати запропоновану модель з іншими моделями обробки природної мови. Наприклад, використати моделі розпізнавання іменованих сутностей чи аналізу емоційного забарвлення разом з класифікацією.

Література: 1. *Cho K., Merrienboer B., Gulcehre C., Bougares F., Schwenk H., Bengio Y.* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Proceedings of the conference on empirical methods in natural language processing (EMNLP). 2014. https://doi.org/10.3115/v1/d14-1179. 2. *Воробйов Є. К., Петров К. Е.* Дослідження методів класифікації діалогових актів . Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: тези доп. дванадцатої міжнар. науково-техн. конф. 2022. С. 14. 3. *Core M., Allen J.* Coding dialogs with the DAMSL annotation scheme. AAAI fall symposium on communicative action in humans and machines. 1997. P. 28-35. 4. *Jurafsky D., Shriberg E., Biasca D.* Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual. 1997. 5. *Shriberg E., Dhillon R., Bhagat S., Ang J., Carvey H.* The ICSI meeting recorder dialog act (MRDA) corpus. Proceedings of the Human Language Technology Conference at the North American Chapter of the Association for Computational Linguistics. 2004. https://doi.org/10.21236/ada460980. 6. *Bunt H., Petukhova V., Traum D., Alexandersson J.* Dialogue Act Annotation with the ISO 24617-2 Standard. Multimodal interaction with W3C standards. Cham: Springer International Publishing. 2016. P. 109-135. https://doi.org/10.1007/978-3-319-42816-1_. 7. *Ang J., Liu Yang, Shriberg E.* Automatic dialog act segmentation

and classification in multiparty meetings // IEEE international conference on acoustics, speech, and signal processing (ICASSP '05). 2005. https://doi.org/10.1109/icassp.2005.1415300. 8. *Stolcke A., Ries K., Coccaro N., Shriberg E., Bates R., Jurafsky D., Meteer M.* Dialogue act modeling for automatic tagging and recognition of conversational speech // Computational linguistics. 2000. № 26(3). P. 339-373. https://doi.org/10.1162/089120100561737. 9. *Chen Z., Yang R., Zhao Z., Cai D., He X.* Dialogue Act Recognition via CRF-Attentive Structured Network. The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '18). 2018. P. 225-234. https://doi.org/10.1145/3209978.3209997. 10. *Berger J. O.* Statistical decision theory and Bayesian analysis. 2nd ed. New York: Springer-Verlag. 1985. 617 p. 11. *Grau S., Sanchis E., Castro J., Vilar D.* Dialogue act classification using a bayesian approach / 9th Conference speech and computer. 2004. P. 495-499. 12. *Mast M., Niemann H., Noth E., Schukat-Talamazzini E.G.* Automatic classification of dialog acts with Semantic Classification Trees and Polygrams. Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing. Springer, Berlin, Heidelberg. 1996. P. 217-229. https://doi.org/10.1007/3-540-60925-3_49. 13. *Wright H.* Automatic utterance type detection using suprasegmental features // ICSPL'98. 1998. Vol. 4. P. 1403. 14. *Bard E. G., Sotillo C., Anderson A. H., Thompson H. S., Taylor M. M.* The DCIEM Map Task Corpus: spontaneous dialogue under sleep deprivation and drug treatment. Speech Commun. 1996. Vol. 20. № 1-2. P. 71-84. https://doi.org/10.1016/S0167-6393(96)00045-3. 15. *Ries K.* HMM and neural network based speech act detection // IEEE international conference on acoustics, speech, and signal processing. Proceedings (ICASSP'99). 1999. Vol. 1. P. 497-500. https://doi.org/10.1109/icassp.1999.758171. 16. *Gang Ji, Bilmes J.* Dialog act tagging using graphical models. IEEE international conference on acoustics, speech, and signal processing (ICASSP'05). 2005. Vol. 1. P. 33-36. https://doi.org/10.1109/icassp.2005.1415043. 17. *Rotaru M.* Dialog act tagging using memory-based learning. 2002. P. 255-276. https://doi.org/10.1.1.116.7922. 18. *Samuel K., Carberry S., Vijay-Shanker K.* Dialogue act tagging with Transformation-Based Learning. Proceedings of the 17th International Conference on Computational Linguistics (COLING-ACL'98).1998. P. 1050-1056. https://doi.org/10.3115/980432.980757. 19. *Levin L., Langley C., Lavie A., Gates, D., Wallace D., Peterson K.* Domain Specific Speech Acts for Spoken Language Translation // Proceedings of the Fourth SIGdial Workshop of Discourse and Dialogue. 2003. P. 208-217. 20. *Andernach T., Poel M., Salomons E.* Finding classes of dialogue utterances with kohonen networks // ECML/MLnet workshop on empirical learning of natural language processing tasks. 1997. P. 85-94. 21. *Cho K., Merrienboer B., Bahdanau D., Bengio Y.* On the properties of neural machine translation: encoder-decoder approaches // Proceedings of SSST-8: Eighth workshop on syntax, semantics and structure in statistical translation. 2014. https://doi.org/10.3115/v1/w14-4012. 22. *Graves A., Wayne G., Reynolds, M. et al.* Hybrid computing using a neural network with dynamic external memory // Nature. 2016. Vol. 538. № 7626. P. 471-476. https://doi.org/10.1038/nature20101. 23. *Devlin J., Chang M., Lee K., Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2019. Vol. 1 P. 4171-4186. https://doi.org/10.18653/v1/N19-1423. 24. *Liu Y., Ott M., Goyal N., Du J.,Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L, Stoyanov V.* RoBERTa: A Robustly Optimized BERT Pretraining Approach. 2019. https://doi.org/10.48550/arXiv.1907.11692. 25. *Lee J. Y., Dernoncourt F.* Sequential short-text classification with recurrent and convolutional neural networks // Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies. 2016. https://doi.org/10.18653/v1/n16-1062. 26. *Bothe C., Weber C., Magg S., Wermter S.* A Context-based Approach for Dialogue Act Recognition using Simple Recurrent Neural Networks. Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC'2018). 2018. https://doi.org/10.48550/arXiv.1805.06280. 27. *Hochreiter S., Schmidhuber J.* Long Short-Term Memory. Neural computation. 1997. Vol. 9. № 8. P. 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Надійшла до редколегії 22.07.2022

Петров Костянтин Едуардович, доктор технічних наук, професор, завідувач кафедри інформаційних управляючих систем Харківського національного університету радіоелектроніки. Наукові інтереси: методи прийняття рішень, оптимізація організаційних систем. Адреса: пр. Науки, 14, м. Харків, 61166, Україна. E-mail: kostiantyn.petrov@nure.ua, тел.: +38(057)7021451.

Воробйов Євген Костянтинович, магістрант кафедри штучного інтелекту Харківського національного університету радіоелектроніки. Наукові інтереси: обробка природної мови, методи машинного навчання. Адреса: пр. Науки, 14, м. Харків, 61166, Україна. E-mail: yevhen.vorobiov@nure.ua, тел.: +38(095)8567606.

Кобзев Ігор Володимирович, кандидат технічних наук, доцент, доцент кафедри інформатики та комп'ютерної техніки Харківського національного економічного університету імені Семена Кузнеця. Наукові інтереси: методи управління організаційними системами. Адреса: пр. Науки, 9-А, м. Харків, 61166 Україна. E-mail: ikobzev12@gmail.com, тел.: +38(057)5038907.

УДК 004.65:005.33 DOI: 10.30837/0135-1710.2022.178.013

М.В. ЄВЛАНОВ, І.О. ЮР'ЄВ, Д.О. МІРОШНИЧЕНКО

ОЦІНЮВАННЯ ДОЦІЛЬНОСТІ ПРОВЕДЕННЯ РЕФАКТОРИНГУ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ, ЯКА ЕКСПЛУАТУЄТЬСЯ

Розглянуто основні особливості такого різновиду ІТ-проєктів, як рефакторинг бази даних інформаційної системи, яка експлуатується. Виділено проблему визначення доцільності проведення робіт з рефакторингу бази даних. Запропоновано об'єктивні оцінки складності актуальної бази даних та сукупності запитів на зміни інформаційної системи, яка експлуатується. Розроблено показник обсягу змін, які слід внести під час рефакторингу бази даних, що пропонується.

1. Вступ

Рефакторинг бази даних (БД) - це зміна схеми БД з метою покращення дизайну бази даних і збереження як інформації, так і семантики поведінки. Зокрема, покращення дизайну БД має на меті створення умов для легкого розширення функціоналу ІТ-проєкту. Рефакторинг є усталеною концепцією в розробці програмного забезпечення, і принципи, якими керується рефакторинг програмного забезпечення, можна також застосувати до рефакторингу БД [1)]. Мета рефакторингу БД - зробити схему БД більш гнучкою, масштабованою та придатною для обслуговування з часом, мінімізуючи ризик появи помилок або втрати даних. Рефакторинг може включати ряд дій, від невеликих змін окремих таблиць або стовпців до більш масштабних змін у загальній архітектурі схеми [2].

Рефакторинг БД може бути складним і трудомістким процесом, особливо у великих або критично важливих системах. Важливо ретельно планувати та виконувати рефакторинг, а також ретельно тестувати отримані зміни, щоб переконатися, що вони не створюють нових проблем і не порушують існуючі функції.

Більшість розповсюджених методів успішного рефакторингу БД включають [2]:

- створення резервної копії або знімка БД перед внесенням будь-яких змін;
- детальне документування поточної схеми та будь-яких запропонованих змін;
- тестування змін у невиробничому середовищі перед застосуванням їх до системи, яка експлуатується;
- зведення до мінімуму обсягу та частоти змін, щоб уникнути порушення нормальної роботи;
- співпраця із зацікавленими сторонами, розробниками та адміністраторами БД, щоб переконатися, що рефакторинг відповідає бізнес-цілям і технічним вимогам.

Ефективність рефакторингу БД зазвичай пропонується визначати як мінімум змін та вибір найкращого шляху реструктуризації БД з метою підтримки розширення функцій системи.

Проведення рефакторингу у подальшому розвитку проєкту буде сприяти підвищенню ремонтопридатності БД та спрощенню її структури. Це спростить додавання нових функцій інформаційної системи, виправлення помилок і адаптацію до бізнес-вимог, що змінюються, без внесення помилок або порушення нормальної роботи. Таким чином, інвестиції в проєкт рефакторингу БД принесуть низку переваг, які покращать результати бізнесу та знизять ризики у подальшому розвитку інформаційної системи.

Але існує дуже багато проблем, пов'язаних з ініціацією, плануванням та управлінням рефакторингом БД як різновидом ІТ-проєкту модернізації інформаційної системи, що експлуатується. Причиною виникнення цих проблем є, перш за все, майже повна відсутність формального апарату, який дозволяв би описати процеси рефакторингу та особливості управління таким різновидом ІТ-проєктів. Тому виникає необхідність проведення досліджень, присвячених розробці нових та вдосконаленню існуючих моделей і методів управління рефакторингом як різновидом ІТ-проєкту та його окремими процесами.

2. Аналіз особливостей сучасних підходів до опису рефакторингу бази даних

Незважаючи на те, що рефакторинг БД як різновид ІТ-проєктів виник вже достатньо давно, рівень формальності опису рефакторингу залишається дуже низьким. Основну увагу дослідники зосереджують на створенні прикладних методик проведення рефакторингу БД.

Як приклад такої методики розглянемо запропоновану Скоттом Амблером та його колегами у [2] методику рефакторингу БД. Вона включає в себе набір процедур та практик, які допомагають безпечно та ефективно проводити рефакторинг БД у рамках проєкту розробки програмного забезпечення. Ця методика пропонує підхід до рефакторингу, який дозволяє поступово покращувати якість БД та знижувати ризики, пов'язані зі зміною БД [2].

Методика рефакторингу БД представлена С. Амблером як процес, наведений на рис. 1 [3].

Переваги використання методики рефакторингу БД включають [2, 3]:

- поліпшення якості БД;
- зниження ризиків, пов'язаних зі зміною БД;
- прискорення процесу розробки БД;
- збільшення гнучкості системи;
- поліпшення розуміння структури БД.

Як недоліки зазначеної методики слід вказати:

- значну трудомісткість та складність для невеликих проєктів або невеликих змін у БД;
- неможливість легкого застосування у випадках, коли програмне забезпечення має високий ступінь залежності від БД;
- можливість виникнення потреби у використанні спеціалізованих інструментів, таких як інструменти для автоматичного тестування БД або інструменти міграції даних, що призводить до різкого збільшення витрат на виконання рефакторингу;
- можливість використання тільки для поліпшення якості БД.

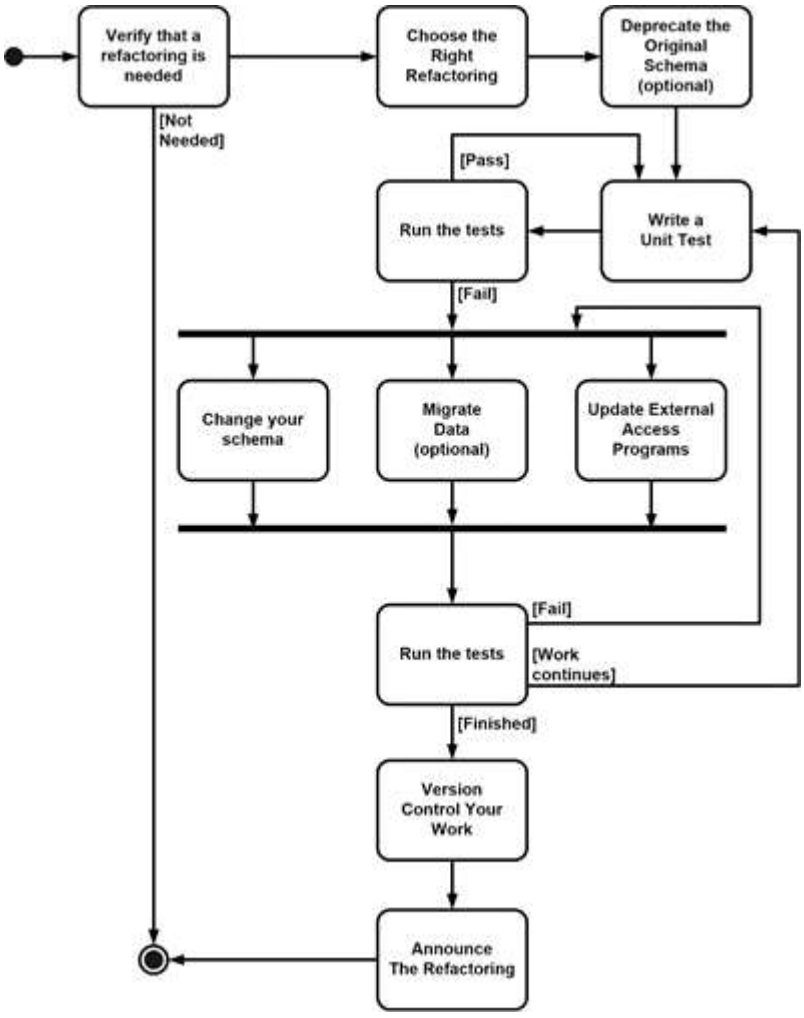

Рис. 1. Схема процесу рефакторингу бази даних

Ще одним підходом до організації і проведення рефакторингу БД є модель управління якістю даних (Data Quality Management Model), розроблена The Data Warehousing Institute. Вона надає докладний план дій для поліпшення якості даних і включає процеси, пов'язані з рефакторингом БД. Ця модель складається з таких елементів, що описують процес управління якістю даних [4]:

- визначення вимог щодо якості даних - визначення критеріїв якості даних та створення механізму для оцінки відповідності даних цим критеріям;
- аналіз якості даних - оцінка якості даних з використанням певних критеріїв якості та визначення причин помилок у даних;
- очищення даних - видалення або коригування некоректних чи неповних даних з метою покращення якості даних;

- управління якістю даних - встановлення процедур та механізмів для безперервного контролю якості даних та забезпечення їх відповідності встановленим критеріям якості;

- моніторинг якості даних - оцінка та контроль якості даних для забезпечення їх відповідності правилам та стандартам.

Використання моделі управління якістю даних для проведення рефакторингу БД дозволяє надати команді виконавців відповідного ІТ-проєкту такі можливості:

- можливість поліпшення розуміння структури БД;
- можливість поліпшення якості БД;
- можливість зниження ризиків, пов'язаних зі зміною БД;
- можливість збільшення гнучкості системи для введення нового функціоналу інформаційної системи.

Головним недоліком моделі управління якістю даних є її декларативність та обмеженість концептуальним рівнем опису. Перехід від задекларованих у цій моделі елементів управління якістю БД вимагає великих витрат часу і ресурсів.

В цілому можна стверджувати, що існуючі підходи до опису рефакторингу БД не проявляють достатньо уваги до формальних описів моделей задач, які виникають під час такого рефакторингу, та методів вирішення цих задач. І перш за все це стосується процесу "Verify that a refactoring is needed" (рис. 1). Головною метою цього процесу є об'єктивне визнання необхідності проведення рефакторингу БД до початку його проведення. Виконання цього процесу дозволяє вчасно відмовитися від проведення рефакторингу БД, якщо обмеження відповідного ІТ-проєкту перевищують очікуваний прибуток. Тому в даному дослідженні основну увагу буде приділено рішенню задачі кількісного оцінювання доцільності проведення рефакторингу БД інформаційної системи, яка експлуатується.

3. Мета і задачі дослідження

Метою даного дослідження є розробка показника кількісного оцінювання обсягу змін, які слід внести під час рефакторингу БД, що пропонується, за результатами порівняльного аналізу БД інформаційної системи, яка експлуатується, та запитів на зміну елементів цієї системи. Досягнення цієї мети дозволить своєчасно прийняти рішення про доцільність проведення рефакторингу БД або про необхідність виконання реінжинірингу БД замість запропонованого рефакторингу.

Для досягнення даної мети в статті пропонується вирішити такі задачі:

- розробити кількісні показники складності актуальної БД та сукупності запитів на зміни інформаційної системи, яка експлуатується;
- розробити показник кількісного оцінювання обсягу змін, які слід внести під час рефакторингу БД, що пропонується.

4. Моделі, які використовуються для формального опису бази даних та запитів на зміни

І для опису обсягу змін, що вносяться, і для опису актуальної БД (тобто БД інформаційної системи, яка експлуатується, до проведення рефакторингу) необхідно використовувати однаковий математичний апарат. Як такий математичний апарат пропонується використати теоретико-множинні описи діаграми "сутність - зв'язок" (Entity - Relation Diagram (ERD)), запропоновані у [5]. Даний апарат обраний тому, що ERD є найпоширенішою формою представлення проєктованої чи змінюваної БД, яка практично не залежить від конкретної моделі даних. Крім того, ERD використовується як засіб опису структур даних у класичній моделі вимог до системи, запропонованій у методології SSADM [6]. Ця класична модель вимог до системи може використовуватися і для опису запитів на зміни інформаційної системи, яка експлуатується.

Будь-яку БД можна описати кортежем [5]

$$ERD_{DB} =< E, R, D >, \quad (1)$$

де $E = \{E_i\}$ - множина сутностей ERD, $i=1,t$; $R = \{R_j\}$ - множина зв'язків, визначених на сутностях множини E та елементах цих сутностей, $i=1,n$; $D = \{D_l\}$ - множина доменів, визначених в ERD, $l=1,p$.

Кожна сутність з множини E_i може бути також описана кортежем вигляду [5]

$$E_i =< n_{E_i}, Tit_{E_i}, B_{E_i} >, \quad (2)$$

де n_{E_i} - ім'я сутності E_i; Tit_{E_i} - заголовок сутності E_i; $B_{E_i} = \left\{ e_{E_i}^k \right\}, k = 1,2,...$ - тіло сутності E_i; $e_{E_i}^k$ - k-й екземпляр сутності E_i.

Заголовок Tit_{E_i} сутності E_i, в свою чергу, може бути описаний кортежем вигляду [5]

$$Tit_{E_i} =< atr_{E_i}^j > \subseteq Atr_E, \quad (3)$$

де Atr_E - множина атрибутів, які використовуються для формування заголовків усієї множини сутностей E; $atr_{E_i}^j$ - j-й атрибут, який використовується для опису заголовку сутності E_i.

Кожен атрибут $atr_{E_i}^j$ сутності E_i може бути описаний кортежем вигляду [5]

$$atr_{E_i}^j =< n_{E_i}^j, D_{E_i}^j >, \quad (4)$$

де $n_{E_i}^j$ - ім'я атрибута $atr_{E_i}^j$; $D_{E_i}^j$ - домен атрибута $atr_{E_i}^j$, причому $D_{E_i}^j \in D$.

Необхідно відзначити, що будь-який атрибут в ERD може приймати тільки атомарні, неподільні (для ERD) значення. Виходячи з цього, кожен екземпляр сутності $e_{E_i}^k$ може бути описаний кортежем вигляду [5]

$$e_{E_i}^k =< val_{E_i}^{kj} >, \quad (5)$$

де $val_{E_i}^{kj}$ - значення, яке приймає в екземплярі сутності $e_{E_i}^k$ атрибут $atr_{E_i}^j$, який використовується для опису заголовку сутності E_i. При цьому $val_{E_i}^{kj} \in D_{E_i}^j$.

Зв'язком $R_i \in R$ в ERD прийнято називати іменовану значну асоціацію між двома сутностями чи сутності із собою. У загальному випадку зв'язок $R_i \in R$ між сутностями E_m та E_n може бути описаний кортежем вигляду [5]

$$R_i =< n_{R_i}, Atr_{E_m}^{R_i}, Atr_{E_n}^{R_i}, Pow_{E_m}^{R_i}, Pow_{E_n}^{R_i}, S_{E_m}^{R_i}, S_{E_n}^{R_i} >, \quad (6)$$

де n_{R_i} - ім'я зв'язку R_i; $Atr_{E_m}^{R_i}$ - підмножина атрибутів сутності E_m, що беруть участь в утворенні зв'язку R_i; $Atr_{E_n}^{R_i}$ - підмножина атрибутів сутності E_n, що беруть участь в утворенні зв'язку R_i; $Pow_{E_m}^{R_i}$ - потужність зв'язку R_i для сутності E_m (кількість екземплярів $e_{E_m}^k \in B_{E_m}$, що беруть участь в утворенні зв'язку R_i); $Pow_{E_n}^{R_i}$ - потужність зв'язку R_i для сутності E_n (кількість екземплярів $e_{E_n}^k \in B_{E_n}$, що беруть участь в утворенні зв'язку R_i); $S_{E_m}^{R_i}$ - ступінь участі сутності E_m у зв'язку R_i, яка визначає обов'язковість участі

екземплярів $e^k_{E_m} \in B_{E_m}$ у зв'язку R_i; $S^{R_i}_{E_n}$ - ступінь участі сутності E_n у зв'язку R_i, яка визначає обов'язковість участі екземплярів $e^k_{E_n} \in B_{E_n}$ у зв'язку R_i.

Аналізуючи множину зв'язків $R = \{R_i\}$ за ступенем обов'язковості участі сутностей в утворенні даних зв'язків, можна розглянути цю множину як сукупність підмножин [5]

$$R = R^{00} \bigcup R^{10} \bigcup R^{11}, \tag{7}$$

де R^{00} - підмножина зв'язків, необов'язкових з боку сутностей E_m та E_n; R^{10} - підмножина зв'язків, обов'язкових з боку сутності E_n і необов'язкових з боку сутності E_m; R^{11} - підмножина зв'язків, обов'язкових з боку сутностей E_m и E_n.

Тоді умови належності зв'язку R_i до однієї з розглянутих вище підмножин, виділених за ступенем обов'язковості участі сутностей в утворенні даного зв'язку, визначаються так [5]

$$R \in R^{00} \text{ за умови } \begin{cases} S^{R_i}_{E_m} = 0; \\ S^{R_i}_{E_n} = 0; \end{cases}; \qquad R \in R^{10} \text{ за умови } \begin{cases} S^{R_i}_{E_m} = 1; \\ S^{R_i}_{E_n} = 0; \end{cases} \tag{8}$$

$$R \in R^{10} \text{ за умови } \begin{cases} S^{R_i}_{E_m} = 1; \\ S^{R_i}_{E_n} = 1; \end{cases}, \tag{9}$$

причому

$$S^{R_i}_{E_m} = \begin{cases} 1 \text{ якщо } \exists e^i_{B_k} \in B_{E_k}; \\ 0 \text{ в противному випадку.} \end{cases} \tag{10}$$

Аналізуючи найпоширеніші типи зв'язків, що утворюють множину $R = \{R_i\}$, виділені за кількістю сутностей, що беруть участь в утворенні екземплярів, дану множину можна описати так

$$R = R^{om} \bigcup R^{mm}, \tag{11}$$

де R^{om} - підмножина зв'язків типу "один - до багатьох"; R^{mm} - підмножина зв'язків типу "багато - до багатьох". Зв'язок типу "один - до одного" розглядається як окремий випадок зв'язку типу "один - до багатьох".

Тоді умови належності зв'язку R_i до однієї з розглянутих вище підмножин, виділених за кількістю сутностей, що беруть участь в утворенні екземплярів; визначаються так [5]:

$$R_i \in R^{0m} \text{ за умови } \begin{cases} Pow^{R_i}_{E_m} = 1; \\ Pow^{R_i}_{E_n} = k, k = 1, p; \end{cases} \tag{12}$$

$$R_i \in R^{0m} \text{ за умови } \begin{cases} Pow^{R_i}_{E_m} = k, k = 1, p; \\ Pow^{R_i}_{E_n} = l, l = 1, p. \end{cases} \tag{13}$$

Для ERD у нотації IDEF1X немає можливості позначати ступінь участі сутності з боку "багато", і при переході від логічної моделі даних до фізичної підмножина зв'язків R^{mm} відображається в підмножину зв'язків R^{om}.

Для того, щоб отримати детальний опис зв'язків типу "один - до багатьох", введемо поняття первинного ключа PK_{E_i} для сутності E_i. Такий ключ описуватиметься виразом [5]

$$PK_{E_i} = Atr_{E_i}^{PK} \subseteq Atr_{E_i}, \tag{14}$$

де $Atr_{E_i}^{PK} =< atr_{E_i}^{jPK} >$ - підмножина атрибутів, що утворюють первинний ключ для сутності E_i. Елементи даної підмножини виділяються таким чином [5]

$$\begin{cases} val_{E_i}^{kPK} = \left\{ val_{E_i}^{kjPK} \right\} \subseteq < val_{E_i}^{kj} >; val_{E_i}^{kjPK} \neq \varnothing; r(Atr_{E_i}^{PK}) \to \min; \\ val_{E_i}^{kPK} \neq val_{E_i}^{lPK}, \text{ якщо } k \neq l; \\ val_{E_i}^{kPK} = val_{E_i}^{lPK}, \text{ якщо } k = l. \end{cases} \tag{15}$$

де $val_{E_i}^{kPK}$ - значення первинного ключа для екземпляра сутності $e_{E_i}^k$; $val_{E_i}^{kjPK}$ - значення j-го атрибуту, який бере участь в утворенні первинного ключа, що є присутнім в екземплярі сутності $e_{E_i}^k$; $r(Atr_{E_i}^{PK})$ - функція, що визначає потужність підмножини атрибутів $Atr_{E_i}^{PK}$, які утворюють первинний ключ сутності E_i.

За аналогією з визначенням первинного ключа, введемо поняття зовнішнього ключа FK_{E_m} для сутності E_m. Такий ключ можна описати виразом [5]

$$FK_{E_m} = Atr_{E_m}^{FK} \subseteq Atr_{E_m}, \tag{16}$$

де $Atr_{E_m}^{FK} =< atr_{E_m}^{jFK} >$ - підмножина атрибутів, що утворюють первинний ключ для сутності E_m. Елементи даної підмножини виділяються таким чином [5]

$$\begin{cases} \exists E_n, \text{ причому для } \mathrm{e}_{E_n}^k \in B_{E_n} \exists \mathrm{e}_{E_m}^l \in B_{E_m}; \\ val_{E_n}^{kPK} = val_{E_m}^{lFK}; val_{E_n}^{kPK} = \left\{ val_{E_n}^{kjPK} \right\}; val_{E_m}^{lFK} = \left\{ val_{E_m}^{ljFK} \right\}; \\ val_{E_m}^{lFK} \in D^j; val_{E_n}^{kPK} \in D^j; D^j \in D, j = \overline{1,t}; \end{cases} \tag{17}$$

де $val_{E_n}^{kPK}$ - значення первинного ключа для екземпляра сутності $\mathrm{e}_{E_n}^k$; $val_{E_m}^{lFK}$ - значення вторинного ключа для екземпляра сутності $\mathrm{e}_{E_m}^l$; $val_{E_n}^{kjPK}$ - значення j-го атрибуту, який бере участь в утворенні первинного ключа, що є присутнім в екземплярі сутності $\mathrm{e}_{E_n}^k$; $val_{E_m}^{ljFK}$ - значення j-го атрибуту, який бере участь в утворенні зовнішнього ключа, що є присутнім в екземплярі сутності $\mathrm{e}_{E_m}^l$; D^j - домен атрибутів $atr_{E_n}^{jPK}$ та $atr_{E_m}^{jFK}$.

Тоді (6) для елементів підмножини R^{om} можна перетворити так [5]

$$R_i^{0m} =< n_{R_i}, FK_{E_m}^{R_i}, PK_{E_n}^{R_i}, Pow_{E_m}^{R_i}, Pow_{E_n}^{R_i}, S_{E_m}^{R_i}, S_{E_n}^{R_i} > \tag{18}$$

за умови

$$\begin{cases} Pow_{E_m}^{R_i} \geq 1; \\ Pow_{E_n}^{R_i} = 1. \end{cases} \quad (19)$$

Зв'язок R_i^{0m} буде обов'язковим з боку сутності E_n, якщо виконується умова $S_{E_n}^{R_i} = 1$. Зв'язок R_i^{0m} буде ідентифікуючим, якщо вірна умова [5]

$$\begin{cases} FK_{E_m}^{R_i} \subseteq PK_{E_n}^{R_i}; \\ S_{E_n}^{R_i} = 1. \end{cases} \quad (20)$$

При цьому сутність E_m є слабкою сутністю (її первинний ключ може бути визначений без участі екземпляра сутності E_n), а E_n - сильною сутністю (за умови, що вона не є слабкою сутністю в іншому зв'язку).

Використання запропонованого математичного апарату дозволяє вирішувати задачі кількісного оцінювання складності актуальної БД, складності сукупності запитів на зміни інформаційної системи, яка експлуатується, а також обсягу змін, які слід внести під час рефакторингу БД, що пропонується, шляхом підрахунків кількості атрибутів, які беруть участь в утворенні сутностей та зв'язків, виділених у описах відповідних БД та сукупності запитів на зміни.

5. Результати розробки кількісних показників складності актуальної бази даних та сукупності запитів на зміни інформаційної системи, яка експлуатується

Виходячи із запропонованого у [5] теоретико-множинного апарату опису ERD, складність актуальної БД можна оцінити через кількість окремих об'єктів, з яких складається ця БД. У загальному випадку, виходячи з (1), цю кількість можна підрахувати таким чином

$$C_{DB} = r(E) + r(R) + r(D). \quad (21)$$

де $r(\bullet)$ - функція, , що визначає потужність відповідних множин.

Однак, у (21) не враховані особливості проведення рефакторингу. Зокрема, передбачається, що для загального домену БД під час рефакторингу, викликаного найчастіше додаванням і зміною окремих послуг, зміни не очікуються. Тому підраховувати кількість описів елементів домену БД як таких, що впливають на складність робіт з рефакторингу цієї БД, загалом недоцільно. Крім того, у (21) відсутня можливість уніфікованого опису сутностей та зв'язків актуальної БД.

Така можливість уніфікованого опису виникає, коли під час кількісного оцінювання складності актуальної БД відбувається перехід від підрахунків окремих сутностей та зв'язків цієї БД до підрахунків окремих атрибутів, що беруть участь в утворенні відповідних сутностей та зв'язків. Так кількісну оцінку складності сутностей, що утворюють актуальну БД, виходячи з (2) та (3), пропонується визначити як кількість атрибутів, які утворюють заголовок кожної конкретної сутності, яка входить до опису актуальної БД. Цю кількість атрибутів можна підрахувати за виразом

$$C_{DB_E} = \sum_{i=1}^{t} r(< atr_{E_i}^j >). \quad (22)$$

За аналогією з (22), кількісну оцінку складності зв'язків, наявних між сутностями актуальної БД, пропонується визначити як кількість атрибутів, які беруть участь в утворенні кожного конкретного зв'язку між батьківською сутністю E_m та дочірньою сутністю E_n. Цю кількість атрибутів можна підрахувати за виразом:

$$C_{DB_R} = \sum_{i=1}^{z} r(Atr_{E_m}^{R_i}) + \sum_{i=1}^{z} r(Atr_{E_n}^{R_i}), \quad (23)$$

де z - в даному випадку кількість зв'язків, визначених на сутностях множини E актуальної БД та елементах цих сутностей.

Тоді кількісну оцінку складності актуальної БД (21) можна, з урахуванням (22) та (23), представити таким чином

$$C_{DB} = \sum_{i=1}^{z} r(< atr_{E_i}^{j} >) + \left(\sum_{i=1}^{z} r(Atr_{E_m}^{R_i}) + \sum_{i=1}^{z} r(Atr_{E_n}^{R_i}) \right). \quad (24)$$

Для кількісної оцінки складності сукупності запитів на зміни інформаційної системи, яка експлуатується, визначимо можливість опису таких запитів на зміну класичною моделлю вимог, запропонованою у методології SSADM [6]. За цією моделлю кожний запит на зміну може бути описаний відповідною ERD. Тому, виходячи з (1), множина ERD, які описують усю сукупність запитів на зміну, може бути описана таким чином

$$ERD_{RFC} = \bigcup_{r=1}^{q} ERD_{RFC_r} = \bigcup_{r=1}^{q} < E_r, R_r, D_r >, \quad (25)$$

де r - ідентифікатор ERD, яка використовується для опису r-го запиту на зміну; q - кількість запитів на зміну у сукупності, на основі якої робиться оцінювання можливості проведення рефакторингу БД.

Виходячи з (21), в загальному випадку кількість окремих об'єктів, з яких складаються ERD, що описують сукупність запитів на зміни, можна підрахувати наступним чином

$$C_{RFC} = \bigcup_{r=1}^{q} r(E_r) + \bigcup_{r=1}^{q} r(R_r) + \bigcup_{r=1}^{q} r(D_r). \quad (26)$$

Однак ця формула не враховує особливості проведення рефакторингу БД. Як показано вище, для загального домену БД під час рефакторингу зміни не очікуються. Тому врахування останнього доданку у (26) перестає бути необхідним. Крім того, у (26) не врахована можливість використання для опису кожного конкретного запиту на зміну таких атрибутів, які будуть залишатися незмінними під час рефакторингу БД. Це означає, що для кількісного оцінювання складності запитів на зміни інформаційної системи, яка експлуатується, треба враховувати тільки ті атрибути, описи яких у цих запитах повністю або частково відрізняються від описів аналогічних атрибутів у актуальній БД. Виходячи з цього положення, визначимо оцінку складності сукупності запитів на зміни як оцінку кількості елементів у множинах, які є результатами різниці множини описів атрибутів з ERD, яка описує кожен запит на зміну, та множини описів атрибутів з ERD актуальної БД.

Для розрахунку такої оцінки введемо такі підмножини:

$$Atr_{E_{DB}} = \bigcup_{i=1}^{t} < atr_{E_i}^{j} >; \qquad Atr_{E_{RFC}} = \bigcup_{r=1}^{q} \bigcup_{i=1}^{t} < atr_{E_{i_r}}^{j} >. \quad (27)$$

Ці підмножини описують, відповідно, підмножину атрибутів актуальної БД та підмножину атрибутів сукупності запитів на зміни інформаційної системи, яка експлуатується.

Тоді оцінку складності сутностей, що утворюють ERD сукупності запитів на зміни, можна розрахувати за формулою

$$C_{E_{RFC}} = r\left(Atr_{E_{RFC}} \setminus Atr_{E_{DB}} \right). \quad (28)$$

Для оцінювання складності зв'язків, що утворюють ERD сукупності запитів на зміни, введемо такі підмножини:

$$Atr_{E_m}^{DB} = \bigcup_{i=1}^{t} < atr_{E_m}^{R_i} >; \qquad Atr_{E_n}^{DB} = \bigcup_{i=1}^{t} < atr_{E_n}^{R_i} >; \quad (29)$$

$$Atr_{E_m}^{RFC} = \bigcup_{r=1}^{q} \bigcup_{i=1}^{t} < atr_{E_m}^{R_{i_r}} >; \qquad Atr_{E_n}^{RFC} = \bigcup_{r=1}^{q} \bigcup_{i=1}^{t} < atr_{E_n}^{R_{i_r}} >. \quad (30)$$

Ці підмножини описують сукупності атрибутів, які беруть участь в утворенні зв'язків між сутностями ERD актуальної БД (вираз (29)) та сутностями сукупності ERD запитів на зміни (вираз (30)).

Тоді оцінку складності зв'язків, що утворюють ERD сукупності запитів на зміни, можна розрахувати за формулою

$$C_{RFC_R} = r\left(Atr_{E_m}^{RFC} \setminus Atr_{E_m}^{DB}\right) + r\left(Atr_{E_n}^{RFC} \setminus Atr_{E_n}^{DB}\right). \quad (31)$$

Виходячи з отриманих результатів, оцінку (26) складності сукупності запитів на зміни інформаційної системи, яка експлуатується, можна описати виразом

$$C_{RFC} = r\left(Atr_{E_{RFC}} \setminus Atr_{E_{DB}}\right) + \left(r\left(Atr_{E_m}^{RFC} \setminus Atr_{E_m}^{DB}\right) + r\left(Atr_{E_n}^{RFC} \setminus Atr_{E_n}^{DB}\right)\right). \quad (32)$$

6. Результати розробки показника кількісного оцінювання обсягу змін, які слід внести під час рефакторингу бази даних

Отримані у вигляді (24) та (32) оцінки складності актуальної БД та сукупності запитів на зміни інформаційної системи, яка експлуатується, пропонується використати для кількісного оцінювання обсягу змін, які слід внести під час рефакторингу БД. Але, на відміну від абсолютних кількісних оцінок (24) та (32), показник, який пропонується використовувати для кількісного оцінювання такого обсягу змін, слід розробити відносним. Це дозволить спростити прийняття рішення щодо верхньої межі доцільності проведення рефакторингу БД, за якою процес власне рефакторингу БД переходить у процес повного реінжинірингу цієї ж БД.

В загальному випадку цей показник буде мати вигляд

$$C_{REF} = C_{RFC} \Big/ C_{DB}. \quad (33)$$

З урахуванням (24) та (32), вираз (33) матиме вигляд

$$C_{REF} = \frac{r(Atr_{E_{RFC}} \setminus Atr_{E_{DB}}) + \left(r(Atr_{E_m}^{RFC} \setminus Atr_{E_m}^{DB}) + r(Atr_{E_n}^{RFC} \setminus Atr_{E_n}^{DB})\right)}{\sum_{i=1}^{t} r(< atr_{E_i}^{j} >) + \left(\sum_{i=1}^{z} r(Atr_{E_m}^{R_i}) + \sum_{i=1}^{z} r(Atr_{E_n}^{R_i})\right)}. \quad (34)$$

Для спрощення технології розрахунку значень показника (34) визнаємо як вірні таки рівняння

$$r(Atr_{E_{DB}}) = \sum_{i=1}^{t} r(< atr_{E_i}^{j} >); \quad r(Atr_{E_m}^{DB}) = \sum_{i=1}^{z} r(Atr_{E_m}^{R_i}); \quad r(Atr_{E_n}^{DB}) = \sum_{i=1}^{z} r(Atr_{E_n}^{R_i}). \quad (35)$$

Тоді правила розрахунку значень за (35) технологічно спростяться за рахунок уніфікації процедур формування окремих підмножин, а сам вираз (35) матиме вигляд

$$C_{REF} = \frac{r(Atr_{E_{RFC}} \setminus Atr_{E_{DB}}) + \left(r(Atr_{E_m}^{RFC} \setminus Atr_{E_m}^{DB}) + r(Atr_{E_n}^{RFC} \setminus Atr_{E_n}^{DB})\right)}{r(Atr_{E_{DB}}) + \left(r(Atr_{E_m}^{DB}) + r(Atr_{E_n}^{DB})\right)}. \quad (36)$$

Використання запропонованого показника (36) дозволяє визначити верхню межу доцільності проведення рефакторингу БД. Виходячи з практичного досвіду, тут і в подальшому пропонується визначати процес рефакторингу доцільним, якщо значення показника (36) не буде перевищувати 0,5. Якщо показник (36) під час визначення доцільності проведення рефакторингу БД прийматиме значення, більші за 0,5 (тобто при виконанні запитів на зміни потрібно змінити більше половини актуальної БД), то слід визнати відповідні роботи роботами з реінжинірингу БД. Але цей висновок слід вважати попереднім і таким, що потребує підтвердження чи спростування шляхом подальших досліджень.

7. Висновки і перспективи подальших досліджень

Розроблені формальні описи оцінок складності актуальної БД та сукупності запитів на зміни інформаційної системи, яка експлуатується, дозволяють оцінити кількість елементів, з

яких складаються ERD, що описують актуальну БД та кожен із сукупності запитів на зміни. В основу запропонованих оцінок покладено теоретико-множинний апарат, який описує, у тому числі, ERD через кортежі та підмножини атрибутів. Використання такого уніфікованого формального опису дозволяє в подальшому використовувати для автоматизованого розрахунку значень цих оцінок одні й ті ж процедури.

На основі розроблених формальних оцінок було запропоновано показник кількісного оцінювання обсягу змін, які слід внести під час рефакторингу БД. Визначено два основних варіанти розрахунку цього показника за виразами (34) та (36) відповідно. Описано особливість використання значень цього показника під час процесу перевірки доцільності проведення робіт з рефакторингу БД.

Слід зазначити, що отримані результати дозволяють визначити складність та обсяг змін, які слід внести під час рефакторингу БД. Але ці результати не дозволяють оцінити доцільність проведення робіт з рефакторингу БД як окремого різновиду ІТ-проєктів. Тому найближча перспектива подальших досліджень полягає у модифікації існуючих моделей оцінювання ІТ-проєктів (зокрема, параметричних моделей COCOMO II) для оцінювання трудовитрат, витрат часу та потреби у персоналі ІТ-проєкту рефакторингу БД на основі отриманих оцінок.

Список літератури: 1. *Мірошниченко Д.О.* Дослідження моделей і методів управління ІТ-проєктом рефакторингу бази даних Інтернет-провайдера задля підтримки гнучкого створення замовлень. 27-й Міжнародний молодіжний форум "Радіоелектроніка та молодь XXI століття". Зб. матеріалів форуму. Т.6. Конференція "Інформаційні інтелектуальні системи". Харків: ХНУРЕ. 2023. С.258-259. *2. Ambler S. W., Sadalage P.* Refactoring Databases: Evolutionary Database Design. Addison-Wesley Longman, Incorporated, 2006. 384 с. 3. *The Process* of Database Refactoring: Strategies for Improving Database Quality. Agile Data. URL: http://agiledata.org/essays/databaseRefactoring.html (дата звернення: 07.04.2023 р.). 4. *Data* Quality Management Model (2015 Update) - Retired // AHIMA's HIM Body of Knowledge. URL: https://library.ahima.org/PB/DataQualityModel#.ZDBZmXvP2Uk (дата звернення: 07.04.2023 р.). 5. *Левыкин В.М., Евланов М.В., Сугробов В.С.* Параллельное проектирование информационного и программного комплексов информационной системы. Радиотехника. 2006. Вып. 146. С. 89-98. 6. *Петров Э.Г., Чайников С.И., Овезгельдыев А.О.* Методология структурного системного анализа и проектирования крупномасштабных ИУС. Концепции и методы. Харьков: Рубикон, 1997. Часть 1. 140 с.

Надійшла до редколегії 17.11.2022

Євланов Максим Вікторович, доктор технічних наук, доцент, професор кафедри ІУС ХНУРЕ. Наукові інтереси: методи, моделі та інформаційні технології інженерії вимог до інформаційних систем. Адреса: Харків, 61166, пр. Науки, 14. Контактний телефон: +38(057)7021451.

Юр'єв Іван Олексійович, кандидат технічних наук, доцент, доцент кафедри ІУС ХНУРЕ. Наукові інтереси: методи, моделі та інформаційні технології управління інформаційними системами та ІТ-сервісами. Адреса: Харків, 61166, пр. Науки, 14. Контактний телефон: +38(057)7021451.

Мірошниченко Дмитро Олександрович, здобувач вищої освіти, група УПГІТм-21-1. Наукові інтереси: дослідження моделей і методів управління ІТ-проєктом рефакторингу бази даних Internet-провайдера. Адреса: Харків, 61166, пр. Науки, 14. Контактний телефон: +38(057)7021451.

УДК 681.5:004.94 DOI: 10.30837/0135-1710.2022.178.023

M. LYTVYNENKO, L. REBEZYUK

MATHEMATICAL MODEL OF ACYCLIC FUZZY CONTROL FOR TRAFFIC SIGNAL SYSTEM WITH ADAPTIVE UNCONDITIONAL TRAM PRIORITY

The mathematical model of automatic control of the traffic signals system is proposed, which implements adaptive unconditional priority for the tram approaching the intersection. The mathematical model is based on a fuzzy control law and considers the dynamics characteristics of the tram from the moment it is registered by the intersection entry detector. An example of the proposed model application in the simulation for a complex intersection is provided and a comparative analysis of the obtained time characteristics of traffic flows for fuzzy control and existing adaptive methods that do not implement the tram signal priority is performed.

1. Introduction

Public transportation (PT) or mass transit is the most effective mean of moving a large number of people according to the criterion of the required space and energy consumption, especially in densely populated regions. The introduction of high-quality PT is a significant factor in the growth of cities as centers of economic activity, because travel time reduction, as well as high reliability of the transport system, allow more participants to be involved in economic processes with lower personal costs. Efficient PT has less negative impact on the urban agglomerations' environment due to more efficient use of energy resources, as well as reducing the congestion of the cities' road network, offering an alternative way of commuting [1,2]. Therefore, the right choice of city's mobility strategy is one of the essential components that improve the quality of the urban environment. And efficient use of energy resources is an additional opportunity to significantly reduce pollution of the environment and improve transport service.

Depending on economic, technological, political, etc. factors, an appropriate degree or degrees of separation (spatial priority) of the PT infrastructure is chosen. V. Vuchik proposed the PT modes classification consisting of three (A, B, C) right of way (RoW) categories [3], decreasing the category reduces the dependence of PT mode on the road situation, but increases construction and maintenance costs of such transport infrastructure type [4]. Thus, the PT of category RoW A is completely independent of the traffic situation with its completely separated infrastructure, most often it is heavy rail (rapid transit/metro, urban railway, such as Kyiv City Express). The cost of building a RoW A transport solution from scratch limits its use to only the most important transport arteries of the agglomeration, leaving large urban areas unserved by quality inter-district connections, which is especially critical for decentralized post-industrial cities with developed service industries. Other categories of both RoW B and RoW C are implemented mainly by trams and buses (trolleybuses) but differ in traffic organization and scope. In contrast to the PT RoW C category, which provides local service, has frequent stops ($\approx 100 \div 400$ m) [4] and moves mainly in the flow of general traffic albeit can have special signals and lanes marked by line, RoW B category is less dependent on traffic conditions, as it has a spatial priority, implemented by physically separated lanes. However, only the RoW A PT is free from any at grade intersections with non-parallel traffic flows, so transit signal priority (TSP) systems are an important part of implementing a transport solution that partially or completely belongs to RoW B category.

To optimally distribute time resources between transport resources, it is necessary to perform modeling of tram dynamics, which will allow to correctly calculate the time of tram's approach to the intersection and, if possible, eliminate the need for a complete halt. This will avoid high tram starting currents, which together with the time efficiency will significantly improve its energy efficiency.

The aim of the study is to develop a mathematical model (MM) of intersection traffic signals control system, which implements the adaptive tram priority, based on tram dynamics so that it crosses the signalized intersection as quickly as possible. To achieve this goal, it is necessary to solve a number of problems:

- obtaining a mathematical model of the tram and determining the approximate time of reaching the intersection by tram;
- anticipation of the possibility of incomplete and inaccurate data being used by the model;
- adaptation to diverse types of intersections, traffic flows intensities and configurations of tram routes;
- unconditional minimization of tram passenger delay and deviation from the schedule;
- ensuring the model's stability in temporary non-standard road situations.

2. Analysis of literature sources

In case of intersection of traffic flows, the tram time priority is applied to reduce its de-lay at traffic signals [5]. If TSP is applied independently at a particular intersection, such a priority is called local. In the case of traffic signals priority control together at several intersections, the priority is considered coordinated. Traditionally, there are passive and active TSP [6]. Also, with the increase of embedded systems computing power, it becomes possible to give priority in real time depending on the traffic situation, this approach is recently referred to as adaptive priority [7]. This priority iteratively calculates the sequence of signals and their duration (traffic signal plan), based on both local characteristics of the PT and the system-wide characteristics of traffic (delays, halts, etc.). Adaptive priority requires early detection of PT vehicle to calculate the time of arrival and can be implemented on the existing system of adaptive traffic signals control, but this is not a necessary condition for its implementation [7]. Typically, the control model of traffic signal system with adaptive priority for PT interacts with the following components:
- means of detection (and interaction with) PT vehicle;
- traffic detection system;
- priority request generator (s) and servers, control system with traffic signals plans in real-time.

The model itself implements a signal control algorithm, considering the impact on other traffic and ensuring the safety of pedestrians. By definition, a TSP-capable traffic signal will not adversely affect the coordination of traffic signals [8].

Most modern systems that implement TSP work in a coordinated manner as part of the urban traffic control (UTC) systems in real time. Existing control systems with real-time TSP can be divided into two categories [9]:
- with a constant length of traffic signal cycle (rule-based) [10,11], time parameters are gradually adapted to fluctuations in traffic conditions in real time;
- with variable cycle length (optimization-based) [12-14], adaptive commands constantly optimize the traffic signal plan using the rolling horizon method [15].

It should be noted that the European approach to the implementation of TSP is quite severe, with higher levels of priority and less attention to possible negative impacts on other traffic [16].

As the process of adaptive control of the traffic light system with priority for general traffic, and the tram in particular, cannot be based on perfect, complete data, the appropriate approach to this issue is soft computing, which seeks to adhere to the principle of tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost [17]. Existing studies mainly consider the application of fuzzy computing to adaptive traffic signal control in general, without a detailed focus on TSP [18-20]. In addition, existing methods that explicitly implement TSP using soft computing use a constant duration of the traffic signal cycle and/or predetermined traffic light stages (phases), which inevitably imposes limitations on adaptive control [21,22]. For the problem of adaptive priority, the most common are methods of evolutionary and fuzzy computing.

Since the description of traffic is naturally associated with imprecision, and traffic control at a particular intersection is conducted according to certain rules, fuzzy control is a natural approach to solving this problem. The application of fuzzy logic to the problem of traffic light control involves control based on expert knowledge, rather than modeling a directly controlled process [23], which reduces computational costs. Nevertheless, the listed above papers do not use the data of PT vehicle dynamics, as there authors mainly consider a PT of RoW C category. Therefore, it is necessary to apply a mathematical model of the longitudinal dynamics of the tram approaching the intersection, slowing down for safety reasons.

3. Mathematical model of tram dynamics

The involved tram MM [24] (Tatra T3) determines its deceleration or acceleration forces by calculating the torque that the wheels receive from the motor, which determines the effect on the linear dynamics of the tram. Modeling of the braking process involves the use of an electrodynamic

braking system applied at speeds > 5 km/h. The main characteristics of the tram car that determine its dynamics are given in Table 1 [25]. The calculation of the current weight of the tram uses the assumption that the average weight of one passenger is 65 kg.

Table 1

Characteristics	Notation	Value
Estimated wheel radius	r	325 mm
Wheel mass	m_{wh}	195 kg
Power of traction motors	P_{max}	4×50 kW
Maximum speed of an empty tram	–	65 km·h^{-1}
Average deceleration during service braking	–	1.4 m·s^{-2}
Mass of empty tram car	–	18.1×10^3 kg
Nominal passenger capacity of the tram car	–	100

The linear dynamics of the tram consists of three forces. The first is the adhesion force, which depends on the tram velocity:

$$F_{ad}(v_s) = \mu(v_s)Mg \quad , \tag{1}$$

where $\mu(v_s) = c_{ad}e^{-b_{ad}v_s}$ is adhesion coefficient; $v_s = r\omega_{wh}(t) - v(t)$ is wheel slip velocity; $\omega(t)$ is angular velocity of a tram wheel rotation; $v(t)$ is linear velocity of the tram $\left[ms^{-1}\right]$; M is current mass of the tram; a_{ad}, b_{ad}, c_{ad}, d_{ad} are adhesion parameters (Table 2) [26], depending on weather conditions (0 means ideal conditions, and 3 are the worst).

Table 2

Parameter	Adhesion			
	0	1	2	3
a_{ad}	0.54	0.54	0.54	0.05
b_{ad}	1.2	1.2	1.2	0.5
c_{ad}	1.0	0.2	0.1	0.08
d_{ad}	1.0	0.2	0.1	0.08

The second force is the rolling resistance, which is determined by the empirical formula [27]:

$$F_r(v) = A + Bv + Cv^2 , \tag{2}$$

where A is coefficient associated with the mass acting on the axle of the tram, combines the frictional resistance of the rail and wheel, resistance due to track defects, as well as friction of bearings; B is coefficient associated with the lateral displacements of the tram, due to which there is a friction force between the flange of the wheel and the inner part of the rail; C is coefficient related to the cross-sectional area of rail vehicle, as well as space in-between vehicle, so its impact becomes more noticeable at speeds above 80 km/h. For the involved model A=0.0147M, B=125.83, C=0.

The third force is a projection of the force of gravity and depends on the slope (gradient) of the track $\theta : F_s(\theta) = Mg\sin(\theta)$.

Linear dynamics is determined by the angular velocity of the tram wheel, which is affected by motor torque and adhesion force torque. The torque of the tram motor depends on the position of the notch $pos \in \{-7,...,0,...,7\}$. The authors of the model [24] experimentally identified the dependence of motor torque on the notch position:

$$\tilde{\tau}_{mot}(pos) = \begin{cases} K_p pos & \text{if } pos > 0 \text{ and } \omega\tilde{\tau}_{mot} < P_{\max}, \\ P_{\max}/\omega & \text{if } pos > 0 \text{ and } \omega\tilde{\tau}_{mot} \geq P_{\max}, \\ K_n pos & \text{else,} \end{cases} \tag{3}$$

where K_p, K_n are experimentally determined proportionality coefficients corresponding to the maximum and minimum acceleration at the highest and lowest position of the notch, respectively (K_p=1449, K_n=1176).

The transition to time space τ_{mot} is using transfer function $H(s) = 3/s + 3$.

So, for the tram model involved, the dynamics equations are:

$$\dot{v} = \frac{F_{ad} - F_r - F_s}{M}, \qquad \dot{\omega} = \frac{\tau_{mot} - \tau_{ad}}{J}, \tag{4}$$

where $J = 0{,}5 m_{wh} r^2$ is the tram wheel moment of inertia.

4. Mathematical model of fuzzy control of traffic signals system

During the process of controlling the traffic light system, it is necessary to ensure the coordination of signals for different directions of movement. The considered systems and methods of traffic signal control system with adaptive TSP use two approaches:

- fixed, based on traffic signal stages, combining several compatible signals for different directions. Signals included in certain phases are determined in advance;

- flexible, based on signal groups (SG) [28]. Signals decomposition allows to adapt the composition of stages in real time, considering the current state of traffic and interoperability of SG (Fig. 1). By determining the start time and duration for the group, and not for each signal, a significant savings in computing resources can be achieved.

ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	2	1	2	2	0	1	0	2	1	1	0	2	1	0	0	0	0	1	0	0	0	0	0
1	5	0	5	2	2	2	2	0	0	2	2	2	3	0	2	0	0	0	0	2	0	0	2	2
2	1	2	0	2	2	1	1	0	2	0	1	2	2	0	0	1	0	0	0	0	0	1	0	0
3	5	3	5	0	0	5	0	0	3	0	0	2	2	0	0	0	0	2	0	0	0	0	0	3
4	5	3	5	0	0	5	0	0	3	0	5	2	0	0	0	0	0	3	0	0	0	0	3	3
5	0	2	1	2	2	0	0	0	2	0	1	0	2	1	0	0	0	0	1	0	0	0	0	0
6	1	2	1	0	0	0	0	0	2	0	1	2	0	0	0	0	0	0	1	0	0	1	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	0	0	0
8	5	0	5	3	3	5	5	0	0	5	5	3	3	0	0	0	2	0	0	0	2	0	0	2
9	1	2	0	0	0	0	0	0	2	0	0	2	0	0	0	0	0	0	1	0	0	1	0	0
10	1	2	1	2	2	1	1	0	2	0	0	2	2	1	0	0	0	0	0	0	0	1	0	0
11	0	3	5	0	2	0	5	0	3	5	5	0	0	0	0	0	0	0	0	3	0	0	0	3
12	5	4	5	2	0	5	0	0	4	0	5	0	0	0	0	0	2	0	0	0	0	0	0	2
13	8	0	0	0	0	8	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	5	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	8	0	0	0	0	8	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	4	0	0	0	0	0	5	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	8	0	0	0	8	0	0	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	4	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	4	0	5	5	0	0	0	5	0	0	6	6	0	0	0	0	0	0	0	0	0	0	0

Fig. 1. Conflict matrix (in seconds)

Figure 2 describes the diagram of the proposed traffic signal control model. Three types of traffic light signals are considered, which control the movement of distinct categories of road users: trams *y1(t)*, pedestrians *y2(t)* and general traffic *y3(t)*. The initial state of the signals of the traffic signals system is determined by the parameter *g(t)*. In the general case, the control input *s(t)* to the traffic signals system is a set of SG identifiers $\bigcup s_i$, which are combined into the stage S_j. The control law actively takes into account the characteristics of traffic and tram(s) for each s_i. The characteristics of the car flow are: n^{in} is the number of vehicles that have passed through the entry detector; n^{out} is the number of vehicles that passed through the exit detector at the stop line; *dN(0)* is initial time distance between vehicles on the entry detector; *dT(0)* is initial time distance between vehicles on the stop line detector. The characteristics of the tram *m* are: *b* is time elapsed from detection by entry detector until detection by exit detector located after the intersection; *p* is passenger occupancy; $\delta(0)$ is cumulative deviation from the schedule at the time of detection by the entry detector; *v* is current speed. The application of traffic signals control affects these characteristics, which are then analyzed by fuzzy rules of the control law. The control law provides for passive consideration of accidental disturbance in the form of a pedestrian, which briefly affects the dynamics of a tram.

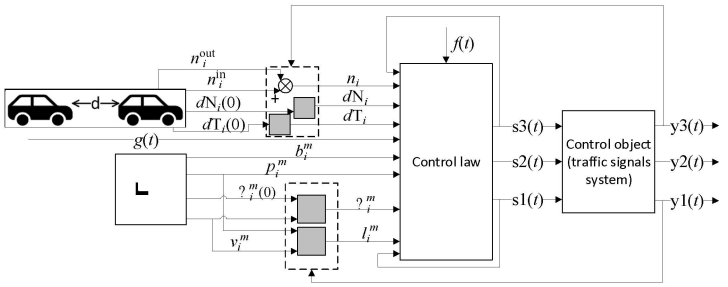

Fig. 2. Diagram of the adaptive control system of intersection traffic signals with tram priority

The only characteristic that determines the order of SG activation is their weights or importances w, which are determined at the second level of the model [29] (Fig. 3). For SG, which control the general traffic, the weight $w(s3_i)$ is determined depending on the number of vehicles between the detectors [29], herewith $\exists w(s1_i) > 0 \,|\, \forall w(s3_i) \triangleq 0.25 \times w(s3_i)$. To determine the weight of tram

SG, in this paper we propose the rules of fuzzy inference (Table 3), which take into account the passenger occupancy and schedule adherence (Fig. 4). Stages S_j are determined using the conflict matrix CM:

$$s_i \in S_j : S_{a=k} \cap \left(\bigcup_{b \neq k} S_b \right) = \varnothing; \forall s_x, s_y \in S_j, CM[x,y] = 0 . \quad (5)$$

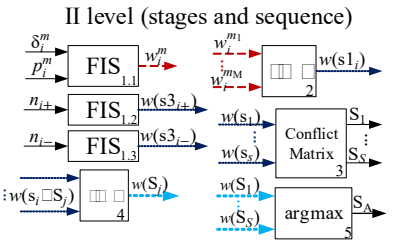

Fig. 3. Diagram of the second level of the model

Table 3

d_shed	*n_pass*			
	zero	a few	medium	many
ahead	zero	low	medium	high
on time	zero	medium	high	highest
behind	medium	high	high	highest

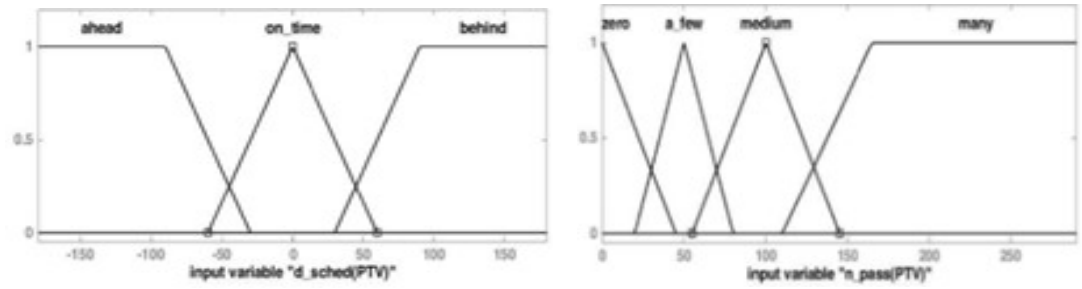

Fig. 4. Membership functions of fuzzy variables *d_shed* and *n_pass*

Next stage $S_A = agr \max_j \left(\left\| \sum_{s_i \in S_j} w[s_i] \right\| \right)$.

The first level of the model (Fig. 5) determines the SG time parameters for the composed phases. The rules for determining the action on a green SG $s3_{i+}$ are defined as in the FUSICO project [29]. For the problem of tram priority, the control rules for several stages [23] have been extended to take into account the tram SG importance $w(s1_i)$ (Fig. 6). The actions of GE (green extension), GR (green recall) are further analyzed fuzzy rules (Fig. 7) taking into account the tram dynamics, where the confidence degree in the action is determined, similar to the rule base for SG $s3_{i+}$.

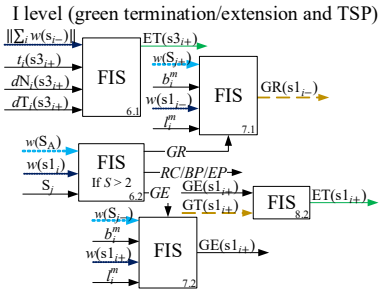

Fig. 5. Diagram of the first level of the model

Variable *l(PTV)* is a relative value of the time the tram needs to reach the intersection, which depends only on the dynamics of the tram and is invariant to the time since detection:

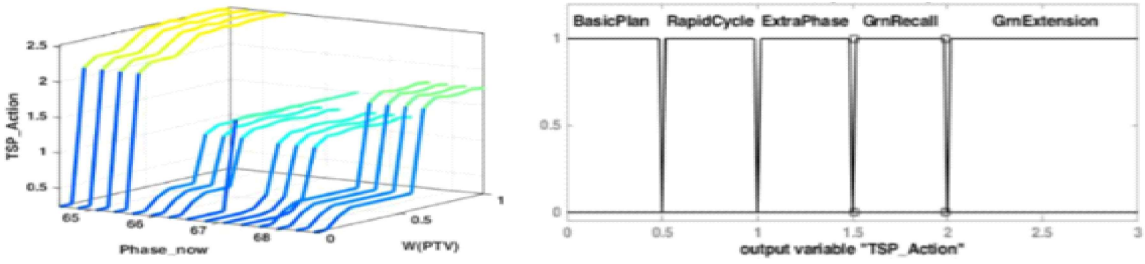

Fig. 6. Reaction function for several stages and membership function of output crisp variable

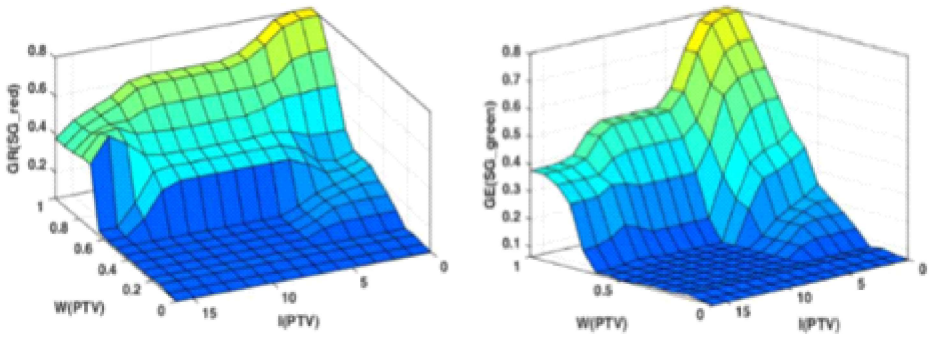

Fig. 7. Reaction functions to determine action on SG $s1_{i-}$ та $s1_{i+}$

$$l(PTV) = l_i^m - \frac{s_{\text{det}}}{v_{\max}} + b_i^m, \tag{6}$$

where l_i^m is time until the tram reaches the intersection, obtained by modeling its linear dynamics; s_{det} is distance between the entry point of the detector and the boundary of the intersection (100 m); $v_{\max}$ is the maximum speed allowed before the intersection.

Table 4

$GE(s1_{i+})$	$GT(s1_{i+})$				
	Wait	RWait	NSure	RTer	Term
Wait	Term	Term	Term	Term	Term
RathWait	RTer	RTer	RTer	Term	Term
NotSure	NSure	NSure	NSure	RTer	Term
RatherExt	RExt	RExt	RExt	RExt	RExt
Extend	Extnd	Extnd	Extnd	Extnd	Extnd

To resolve potential conflicts between priority requests from the inhibiting SG $s1_{i-}$ during active $s1_{i+}$ relevant rules have been added (Table 4). Also, for the correct service order of priority requests, during the composition of stages, in the SG $s1_{i-}$, allowing movement from the common lane, the value b_i^m is analyzed.

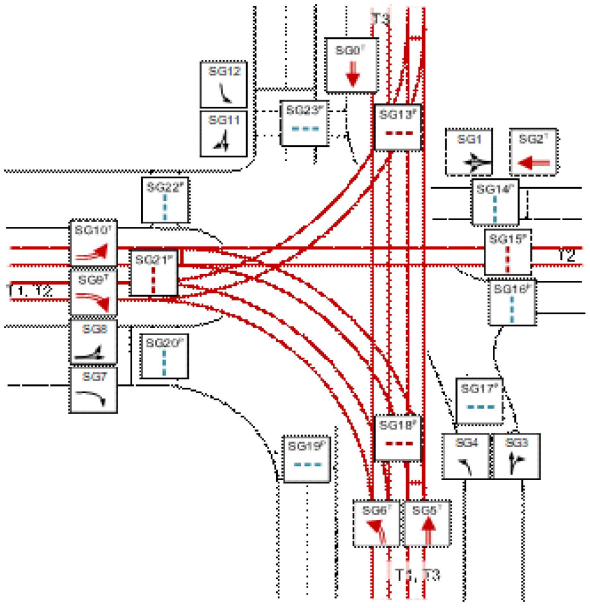

Fig. 8. Layout of the created intersection

5. Analysis of modeling results

The modeling was performed for an artificial intersection with tram lines branching and intersections of adjacent directions for trams and general traffic, which complicates the task of optimal adaptive control with a fixed stages approach (Fig. 8).

SUMO tool was utilized for modeling [30], in which reference control methods for in-tersection traffic signals system are available: gap-based or vehicle-actuated (VA) and delay-based (DB). These methods are based on a fixed predefined sequence of stages and adjust the duration of the permissive signal depending on the time interval between consecutive vehicles for the VA method and the delay of detected vehicles depending on their speed in the range of lane detectors for the DB method.

The study of the applicability of the developed method fuzPrioPro for traffic

control is performed for non-peak and peak conditions of transport demand. Taking into account the geometric features of the intersection, the following demand parameters are set for general traffic in non-peak conditions (Table 5).

Table 5

Flow origin	Intensity, veh/h	Flow direction probability, %			
		N	E	S	W
North	103	—	11	53	36
East	80	7	—	41	52
South	240	8	17	—	75
West	180	18	10	72	—

For peak conditions, the value of the traffic intensity increases 1.5 times. The tram is characterized by deterministic demand and has a defined schedule with intervals. Expected passenger traffic is determined in passengers per hour per direction, p/h/d. For non-peak and peak conditions, the parameters of passenger flow for the specified tram lines are given in Tables 6 and 7 respectively.

Table 6

Line	Origin	Occupancy, pas/tram	Avg trams per tram set	Intensity, sets/h
T1	West	77	1.3	13
	South	78	1.4	15
T2	West	40	1.0	7
	East	65	1.0	9
T3	North	73	1.1	10
	South	80	1.2	11

The measured characteristics of vehicles traffic flows that accumulate in front of traffic signals are their number. For trams of each direction of each line, the analyzed metrics are time loss and number of halts. The modeling is performed for 3600 s, with a step of 3 s for the control method. The period of averaging of data on the general traffic is 60 s, for trams data the period is 30 s. Passenger occupancy of tram sets is determined by the normal distribution with a standard deviation $\sigma = 20$ and the mean value of μ according to defined parameters (Table 6, 7).

Table 7

Line	Origin	Occupancy, pas/tram	Avg trams per tram set	Intensity, sets/h
T1	West	100	2.0	16
	South	115	2.0	18
T2	West	90	1.0	10
	East	100	1.0	10
T3	North	93	1.5	13
	South	94	1.7	14

Figure 9 shows the delays of trams of lines T1 and T3 (SG 5, 6), as well as the number of complete halts within the detectors range in off-peak conditions. Compared to the DB and VA methods, the proposed method with absolute tram priority significantly reduces the mean delay of both the tram vehicles and all their passengers. In addition, there is a lower standard deviation of tram delays on both lines, as well as a lower mean value of the number of halts when using a fuzzy control method.

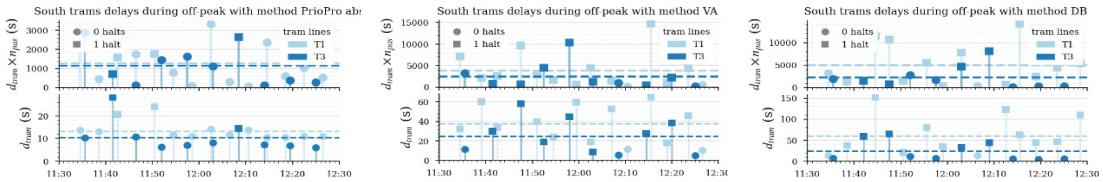

Fig. 9. Comparison of time characteristics of trams from the south in non-peak conditions

A comparable situation is observed for trams of lines T1 and T2 (SG 9, 10) (Fig. 10). It can be noted that the average delay of T2 line's passengers is lower despite the higher mean delay of the

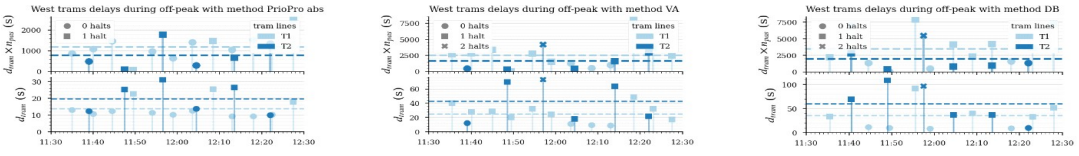

Fig. 10. Comparison of time characteristics of trams from the west in non-peak conditions

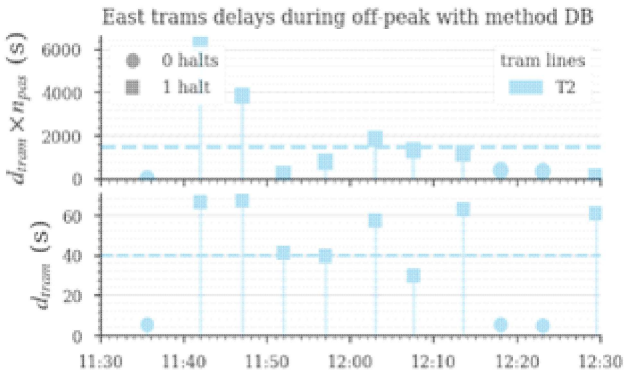

Fig. 11. The drawback of the fixed stages approach for complex intersections

trams themselves, which is due to its lower passenger flow. For the PrioPro method there are no cases with a double halt at the intersection. For other methods, this occurs approximately at the same time for the tram line T2, for which there is no possibility of conflict-free crossing with the approach of a fixed sequence of stages (Fig. 8) (conflict with trams of the T2 line in counter direction).

Indeed, analyzing the delays of trams from the east, one can see the passage of the tram, which is likely to be the cause of several halts (Fig. 11). The following Table 8 generalizes characteristics of tram traffic in non-peak conditions.

Table 8

line	TramTimeLoss		PasTimeLoss		HaltsPerTram		Speed		method
	mean	std	mean	std	mean	std	mean	std	
T1	48.79	37.0	4291.4	3790.16	0.77	0.43	12.62	6.57	DB
T2	47.18	30.86	1668.47	1875.29	0.82	0.53	13.51	8.82	DB
T3	22.77	23.1	2108.37	2446.82	0.42	0.51	21.14	9.04	DB
T1	32.06	17.43	3212.27	3359.43	0.81	0.4	14.48	5.36	VA
T2	36.03	22.61	1495.24	1484.26	0.88	0.49	14.48	7.11	VA
T3	27.54	17.86	2730.65	3073.45	0.74	0.45	17.3	7.88	VA
T1	13.54	4.6	1232.67	868.34	0.19	0.4	21.21	2.83	PrioPro
T2	12.87	7.95	486.03	478.16	0.24	0.44	22.85	5.49	PrioPro

One can see consistently lower time delays for all tram lines under PrioPro control. Compared to the DB method, the expected delay of tram vehicles was reduced by 67%, while against the VA method reduction is by 62%. If we compare the delay of passengers, the biggest gain is observed for the T1 line and reaches 71% compared to the DB method. The VA method has less difference, for all lines it is about 65%. In terms of the number of halts, their expected value is consistently lower for the PrioPro method. The mean speed of the passage and regularity also increased.

Analyzing the characteristics of the general traffic flow, it can be stated that the pro-posed method does not have a significant negative impact on traffic in non-peak condi-tions. It is seen that the VA method leaves the stationary mode in the middle of modeling, yet, in the end, the queue begins to discharge (Fig. 12).

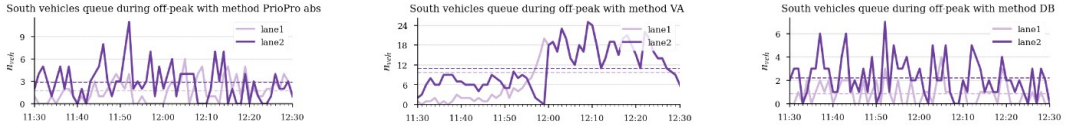

Fig. 12. Comparison of time characteristics of general traffic from the south under non-peak conditions

Table 9

origin	lane	nVehWithin		method	nVehWithin		method	nVehWithin		method
		mean	std		mean	std		mean	std	
S	1	0.85	0.97	DA	9.73	7.9	VA	1.75	1.57	PrioPro
S	2	2.23	1.69	DA	10.84	6.6	VA	2.87	2.35	PrioPro
W	1	0.76	1.0	DA	0.62	0.75	VA	0.52	0.85	PrioPro
W	2	0.65	0.89	DA	0.72	0.84	VA	0.52	0.81	PrioPro

The following Table 9 is a summary of the queue length on the busiest approaches to the intersection in non-peak conditions. The

PrioPro method is slightly inferior to the DB method in throughput for the southern approach but is best for the traffic flow from the west.

Compared to off-peak conditions, the delay of trams from the south varies even less, especially for the T3 line. However, under fuzzy control there is a case of double tram halt (Fig. 13).

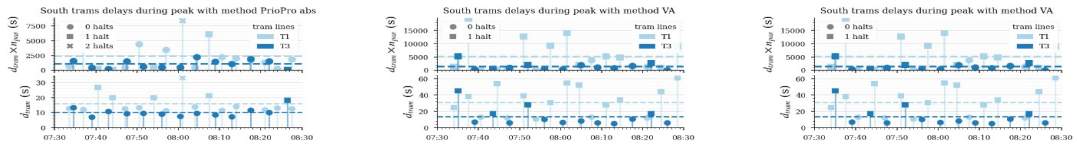

Fig. 13. Comparison of time characteristics of trams from the south under peak conditions

Table 10 shows the generalized results of modeling the trams flow under saturated conditions of traffic. Compared with the DB method, the average reduction in both types of delays is 80%, for the VA method is less: 57%. The number of halts has significantly decreased, and reduction is observed even in comparison with non-peak conditions. The expected value of the passage speed has also increased.

Table 10

line	TramTimeLoss		PasTimeLoss		HaltsPerTram		Speed		method
	mean	std	mean	std	mean	std	mean	std	
T1	73.68	50.3	14425.16	16986.18	0.75	0.44	10.85	7.51	DB
T2	82.18	46.87	10913.72	11201.96	1.14	0.36	7.91	4.29	DB
T3	41.98	35.68	4254.85	6065.89	0.64	0.49	16.05	8.82	DB
T1	34.85	18.36	6761.53	7769.66	0.72	0.46	14.55	6.0	VA
T2	34.45	21.07	4363.26	5478.01	0.67	0.48	15.05	7.66	VA
T3	18.34	16.82	2032.48	2863.9	0.48	0.51	21.92	6.98	VA
T1	15.26	5.79	2885.98	3274.74	0.28	0.58	20.8	2.89	PrioPro
T2	10.97	4.2	1338.95	1244.44	0.14	0.36	23.43	3.68	PrioPro
T3	10.22	3.39	1095.28	956.78	0.08	0.28	25.04	2.5	PrioPro

As for the general traffic, under saturated conditions, all control methods at some mo-ment go into non-stationary mode, but stabilize relatively quickly, except for the VA method (Fig. 14).

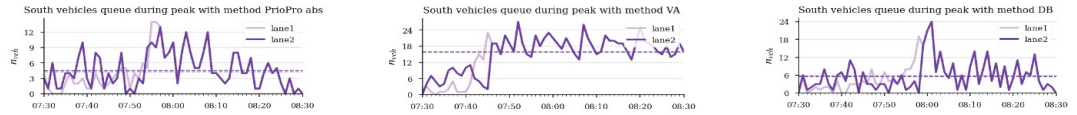

Fig. 14. Comparison of time characteristics of general traffic from the south under peak conditions

Analyzing modeling results for general traffic under peak conditions (Table 11), one can see that the proposed PrioPro method provides stable control for the busiest traffic directions.

Table 11

origin	lane	nVehWithin		method	nVehWithin		method	nVehWithin		method
		mean	std		mean	std		mean	std	
S	1	5.79	5.37	DA	15.78	7.04	VA	4.21	3.61	PrioPro
S	2	5.56	5.11	DA	15.88	6.45	VA	4.49	3.47	PrioPro
W	1	1.66	1.72	DA	1.0	1.48	VA	1.08	1.69	PrioPro
W	2	1.79	1.94	DA	2.1	2.46	VA	3.0	2.55	PrioPro

6. Conclusions

The proposed mathematical model of acyclic traffic signal control system with uncon-ditional priority for the tram as a category RoW B public transportation, considering the dynamics of its movement as it approaches the intersection, significantly reduces passen-ger delay, and allows to avoid complete tram halt. At the same time, there is a slight deterioration in general traffic flow, which is expected with the application of absolute transit signal priority.

A further direction of research may be to expand the rules for dealing with the case of interruption of an active signal group for general traffic by priority request from waiting tram for more smooth traffic flows control.

References: 1. *American* Public Transportation Association. Economic Impact of Public Transportation Investment 2020 UP-DATE. APTA: American Public Transportation Association, 2020. 2. *Litman T.* Evaluating Public Transit Benefits and Costs. Victoria Transport Policy Institute Victoria, BC, Canada, 2021. 3. *Vuchic V.R.* Urban transit systems and tech-nology. John Wiley & Sons, 2007. 4. *Bruun E., Allen D., Givoni M.* Choosing the right public transport solution based on performance of components: 4 / Transport. 2018. Vol. 33. № 4. P. 1017-1029. 5. *M. Lytvynenko, O. Shkil, I. Filippenko, L. Rebezyuk.* Model and Means of Timed Automata-based Real-time Adaptive Transit Signal Control. IEEE East-West Design & Test Symposium (EWDTS). 2020. P. 1-4. 6. *Transportation* Research Board, National Academies of Sciences, Engineering, and Medicine. Transit Signal Priority: Current State of the Practice. Washington, DC: The National Academies Press, 2020. 7. *Smith H.R., Hemily B., Ivanovic M.* Transit Signal Priority (TSP): A Planning and Implementation Handbook. Washington, DC, USA: ITS America. 2005. 212 p. 8. *American* Association of State Highway and Transportation Officials (AASHTO), Institute of Transportation Engineers (ITE), National Electrical Manufacturers Association (NEMA). NTCIP 1211 Object Definitions for Signal Control and Prioritization (SCP). 2014. 9. *Bhouri N.,Mayorano F.,Lotito P.,Haj Salem H., Lebacque J.* Public Transport Priority for Multimodal Urban Traffic Control / Cybernetics and Information Technologies. 2015. Vol. 15. № 5. P. 17-36. 10. *Jiang P.P., Poschinger A., Qi T.Y.* MOTION - A Developing Urban Adaptive Traffic Signal Control System . Advanced Materials Research. 2013. Vol. 779. P. 788-791. 11. *Oliveira-Neto F.M., Loureiro C.F.G., Han L.D.* Active and passive bus priority strat-egies in mixed traffic arterials controlled by SCOOT adaptive signal system: Assessment of performance in Fortaleza, Brazil. Transportation research record. 2009. Vol. 2128. № 1. 12. *Farges J.-L., Henry J.-J.* PT priority and PRODYN. Artech House, 1994. P. 3086-3093. 13. *Mirchandani, P., Knyazyan, A., Head, L., Wu, W.* An Approach Towards the Integration of Bus Priority, Traffic Adaptive Signal Control, and Bus Infor-mation/Scheduling Systems. Lecture Notes in Economics and Mathematical Systems. 2001. Vol. 505. P. 319-334. 14. *Niittymaki J., Maenpaa M.* The Role of Fuzzy Logic Public Transport Priority in Traffic Signal Control / Traffic Engineering+ Control. 2001. Vol. 42. № 1. P. 22-26. 15. *Gartner N.H., Tarnoff P.J., Andrews C.M.* Evaluation of optimized policies for adaptive control strategy. Transportation research record. 199. № 1324. P. 105-114. 16. *Kaparias, I., K. Zavitsas, M.G.H. Bell, M. Tomassini* State-of-the-art of urban traffic management policies and technologies. ISIS. 2010. Vol. 13. P. 08. 17. *A Definition* of Soft Computing - adapted from L.A. Zadeh [Electronic resource]. URL: http://www.soft-computing.de/def.html. 18. *Greguric, M., Vujic, M., Alexopoulos, C. and Miletic, M.* Application of Deep Reinforcement Learning in Traffic Signal Control: An Overview and Impact of Open Traffic Data: 11. Applied Sciences. 2020. Vol. 10. № 11. P. 4011. 19. *Jin J., Ma X.* A group-based traffic signal control with adap-tive learning ability. Engineering applications of artificial intelligence. 2017. Vol. 65. P. 282-293. 20. *Odeh, S.M., Mora, A.M., Moreno, M.N., Merelo, J.J.* A Hybrid Fuzzy Genetic Algorithm for an Adaptive Traffic Signal System . Ad-vances in Fuzzy Systems. Hindawi. 2015. Vol. 2015. P. e378156. 21. *Ghanim M.S., Abu-Lebdeh G.* Real-Time Dynamic Transit Signal Priority Optimization for Coordinated Traffic Networks Using Genetic Algorithms and Artificial Neural Networks / Journal of Intelligent Transportation Systems. Taylor & Francis, 2015. Vol. 19, № 4. P. 327-338. 22. *Kuang X., Xu L.* Real-Time Traffic Signal Intelligent Control with Transit-Priority / Journal of Software. 2012. Vol. 7. № 8. P. 1738-1743. 23. *Niittymaki J.* Fuzzy Logic Application to Public Transport Priorities at Signalized Intersections. Mathematics in Transport Planning and Control. Emerald Group Publishing Limited. 1998. P. 47-57. 24. *Do L., Herman I., Hurak Z.* Onboard Model-based Prediction of Tram Braking Distance. IFAC-PapersOnLine. Elsevier. 2020. Vol. 53. № 2. P. 15047-15052. 25. *CKD* Tatra T3 imhd.sk Bratislava [Electronic resource]. URL: https://imhd.sk/ba/popis-typu-vozidla/20/%C4%8CKD-Tatra-T3. 26. *Takaoka Y., Kawamura A.* Disturbance observer based adhesion control for Shinkansen. Nagoya, Japan. IEEE. 2000. P. 169-174. 27. *Hay W.W.* Propulsive Resistance / Railroad engineering. 2nd ed. Wiley. 1982. P. 69-89. 28. *Heydecker B.G.* A decomposition approach for signal opti-misation in road networks. Transportation Research Part B: Methodological. Pergamon. 1996. Vol. 30. № 2. P. 99-114. 29. *Niittymaki J., Pursula M.* Signal control using fuzzy logic / Fuzzy sets and systems. North-Holland. 2000. Vol. 116. № 1. P. 11-22. 30. *Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flotterod, Y.P., Hilbrich, R., Lucken, L., Rummel, J., Wagner, P. and Wiesner, E.* Microscopic Traffic Simulation using SUMO. 2018 21st International Conference on Intelligent Transportation Systems (ITSC). 2018. P. 2575-2582.

Надійшла до редколегії 08.06.2022

Lytvynenko Mykhailo, Master of Automation and Computer-integrated Technologies (System Engineering Department) KhNURE. Address: Nauky Ave., 14, Kharkiv, Ukraine, 61166; tel. +38(0 98)915 14 95; email: mykhailo.lytvynenko1@nure.ua

Rebezyuk Leonid, Associate Professor of System Engineering Department KhNURE, Candidate of Technical Sciences, Associate Professor. Address: Nauky Ave., 14, Kharkiv, Ukraine, 61166; tel. +38(057)702 10 06, +38(068)9613676, email: leonid.rebezyuk@nure.ua.

УДК 004.8:004.032.26 DOI: 10.30837/0135-1710.2022.178.033

А.Ю. ШАФРОНЕНКО, Є.В. БОДЯНСЬКИЙ

АДАПТИВНА КЛАСТЕРИЗАЦІЯ БАГАТОЕКСТРЕМАЛЬНИХ МАСИВІВ ДАНИХ З ВИКОРИСТАННЯМ МОДИФІКОВАНОГО АЛГОРИТМУ РИБ'ЯЧОЇ ЗГРАЇ

Розглянуто задачу кластеризації багатоекстремальних масивів даних. Для оптимізації функцій пошуку локальних екстремумів запропоновано алгоритм, що є по суті оптимізаційною функцією модифікованого алгоритму риб'ячої зграї, випадкового пошуку та еволюційної оптимізації. Цей алгоритм пришвидшує пошук глобальних екстремумів, не потребує додаткових обчислень, дозволяє скоротити кількість запусків процедури оптимізації, знаходити екстремуми функцій складної форми у випадку, коли класи перетинаються, та є простим у числовій реалізації.

1. Вступ

Задача кластеризації масивів спостережень довільної природи є невід'ємною частиною Data Mining, а у більш загальному випадку - Data Science. З обчислювальної точки зору, найпростішими є так звані ієрархічні методи та алгоритми, засновані на розбиттях [3], серед яких слід відзначити процедуру *k*-середніх, що набула дуже широкого розповсюдження для вирішення найрізноманітніших задач. Слід відзначити, що найбільш адекватним математичним апаратом для вирішення задач кластеризації є методи обчислювального інтелекту [5-7] і, перш за все, штучні нейронні мережі, нечіткі системи, еволюційна оптимізація та так звані гібридні системи обчислювального інтелекту, що об'єднують ці три напрями.

В загальному випадку вирішення задачі кластеризації суттєво ускладнене, якщо вихідні вектори-спостереження викривлені збуреннями, містять пропуски, самі вихідні масиви або занадто великі, або занадто короткі, кластери можуть мати досить складну форму, а їх кількість апріорі невідома. У цьому випадку найефективнішими є алгоритми, що базуються на аналізі щільностей розподілу даних [9-12], які були запропоновані для вирішення задач кластеризації великих масивів векторних даних високої розмірності. В основі цих алгоритмів лежить пошук екстремумів-максимумів функції щільності розподілу даних в масиві, що аналізується, при цьому ця функція формується як суперпозиція ядерних функцій, пов'язаних з кожним спостереженням. Фактично ця функція будується на основі вікон Парзена [13] та оцінок Надарая-Ватсона [14,15].

З обчислювальної точки зору, задача кластеризації перетворюється на проблему пошуку локальних екстремумів багатоекстремальної функції векторного аргументу щільності за допомогою градієнтних процедур, які багаторазово запускаються з різних точок вихідного масиву даних. Зрозуміло, що це займає досить багато часу, оскільки апріорі невідомо, скільки ж екстремумів має сформована функція щільності.

Пришвидшити процес пошуку цих екстремумів можна, скориставшись ідеями еволюційної оптимізації, що включає в себе алгоритми, інспіровані природою, ройові алгоритми, популяційні алгоритми тощо [16-18]. При цьому пошук ведеться одночасно групою агентів, які діють або незалежно, або у взаємодії, що дозволяє суттєво пришвидшити процес пошуку екстремумів, кожен з яких "відповідає" тому або іншому кластеру, що формується.

2. Формування функції щільності розподілу даних у масиві, що підлягає кластеризації

Вихідною інформацією для вирішення задачі кластеризації традиційно є масив векторів-спостережень $X = \{x(1), x(2), ..., x(k), ..., x(N)\}$, $x(k) = \{x_i(k)\} \in R^n$, при цьому дані попередньо відцентровано на гіперкуб так, що $x(k) = \{x_{i,i_2}(k)\} \in R^{n_1 \times n_2}$. Така ситуація може виникати у випадку обробки масивів зображень. За таких умов одними з найбільш ефективних є алгоритми, що базуються на аналізі щільностей розподілу даних у вихідних масивах, серед яких можна відзначити DENCLUE та його модифікації.

DENCLUE (DENsity-based CLUstEring) розглядається як окремий випадок оцінки ядерної щільності - це техніка непараметричного оцінювання, спрямована на пошук точок щільних областей. Основними поняттями, на яких базується DENCLUE, є функція впливу, функція щільності та атрактори щільності, що по суті є локальними екстремумами функції щільності. В загальному випадку функція впливу для будь-якого векторного спостереження $x(\bullet)$ з вихідного масиву X є ядерною дзвонуватою функцією $f^{x(\bullet)}(x)$, при цьому найпопулярнішою є традиційна гаусівська функція

$$f_G^{x(\bullet)}(x) = \exp\left(-\frac{d^2(x, x(\bullet))}{2\sigma^2}\right) = \exp\left(-\frac{\|x - x(\bullet)\|^2}{2\sigma^2}\right), \quad (1)$$

(тут $d^2(x, x(\bullet))$ - евклідова відстань; σ^2 - параметр ширини функції впливу) завдяки простоті обчислення її похідних.

У матричному випадку замість евклідової відстані можна використати метрику Флобеніуса, при цьому функція впливу набуває вигляду

$$f_G^{x(\bullet)}(x) = \exp\left(-\frac{d^2(x, x(\bullet))}{2\sigma^2}\right) = \exp\left(-\frac{Tr\left(x - x(\bullet)\right)\left(x - x(\bullet)\right)^T}{2\sigma^2}\right), \quad (2)$$

де $Tr(\bullet)$ - символ сліду матриці.

На основі функцій впливу формується функція щільності розподілу даних у масиві X у вигляді

$$f^x(x) = \sum_{k=1}^{N} f\left(x, x(k)\right). \quad (3)$$

Власне процес формування кластерів пов'язаний з відшуканням усіх екстремумів функції щільності (3) за допомогою градієнтної процедури

$$x^l = x^{l-1} + \eta^l \frac{\nabla f^x\left(x^l, x^{l-1}\right)}{\left\|\nabla f^x\left(x^l, x^{l-1}\right)\right\|}, \quad x_0 = x(k), l = 0,1,2,...; \forall k = 1,2,...,N, \quad (4)$$

тобто кількість запусків алгоритму (4) визначається обсягом навчальної вибірки N. Зрозуміло, що при великих N процес кластеризації - пошуку локальних екстремумів - може потребувати дуже багато часу. Тому запропоновані модифікації DENCLUE пов'язані з пришвидшенням процесу пошуку локальних екстремумів (3) шляхом модифікації градієнтної процедури (4) [10-12].

Пришвидшити процес відшукання локальних екстремумів можна, використовуючи замість градієнтного пошуку методи еволюційної оптимізації, серед яких як достатньо ефективний, чисельно простий і швидкий можна відзначити так званий пошук на основі косяків риб [19-21], що повинен бути модифікований для вирішення задачі кластеризації.

3. Модифікований метод оптимізації на основі косяків риб

При використанні методів еволюційної оптимізації, що по суті є методами оптимізації нульового порядку, припускається, що при відшуканні екстремумів деякої функції $f^x(x)$ застосовується популяція агентів, кожен з яких діє або самостійно, або взаємодіючи з іншими, при цьому рух кожного q-го агента $(q = 1,2,...,Q)$ на l-й ітерації пошуку може бути записаний за допомогою співвідношення

$$x_q^l = x_q^{l-1} + \eta_q^l Dir_q^l, \quad q = 1,2,...,Q,$$

де $x_q^l = \left(x_{q1}^l, x_{q2}^l, ..., x_{qn}^l\right)^T$, Dir_q^l - вектор, що задає напрямок руху q-го агента на l-й ітерації пошуку.

У великій родині таких методів слід відзначити метод на основі косяків риб, де кожен агент популяції імітує рух окремої риби [19-21]. Основною перевагою цього методу є

достатня ефективність відшукання глобального екстремуму досить складних функцій, до яких можна віднести і функцію щільності розподілу даних в задачах кластеризації.

Автори методу вводять у розгляд ітерації, пов'язані з рухом косяка: годування та плавання.

Оператор годування відповідає за вагу кожної риби як елемента косяка - агента. Чим важче риба, тим ближче вона до екстремума-максимума. Вага кожної риби w_q налаштовується згідно із виразом

$$w_q^l = w_q^{l-1} + \frac{f^x\left(x_q^l\right) - f^x\left(x_q^{l-1}\right)}{\max\limits_p \left\{ f^x\left(x_q^l\right) - f^x\left(x_q^{l-1}\right) \right\}} \quad \forall q = 1,2,...,Q, \tag{5}$$

при цьому

$$0 < w_q^l < w_{\max}, \;\; w_l^0 = 0,5 w_{\max}.$$

Оператор плавання описує як індивідуальний рух кожної риби, так і колективний рух косяка в цілому. Тут розглядаються три типи руху: індивідуальний, інстинктивно-колективний та колективно-рольовий. Індивідуальний рух описується співвідношенням

$$x_{qi}^l = \begin{cases} x_{qi}^l + \eta_q^l Rand\left\{0,1\right\}, \text{ if } f^x\left(x_q^l\right) > f^x\left(x_q^{l-1}\right), \\ x_q^{l-1} \;\; \text{else}, \end{cases} \tag{6}$$

де $Rand\left\{0,1\right\}$ - рівномірно розподілене у інтервалі $(0,1)$ випадкове число. Фактично це процедура "зондування" функції $f^x(x)$ в околі точки x_q^{l-1}, при цьому крім (6) тут може бути застосований будь-який інший алгоритм випадкового пошуку.

На базі зондування функції щільності за допомогою індивідуального руху (6) реалізується інстинктивно-колективний рух у напряму зростання цієї функції

$$x_q^l = x_q^{l-1} + \frac{\left(\sum\limits_{p=1}^{Q} \left(x_p^l - x_p^{l-1} \right) \right) \left(f^x\left(x_q^l\right) - f^x\left(x_q^{l-1}\right) \right)}{\sum\limits_{p=1}^{Q} \left(f^x\left(x_p^l\right) - f^x\left(x_p^{l-1}\right) \right)}. \tag{7}$$

Вводячи у розгляд зважений центр ваги косяка риб

$$Bar^l = \frac{\sum\limits_{p=1}^{Q} x_p^l w_p^l}{\sum\limits_{p=1}^{Q} w_p^l}, \tag{8}$$

можна записати цей рух у вигляді

$$x_q^l = \begin{cases} x_q^l - \eta_q^l Rand\left\{0,1\right\} \dfrac{x_q^{l-1} - Bar^{l-1}}{\left\| x_q^{l-1} - Bar^{l-1} \right\|}, \text{ if } \sum\limits_{p=1}^{Q} w_p^l > \sum\limits_{p=1}^{Q} w_p^{l-1}, \\ x_q^l + \eta_q^l Rand\left\{0,1\right\} \dfrac{x_q^{l-1} - Bar^{l-1}}{\left\| x_q^{l-1} - Bar^{l-1} \right\|}, \text{ if } \sum\limits_{p=1}^{Q} w_p^l < \sum\limits_{p=1}^{Q} w_p^{l-1}. \end{cases} \tag{9}$$

Для підвищення ефективності FSS у розгляд може бути введений додатковий оператор розведення, що дозволяє створювати нових риб-агентів, які мають покращені характеристики у порівнянні зі вже існуючими членами косяка. Для цього можна скористатися ідеями еволюційних операцій [23], серед яких з обчислювальної точки зору та з точки зору ефективності - надійності відшукання екстремуму можна відзначити послідовний симплекс-метод [24] та його модифікації [25-26].

Сформуємо косяк, що містить $Q = n+1$ риб-агентів, при цьому ця кількість залишається незмінною у процесі пошуку, тобто популяція $x_1^0, x_2^0, ..., x_Q^0$ генерується випадковим чином. В цій популяції знайдемо "найгіршу" рибу x_{qworst}^0, що має найменшу вагу $w_{q\min}^0$, та "найкращу" рибу x_{qbest}^0 з найбільшою вагою $w_{q\max}^0$. Основна операція руху симплекса полягає у відображенні x_{qworst}^0 через центр ваги n риб (без найгіршої), який може бути записаний у вигляді

$$\overline{x}^0 = \frac{1}{n}\sum_{q=1}^{Q}\left(x_q^0 - x_{qworst}^0\right).$$

В результаті цієї операції створюється нова риба

$$x_q^{1*} = \overline{x}^0 + \alpha\left(\overline{x}^0 - x_{qworst}^0\right),$$

яка заміняє у косяку найгіршу особину x_{qworst}^0. Так формується нова популяція $x_1^1, x_2^1, ..., x_Q^1$.

Таким чином, рух косяка-симплекса може бути описаний за допомогою співвідношень

$$\begin{cases} \overline{x}^{l-1} = \dfrac{1}{n}\sum\limits_{q=1}^{Q}\left(x_q^{l-1} - x_{qwost}^{l-1}\right), \\ x_q^l = \overline{x}^{l-1} + \alpha\left(\overline{x}_q^{l-1} - x_{qwost}^{l-1}\right), \end{cases} \tag{10}$$

що у загальному випадку є за своєю суттю алгоритмом оптимізації Нелдера-Ліда [25]. Таким чином, з косяка у процесі пошуку екстремуму вилучаються найгірші риби з найнижчою вагою та створюються нові агенти з більшою вагою.

Оскільки задача, що розглядається, є за своєю суттю проблемою багатоекстремальної оптимізації, необхідно відшукати множину екстремумів, кожен з яких є центроїдом деякого кластера. При знаходженні якогось з екстремумів з вихідної вибірки X виключаються спостереження, що розташовані безпосередньо в його околі. Після цього вилучення запропонована процедура комбінованої еволюційної оптимізації повторюється до відшукання всіх екстремумів - центроїдів.

4. Висновки

Розглянуто задачу кластеризації багатоекстремальних масивів даних. Для оптимізації функцій пошуку локальних екстремумів запропоновано алгоритм, що є по суті оптимізаційною функцією модифікованого алгоритму риб'ячої зграї, випадкового пошуку та еволюційної оптимізації. Цей алгоритм пришвидшує пошук глобальних екстремумів, не потребує додаткових обчислень, дозволяє скоротити кількість запусків процедури оптимізації, знаходити екстремуми функцій складної форми у випадку коли класи перетинаються, та є простим у числовій реалізації. Запропонований підхід дозволяє скоротити кількість запусків процедури оптимізації, дозволяє знаходити екстремуми функцій складної форми та є простим в чисельній реалізації.

Список літератури: 1. *Gan G., Ma Ch., Wu J.* Data Clustering: Theory, Algorithms and Applications. Philadiphia, Pensilvania: SIAM: 2007. 455 p. 2. *Abonyi J., Feil D.* Cluster Analisis for Data Mining and System Identification. Basel: Birlhause. 2007. 303 p. 3. *Xu R., Wunsch D.C.* II - Clustering. - Hoboken, N.J.: John Wiley Sons, Inc., 2009. 341p. 4. *Aggarwal, C.C.* Data Mining: Text Book. Springer. 2015. 5. *Engelbrecht A.P.* Computational Intelligence an Introducion. John Willey& Sons, 2007. 597 p. 6. *Rukowski L.* Computational Intelligence Methods and Techniques. Berlin Heidelberg: Springer - Verlag. 2008. 514 p. 7. *Kroll A.* Computational Intelligence. Eine Einfurming in Problelme, Methoden and Tchnische Anwen-dungen. Munchen: Oldenbourg Verlag. 2013. 428.p. 8. *Kohonen T.* Self-Organizing Maps/ Kohonen T. Berlin: Springer, 1995. 362 p. DOI: 10.1007/978-3-642-56927-2. 9. *Hinneburg A., Klim D.A.* An efficient approach to clustering in large multimedia databases with noise. Proc. 4th Int. Conf. in Knowkedge Discovery and Data Mining (KDD 98). N.Y.: AAAI Press. 1998. P. 58-65. 10. *Hinneburg A., Gabriel H.-H.* DENCLUE 2.0: Fast clustering based on kernel density estimation. 11. *Hinneburg A., Klim D.A.* A general approach to clustering in large databases with noise - kniwledge and Identification Systems. 2003. 5 (5). P. 387-415. 12. *Rehhioni H.,Idrissi A., Abourezq M., Zegrary F.* DENCLUE-IM: A new approachfor big data clustering. Procedia Computer Science. 2016. 83. P. 560-567. 13. *Parzen E.* On estimation of a proobably density function and mode. The Annalis of Math Statistics. 1962. 33. №3. P. 1065-1076. 14. *Nadaraya E.A.* On nonparametric estimation of density function and regressiion curves. Theory of Probab. Appl. 1965. 10. P. 186-190. 15.

Wantson G.S. Smooth regression analysis. Sankhya: The Indian Journal of Statistics. 1964. Ser. A. 26. № 4. P. 359-372. 16. *Kennedy J., Eberhart R.* Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks. Perth, Australia, 1995. P. 1942-1948. 17. *Eiben A., Smith J.* Introduction to Evolutionary Computing. Heidelberg: Springer. 2003. 18. *Karpenko A. P.* Population algorithms for global continious optimization. Review of new and little - known algorithms. Приложение к журналу" Информационные технологии". 2012. № 7. 32 p. 19. *Bastos-Felino C.J.A., Lima Neto C.J.A., Lins A.J.C.C., Nascimento A.I.S., Lima M.P.* Fish School Search. Nature. Insperiod Algorithms for Optimization. Berlin Hedelberg: Springer Verlag. 2009. SCI 193 . P. 261-277. 20. *Cavalcanti Jr. G.M., Bastos-Felino C.J.A., Lima Neto F.B., Castro R.M.C.S.* A hybrid algorithm based on fish school search and particle swarm optimization for dynamic problems. Proc. Int. Conf. in Swarm Intelligence (ICSI). 2011. V. 2. P.543-552. 21. *Janecek A.,Tan Y.* Feeding the fish-weight update strategies for the fish school seach algorithm. Berlin Heidelberg: Springer - Verlag. Lecture Nodes in Computer Scince. 2011. V. 6729. Part II. P. 553-562. 22. *Растригін Л.А.* Випадковий пошук у процесах адаптації. Рига: Зінатне. 1973. 132 с. 23. *Box Y.E.P.* Evolutionary operation: A method for increasity industrial productivity. Applied Statistics. 1957. 6. P. 81-101. 24. *Spendley W., Hext G.R., Hinswath F.R.* Sequential application of simplex design in optimization and evolutionary operation. Tehnometrics. 1962. 4. P. 441-461. 25. *Nelder J.A., Mead R.* A simplex method for function minimization. Computer J. 1965. 7. P. 308-313.

Надійшла до редколегії 08.12.2022

Шафроненко Аліна Юріївна, кандидат технічних наук, доцент, доцент кафедри інформатики, ХНУРЕ. Наукові інтереси: нейронні мережі, нечітка кластеризація, еволюційні алгоритми, Data Mining, Big Data. Адреса: Україна, м. Харків, пр. Науки 14, тел. +38(068)8922587

Бодянський Євгеній Володимирович, доктор технічних наук, професор, професор кафедри штучного інтелекту, науковий керівник Проблемної НДЛ АСУ, ХНУРЕ. Наукові інтереси: обчислювальний інтелект, Data Mining, Big Data, Data Stream Mining. Адреса: Україна, м. Харків, пр. Науки 14.

УДК 004.65; 004.91 DOI: 10.30837/0135-1710.2022.178.037

Н.В. ВАСИЛЬЦОВА. І.Ю. ПАНФЬОРОВА

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

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

1. Вступ

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

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

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

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

2. Аналіз літературних даних і постановка проблеми дослідження

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

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

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

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

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

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

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

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

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

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

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

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

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

3. Мета і задачі дослідження

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

Для досягнення цієї мети у статті вирішуються такі задачі:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

а) у дочірньому кластері залишається лише один елемент;

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

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

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

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

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

5.1. Опис задачі аналізу конфігурації ІТ-продукту

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

Таблиця 1

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

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

Продовження таблиці 1

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

Кінець таблиці 1

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

Таблиця 2

Множина описів сутностей функціональної задачі

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

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

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

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

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

Під час виконання першої ітерації Кроку 2 було виділено пару найближчих один до одного кластерів *С2* и *С3*. Відстань між цими кластерами дорівнює 0. Вибір здійснювався під час розглядання матриці відстаней (табл. 4) зліва направо і зверху донизу. Було сформовано новий кластер *С11={С2, С3}*.

Таблиця 3

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

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

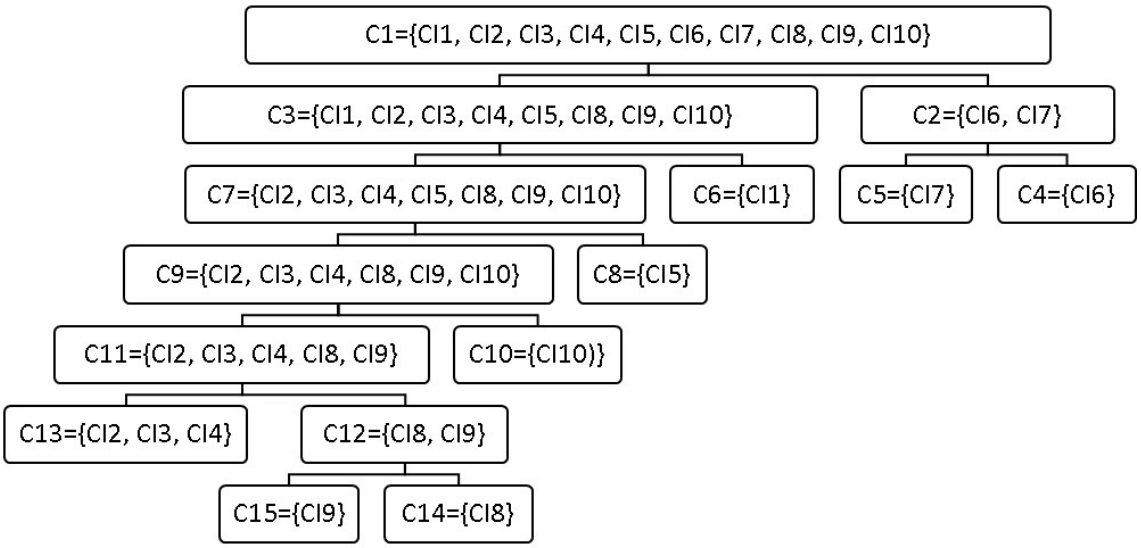

Рис. 1. Дендрограма кластерів конфігураційних елементів, сформована в результаті застосування дивізимного алгоритму

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

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

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

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

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

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

Таблиця 4

Вихідна матриця відстаней
(алгоритм найближчого сусіда)

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

Таблиця 5

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

Таблиця 6

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

Таблиця 7

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

Таблиця 8

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

Таблиця 9

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

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

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

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

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

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

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

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

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

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

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

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

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

Таблиця 10

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

Таблиця 11

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

Таблиця 12

	C18	C16
C18	0	7
C16	7	0

7. Порівняльний аналіз отриманих рішень

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

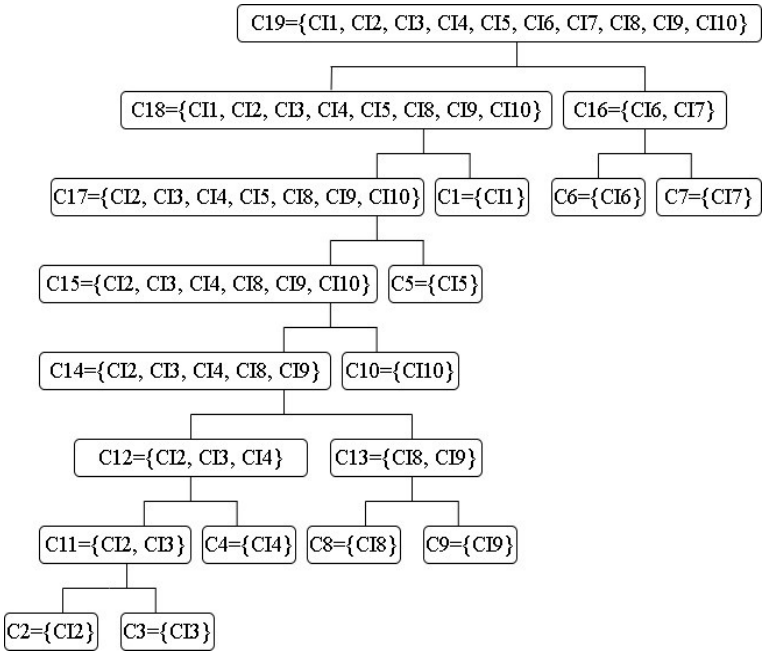

Рис. 2. Дендрограма кластерів конфігураційних елементів, сформована в результаті застосування агломеративного алгоритму найближчого сусіда

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

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

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

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

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

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

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

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

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

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

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

Крок 7. Сформувати підсумкову дендрограму та завершити роботу алгоритму.

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

8. Висновки та перспективи подальших досліджень

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

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

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

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

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

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

Надійшла до редколегії 30.11.2022

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

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

УДК 004.738 DOI: 10.30837/0135-1710.2022.178.050

Д.А. НЕФЬОДОВ, С.Г. УДОВЕНКО, Л.Е. ЧАЛА

МІКРОСЕРВІСНА АРХІТЕКТУРА СИСТЕМИ ПОТОКОВОЇ ОБРОБКИ ВЕЛИКИХ ДАНИХ

Розглянуто можливі шляхи використання мікросервісної архітектури у системах потокової обробки великих даних. Досліджено переваги і недоліки існуючих архітектур аналізу великих даних. Запропоновано варіант системи мікросервісної обробки великих даних з використанням розподіленої потокової платформи подій Kafka, платформи розробки та запуску програм Docker, об'єктно-реляційної системи управління базами даних Postgres, а також веб-платформи для створення додатків FastAPI. Розроблено архітектурні шаблони, які можуть спростити розробку застосунків для обробки великих даних з використанням мікросервісів, та концепти програм з використанням отриманих шаблонів. Наведено результати моделювання, які свідчать про те, що мікросервісна архітектура з розподіленим навантаженням на декілька реплік може забезпечити кращу масштабованість і швидкодію в порівнянні з монолітною архітектурою.

1. Вступ

Через постійно зростаючу кількість даних, які накопичуються у сучасному світі, за останні роки набула популярності концепція великих даних (ВД). Однак реалізація відповідних програм обробки ВД все ще залишається складним завданням. Застосування централізованих систем обробки та аналізу значно підвищує час розробки таких програм для реалізації інформаційних технологій. Одним з потенційних підходів до полегшення цієї задачі є використання нових децентралізованих технологій для створення аналітичних рішень для ВД. Зокрема, останнім часом отримали розвиток дослідження можливостей використання мікросервісів у системах обробки ВД. У зв'язку зі збільшенням вимог до аналізу ВД (АВД) з'явилися різноманітні технології та платформи для здійснення такого аналізу [1]. Відповідно до функціональних специфікацій, ці платформи АВД загалом поєднують логіку обробки даних із керуванням обчислювальними ресурсами під час обробки.

Актуальність використання мікросервісів для АВД пояснюється тим, що на сьогоднішній день існує великий попит на системи, здатні працювати з ВД, а мікросервісна архітектура дозволяє розробляти гнучкіші варіанти обробки потокових даних в реальному часі (зокрема, з використанням контейнерних технологій).

Традиційним підходом до побудови систем потокової обробки даних (СПОД) досі залишається використання монолітної архітектури [2]. Монолітна архітектура - це традиційний спосіб побудови програмних додатків СПОД, згідно з яким ці додатки будуються як неподільний блок, основними складовими якого є бази даних (БД), інтерфейс на стороні клієнта та сервер для обробки запитів. Компоненти такої архітектури тісно пов'язані один з одним і мають розроблятися та керуватися як одна сутність, оскільки їх виконання відбувається в рамках одного процесу операційної системи. При цьому навіть незначна зміна в одній частині будь-якого компонента вимагає нового релізу всього програмного забезпечення СПОД.

Функціонування складних організаційно-технічних систем зазвичай пов'язане з проблемами, обумовленими зростанням обсягу потоків даних, необхідністю інтеграції з новими інформаційними об'єктами, оновленням програмної платформи тощо. Зі зростанням таких проблем монолітні архітектури потроху втрачають свою актуальність через необхідність підтримувати їх та вносити зміни (наприклад, в разі розширення функцій систем обробки даних). У зв'язку з цим все більшої популярності набувають так звані мікросервісні архітектури (МСА). Головний принцип таких архітектур - розбиття монолітного програмного додатку деякої інформаційної системи на сукупність модулів (мікросервісів), що можуть бути в разі потреби вилучені або додані (за умови дотримання певних вимог) без порушення функціональності всієї системи. Такий підхід дає можливість розширювати інформаційну систему, масштабувати її та забезпечувати швидку та гнучку оркестрацію всіх модулів.

Можна визначити такі позитивні властивості МСА: модульність (кожний мікросервіс призначений для вирішення конкретної задачі); мінімальний об'єм програмного коду для

кожного з мікросервісів; багатоплатформеність (можливість використання різних засобів та технологій для модулів); відмовостійкість тощо. Цей перелік характеристик підкреслює виграшну позицію мікросервісних архітектур в порівнянні з монолітними. Однак МСА мають також і певні недоліки, зокрема: розробка складних програмних проєктів з використанням мікросервісів є довготривалим процесом; існують певні труднощі при перерозподілі функцій між мікросервісними компонентами системи; кожен мікросервіс потребує окремого обслуговування, тому необхідним є постійний автоматизований моніторинг та логування даних.

Слід зазначити, що на сьогодні немає універсальних рекомендацій щодо застосування мікросервісного підходу для деяких завдань обробки ВД в інформаційних системах різного функціонального призначення. Зокрема, потребує додаткових досліджень проблема децентралізації потокової обробки ВД, що може вирішуватися із застосуванням мікросервісного підходу. При цьому доцільним і актуальним є проведення аналізу МСА для дослідження їх переваг в порівнянні з використанням монолітних сервісів для обробки потоків ВД.

Розглянуті вище особливості побудови систем обробки ВД із застосуванням різних концептуальних підходів (моносервісного та мікросервісного) обумовили характер досліджень, результати яких обговорюються у даній роботі.

Метою цих досліджень є розробка варіанту МСА системи потокової обробки ВД, яка основана на використанні сучасних засобів розробки та запуску додатків в середовищах з підтримкою контейнеризації.

Відповідно до поставленої мети, необхідно вирішити наступні завдання:

- аналіз існуючих МСА потокової обробки ВД;
- розроблення варіанту системи мікросервісної обробки ВД з використанням розподіленої потокової платформи подій Kafka; платформи розробки, доставки та запуску програм Docker; об'єктно-реляційної системи управління БД Postgres; веб-платформи для створення додатків FastAPI;
- моделювання запропонованої МСА СПОД з метою доведення її переваг в порівнянні з монолітною архітектурою СПОД для обробки потоків ВД.

2. Сучасні архітектури потокової обробки великих даних з використанням мікросервісів

Для визначення сукупності компонентів, що є доцільним використати в подальшому у варіанті багатокомпонентної архітектури потокової обробки ВД, розглянемо властивості, переваги і недоліки існуючих схем та засобів такої обробки [3-5].

Відзначимо, що термін "великі дані" (Big Data) є досить розпливчастим. З еволюційної точки зору розмір "великих даних" постійно змінюється. Якщо, наприклад, використовувати поточну глобальну пропускну здатність Інтернет-трафіку як вимірну одиницю, то значення обсягу ВД будуть розташовані між терабайтним і зетабайтним діапазонами. Компанія IBM запропонувала концепт 4V для визначення ознак ВД: обсяг (Volume), що означає масштабність даних; щвидкість обробки (Velocity), що означає аналіз потокових даних; різноманіття (Variety), що вказує на різні форми ВД; правдивість (Veracity), що передбачає можливість невизначеності ВД. Деякі аналітики включають додаткові визначення ВД на основі літери V: варіативність (Variability), видимість (Visibility) та значимість (Value).

Здебільшого, ВД використовуються для АВД. Крім розвитку методів та алгоритмів АВД, необхідним і важливим є створення високопродуктивних систем для ефективної обробки ВД. На практиці використання таких систем має забезпечувати обчислювальні вимоги механізмів обробки даних для виконання завдань АВД під час виконання програм.

Більшість алгоритмів та систем АВД пристосовані для використання даних, що зберігаються в існуючих базах або сховищах даних. Це означає, що такі дані готові для безпосередньої роботи з ними. Проте існують ситуації, коли вхідні дані надходять потоками до систем їх аналізу та обробки з великою швидкістю оновлення, що не завжди робить можливим та доцільним їх зберігання в БД з метою подальшого опрацювання. Більшість алгоритмів потокової обробки використовують відсортування максимально небажаних/непотрібних даних на вході та подальшу роботу з вхідними даними. Існує також можливість подачі на вхід даних фіксованого розміру, що пришвидшить роботу першочергово, але разом з тим виникне ймовірність потрапляння есенційних даних до алгоритму, що призведе до втрати точності фінального аналізу. Різні алгоритми і підходи мають різні ідеї і можли-

вості для застосування, оскільки має місце питання пріоритетності (оптимальність/затратність).

Потокові системи достатьо часто використовуються в сучасних технологіях обробки даних, але існують суттєві обмеження щодо реалізаці потокової обробки. Перш за все, необхідно враховувати швидкість отримання потоків даних: якщо такі потоки надходять надто швидко, слід обрахувати надісланий пакет даних максимально швидко (при цьому треба брати до уваги ймовірність нерелевантності даних при надто великому очікуванні), тому слід підбирати алгоритм, що матиме змогу виконувати усі операції виключно в операційній системі, не використовуючи допоміжних пристроїв, адже це негативно позначається на швидкості самої системи. В той же час необхідно враховувати те, що даних може бути настільки багато, що не буде можливості отримувати ідеальний варіант, достатньо лише використовувати наближення, котрі задовольняють вимоги модуля: для цього слід використовувати наближені методи або методи, що працюють з хешами. Подібна практика використання є прийнятним компромісом між швидкістю і точністю.

Розглянемо основні операції, які зазвичай використовують для оптимізації роботи з даними. Базовий процес управління даними передбачає, насамперед, операцію вибірки даних, що у СПОД здійснюється з застосуванням хешування. Це дозволяє отримати максимально репрезентативну вибірку даних, а результат хешування буде застосовано для всього потоку.

Іншим типом операцій попередньої обробки потоків є фільтрація вхідних даних. Сама по собі фільтрація даних не є складним процесом, проте необхідно створити максимально швидкий фільтр який виконував би свою функцію максимально швидко [6]. Для цього в СПОД ефективним вважається використання фільтра Блума, що має такі компоненти: масив n біт, де кожне значення дорівнює 0; набір хеш-функцій $h1$, $h2$, ..., hk, що відображають значення ключів для n осередків; множину S, що містить m ключів. Головне призначення фільтра Блума - пропускати всі елементи потоку, ключі яких належать множині S, і відкидати більшість елементів з ключами, що не належать S. Після ініціалізізації бітового масиву встановимо в 1 біти, номери яких співпадають з hi(K) для деякої хеш-функції hi і деякого ключа K з множини S. При обробці ключа K, що надійшов з потоку, перевіряємо, чи значення всіх бітів з номерамі $h1(K)$, $h2(K)$, ..., hk(K) дорівнюють одиниці. Це означає, що при присутності усіх значень хеш-функцій у множині пропускаємо черговий елемент. Якщо хоча б один біт дорівнює 0, то ключ K не може належати S, тому цей елемент відкидаємо. Однак такий елемент може пройти і тоді, коли його ключ відсутній в S. Це може відбутися при недостатній кількості хеш-функцій або недостатній потужності множини S.

Ще однією важливою операцією потокової обробки є знаходження елементів в потоці. Найочевиднішим підходом для вирішення подібної задачі буде збереження в оперативній пам'яті тих елементів, які вже були попередньо зафіксовані. Більш того, можливо зберігати їх в структурі даних, що забезпечує ефективний пошук (наприклад, у хеш-таблиці або дереві пошуку, які дозволяють легко додавати нові елементи і перевіряти їх відповідність черговому елементу з потоку).

При відносно невеликій кількості елементів в потоках даних розглянуті операції можуть бути реалізовані в СПОД без особливих зусиль. Проте, при великій кількості елементів в потоці (або при опрацюванні занадто великої кількості потоків) лишається малоймовірною можливість збереження усіх потрібних даних в оперативній пам'яті СПОД з монолітною архітектурою. Існують різні рішення для подолання цієї проблеми. Можна, наприклад, використовувати декілька машин, кожна з яких буде відповідати за обробку одного або декількох потоків. Крім того, можна зберігати значну кількість даних у зовнішній пам'яті та збирати елементи потоку до пакету, виконуючи багато перевірок та оновлень даних в цьому пакеті та переміщаючи його з диска до оперативної пам'яті. Це деякою мірою відповідає концепції використання МСА для потокової обробки ВД, що досліджується в даній роботі.

Розглянемо особливості уніфікованої архітектури ВД для обробки ВД, що дозволяє приймати, обробляти та аналізувати дані, які є надто об'ємними або надто складними для традиційних систем БД [7].

Уніфікована архітектура ВД (Unified Big Data Architecture) - це архітектура ВД, що включає інфраструктуру, інструменти та технології, які створюють, керують і підтримують збір даних, їх обробку, аналітику та елементи машинного навчання. Уніфікована архітектура передбачає централізовану інфраструктуру даних, щоб уникнути дублювання даних і додаткових програмних зусиль. Рішення для обробки ВД зазвичай призначені для одного або кількох з таких типів

робочого навантаження: пакетне оброблення джерел неактивних ВД; обробка ВД в реальному часі; інтерактивне вивчення ВД; прогнозна аналітика та машинне навчання.

Архітектура потокової обробки ВД (Streaming Big Data Processing Architecture) - це структура програмних компонентів, створена для прийому та обробки великих обсягів потокових даних із багатьох джерел [8]. У той час, як традиційні рішення для обробки даних зосереджені на записі та зчитуванні даних пакетами, архітектура потокових даних споживає дані відразу після їх створення, зберігає їх у сховищі та може включати різноманітні додаткові компоненти для кожного випадку використання, такі як інструменти для обробки в реальному часі, дані маніпуляції та аналітика (рис. 1).

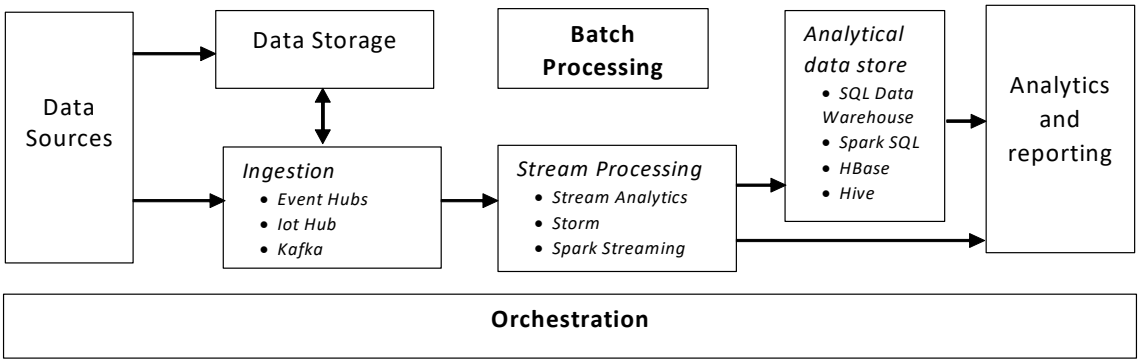

Рис. 1. Базова архітектура потокової обробки великих даних

Варіант базової архітектури потокової обробки ВД, наведений на рис. 1, включає практично всі компоненти уніфікованої архітектури ВД. Основним компонентом є джерела даних (Data Sources). Всі рішення для обробки ВД починаються з одного або кількох джерел даних (це можуть бути: сховища даних додатків, наприклад, реляційні БД; статичні файли, які створюються програмами, наприклад, файли журналу вебсервера; джерела даних з передачею в режимі реального часу, наприклад, пристрої Інтернету речей (IoT Hub). Для систем обробки ВД характерна наявність сховища даних (Data Storage). Дані пакетної обробки зазвичай зберігаються в розподіленому сховищі файлів, де можуть міститися значні обсяги великих файлів у різних форматах. Цей тип сховищ часто називають озером даних (Data Lake).

Слід також розглянути інші компоненти, наявні у більшості архітектур ВД. До таких компонент можна віднести засоби пакетного оброблення (Batch Processing). Оскільки набори даних зазвичай великі, то часто у СПОД обробляються пакетні завдання. Над даними виконуються різні операції, такі як фільтрація, статистична обробка та інші процеси підготовки даних до аналізу. Зазвичай у ці завдання входить також читання вихідних файлів, їх обробка і запис вихідних даних у нові файли.

Ще одним важливим компонентом є отримання повідомлень у режимі реального часу. Якщо рішення містить джерела, що працюють у реальному часі, в архітектурі має бути передбачений спосіб збирання та збереження повідомлень у режимі реального часу для потокової обробки (Stream Processing). Це може бути просте сховище даних з папкою, в яку вхідні повідомлення розміщуються для обробки. Але для прийому повідомлень багатьом рішенням потрібне сховище, яке можна використовувати як буфер. Таке сховище має підтримувати обробку з горизонтальним масштабуванням, надійну доставку та іншу семантику черги повідомлень.

Важливим компонентом систем ВД є також сховища аналітичних даних (Analytical data store). У багатьох рішеннях обробки ВД дані готуються до аналізу. Потім оброблені дані структуруються відповідно до формату запитів для засобів аналітики. Сховище аналітичних даних, яке використовується для обробки таких запитів, може бути і реляційною БД, наприклад, Kimball, що можна побачити в більшості традиційних рішень бізнес-аналітики.. Крім того, дані можна представити за допомогою технологій NoSQL (SQL Data Warehouse, Spark SQL), що мають низьку затримку на читання і запис, а також технології HBase або інтерактивної БД Hive, яка надає абстракцію метаданих для файлів даних у розподіленому сховищі.

Слід відзначити також аналіз та створення звітів (Analytics and reporting). Більшість рішень для обробки ВД дозволяють отримати уявлення про дані за допомогою аналізу та звітів. Для того, щоб розширити можливості аналізу даних, можна включити в архітектуру шар моделювання, наприклад, модель багатовимірного куба OLAP (Online analytical

processing), що сприяє організації великх БД і підтримує їх комплексний аналіз [9]. Їх можна використовувати для виконання складних аналітичних запитів без негативного впливу на транзакційні системи. Останнім компонентом розглянутої архітектури є оркестрація (Orchestration). Більшість рішень для обробки ВД складаються з повторюваних робочих процесів, під час яких перетворюються вихідні дані, дані переміщаються між декількома джерелами і приймачами, оброблені дані завантажуються в сховища аналітичних даних або результати передаються безпосередньо у звіт або на панель моніторингу.

Microsoft рекомендує використовувати подібну архітектуру для таких сценаріїв: зберігання та обробка даних в обсягах, надто великих для традиційної БД; перетворення неструктурованих даних для аналізу та створення звітів; запис, обробка та аналіз асінхронних потоків даних у режимі реального часу або з низькою затримкою;

До переваг такої архітектури можна віднести: вибір технологій та їх комбінацій (наприклад, можна комбінувати та зіставляти керовані сервіси Azure та технології Apache у кластерах HDInsight, щоб з максимальною вигодою застосовувати існуючі навички); підвищення продуктивності за допомогою паралелізму (у рішеннях обробки ВД використовується перевага паралелізму, що дозволяє застосовувати високопродуктивні рішення, які можуть масштабуватися до роботи з великими обсягами даних); еластичне масштабування (всі компоненти архітектури для обробки ВД підтримують горизонтальне масштабування для адаптування рішень для малих і великих робочих навантажень і плати лише за реально використовувані ресурси мережі); взаємодію з поточними рішеннями (компоненти архітектури для обробки ВД також використовуються у відповідних рішеннях Інтернету речей та корпоративних рішеннях бізнес-аналітики, що дозволяє створювати інтегровані засоби для робочих навантажень обробки даних).

Втім розглянута базова архітектура несе додаткову складність, адже практична реалізація обробки ВД може потребувати значної кількості компонентів для прийому даних з декількох джерел. Створення, тестування та усунення неполадок процесів обробки ВД також може стати непростим завданням через велику кількість параметрів конфігурації для оптимізації продуктивності.

Слід також врахувати набір навичок, необхідних для розробників (велика кількість технологій для обробки ВД є вузькоспеціалізованими та використовують платформи і мови, які є стандартними для більш загальних архітектур додатків). З іншого боку, технології обробки ВД сприяють розвитку нових інтерфейсів API з урахуванням більш традиційних мов.

Суттєвою проблемою може стати зрілість технологій, адже багато технологій, що використовуються для обробки ВД, ще знаходяться в розвитку (якщо основні технології Hadoop, Hive та Pig вже сформувалися та широко використовуються, то такі нові технології, як Spark, з кожним релізом зазнають значних змін і вдосконалень).

Важливою характеристикою архітектури ВД є також безпека. У рішеннях з обробки ВД всі статичні дані зазвичай зберігаються у централізованому озері даних Data Lake. Захист доступу до цих даних є нетривіальним завданням, якщо дані повинні прийматися та використовуватися кількома програмами та платформами. У таких системах рекомендується використання паралелізму. У більшості технологій обробки ВД робоче навантаження розподіляється між кількома одиницями обробки, тому статичні файли даних створюються та зберігаються у доступному для розбивки форматі. Розподілені файлові системи типу HDFS мають забезпечити оптимізацію продуктивності читання та запису (зазвичай фактична обробка виконується тут паралельно в кількох вузлах кластера, що скорочує загальний час виконання завдань).

У таких системах пакетна обробка (Batch processing) виконується за регулярним розкладом (наприклад, щотижня або щомісяця). Секціоновані файли та табличні структури даних ґрунтуються на темпоральних періодах, що відповідають розкладу обробки. Це спрощує прийом даних, планування завдань та усунення помилок. Крім того, таблиці розділів, які використовуються у запитах Hive, U-SQL або SQL, можуть значно підвищити їхню продуктивність.

У традиційних рішеннях бізнес-аналітики для переміщення даних до сховища використовується процес екстракції, перетворення та завантаження ETL (Extract-Transform-Load) (рис. 2).

Першим кроком цього процесу є екстракція (Extract) даних із цільових джерел, які зазвичай неоднорідні (наприклад, бізнес-системи, API, дані датчиків, маркетингові інструменти та БД транзакцій тощо). Деякі з цих типів даних є структурованими виходами широко використовуваних систем, тоді як інші є напівструктурованими серверними журналами JSON.

Другий крок полягає в перетворенні (Transform) необроблених даних, отриманих із джерел, у формат, який можна використовувати різними програмами. На цьому етапі дані очищуються, відображаються та перетворюються, щоб відповідати оперативним потребам. Цей процес передбачає кілька типів перетворень, які забезпечують якість і цілісність даних. Дані зазвичай не завантажуються безпосередньо в цільове джерело даних, натомість їх завантажують у проміжну БД. Цей крок забезпечує швидкий відкат, якщо щось піде не за планом.

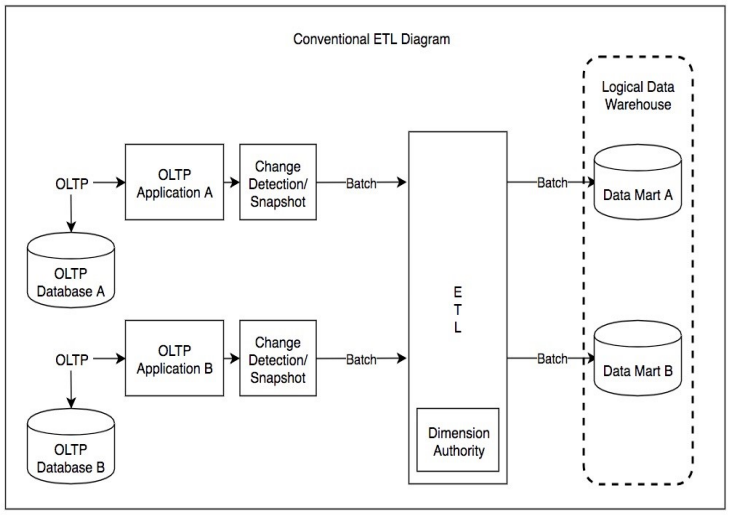

Рис. 2. Уніфікована ETL-діаграма

Останньою функцією ETL є завантаження (Load) - це процес запису перетворених даних із проміжної області в цільову БД. Залежно від вимог програми, цей процес може бути простим або складним. Кожен із цих кроків можна виконати за допомогою інструментів ETL або спеціального коду.

Для великих обсягів даних і різноманітних форматів у рішеннях обробки ВД зазвичай використовуються різні варіації ETL, наприклад, схема перетворення, екстракції і завантаження TEL (Transform-Extract-Load). В цьому випадку дані обробляються в розподіленому сховищі даних та перетворюються на необхідну структуру перед переміщенням до сховища аналітичних даних.

За даної архітектури слід також регулювати витрати при тарифікації на основі обсягу та часу використання. Для завдань пакетної обробки дуже важливо враховувати два фактори: витрати на одиницю обчислювальних вузлів та щохвилинну вартість використання цих вузлів для виконання завдання. Наприклад, виконання пакетного завдання може тривати вісім годин під час використання чотирьох вузлів кластера. При цьому може виявитись, що всі чотири вузли використовуються для завдання лише протягом перших двох годин, а після цього достатньо двох вузлів. У такому разі виконання всього завдання на двох вузлах збільшить загальний час, але не подвоїть його, що зменшить сукупну вартість виконання. У деяких бізнес-сценаріях триваліший час обробки може бути кращим, ніж вища вартість використання ресурсів кластера, що не використовуються.

Таким чином, уніфікована архітектура ВД щільно пов'язана з ETL-процесами, які забезпечують потрапляння даних в реляційне сховище. Перевагою такої моделі є відносна простота і легкість реалізації, а її відмінність від традиційних систем обумовлюється лише вибором технології реалізації, де компоненти бізнес-системи замінені на інструменти Big Data.

Однак, розглянуті технології вимагають багато часу та зусиль для налаштування аналітичних звітів. Сама уніфікована архітектура призначена для пакетної обробки даних і не завжди підтримує потокову передачу подій у реальному часі. Усунути цей недолік можна за допомогою удосконалення архітектурних моделей.

Потокові архітектури мають враховувати унікальні характеристики потоків ВД, які генерують величезні обсяги даних (від терабайтів до петабайтів). Ці дані в кращому випадку є напівструктурованими та потребують значної попередньої обробки та застосування схеми ETL для того, щоб стати інформаційно корисними.

Завдання практичної реалізації потокової обробки ВД рідко вирішується за допомогою однієї БД або ETL-інструментів, тому доцільно спроєктувати рішення, що складається з кількох будівельних блоків. Наприклад, ідея Upsolver SQLake полягає в тому, щоб замінити точкові продукти інтегрованою платформою, яка забезпечує самоорганізовані конвеєри декларативних даних. Далі буде продемонстровано, як цей підхід проявляється в кожній частині ланцюга постачання потокових даних.

Раніше потокова обробка була нішевою технологією, яка використовувалася лише невеликою групою компаній. Однак з розвитком концепції програмного забезпечення як послуги SaaS (Software as a Service), а також IoT і машинного навчання в різних галузях все частіше

використовують потокову аналітику. Оскільки трафік до цифрових активів (додатків або вебсайтів) сучасних компаній зростає, застосування сучасної інфраструктури даних та комплексної аналітики в реальному часі стає необхідним.

У порівнянні з традиційними пакетними архітектурами, що не завжди можуть бути цілком достатніми, системи потокової обробки забезпечують кілька важливих переваг.

Головною з таких переваг є здатність працювати з нескінченними потоками подій. Традиційні інструменти пакетної обробки вимагають зупинки потоку подій, збору пакетів даних і об'єднання пакетів для отримання загальних висновків. Потокова обробка, незважаючи на складність об'єднання та захоплення даних із кількох потоків, дозволяє отримати оброблену інформацію з великих обсягів потокових даних. Іншою перевагою потокових систем є можливість обробки в режимі реального часу (або майже в режимі реального часу). Потокова обробка також спрощує виявлення закономірностей у даних часових рядів (виявлення закономірностей у часі, наприклад, пошук тенденцій у даних трафіку веб-сайту, потребує постійної обробки та аналізу даних). Останньою перевагою є легкість масштабування даних (це об'єднує потокову обробку з MCA), адже зростання обсягів даних може порушити роботу системи пакетної обробки, вимагаючи надання додаткових ресурсів або зміни архітектури. Сучасна інфраструктура потокової обробки є гіпермасштабованою, тобто здатною працювати з гігабайтами даних на секунду за допомогою одного потокового процесора. Це дає змогу обробляти зростаючі обсяги даних без змін інфраструктури.

Більшість потокових систем все ще будуються на конвеєрі з відкритим вихідним кодом та на пріоритетних рішеннях для проблем прийому і зберігання даних, обробки потоку та оркестровки завдань. Будь-яка потокова архітектура має включати чотири ключові архітектурні блоки.

Першим з них є брокер повідомлень (The Message Broker). Це елемент, що отримує дані з джерела, яке називається виробником (Producer), перетворює їх у стандартизований формат повідомлення та постійно передає їх потоком, після чого інші компоненти можуть використовувати повідомлення, передані брокером.

Перше покоління брокерів повідомлень, таких як RabbitMQ і Apache ActiveMQ, покладалося на парадигму підпрограмного забезпечення, орієнтованого на обробку повідомлень MOM (Message Oriented Middleware). Пізніше з'явилися надпродуктивні платформи обміну повідомленнями, або потокові процесори (Stream Processor), які більше підходять для потокової парадигми. Найбільш популярними інструментами обробки потоків є Apache Kafka та Amazon Kinesis Data Streams.

На відміну від MOM-брокерів, потокові брокери підтримують дуже високу продуктивність, зберігаючи цілісність даних, мають величезну пропускну здатність трафіку повідомлень (гігабайт за секунду або навіть більше) і зосереджені на потоковій передачі з незначною підтримкою перетворення даних або планування завдань.

Другим блоком є платформи ETL. Потоки даних від одного чи кількох брокерів повідомлень необхідно агрегувати, трансформувати та структурувати, перш ніж дані можна буде проаналізувати за допомогою інструментів аналітики на основі SQL. Це можна зробити за допомогою платформи ETL, яка отримує запити від користувачів, витягує події з черг повідомлень, а потім застосовує запит для генерації результату (у цьому процесі часто виконуються додаткові об'єднання, перетворення або агрегації даних). Результатом може бути виклик API, дія, візуалізація, сповіщення або новий потік даних. Прикладами інструментів ETL з відкритим кодом для потокової передачі даних є Apache Storm, Spark Streaming і WSO2 Stream Processor. Хоча ці структури працюють по-різному, усі вони здатні прослуховувати потоки повідомлень, обробляти дані та зберігати їх у пам'яті. Деякі потокові процесори, включаючи Spark і WSO2, забезпечують синтаксис SQL для запитів і обробки даних.

Після того, як потокові дані підготовлені для використання потоковим процесором, їх необхідно проаналізувати. Існує багато різних інструментів аналізу потокових даних, найпоширенішими з яких є Amazon Athena, Amazon Redshift, Elasticsearch та Cassandra.

З появою відносно недорогих технологій зберігання ВД більшість організацій сьогодні зберігають дані своїх потокових подій. Ці дані можна зберігати або у базах та сховищах даних (наприклад, Postgre SQL або Amazon Redshift), або у брокерах повідомлень (наприклад, за допомогою постійного сховища Kafka) або у озерах даних (наприклад, Amazon S3). Озеро даних є найбільш гнучким і недорогим варіантом для зберігання даних про події, але його створення та обслуговування часто є дуже технічно складним завданням.

Розглянемо приклад організації системи потокової обробки ВД за допомогою мікросервісів на прикладі геоплатформи, запропонованої компанією MTS IT (рис. 3) [4].

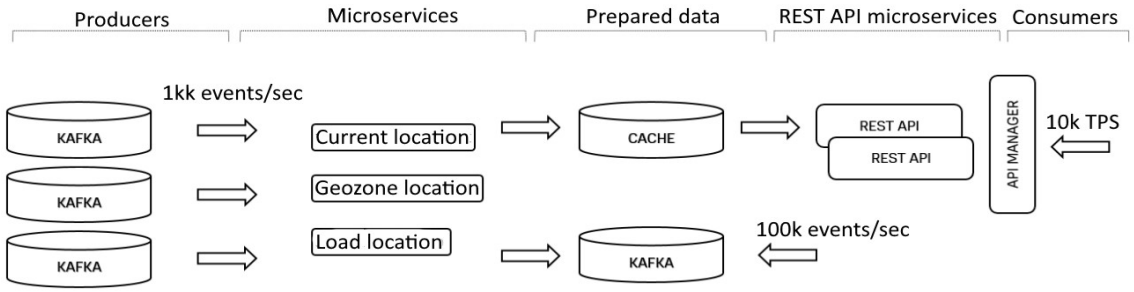

Рис. 3. Схематична взаємодія сервісів у геоплатформі MTS IT

В даній платформі джерелом даних є кластер Kafka, а далі йде набір сервісів, кожен з яких реалізує окремий бізнес-процес (наприклад, сервіс визначення поточного місцезнаходження клієнта, сервіс моніторингу входу клієнтів в дану геозону або визначення завантаженості станцій в реальному часі). Після чого ці дані можна запросити на вимогу через API, які також поділяються на сервіси за бізнес-кейсами, або зчитати з черги повідомлень кластеру Kafka.

Серед переваг такої архітектури можна підкреслити самостійне розгортання та оновлення служб [10]. Так система дозволяє швидко запустити в комерційну експлуатацію нову бізнес-функцію або цілий сервіс, не боячись порушити роботу інших сервісів. Це особливо стосується обробки ВД, адже великий обсяг вхідних даних і їх різноманітність призводять до формування великої кількості нових бізнес-кейсів і частої зміни вже існуючих.

Слід також відзначити і ефективність горизонтального масштабування. Індивідуальне масштабуваня послуг як окремий сервіс є ефективнішим, ніж масштабування послуг як частини моноліту.

Вагомою перевагою є також те, що сервіси можуть бути реалізовані за допомогою різних мов програмування та фреймворків. Це дуже важливо в Big Data, де через велику кількість бізнес-процесів багато різних команд одночасно працюють на одній платформі [11]. З точки зору розробки слід відзначити низький поріг входу для нових розробників, адже маленьку кодову базу легко зрозуміти. Зазвичай, сервіси можна переписати з нуля за кілька SCRUM-спринтів.

Нарешті, останньою вагомою перевагою з точки зору управління проєктами є масштабування команд розробників. Якщо відразу зрозуміло, як розділити систему на мікросервіси, то за наявності ресурсів можна розпаралелити розробку та значно скоротити загальний час впровадження платформи. Але, як і будь-яка інша архітектура, МСА має і свої недоліки, головним з яких є розподілення системи. Систему, що складається з багатьох сервісів, стає важко спроєктувати та розробити (для таких комплексних задач потрібні архітектори та забудовники високої кваліфікації). Слід відзначити ненадійність мережних взаємодій в МСА (сервіси спілкуються один з одним через мережу, а мережеві з'єднання є схильними до збоїв). Останнім суттєвим недоліком використання МСА є складність оперування, адже розподілену систему важче підтримувати.

Порівняємо МСА (рис. 3) з рішенням, реалізованим згідно з концепцією монолітної архітектури (рис. 4). Серед переваг монолітної архітектури над МСА можна зазначити лише швидкість

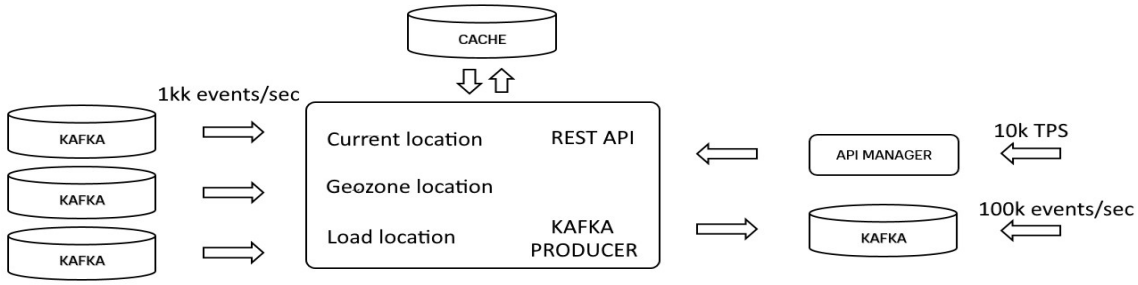

Рис. 4. Схематична взаємодія монолітної геоплатформи MTS IT

розгортання програми та відсутність мережевої взаємодії, що наявна у рішенні з використанням МСА [48], однак кількість недоліків монолітної платформи значно переважає ці два пункти.

Головним з таких недоліків слід вважати знижену загальну надійність (будь-яке доопрацювання одного з модулів може порушити дієздатність цілої низки інших сервісів).

Застосування монолітної архітектури передбачає також сильний зв'язок модулів (часто деякі модулі використовують методи інших модулів у своїй реалізації, тому для модифікації одного модуля потрібно модифікувати і кілька інших, щоб не зламати зворотню сумісність).

Ще одним важливим недоліком використання монолітної архітектури є великий вхідний поріг для нових розробників (розробнику потрібний значний час, щоб зрозуміти чималу кодову базу), що значно ускладнює впровадження нових технологій.

Суттєво різняться і вимоги до інфраструктури МСА та монолітної архітектури для систем потокової обробки ВД. В таблиці 1 наведені оцінки вимог до апаратних ресурсів інфраструктури систем обробки ВД з монолітною та мікросервісною архітектурами. Тут використані позначення: S - small (низькі вимоги), M - middle (середні вимоги), L - large (значні вимоги) для таких типів ресурсів як RAM (об'єм оперативної пам'яті), CPU (ресурси центрального процесора), HDD / SSD (потрібне місце на дисках) та LAN (мережа комунікації).

Таблиця 1

Характеристика вимог до апаратних ресурсів інфраструктури систем обробки великих даних з монолітною та мікросервісною архітектурами

Архітектура	RAM	CPU	HDD/SSD	LAN
Монолітна	M	M	M	S
Мікросервісна	L	L	S	L

3. Пропонований варіант архітектури потокової обробки великих даних з використанням мікросервісів

Для визначення сукупності компонентів, що є доцільним використати в подальшому у варіанті МСА системи потокової обробки ВД, були розглянуті та обрані такі мови програмування і технології: інтерпретована об'єктно-орієнтована мова програмування Python; програмне забезпечення для автоматизації розгортання та керування програмами в середовищах з підтримкою контейнеризації; контейнеризатор додатків Docker; розподілена потокова платформа подій Apache Kafka, об'єктно-реляційна СУБД PostgreSQL; веб-платформа для створення API FastAPI [12].

Наведемо короткий опис цих елементів.

Python є інтерпретованою високорівневою об'єктно-орієнтованою кросплатформеною мовою програмування. На сьогодні ця мова є провідною у сферах штучного інтелекту, машинного навчання та нейромережевого моделювання.

Docker є відкритою платформою для розробки, доставки та запуску програм. Скориставшись методологіями Docker для швидкої доставки, тестування та розгортання коду в середовищі контейнерів, можна значно зменшити затримку між написанням коду та його використанням в МСА.

Apache Kafka є розподіленою потоковою платформою подій із відкритим кодом, що використовується для високопродуктивних конвеєрів даних, потокової аналітики, інтеграції даних і критично важливих програм, та складається з серверів і клієнтів, які спілкуються через високопродуктивний мережевий протокол TCP. Kafka працює як кластер з одного або кількох серверів, які можуть охоплювати кілька центрів обробки даних або хмарних регіонів. Kafka дозволяє створювати розподілені програми та мікросервіси, які зчитують, записують і обробляють потоки подій паралельно у відмовостійкий спосіб.

Postgres (повна назва PostgreSQL) є об'єктно-реляційною СУБД, що характеризується масштабованістю і відповідає стандартам SQL. Його основною функцією є безпечне зберігання даних та подальше їх отримання за запитом інших програм (тих, що знаходяться на тому самому комп'ютері або тих, що працюють на інших комп'ютерах мережі. Postgres може обробляти робочі навантаження від невеликих однопоточних додатків до великих Інтернет-додатків з багатьма одночасними користувачами.

FastAPI є сучасною високопродуктивною веб-платформою для створення API на основі стандартного синтаксису мови Python. Ключовими особливостями цієї платформи є висока продуктивність (це один із найшвидших фреймворків Python); висока швидкість програмування; інтуїтивно зрозумілий синтаксис; мінімалістичність (дублювання коду зведено тут до мінімуму).

Отже, з огляду на всі переваги розглянутих засобів реалізацій, такий технологічний стек дозволяє швидко розробити систему потокової обробки ВД (з урахуванням перспектив подальшого розвитку системи). Усі обрані технології є загально визнаними та широко застосовуваними.

Аналіз різних архітектур ВД, що організовані за допомогою мікросервісів, дозволив розробити прототип системи ВД з використанням розглянутих вище методів і технологій.

Схему взаємодії мікросервісів у запропонованому рішенні наведено на рис. 5.

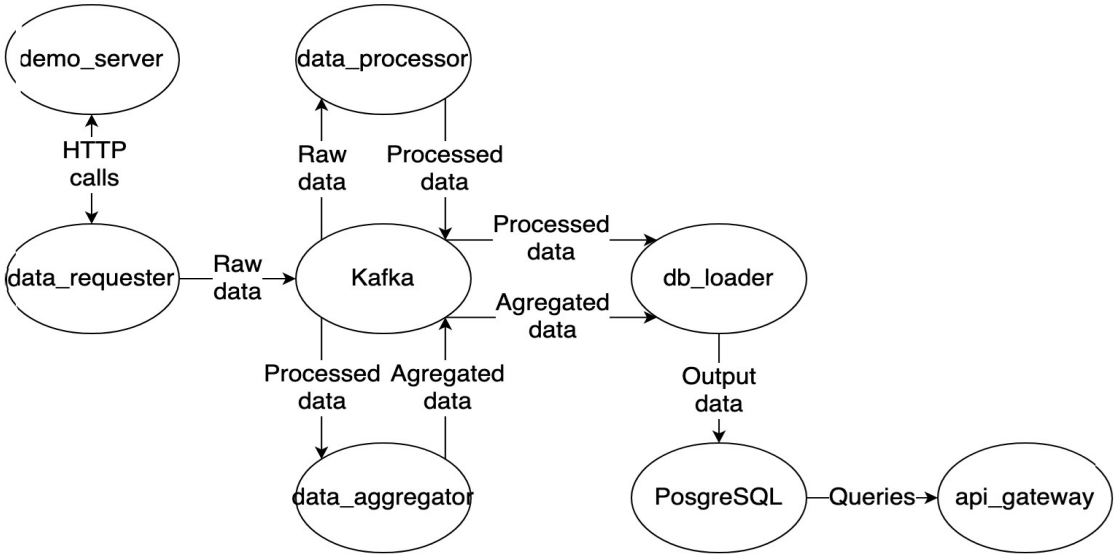

Рис. 5. Схема взаємодії мікросервісів у запропонованому рішенні

Розглянемо працездатність цієї схеми на прикладі її використання для створення системи бізнес-аналізу, яка приймає в потоковому режимі велику кількість записів, що являють собою пару курсів валют, та обчислює середні значення цих пар для деякої кількості запитів. Даний прототип системи розглянуто для перевірки концепції (Proof of Concept) застосування МСА у системах потокової обробки ВД.

Іншим напрямом проведення експериментальних досліджень було порівняння характеристик систем потокової обробки ВД для монолітного та мікросервісного варіантів їх реалізації.

4. Результати розробки та експериментального дослідження пропонованого варіанту МСА для систем потокової обробки великих даних

Для прототипизації мікросервісного застосунку, що працює з ВД згідно з запропонованою схемою взаємодії мікросервісів, було обрано потокову архітектуру.

За допомогою інструментарію Docker було створено систему Docker-контейнерів, кожен з яких відповідає за поставлене йому завдання:

- контейнер infrastructure, що містить базову та запропоновану інфраструктури, а також усі інфраструктурні послуги (такі як Kafka та Postgres);
- контейнер demo_server, що містить демонстраційну серверну програму, що створює випадкові тестові дані, необхідні для демонстрації;
- контейнер data_requester, що виконує мікросервісну програму запиту даних та запитує дані з демонстраційного сервера та надсилає їх до Kafka;
- контейнер data_processor, що відповідає за читання повідомлення, отриманого від сервісу data_requester через Kafka, після чого розділяє дані та завантажує оброблені дані назад у Kafka;
- контейнер data_aggregator, що містить програму агрегатора даних; служба агрегатора даних читає повідомлення, отримані від процесора даних через Kafka, і обчислює середнє значення для останніх 10 відповідей;

- контейнер db_loader, що завантажує БД та службу завантажувача БД, читає повідомлення від сервісів data_processor і data_aggregator і зберігає ці дані в БД Postgres;

- контейнер api_gateway, що надає HTTP-ендпоінти для запиту даних із бази даних Postgres.

Для реалізації процедур створення та запуску зазначених Docker-контейнерів були написані відповідні скрипти та виконавчі файли.

Тестові дані формуються у мікросервісі demo_server, з якого щосекундно дані запитуються мікросервісом data_requester. Сервіс data_requester є першим мікросервісом системи. Після запиту цей сервіс зберігає відповідь у Kafka.

Далі це повідомлення вичитує data_processor, обробляє його та теж зберігає у Kafka. Потім його вичитують сервіси data_aggregator та db_loader. Перший сервіс вираховує середні значення, а другий переміщає до бази всі повідомлення, згенеровані сервісом data_aggregator. Сервіс api_gateway формує запити до БД за вимогою користувача. Весь проєкт та всі додаткові сервіси запускатимуться у сервісі docker-compose.

Для тестування та демонстрації роботи всієї системи розроблено сервіс, який формує значення валютних пар у JSON-форматі з залученням бібліотеки FastAPI (лістинг 1, див. рис. 6).

Дані з сервера запитує мікросервіс api_requester. Під час створення було визначено ім'я програми, що є обов'язковим аргументом. Якщо ми запустимо кілька екземплярів сервісу з однаковим ім'ям, Kafka розподілить партиції між ними, що дозволить масштабувати у систему горизонтально.

```
@router.get("/pairs", tags=["pairs"])
async def get_pairs() -> Dict:
  metrics.GET_PAIRS_COUNT.inc()
  return {
     "USDUAH": round(random.random() * 100, 2),
     "EURUAH": round(random.random() * 100, 2)
  }
```

Рис. 6. Лістинг 1. Ендпоінт для генерації пари двох курсів валют

Сервіс data_requester не читає повідомлень з тем Kafka, але кожну секунду відправляє запит до емулятора даних і обробляє відповідь. Для опису функції, що викликається за таймером із заданим інтервалом, розроблено відповідний сервіс (лістинг 2, див. рис. 7).

```
@app.timer(interval=1.0)
async def request_data() -> None:
  provider = data_provider.DataProvider(
     base_url=config.get(config_loader.BASE_URL))
  pairs = await provider.get_pairs()
  metrics.REQUEShbc/T_CNT.inc()
  logger.info(f"Received new pairs: {pairs}")
  if pairs:
     await src_data_topic.send(key=uuid.uuid1().bytes,
        value=json.dumps(pairs).encode())
```

Рис. 7. Лістинг 2. Циклічна функція, що запрошує дані з data_provider

Періодичність виконання функції контролює декоратор @app.timer. Функція створює екземпляр класу DataProvider, який відповідає за запит даних.

Наступним мікросервісом є data_processor, що займається обробкою пар, отриманих від мікросервісу api_requester. Цей сервіс отримує повідомлення з топіка та обробляє їх.

Для цієї функції використано декоратор @app.agent(src_data_topic), який контролює те, що функція реагує на повідомлення в темі src_data_topic. Повідомлення вичитуються в циклі async for msg_key, msg_value in stream.items(). Далі здійснюються серіалізація отриманого повідомлення та формування валютних пар та їх значень, причому кожна пара записується в наступний топік окремо (лістинг 3, див. рис. 8).

Два попередні сервіси читають та пишуть дані в Kafka потоками. Кожне нове повідомлення вони опрацьовують незалежно від попередніх, але періодично виникає необхідність обробки повідомлення разом із попередніми. Система має порахувати середнє значення для останніх 10 значень пар, що мають бути збережені.

Якщо запустити кілька примірників сервісу агрегацій, то кожен з них зберігатиме локально лише свої значення, що призведе до формування некоректних середніх значень. Тому необхідно реалізувати функцію збереження історії та вирахування середнього значення (лістинг 4, див. рис. 9).

Мікросервіс db_loader прослуховує одразу дві теми - processed_data і data-aggregator-average-changelog. До першої теми пише повідомлення data_processor, а до другої -

```
@app.agent(src_data_topic)
async def on_event(stream) -> None:
  async for msg_key, msg_value in stream.items():
    metrics.SRC_DATA_RECEIVED_CNT.inc()
    logger.info(f"Received new pair message {msg_value}")
    serialized_message = json.loads(msg_value)
    for pair_name, pair_value in serialized_message.items():
      logger.info(f"Extracted pair: {pair_name}: {pair_value}")
      metrics.PROCESSED_PAIRS_CNT.inc()
      await processed_data_topic.send(key=msg_key,
        value=json.dumps({pair_name: pair_value}).encode())
      metrics.PROCESSED_DATA_SENT_CNT.inc()
      yield msg_value
```

Рис. 8. Лістинг 3. Код отримання пар даних

```
@app.agent(processed_data_topic)
async def on_event(stream) -> None:
  async for msg_key, msg_value in stream.items():
    metrics.PROCESSED_DATA_RECEIVED_CNT.inc()
    logger.info(f'Received new processed data message {msg_value}')
    serialized_message = json.loads(msg_value)
    for pair_name, pair_value in serialized_message.items():
      average_value = average_table.get(pair_name, {})
      if average_value:
        average_value['history'].append(pair_value)
        average_value['history'] = average_value['history'][-10:]
        average_value['average'] = round(sum(average_value['history']) / \
          len(average_value['history']), 2)
      else:
        average_value['history'] = [pair_value]
        average_value['average'] = pair_value
      logger.info(f"Aggregated value: {average_value}")
      average_table[pair_name] = average_value
      metrics.PAIRS_AVERAGE_AGGREGATED_CNT.inc()
```

Рис. 9. Лістинг 4. Функція підрахунку середнього значення двох пар

data_aggregator, що викликає необхідність опису двох функцій обробки повідомлень. За аналогією з іншими сервісами, ці повідомлення читаються з Kafka та передаються до БД. Для роботи з базою скористаємося інструментом об'єктно-реляційного відображення ORM (Object-relational mapping) SQLAlchemy.

Для запиту результатів було написано сервіс на FastAPI, який вичитує дані з БД PosgreSQL та повертає їх у JSON форматі (рис. 10).

Для роботи з БД при цьому використовується ORM, як і на попередньому кроці (лістинг 5, див. рис. 11).

Для тестування можливостей запропонованої МСА було здійснено експериментальне порівняння характеристик систем потокової обробки ВД для монолітного та мікросервісного варіантів їх реалізації (на прикладі бізнес-системи, що здійснює управління поставкою продуктів, а також обробку відповідних замовлень і платежів).

Порівняння здійснювалося для двох вебзастосунків з різними архітектурами: перший з монолітною архітектурою (рис. 12), а другий - з МСА (запропонований варіант) системи (рис. 13), яка складається з трьох компонентів: auth (автентифікація), products (продукти) і checkout (оформлення замовлення).

Монолітна архітектура вебзастосунку передбачає, що весь код і функціональність знаходяться в одній програмі або модулі. Всі компоненти та функції, такі як обробка запитів, доступ

```
dnefodov@DNEFODOV-M-Q33X streaming % curl 'http://127.0.0.1:8007/pairs/average'
[{"id":"1","create_date":"2022-11-26 12:07:03.513934","pair_name":"\"USDUAH\"","value":"54.1"},{"
id":"2","create_date":"2022-11-26 12:07:03.898984","pair_name":"\"EURUAH\"","value":"39.57"}]%

dnefodov@DNEFODOV-M-Q33X streaming % curl 'http://127.0.0.1:8007/pairs/currencies?pair_name=USDUA
H'
[{"id":"667","create_date":"2022-11-26 12:07:38.168654","pair_name":"USDUAH","value":"78.55"},{"i
d":"665","create_date":"2022-11-26 12:07:37.148802","pair_name":"USDUAH","value":"26.21"},{"id":"
663","create_date":"2022-11-26 12:07:36.164853","pair_name":"USDUAH","value":"0.68"},{"id":"661",
"create_date":"2022-11-26 12:07:35.141548","pair_name":"USDUAH","value":"57.71"},{"id":"659","cre
ate_date":"2022-11-26 12:07:34.147308","pair_name":"USDUAH","value":"51.88"},{"id":"657","create_
date":"2022-11-26 12:07:33.143946","pair_name":"USDUAH","value":"27.58"},{"id":"655","create_date
":"2022-11-26 12:07:32.114102","pair_name":"USDUAH","value":"8.65"},{"id":"653","create_date":"20
22-11-26 12:07:31.095606","pair_name":"USDUAH","value":"37.89"},{"id":"651","create_date":"2022-1
1-26 12:07:30.095508","pair_name":"USDUAH","value":"63.13"},{"id":"649","create_date":"2022-11-26
 12:07:29.086311","pair_name":"USDUAH","value":"74.85"}]%
```


Рис. 10. Фрагмент результатів роботи системи з мікросервісною архітектурою

```
@router.get("/pairs/currencies", tags=["pairs"])
async def get_currencies(pair_name: str, limit: int = 10) -> List[Dict]:
metrics.GET_CURRENCIES_CNT.inc()
  return await db.get_currencies(pair_name, limit)
@router.get("/pairs/average", tags=["pairs"])
async def get_average() -> List[Dict]:
 metrics.GET_AVERAGE_CNT.inc()
 return await db.get_averages()
```

Рис. 11. Лістинг 5. Ендпоінти отримання курсів валют та середнього значення останніх 10 відповідей

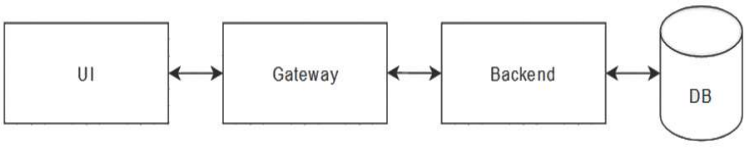

Рис. 12. Структурна схема застосунку з монолітною архітектурою

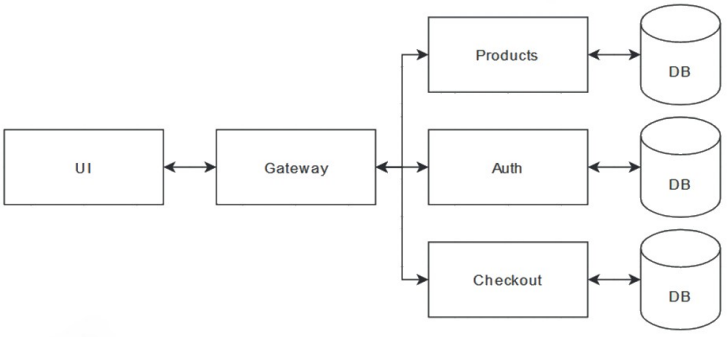

Рис. 13. Структурна схема застосунку з мікросервісною архітектурою

до БД і бізнес-логіка, розташовані в одному монолітному середовищі. Це означає, що всі запити на сервер обробляються одним програмним кодом.

Вебзастосунок з розділеною МСА розбитий на три компоненти: auth, products і checkout. Кожен компонент виконує конкретні функції, пов'язані зі своєю областю відповідальності. Компонент auth відповідає за автентифікацію і авторизацію користувачів, products - за управління продуктами та інформацією про них, а checkout - за обробку замовлень і платежів.

Під час тестування з великою кількістю асинхронних запитів (50 000 та 500 000) до трьох різних варіантів систем (монолітного, мікросервісного з однією реплікою та мікросервісного з двома репліками) були отримані наступні результати щодо середнього часу відповіді.

Для монолітного застосунку:

- при 50 000 запитів середній час відповіді склав 13 мс;
- при 500 000 запитів середній час відповіді збільшився до 110 мс.

Для мікросервісного застосунку з однією реплікою:
- при 50 000 запитів середній час відповіді становив 14 мс;
- при 500 000 запитів середній час відповіді скоротився до 99 мс.
Для мікросервісного застосунку з двома репліками:
- при 50 000 запитів середній час відповіді склав 16 мс;
- при 500 000 запитів середній час відповіді скоротився до 87 мс.
На основі цих результатів можна зробити декілька спостережень.

Монолітний застосунок мав найшвидшу середню відповідь при 50 000 запитів, але при збільшенні кількості запитів до 500 000 його час відповіді значно зріс.

Мікросервісний застосунок з однією реплікою показав хороші показники швидкості відповіді як при 50 000, так і при 500 000 запитів.

Мікросервісний застосунок з двома репліками демонстрував найкращу продуктивність, зменшуючи час відповіді зі зростанням кількості запитів (він дозволяє відстежити середнє значення курсів валют за останні 400 тисяч випадково згенерованих записів в середньому за 30 секунд).

Це свідчить про те, що МСА з розподіленим навантаженням на декілька реплік може забезпечити кращу масштабованість і швидкодію в порівнянні з монолітною архітектурою. Застосування реплік дозволяє розділити навантаження між декількома екземплярами сервісу, що призводить до поліпшення продуктивності та зниження часу відповіді. Слід зазначити, що у варіанті тестування застосунка з однією реплікою використання мікросервісної арзітектури характеризується певними затратами через застосування проксі-серверу для розподілу запитів по мікросервісах.

5. Висновки та перспективи подальших досліджень

На сьогодні не існує універсальних рекомендацій щодо застосування мікросервісного підходу для деяких завдань обробки ВД в інформаційних системах різного функціонального призначення. Зокрема, потребує додаткових досліджень проблема децентралізації потокової обробки ВД, що може вирішуватися із застосуванням мікросервісного підходу. Актуальність використання мікросервісів у ВД пояснюється тим, що на сьогоднішній день існує великий попит на системи, здатні працювати з ВД, а МСА дозволяє отримати більш гнучкі для підтримки довготривалі рішення та спростити управління розробкою.

Аналіз різних архітектур ВД, організованих за допомогою мікросервісів, дозволив запропонувати варіант системи мікросервісної обробки ВД з використанням розподіленої потокової платформи подій Kafka, платформи розробки та запуску програм Docker; об'єктно-реляційної системи управління БД Postgres, а також веб-платформи для створення додатків FastAPI. З огляду на всі можливості розглянутих засобів, такий технологічний стек дозволяє прискорити розробку працездатної та ефективної схеми взаємодії мікросервісів для обробки потоків ВД (з урахуванням перспектив подальшого розвитку системи). Розроблені архітектурні шаблони можуть спростити розробку систем обробки ВД з використанням мікросервісів. Результати моделювання запропонованої МСА доводять її переваги в порівнянні з монолітною архітектурою в системах аналізу обробки потоків ВД. МСА з розподіленим навантаженням на декілька реплік може забезпечити кращу масштабованість і швидкодію в порівнянні з монолітною архітектурою.

Перспективою продовження досліджень, повязаних з розробкою МСА обробки ВД, може бути удосконалення схеми взаємодії компонентів запропонованого варіанту архітектури, поліпшення його експлуатаційних характеристик та розширення функціональних можливостей.

Список літератури: 1. *Efremov A.* Application of microservice architecture in Big Data streaming https://prog.world/application-of-microservice-architecture-in-big-data-streaming (Last accessed: 26.11.2022). 2. *Newman S.* Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. O'Reilly Media, Incorporated, 2019. 272 p. 3. *Zikopoulos P.* Harness the power of big data, the IBM Big data platform. US: McGraw-Hill. 2013. P. 4. *Demchenko Y.* Defining architecture components of the big data ecosystem. In: IEEE Collaboration Technologies and System (CTS). 2014. 104 p. 5. *Стиль* архитектуры для обработки больших данных. 2022. URL: https://learn.microsoft.com/ru-ru/azure/architecture/guide/architecture-styles/big-data (Last accessed: 26.11.2022). 6. *Miliauskas E.* A guide to data warehousing clickstream data [Електронний ресурс] / Evaldas Miliauskas. Stacktome. 2019. Режим доступу до ресурсу: https://stacktome.com/blog/a-guide-to-data-warehousing-clickstream-data. 7. *Miao K., Li J., Hong W.,Chen M. A.* Microservice-

Based Big Data Analysis Platform for Online Educational Applications. 2020. URL: https://www.hindawi.com/journals/sp/2020/6929750 (Last accessed: 26.11.2022). 8. *Levy E.* Streaming Data Architecture in 2022: Components and Examples. 2022. URL: https://www.upsolver.com/blog/streaming-data-architecture-key-components (Last accessed: 26.11.2022). 9. *Tejada Z.* Online analytical processing (OLAP). 2022. URL: https://learn.microsoft.com/en-us/azure/architecture/data-guide/relational-data/online-analytical-processing (Last accessed: 26.11.2022). 10. *Dean Z.* How to Overcome Data Order Issues in Apache Kafka [Електронний ресурс] / Zeke Dean. 2020. Режим доступу до ресурсу: https://www.dataversity.net/how-to-overcome-data-order-issues-in-apache-kafka/. 11. *Soylemez M., Tekinerdogan B., Tarhan A.* Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review. URL: https://doi.org/10.3390/app12115507. (Last accessed: 26.11.2022). 12. *Richards M., Ford N.* Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media. 2020. 432 p.

Надійшла до редколегії 14.11.2022

Нефьодов Даніїл Андрійович, аспірант кафедри штучного інтелекту ХНУРЕ. Наукові інтереси: розробка архітектури розподілених систем, методи стримінгової класифікації. Адреса: Україна, 61000, Харків, проспект Науки, 14, тел. +38(095)3643243, e-mail: danyil.nefodov@nure.ua, ORCID: 0009-0008-3171-1397

Удовенко Сергій Григорович, доктор технічних наук, професор, завідувач кафедри інформатики та комп'ютерної техніки ХНЕУ ім. С. Кузнеця. Наукові інтереси: інтелектуальні системи керування та обробки інформації. Адреса: Україна, 61000, Харків, проспект Науки, 9-А, тел. +38(067)9098331, e-mail: udovenkosg@gmail.com, ORCID: 0000-0001-5945-8647

Чала Лариса Ернестівна, кандидат технічних наук, доцент, доцент кафедри штучного інтелекту ХНУРЕ. Наукові інтереси: проєктування інформаційних систем, обробка природномовної інформації. Адреса: Україна, 61099, Харків, проспект Науки, 14, тел. +38(068)3511405, e-mail: larysa.chala@nure.ua, ORCID: 0000-0002-9890-4790

УДК 004.02:005.2 DOI: 10.30837/0135-1710.2022.178.064

В.В. ЯРМАК

МОДИФІКАЦІЯ МЕТОДІВ ОЦІНЮВАННЯ ТРУДОВИТРАТ ІТ-ПРОЄКТУ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ

Розглянуто проблему оцінювання трудовитрат ІТ-проєкту розробки мобільного додатку. Проведено аналіз інтуїтивних та параметричних моделей і методів оцінювання трудовитрат ІТ-проєктів, виділено їх основні недоліки. Запропоновано вдосконалити існуючі інтуїтивні методи оцінювання трудовитрат ІТ-проєктів, збільшуючи об'єктивність оцінок. Розроблено модифікований метод секторів, який враховує основні недоліки похідних інтуїтивних методів оцінювання трудовитрат.

1. Вступ

ІТ-проєкти розробки мобільних додатків слід віднести до окремого підкласу ІТ-проєктів. Це рішення базується на відмінностях, які вирізняють ІТ-проєкти розробки мобільних додатків серед інших різновидів ІТ-проєктів. Серед цих відмінностей у [1] особливо виділяються відмінності в технічному завданні на проєктування, графічному дизайні та додаткових апаратних можливостях, які слід враховувати під час розробки мобільних додатків. Як показано у [1], ці відмінності є головними причинами збільшення вартості ІТ-проєктів розробки мобільних додатків.

Однак оцінити вартість ІТ-проєкту розробки мобільного додатку неможливо без проведення оцінювання трудовитрат на виконання подібного проєкту. Між тим, існуючі моделі і методи оцінки трудовитрат ІТ-проєктів не підходять для об'єктивної оцінки трудовитрат ІТ-проєкту розробки мобільного додатку. Ці моделі і методи створювалися раніше за виникнення подібного різновиду ІТ-проєктів і недостатньо пристосовані для оцінювання мобільних додатків.

Часто оцінка трудовитрат ІТ-проєкту розробки мобільного додатку проводиться суб'єктивно. Суб'єктивність оцінки залежить від рівня навичок ІТ-фахівця, який проводить оцінювання. Значна кількість існуючих моделей і методів оцінювання трудовитрат мають

високу швидкість оцінки і можуть бути досить ефективними, проте вони мають значний недолік - низький рівень точності оцінки завдання.

Тому дуже важливо якомога точніше оцінювати трудовитрати ІТ-проєкту розробки мобільного додатку, щоб у ІТ-компанії вистачило ресурсів для успішного його завершення. Необ'єктивна оцінка трудовитрат ІТ-проєкту може привести до марної трати фінансових ресурсів, а також до зайвих трудовитрат і виснаження команди проєкту, що в свою чергу може збільшити ризик банкрутства ІТ-компанії та її подальшого закриття.

2. Аналіз особливостей використання існуючих моделей і методів для оцінювання трудовитрат ІТ-проєкту розробки мобільного додатку

Використання для управління сучасними ІТ-проєктами методології Agile призводить до застосування переважно інтуїтивних методів оцінки завдань. Даний вибір обумовлений наступними принципами [2]:

- в Agile дуже цінується швидкість оцінки задач (сама по собі оцінка не має ніякої бізнес-цінності, тому в динамічному ітеративному процесі логічно зробити її з найменшими затратами праці, якомога дешевше і оперативніше);
- акцентування на командну роботу (у методах Agile дуже важливо використовувати думку різних фахівців команди);
- використання відносних одиниць вимірювання.

Як приклади інтуїтивних методів оцінювання трудовитрат ІТ-проєктів було проаналізовано особливості таких моделей і методів: метод "Price-to-win"; оцінка Паркінсона; метод "T-Shirt Sizes"; метод "Planning Poker"; метод "Bucket System"; метод "Dot-voting"; метод "Maximum Size or Less"; метод "Big/Small/Uncertain" та метод "Ordering Rule".

Слід також зазначити, що в даний час існує досить велика кількість параметричних моделей і методів оцінювання витрат на виконання ІТ-проєкту створення різноманітних ІТ-продуктів. Серед цих моделей і методів можна виділити [3]:

а) спрощений метод функціональних точок;
б) метод функціональних точок;
в) метод об'єктних точок;
г) метод Де-Марко;
д) метод точок властивостей;
е) лінійні методи;
ж) метод Wideband Delphi;
і) методи, засновані на моделях COCOMO і COCOMO II.

Незважаючи на різноманітність даних методів, більшість їх не можна застосовувати для оцінювання трудовитрат ІТ-проєкту розробки мобільного додатку. Так, застосування методів Де-Марко, точок властивостей, лінійних методів, а також методу Wideband Delphi в цьому випадку практично неможливе через неможливість отримання необхідних оцінок з прийнятним рівнем достовірності. У той же час, використання методів спрощених функціональних точок, класичних функціональних точок і об'єктних точок дозволяє отримати наближені, не зовсім точні, але обґрунтовані та придатні для прийняття рішень оцінки [4].

Опис загальних недоліків інтуїтивних та параметричних моделей і методів оцінювання трудовитрат з точки зору ІТ-проєкту розробки мобільного додатку наведений у табл. 1.

За результатами аналізу недоліків існуючих моделей і методів оцінювання трудовитрат, наведеними у табл. 1, слід зробити висновок про необхідність вдосконалення існуючих інтуїтивних моделей і методів шляхом підвищення об'єктивності оцінок трудовитрат.

3. Мета і задачі дослідження

Метою даного дослідження є розробка нового методу оцінювання трудовитрат ІТ-проєкту розробки мобільного додатку. Цей метод повинен бути простим у використанні та більш ефективним, ніж існуючі моделі та методи оцінювання трудовитрат.

Для досягнення даної мети в статті пропонується вирішити такі задачі:

- дослідити існуючі інтуїтивні методи оцінювання трудовитрат ІТ-проєктів;
- розробити модифікований метод оцінювання трудовитрат ІТ-проєкту розробки мобільного додатку.

Таблиця 1

Зведена таблиця недоліків існуючих моделей та методів оцінювання трудовитрат IT-проекту розробки мобільних додатків

Вид моделей/методів	Приклади моделей/методів	Загальні недоліки оцінок
Параметричні оцінки	а) спрощений метод функціональних точок; б) метод функціональних точок; в) метод об'єктних точок; г) метод Де-Марко; д) метод точок властивостей; е) лінійні методи; ж) метод Wideband Delphi; і) методи, засновані на моделях COCOMO і COCOMO II.	– занадто складні для оцінки задач, що пов'язані з розробкою мобільних додатків; – занадто громіздкі (необхідний значний час для розрахунку оцінки); – необхідно збирати значну кількість даних про продукт.
Інтуїтивні оцінки	а) price-to-win; б) оцінка Паркінсона; в) метод T-Shirt Sizes; г) метод Planning Poker; д) метод Bucket System; е) метод Dot-voting; ж) метод Maximum Size or Less; і) метод Big/Small/Uncertain; й) метод Ordering Rule.	– неможливість виключити повністю суб'єктивізм в оцінці експертів; – неможливість забезпечити об'єктивну оцінку компетентності експертів.

4. Дослідження існуючих інтуїтивних методів оцінювання трудовитрат IT-проєктів

Серед інтуїтивних методів оцінювання трудовитрат IT-проєктів було виділено методи "Maximum Size or Less", "Big/Small/Uncertain" та "Ordering Rule". Розглянемо ці методи детальніше.

Суть методу "Maximum Size or Less" (поділ до максимального розміру або менше) полягає в тому, що учасники процесу оцінювання спочатку визначають максимально можливий розмір для завдання у беклогу. Найчастіше як максимальне значення вибирається 1 людино-день. В цьому випадку найбільші завдання повинні вимагати для їх виконання не більше 1 дня [2].

Метод "Maximum Size or Less" складається з таких етапів [2]:

- Етап 1, на якому кожна історія обговорюється всіма учасниками, щоб відповісти на запитання: розмір задачі, що оцінюється, більший за максимальне значення або менший;

- Етап 2, на якому, якщо розмір даної історії більший за максимальний, група декомпозує її на підзадачі і повторює процес оцінки для складових частин.

Слід зазначити, що виконання Етапу 2 триває, доки все оцінювані завдання не опиняються в дозволеному діапазоні розмірів - дорівнюватимуть або будуть менше значення, обраного за максимальне.

Цей метод оцінки дуже простий у використанні, і з його допомогою команда здатна впоратися з 15-30 задачами (в залежності від складності та досвіду декомпозиції) за одну сесію [2].

Метод "Big/Small/Uncertain" (великий/малий/невизначений) схожий на техніку Bucket System. Основна відмінність цього методу полягає в тому, що в ньому використовується тільки 3 відерка (місця групування задач): відерко "Великий розмір", відерко "Малий розмір" та відерко "Невизначений розмір завдання".

Метод "Big/Small/Uncertain" складається з таких етапів [2]:

- Етап 1: всі оцінювані історії обговорюються учасниками і поміщаються в одну з трьох категорій Big / Small / Uncertain;

- Етап 2: група проводить обговорення кількох перших завдань (3-5), визначаючи масштаб і орієнтири для кожної категорії;

- Етап 3: історії, які залишилися нерозглянутими, розподіляються між учасниками і оцінюються самостійно, що сильно прискорює процес.

Цей метод є одним з найшвидших методів оцінювання трудовитрат. Він дозволяє оцінити за одну сесію велику кількість історій (50 та більше) і дозволяє залучати до процесу одночасно багато учасників [2].

Метод "Ordering Rule" (вибудовування порядку) являє собою покрокову гру, мета якої - вибудувати всі задачі одна відносно іншої на єдиній шкалі розміру.

Метод "Ordering Rule" складається з таких етапів [2]:

- Етап 1: всі оцінювані історії виписуються на картки;

- Етап 2: картки з завданнями випадковим чином розміщуються на столі або дошці зі шкалою, на кордонах якої вказані "малий розмір" і "великий розмір";

- Етап 3: кожен учасник по черзі робить свій "хід" оцінки. Такий "хід" включає одну з таких можливих дій: перемістити будь-яку історію за шкалою на одну поділку (тобто поміняти оцінку на нижчу або вищу), обговорити історію з колегами, пропустити свій "хід". В результаті "ходів" співробітників задачі можуть переміщатися по дошці, їх оцінка одна відносно одної уточнюється;

- Етап 4: у випадку, коли всі учасники пропускають свій "хід", процес оцінювання завершується. Всі задачі розподілені за шкалою між значеннями "малий розмір" і "великий розмір".

Цей метод досить ефективний для оцінки невеликої кількості задач (5-15). Учасники залучені до загального гейміфікованого процесу і, змінюючи положення історій відносно одне одного, домагаються високої точності оцінки [2].

Загальний недолік інтуїтивних методів полягає в тому, що вони надають значною мірою приблизний результат оцінювання, оскільки ґрунтуються на інтуїції експертів. Таким чином, виключити повністю суб'єктивізм в оцінці експертів з використанням даних методів неможливо.

5. Розробка модифікованого методу секторів

Базуючись на результатах аналізу розглянутих у розділі 4 методів оцінювання трудовитрат ІТ-проєктів, було розроблено новий комбінований метод, який отримав назву "Модифікований метод секторів". Цей метод складається з таких етапів.

Етап 1. Розподілення всією командою виконавців ІТ-проєкту (Scrum Team) разом перших 3-6 історій користувачів (User Stories) між беклогів, які позначені колами S, M та L. У разі дуже великих чи незрозумілих User Stories - відкладення їх до беклогу для декомпозиції. Приклад результату подібного розподілення показаний на рис. 1.

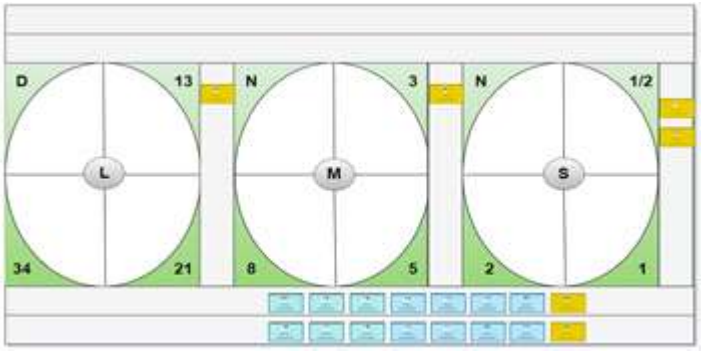

Рис. 1. Приклад результату виконання Етапу 1 модифікованого методу секторів

Етап 2. Індивідуальне та паралельне розподілення членами Scrum Team User Stories, що залишились у вхідному беклогу, між беклогами кіл S, M та L. У разі дуже великих чи незрозумілих User Stories - відкладення їх до беклогу для декомпозиції. Приклад результату виконання Етапу 2 показаний на рис. 2.

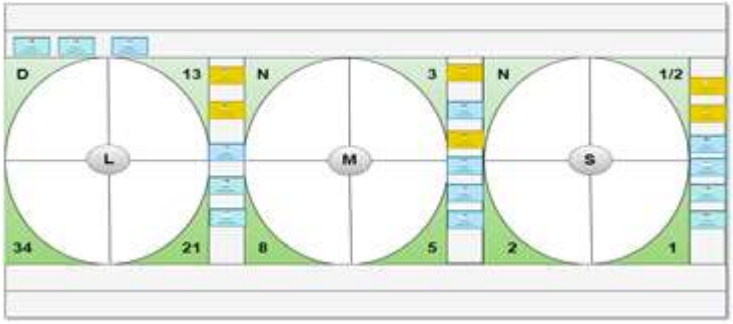

Рис. 2. Приклад результату виконання Етапу 2 модифікованого методу секторів

Етап 3. Розподілення членами Scrum Team User Stories між секторами кола S з використанням певного набору правил та дій.

Етап 4. Переміщення User Stories з сектору N кола S до беклогу кола M.

Етап 5.Розподілення членами Scrum Team User Stories між секторами кола M з використанням певного набору правил та дій.

Етап 6. Переміщення User Stories з сектору N кола M до беклогу кола L.

Етап 7. Розподілення членами Scrum Team User Stories між секторами кола L з використанням певного набору правил та дій..

Етап 8. Переміщення User Stories з сектору D кола L до беклогу для декомпозиції.

Етап 9. Декомпозиція усіх User Stories, що знаходяться у беклогу для декомпозиції.

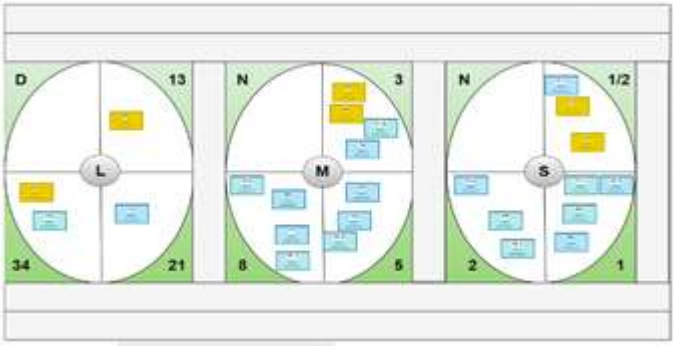

Рис. 3. Приклад результату виконання Етапу 10 модифікованого методу секторів

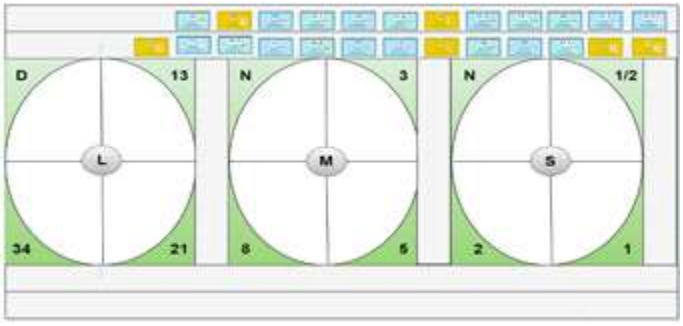

Рис. 4. Приклад результату виконання Етапу 11 модифікованого методу секторів

Етап 10. Повторення послідовності дій, починаючи з Етапу 2, доки кожен член Scrum Team не пропустить свій хід. Приклад результату виконання Етапу 10 показаний на рис. 3.

Етап 11. Розміщення оцінених User Stories у вихідному беклогу (верхня частина дошки) у порядку зростання складності. Приклад результату виконання Етапу 11 показаний на рис. 4.

Етап 12. У разі необхідності - уточнення під час обговорення певних User Stories членами Scrum Team та зміна кількості Story Points у них.

Даний метод є результатом комбінації похідних методів "Big/Small/Uncertain", "Maximum Size or Less" та "Ordering Rule" з вдосконаленнями цих методів та їх окремих етапів. Суть цього вдосконалення наведена у табл. 2.

6. Висновки і перспективи подальших досліджень

Розроблений модифікований метод секторів надає змогу оцінити значну кількість робіт за незнач-

Таблиця 2

Опис модифікацій похідних методів у модифікованому методі секторів

№	Вихідний метод та його етап	Суть модифікації вихідного методу	Етап методу секторів, в якому використовується етап вихідного методу
1	Метод «Big/Small/Uncertain». Етап 1.	Запропоновано розподіляти перші 3-6 User Stories між колами замість ведер	Етап 1
2	Метод «Big/Small/Uncertain». Етап 2.	Запропоновано розподіляти індивідуально та паралельно членами Scrum Team тих User Stories, що залишились між колами замість ведер	Етап 2
3	Метод «Maximum Size or Less». Весь метод.	Замість людино-днів для визначення трудовитрат будемо використовувати числа Фібоначчі, максимальне з котрих дорівнюватиме 34 Story Points. Якщо трудовитрати будуть більше, то User Story повинна бути декомпозована	Етап 1 та Етап 2

Кінець таблиці 2

№	Вихідний метод та його етап	Суть модифікації вихідного методу	Етап методу секторів, в якому використовується етап вихідного методу
4	Метод «Ordering Rule». Весь метод.	1. Поява чисел Фібоначчі, що дозволяє достатньо точно оцінити трудовитрати User Stories. 2. Використання замість прямої шкали трьох кіл (S,M,L), поділених на сектори	Етапи 3-9

ний короткий проміжок часу, до того ж він досить швидкий й простий у використанні. Можна також зробити попереднє припущення про універсальність даного методу, що робить можливим його використання у різних сферах, у тому числі для оцінювання трудовитрат ІТ-проєктів розробки мобільних додатків. Перевірка цього припущення є одним з перспективних напрямів подальших досліджень розробленого методу.

До недоліків модифікованого методу секторів можна віднести те, що у порівнянні з існуючими інтуїтивними методами оцінки трудовитрат задач даний метод потребує більше часу для оцінювання.

Перспективами подальших досліджень є проведення детальних випробувань розробленого модифікованого методу секторів під час оцінювання різних ІТ-проєктів розробки мобільних додатків. Крім того, пропонується провести дослідження з можливості використання розробленого методу для оцінювання трудовитрат ІТ-проєктів розробки інших різновидів ІТ-продуктів.

Список літератури: 1. *Розробка* мобільних додатків від А до Я: повний гайд . Dan-it. URL: https://dan-it.com.ua/blog/razrabotka-mobilnyh-prilozhenij-ot-a-do-ja-polnyj-gajd/ (дата звернення: 09.11.2020 р.). 2. *Моделі,* методи та засоби оцінки вартості програмного забезпечення. Dspace. URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/1541/36-Sidorov.pdf (дата звернення: 20.11.2020 р.). 3. *Євланов М.В., Соловйова К.І.* Уніфікація методів оцінювання витрат на створення сучасних інформаційних систем. Вісник Кременчуцького національного університету ім. Михайла Остроградського. 2014. Вип. 5/2014 (88). С. 62-67. 4. *COCOMO II Model Definition Manual // "Center for Systems and Software Engineering". URL: ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo99.0/modelman.pdf (дата звернення: 20.11.2020 р.).*

Надійшла до редколегії 02.12.2021

Ярмак Валерій Вячеславович, здобувач вищої освіти гр. УПГІТм-20-1 ХНУРЕ. Наукові інтереси: методи, моделі та інформаційні технології управління ІТ-проєктами розробки мобільних додатків. Адреса: Харків, 61166, пр. Науки, 14. Контактний телефон: +38(073)4775099.

УДК 004.89 DOI: 10.30837/0135-1710.2022.178.070

В.О. ЛЕЩИНЬСКИЙ

ПРОЦЕСНИЙ ПІДХІД ДО ПОБУДОВИ МОДЕЛІ КОНТЕКСТУ ПОЯСНЕНЬ В ІНТЕЛЕКТУАЛЬНІЙ ІНФОРМАЦІЙНІЙ СИСТЕМІ

Розглянуто проблему побудови пояснень з урахуванням контексту прийняття рішень в інтелектуальній системі. Виконано структуризацію контексту пояснення з урахуванням явних та неявних знань щодо предметної області, а також об'єктних та темпоральних обмежень щодо процесу прийняття рішення. Запропоновано артефактну модель контексту пояснень, що враховує сукупність каузальних та темпоральних обмежень по роботі із об'єктами предметної області. Розроблено процесно-орієнтований метод побудови моделі контексту пояснень, який дає можливість сформувати модель з урахуванням як явних, так і неявних знань щодо предметної області.

1. Вступ

Сучасні інтелектуальні інформаційні системи використовують складні алгоритми, що базуються на методах машинного навчання і тому не завжди є зрозумілими для користувачів. Така непрозорість процесу прийняття рішень може призвести до недовіри користувачів до отриманих результатів, і, як наслідок, затримок у їх практичному використанні. Наприклад, недовіра до системи розпізнавання образів може знизити рівень безпеки підприємства, яке використовує ці системи для відеонагляду; недовіра до рекомендованих товарів у рекомендаційній системі може привести до відтоку користувачів із відповідної системи електронної комерції [1, 2].

Для вирішення проблеми непрозорості процесу отримання рішення в інтелектуальній системі використовується механізм пояснення. Пояснення дозволяють користувачам зрозуміти ключові каузальні або темпоральні залежності, які привели до відповідного рішення інтелектуальної системи як "чорного ящику" [3, 4].

Такі залежності є мультиваріантними, оскільки вони залежать від контексту прийняття рішень. Останній визначає конкретну ситуацію, в якій безпосередньо приймається те чи інше рішення. Ситуація прийняття рішення являє собою середовище, що визначає множину умов, в яких формується та приймається рішення. Фактично контекст надає обмеження, в межах яких оцінюється та реалізується отримане рішення.

Контекст як опис ситуації прийняття рішення може містити у собі: цілі реалізації рішення; завдання, що потрібні для досягнення цілі; наявні ресурси, умови та обмеження на вирішення задач. Останні визначаються часовими рамками прийняття та реалізації рішення, потребами зацікавлених сторін, бажаними правовими та етичними характеристиками рішення.

Контекст прийняття рішень відіграє ключову роль у побудові процесу прийняття рішень, визначаючи допустимі у визначеній ситуації варіанти результуючих рішень. Тому врахування контексту є необхідним для ефективного прийняття рішень.

Аналіз контексту дає можливість визначити потенційні ризики імплементації рішення і, на цій основі, розглянути альтернативні варіанти рішення. Крім того, контекст, який містить умови та обмеження конкретного користувача, забезпечує індивідуалізацію отриманих рішень.

Тому врахування контексту при побудові пояснень має практичну значущість, а побудова моделей контексту прийняття рішень щодо пояснень в системах штучного інтелекту є актуальною задачею.

2. Аналіз літературних даних і постановка проблеми дослідження

Сучасні підходи до формування пояснень були сформовані в рамках програми самопояснювального штучного інтелекту - Explanable Artificial Intelligence (XAI) [5]. В цій програмі суттєва увага приділяється напрямку побудови пояснень з використанням причинно-наслідкових залежностей. Такі залежності між змінними визначаються з використанням байєсовського підходу [6]. Крім того, каузальні закономірності можуть бути встановлені й для структурованих об'єктів, що складаються з набору змінних [7]. Проте при поясненні процесу прийняття рішення в інтелектуальній системі розглядається і темпоральний аспект, який є відображен-

ням каузальних залежностей у динамічному середовищі. Як правило, темпоральний аспект враховується на основі виділення підмножини даних для певного проміжку часу [8].

Темпоральні залежності були виділені у формі зважених правил та проаналізовані в роботах [9-12]. В роботах [9, 10] було визначено темпоральні правила та запропоновано використати їх для формування множини альтернатив як складових процесу підтримки прийняття управлінських рішень. В роботі [11] запропоновано використати темпоральні правила та відношення між об'єктами для опису процесу прийняття рішень в інтелектуальній системі. В роботі [12] розглянуто проблему формування каузальних залежностей на основі темпоральних правил.

Таким чином, сучасні дослідження у сфері побудови пояснень орієнтовані, в першу чергу, на встановлення причинно-наслідкових залежностей між вхідними та проміжними даними, а також отриманим результатом. Частково враховується також темпоральний аспект на основі виділення часового інтервалу актуальності даних. Дослідження щодо опису процесу прийняття рішень у темпоральному аспекті свідчать про важливість використання темпоральних правил для формування процесу прийняття рішень. У сукупності результати проведених досліджень дають можливість описати процес прийняття рішень з використанням каузальних та темпоральних залежностей, що може служити основою для формування пояснень.

Однак існуючі дослідження не приділяють достатньо уваги побудові пояснень з урахуванням контексту, що має суттєвий вплив на їх актуальність та відповідність вимогам користувача. Тому розробка процесного підходу до моделювання контексту пояснень є актуальною задачею.

3. Мета і задачі дослідження

Метою даного дослідження є розробка моделі контексту пояснень інтелектуальної системи та методу її побудови з урахуванням процесного опису об'єктів, які використовуються в рамках пояснення, а також актуальності даних щодо цих об'єктів. Досягнення цієї мети створює умови для побудови пояснення за принципом чорного ящика, на основі лише інформації про підмножину подій, які виникали у процесі прийняття рішення в інтелектуальній інформаційні системі.

Для досягнення мети роботи вирішуються такі задачі:

- структуризація контексту пояснення з урахуванням явних та неявних знань щодо предметної області;
- розробка моделі контексту пояснень на основі сукупності умов та обмежень на процес прийняття рішення в інтелектуальній системі;
- розробка процесно-орієнтованого методу побудови моделі контексту пояснення з урахуванням темпорального аспекту обробки артефактів у інтелектуальній системі.

4. Структуризація контексту пояснень в системі штучного інтелекту

Контекст пояснень відноситься до конкретних обставин прийняття рішення в інтелектуальній системі і представлений інформацією та знаннями, що мають вплив на створення та використання пояснень. Фактично контекст надає додаткову інформацію, яка дає можливість користувачеві інтелектуальної системи зрозуміти конкретний варіант пояснення, причини отриманого рішення та впевнитись в релевантності останнього. Контекстна інформація також впливає безпосередньо на вибраний варіант пояснення.

Аналіз структури контексту пояснень дає можливість виділити такі його ключові складові: дані; явні знання; потреби користувача системи штучного інтелекту; неявні знання; обмеження на формування та на використання пояснень.

Дані характеризують стан предметної області та містять опис фактів, подій, а також умов і обмежень щодо їх використання, що суттєво впливають на пояснення і тому мають бути розтлумачені за додатковим запитом користувача.

Явні знання містять залежності, які обумовлюють або обмежують процес прийняття рішення в інтелектуальній системі і тому мають бути враховані в рамках пояснення.

Потреби користувача системи штучного інтелекту визначають спосіб використання отриманого результату і тому мають бути враховані у поясненні.

Неявні знання є неформалізованими та недокументованими знаннями, які мають практичне застосування в предметній області і, відповідно, впливають на процес прийняття рішення в інтелектуальній системі. Тому ці знання мають бути формалізовані для подальшого використання при побудові пояснення.

Неявні знання можна розділити на безпосередньо знання та досвід. Підходи до формалізації цих знань відрізняються. Перша категорія неявних знань (implicit-знання) може бути формалізована шляхом опитування експертів в предметній області. Досвід же не має вербального опису і тому не може бути формалізований шляхом інтерв'ювання. Досвід (tacit-знання) може бути, як правило, отриманий на основі фіксації та повторення дій користувача або аналізу логів інтелектуальної інформаційної системи.

Обмеження на формування та використання пояснень мають об'єктний та темпоральний аспекти. Темпоральний аспект задає часові рамки для використання пояснення. Об'єктний аспект охоплює ресурсні обмеження, юридичні вимоги, формалізовані вимоги до практичного застосування результуючого рішення інтелектуальної інформаційної системи.

Узагальнено ключові складові контексту пояснень представлено в таблиці 1.

Таблиця 1

Складові контексту пояснень

Елемент	Властивості
1. Дані	Факти, стани, події, умови та обмеження по них, що мають вплив на пояснення
2. Явні знання	Формалізовані залежності, які обумовлюють, визначають альтернативи або обмежують процес прийняття рішення
3. Потреби користувача	Способи використання отриманого результату, які є прийнятними для користувача інтелектуальної системи
4. Неявні знання	Implicit-знання, які можуть бути отримані шляхом опитування користувачів або експертів у предметній області, мозкових штурмів, тощо
	Tacit-знання, які зазвичай не переводяться у вербальну форму; такі знання можуть бути отримані шляхом аналізу поведінки користувачів, логів інформаційної системи, з якою працює користувач, тощо
5. Обмеження на формування та використання пояснень	Об'єктний аспект визначає обмеження на дії з об'єктами (даними про об'єкти) предметної області у процесі прийняття рішень в інтелектуальній системі
	Темпоральний аспект обмеження визначає часові рамки для використання пояснення, а також часові рамки дій в інтелектуальній системі, які потребують пояснення

Між наведеними в таблиці 1 властивостями контексту пояснень існує ряд залежностей.

По-перше, дані зазвичай є елементами знань про предметну область і можуть виступати як антецеденти причинно-наслідкових залежностей.

По-друге, для побудови та персоналізації пояснень необхідно поєднати знання про предметну область: явні, неявні, знання про використання рішення інтелектуальної системи. Явні та неявні знання про процес прийняття рішення дають можливість відібрати підмножину пояснень щодо поточного результату, а знання про спосіб використання результату - персоналізувати пояснення.

По-третє, розглянута сукупність знань визначає умови та обмеження для пояснення. Умови визначають підмножину варіантів пояснення для користувача, а обмеження мають бути виконані для всіх альтернатив пояснення.

Об'єктний та темпоральний аспекти обмежень для пояснення дають можливість розглядати та тлумачити процес прийняття рішення в системі штучного інтелекту у статиці та динаміці. Об'єктний аспект дає можливість визначити підмножину об'єктів для пояснення, а також підмножину допустимих дій з цими об'єктами. Темпоральний аспект визначає час, коли можна

виконувати вказані дії. Тобто при врахуванні темпорального аспекту ми можемо надавати різні пояснення щодо отриманого в інтелектуальній системі рішення на різних інтервалах часу.

Залежності між розглянутими властивостями контексту пояснень наведено на рис. 1.

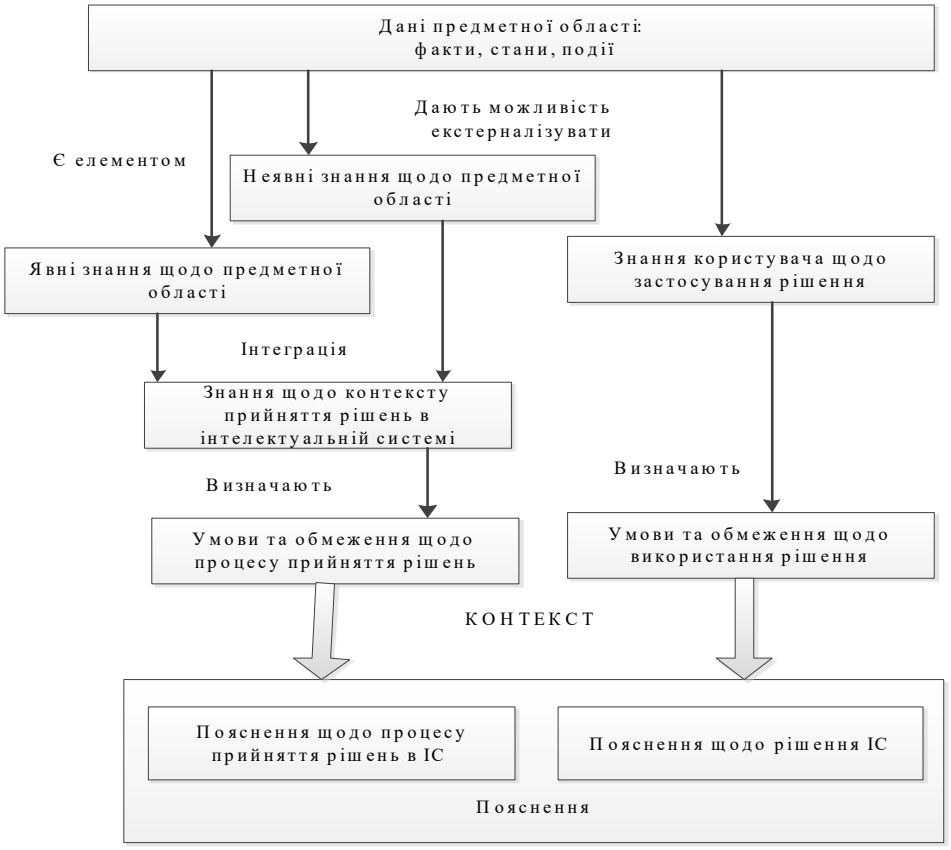

Рис. 1. Залежності між властивостями контексту пояснень

Із представленої на рис. 1 схеми можна зробити висновок про те, що суттєва для пояснення контекстна сукупність знань про процес прийняття рішення, а також про його використання може буде зведена до опису умов та обмежень щодо цього процесу у об'єктному та темпоральному аспектах.

5. Модель контексту пояснень в інтелектуальній системі

Представлена структура контексту пояснень відображає процедури оперування з об'єктами предметної області. Згідно з визначенням фірми IBM, в подальшому такі об'єкти будемо іменувати артефактами. Кожен артефакт має набір властивостей і множину допустимих дій. Крім того, для артефактів визначаються процеси обробки, які задають допустимі послідовності дій. Тобто можливий набір властивостей артефактів та дій з ними визначають контекстні умови та обмеження щодо процесу прийняття рішення у об'єктному аспекті, а допустимі послідовності дій відображають обмеження у темпоральному аспекті.

Оскільки при описі об'єктного та темпорального аспектів ми задаємо послідовність обробки артефактів у часі, такий опис контексту має процесну природу. Тобто послідовності обробки артефактів можна розглядати як підпроцеси єдиного процесу прийняття рішення в інтелектуальній системі. Відповідно, знання про контекст прийняття рішення мають процесну природу і визначають умови та обмеження у процесі отримання результату в системі штучного інтелекту. При використанні процесного опису контексту для пояснення достатньо виділити фрагменти підпроцесів у об'єктному та темпоральному аспектах і далі використати отримані знання для уточнення пояснення.

Розроблена артефактна модель контексту пояснень містить у собі множину артефактів $A = \{a_n\}$ та множину процесів обробки цих артефактів $P = \{p_l\}$:

$$M = \langle A, P : \forall p_l \in P \exists a_n \in A \rangle . \quad (1)$$

Кожен артефакт a_n задається множиною властивостей $V = \{V_n\}$ та множиною взаємопов'язаних дій з обробки артефакту:

$$D = \left\{ D_n : \left(\forall d_{n,s}, d_{n,k} \in D_i \right) \exists f_s^k \right\} , \quad (2)$$

де f_s^k - знання щодо залежності між діями $d_{n,s}, d_{n,k}$ з артефактом a_n.

Процес обробки цих артефактів складається із множини залежностей f_s^k:

$$P = \left\langle f_{k+1}^k, f_{k+2}^k, ..., f_s^k \right\rangle . \quad (3)$$

Вказані залежності можуть бути як каузальними, так і темпоральними. Темпоральні залежності задають послідовність у часі для станів артефакту в цілому, а каузальні - зв'язки між властивостями станів.

При кожній *i*-реалізації процесу обробки артефакту a_n отримуємо послідовність його станів $\pi_{n,i}$. Сукупність цих послідовностей Π_n містить спільні темпоральні залежності. Якщо ці залежності виконуються для всіх послідовностей, то їх можна розглядати як обмеження. В іншому випадку - як контекстні умови реалізації процесу прийняття рішення в інтелектуальній системі.

На базі темпоральних залежностей шляхом визначення ваг можуть бути отримані каузальні залежності. Вказані залежності об'єднують дії з обробки артефактів у єдиний процес.

Таким чином, представлена артефактна модель дає можливість врахувати як об'єктний, так і темпоральний аспекти контекстних умов прийняття рішення, що і становить контекст пояснення в інтелектуальній системі.

6. Процесно-орієнтований метод побудови моделі контексту пояснень

При реалізації розглянутої моделі дані щодо предметної області відображаються через властивості артефактів, з якими оперує процес прийняття рішення. Явні та неявні контекстні знання щодо процесу прийняття рішення відображаються через опис процесу обробки кожного з цих артефактів. Знання щодо результату представляються через опис процесу використання отриманого рішення як специфічного артефакту. У об'єктному аспекті визначаються можливі та допустимі дії із обробки об'єктів, а у темпоральному - послідовність цих дій у часі.

Приклад виділення фрагментів процесного опису контексту наведено на рис. 2.

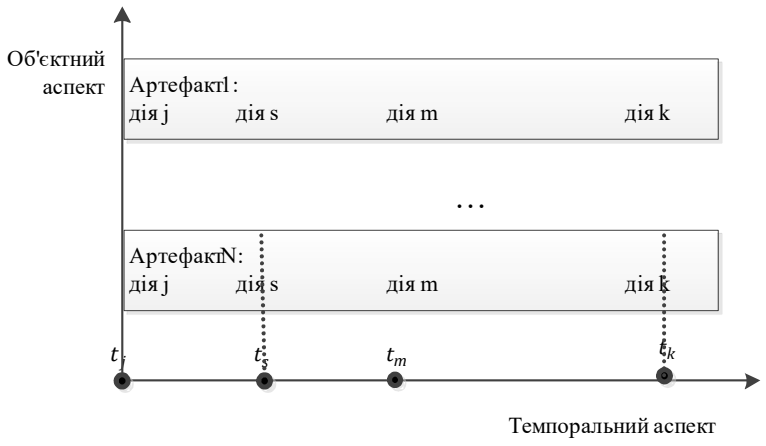

Рис. 2. Відбір обмежень і умов за об'єктним та темпоральним аспектами

Згідно з представленим прикладом, з підмножини артефактів в об'єктному аспекті виділяється a_N - N-артефакт. У темпоральному аспекті виділяються дії з артефактом $d_{n,s}, ..., d_{n,k}$ на інтервалі $[t_s, t_k]$. Ці дії змінюють відповідні підмножини властивості артефактів $\{v_{n,q}\}$.

Для побудови пояснень суттєвою є підмножина залежностей f_s^k між діями d_s та d_k та проміжними діями.

Представлені характеристики процесного опису пояснень дають можливість визначити основні етапи побудови контекстної моделі пояснень в системі штучного інтелекту.

Процесно-орієнтований метод побудови моделі контексту пояснення містить таку послідовність етапів.

Етап 1. Формування множини артефактів $A = \{a_n\}$, що є суттєвими для побудови пояснення.

Крок 1.1. Визначення множини властивостей артефактів $V = \{V_n\}$.

Крок 1.2. Визначення множини дій D над артефактами.

Етап 2. Відбір відомих *i*-реалізацій процесів обробки артефактів .

На даному етапі фіксуються відомі послідовності дій із обробки кожного з артефактів. Це дає можливість врахувати не лише явні знання щодо залежностей f_s^k, але й неявні tacit-знання, оскільки послідовність обробки артефактів відображає неформалізовані зв'язки між діями процесу обробки артефакту.

Етап 3. Побудова процесів обробки артефактів на основі узагальнення реалізацій $\pi_{n,i}$ з використанням темпоральних залежностей між діями з обробки артефактів.

Крок 3.1. Узагальнення реалізацій $f_{n,s}^{n,k}$ залежностей f_s^k у вигляді темпоральних правил r_s^k, що визначають послідовність обробки артефакту у часі, на інтервалі $[t_s, t_k]$.

Крок 3.2. Виділення підмножин темпоральних правил-обмежень та правил-умов виконання дій з обробки артефактів. Обмеження мають виконуватись на всіх реалізаціях $\pi_{n,i}$.

Крок 3.3. Побудова послідовності робіт процесу обробки кожного артефакту у часі на основі інтеграції темпоральних правил-умов при задоволенні правил-обмежень.

Етап 4. Формування каузальних залежностей, що визначають процес обробки кожного з артефактів. На відміну від темпоральних правил r_s^k, каузальні залежності задають причинно-наслідкові зв'язки між підмножинами властивостей артефактів предметної області.

Представлений метод дає можливість формувати суттєві для побудови пояснень контекстні умови і обмеження з використанням темпоральних та каузальних залежностей, представлених у процесах обробки артефактів предметної області. Це створює умови персоналізації пояснень для представленої у вигляді чорного ящика інтелектуальної системи з урахуванням підмножини актуальних вхідних та проміжних даних щодо процесу прийняття рішення.

7. Висновки та перспективи подальших досліджень

Виконано структуризацію контексту пояснень в інтелектуальній інформаційній системі. За результатами структуризації обґрунтовано побудову опису контексту в об'єктному та темпоральному аспектах. Об'єктний аспект охоплює підмножину об'єктів предметної області, які використовуються для формування пояснення. Темпоральний аспект визначає інтервал часу, коли дії з виділеними об'єктами щодо формування рішення є актуальними. Показано, що суттєві для пояснення знання визначають умови та обмеження у об'єктному та темпоральному аспектах щодо процесу прийняття рішень в інтелектуальній системі.

Запропоновано артефактну модель контексту пояснень. Модель містить опис артефактів предметної області та процесів обробки цих артефактів. Кожен процес обробки артефактів складається із послідовності темпоральних та каузальних залежностей між станами та властивостями артефактів відповідно. Модель дає можливість персоналізувати пояснення з урахуванням їх актуальності на основі використання комбінації явних та неявних знань, що складають умови та обмеження щодо процесу прийняття рішення в інтелектуальній системі.

Розроблено процесно-орієнтований метод побудови артефактної моделі контексту пояснень, який містить етапи формування множини артефактів, процесів обробки артефактів та

залежностей, що лежать в основі цих процесів. Метод дає можливість сформувати опис контексту з використанням не лише явних, але й неявних знань шляхом узагальнення та виявлення темпоральних залежностей для упорядкованих у часі даних, що відображають процес обробки артефактів при прийнятті рішення в інтелектуальній системі.

Подальші дослідження щодо контексту пояснень пов'язані із розробкою підходу до формування каузальних контекстних залежностей на основі темпоральних правил, що описують процеси обробки артефактів. Вирішення цієї задачі орієнтовано на екстерналізацію неявних знань щодо процесу прийняття рішень в інтелектуальній інформаційній системі.

Список літератури: 1. *Miller T.* Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence. 2019. Vol. 267. P.1-38. DOI: https://doi.org/10.1016/j.artint.2018.07.007 2. *Camburu O.-M., Giunchiglia E., Foerster J., Lukasiewicz T., Blunsom Ph.* Can I trust the explainer? Verifying post-hoc explanatory methods. arXiv:1910.02065. 2019. 3. Castelvecchi D. Can we open the black box of AI? Nature. 2016.Vol. 538 (7623). P. 20-23. 4. *Tintarev N., Masthoff J.* A survey of explanations in recommender systems. The 3rd international workshop on web personalisation, recommender systems and intelligent user interfaces (WPRSIUI'07). 2007. P. 801-810. 5. *Gunning D., Vorm E., Wang J., Turek M.* darpa's Explainableai(xai) Program: a Retrospective. Applied AI Letters. 2021. Vol. 2. No. 4. https://doi.org/10.1002/ail2.61. 6. *Pearl J.* Causality: Models, Reasoning and Inference. 2nd ed. Cambridge University Press, USA. 2009. 7. *Maier M., Marazopoulou K., Jensen D.* Reasoning about Independence in Probabilistic Models of Relational Data. arXiv. 2014. 8. *Marazopoulou K., Maier M., Jensen D.* Learning the structure of causal models with relational and temporal dependence. Proceedings of the Thirty-First Conference on Uncertainty in Articial Intelligence. 2015. 9. *Чала О.В.* Модель узагальненого представлення темпоральних знань для задач підтримки управлінських рішень. Вісник Національного технічного університету "ХПІ". Системний аналіз, управління та інформаційні технології. 2020. № 1(3). С. 14-18. DOI: 10.20998/2079-0023.2020.01.03. 10. *Chala O.* Models of temporal dependencies for a probabilistic knowledge base. Econtechmod. An International Quarterly Journal. 2008. Vol. 7. No. 3. P. 53 - 58. 11. *Чалий С.Ф.* Реляційно-темпоральна модель набору сутностей предметної області для процесу формування рішення в інтелектуальній інформаційній системі / С. Ф. Чалий, В. О. Лещинський, І. О. Лещинська. Вісник Національного технічного університету "ХПІ". Сер. : Системний аналіз, управління та інформаційні технології = Bulletin of the National Technical University "KhPI". Ser. : System analysis, control and information technology : зб. наук. пр. Харків: НТУ "ХПІ", 2022. № 1 (7). С. 84-89. 12. *Chalyi S., Leshchynskyi V.* Temporal representation of causality in the construction of explanations in intelligent systems. Advanced Information Systems. Kharkiv: NTU "KhPI". 2020. Vol. 4. № 3. P. 113-117.

Надійшла до редколегії 10.12.2022 р.

Лещинський Володимир Олександрович, канд. техн. наук, доцент, докторант кафедри ІУС ХНУРЕ. Наукові інтереси: рекомендаційні системи, самопояснювальний штучний інтелект. Адреса: 61166, Харків, пр. Науки 14, ХНУРЕ, каф. ІУС, контактний телефон: +38(057)7021451.

РЕФЕРАТИ

УДК 004.94:004.78

Синтез моделі класифікації діалогових актів на основі використання рекурентних нейронних мереж / К. Е. Петров, Є. К. Воробйов І. В. Кобзев. АСУ і прилади автоматики. 2022. № 178. С. 4-12.

Проаналізовано основні етапи та технології побудови інтелектуальних діалогових систем. Розглянуто проблему генерування діалогів у системах обробки природної мови. Запропоновано математичну модель класифікації діалогових актів, яка базується на використанні рекурентної нейронної мережі та механізму уваги. Детально описана архітектура нейронної мережі, процес її навчання і тестування. Наведено результати комп'ютерного моделювання, які демонструють практичну реалізовність та ефективність запропонованої моделі.

Табл. 1. Іл. 3. Бібліогр.: 27 назв.

UDC 004.94:004.78

Synthesis of a model for the classification of dialogue acts based on the use of recurrent neural networks / K. E. Petrov, Yev. K. Vorobiov, I. V. Kobzev. Management Information System and Devices. 2022. № 178. P. 4-12**.**

The main stages and technologies of construction of intelligent dialogue systems are analyzed. The problem of generating dialogues in natural language processing systems is considered. A mathematical model for the classification of dialog acts is proposed, which is based on the use of a recurrent neural network and the mechanism of attention. The architecture of the neural network, the process of its training and testing are detail described. The results of computer simulation, which demonstrate the practical feasibility and efficiency of the proposed model are presented.

Tab. 1. Fig. 3. Ref.: 27 items.

УДК 004.65:005.331

Оцінювання доцільності проведення рефакторингу бази даних інформаційної системи, яка експлуатується / М.В. Євланов, І.О. Юр'єв, Д.О. Мірошниченко. АСУ і прилади автоматики. 2022. № 178. С. 13-22.

Розглянуто питання вдосконалення існуючого типового процесу проведення рефакторингу бази даних інформаційної системи, яка експлуатується. Визначено основні переваги та недоліки цього процесу. Встановлено значну потребу у об'єктивному визнанні необхідності проведення рефакторингу бази даних до початку робіт з його проведення. На основі теоретико-множинного апарату розроблено об'єктивні оцінки складності актуальної бази даних та сукупності запитів на зміни інформаційної системи, яка експлуатується. Отримано показник обсягу змін, які слід внести під час рефакторингу бази даних, що пропонується.

Табл. 0. Іл. 1. Бібліогр.: 6 назв.

UDC 004.65:005.33

Assessing the expediency of refactoring the database of the operated information system / M.V. Yevlanov, I.O. Yur`ev, D.O. Miroschnichenko. Management Information System and Devices. 2022. № 178. P. 13-22.

The issue of improving the existing standard process of refactoring the database of the information system in operation is considered. The main advantages and disadvantages of this process are determined. A significant need for objective recognition of the need for database refactoring before starting all other work on its implementation has been established. On the basis of the set-theoretical apparatus, objective assessments of the complexity of the current database and the set of requests for changes to the information system in operation have been developed. An indication of the amount of changes to be made during the proposed database refactoring is obtained.

Tab. 0. Fig. 1. Ref.: 6 items.

УДК 681.5:004.94

Математична модель ациклічного нечіткого керування системою світлофорів з адаптивним безумовним пріоритетом трамваю / М. Литвиненко, Л. Ребезюк. АСУ і прилади автоматики. 2022. № 178. С. 23-32.

Розглянуто питання врахування динаміки трамваю під час його наближення до перехрестя для надання безумовного світлофорного пріоритету. Розроблено математичну модель нечіткого керування системою світлофорів перехрестя. Запропоновані правила для формування значень важливостей світлофорних сигналів для трамваю та для ухвалення рішень щодо адаптації часових параметрів світло-

форного плану на основі сумісности сигналів та їхніх важливостей. Математична модель реалізована та апробована за допомогою засобу моделювання міської мобільности SUMO для помірного та насиченого транспортних попитів на штучному перехресті.

Табл. 11. Іл. 14. Бібліогр.: 30 назв.

UDC 681.5:004.94

Mathematical model of acyclic fuzzy control for traffic signal system with adaptive unconditional tram priority/ M. Lytvynenko, L. Rebezyuk. Management Information System and Devices. 2022. № 178. P. 23-32.

The problem of considering the dynamics of the tram as it ap-proaches the intersection to provide unconditional traffic signal priority is considered. A mathematical model of fuzzy control of the intersection traffic signals system has been developed. Rules have been proposed for the inference of traffic signal importance values for a tram vehicle and for decision on the adaptation of the traffic signal plan time parameters based on the compatibility of signals and their importance. Mathematical model implemented and validated using SUMO urban mobility simulation tool for moderate and saturated transport demands on artificial intersection.

Tab. 11. Fig. 14. Ref .: 30 items.

УДК 004.8:004.032.26

Адаптивна кластеризація багатоекстремальних масивів даних з використанням модифікованого алгоритму риб'ячої зграї / А.Ю. Шафроненко, Є.В. Бодянський. АСУ і прилади автоматики. 2022. № 178. С. 33-37.

Розглянуто задачу кластеризації багатоекстремальних масивів даних. Для оптимізації функцій пошуку локальних екстремумів запропоновано алгоритм, що є по суті оптимізаційною функцією модифікованого алгоритму риб'ячої зграї (Fish School Search), випадкового пошуку та еволюційної оптимізації.

Табл. 0. Іл. 0. Бібліогр. 23 назв.

UDC 004.8:004.032.26

Adaptive clustering of multiextrema data arrays using a modified fish school algorithm / A. Shafronenko, Ye. Bodyanskiy. Management Information System and Devices. 2022. № 178. P. 33-37

The problem of clustering multiextrema data arrays is considered. To optimize the local extrema search functions, an algorithm is proposed, that is essentially an optimization function of the modified Fish School Search algorithm, random search, and evolutionary optimization.

Tab. 0. Fig. 0. Ref .: 23 items.

УДК 004.65; 004.91

Дослідження використання методів ієрархічної кластеризації під час вирішення задачі аналізу конфігурації ІТ-продукту / Н.В. Васильцова, І.Ю. Панфьорова. АСУ і прилади автоматики. 2022. № 178. С. 37-49.

Розглянуто основні особливості існуючих способів рішення задачі аналізу конфігурації ІТ-продукту. Виділено основні недоліки цих способів. Запропоновано розділити задачу аналізу конфігурації ІТ-продукту на дві підзадачі. Розглянуто рішення підзадачі формування множини варіантів декомпозиції опису архітектури системи на окремі функціональні конфігураційні елементи з використанням дивізимного та агломеративного алгоритмів. Проведено порівняльний аналіз особливостей використання ієрархічних алгоритмів кластеризації для вирішення даної підзадачі. Запропоновано модифікацію алгоритму найближчого сусіда, яка дозволяє своєчасно виявляти конфігураційні елементи з повністю співпадаючими описами.

Табл. 12. Іл. 2. Бібліогр.: 12 назв.

UDC 004.65; 004.91

Research on the use of hierarchical clustering methods when solving the task of IT product configuration analysis / N.V. Vasiltcova, I.Yu. Panforova. Management Information System and Devices. 2022. № 178. P. 37-49.

The main features of existing methods for solving the problem of IT product configuration analyzing are considered. The main disadvantages of these methods are highlighted. It is proposed to divide the task of IT product configuration analyzing into two subtasks. Solutions to the subtask of forming a set of options for decomposing the description of the system architecture into separate functional configuration elements using divisive and agglomerative algorithms are considered. A comparative analysis of the application features of hierarchical clustering algorithms for solving this subtask is carried out. A modification of the nearest neighbor algorithm is proposed, which allows timely detection of configuration elements with completely matching descriptions.

Tab. 12. Fig. 2. Ref.: 12 items.

УДК 004.738

Мікросервісна архітектура системи потокової обробки великих даних / Д.А. Нефьодов, С.Г. Удовенко, Л.Е. Чала. АСУ і прилади автоматики. 2022. № 178. С. 50-64.

Розглянуто питання використання мікросервісної архітектури у системах потокової обробки великих даних. Досліджено переваги і недоліки існуючих архітектур аналізу великих даних. Запропоновано варіант системи мікросервісної обробки великих даних з використанням розподіленої потокової платформи подій Kafka, платформи розробки та запуску програм Docker, об'єктно-реляційної системи управління базами даних Postgres, а також веб-платформи для створення додатків FastAPI. Розроблено архітектурні шаблони, які можуть спростити розробку застосунків для обробки великих даних з використанням мікросервісів, та концепти програм з використанням отриманих шаблонів. Наведено результати моделювання, які свідчать про те, що мікросервісна архітектура з розподіленим навантаженням на декілька реплік може забезпечити кращу масштабованість і швидкодію в порівнянні з монолітною архітектурою.

Табл. 1. Іл. 13. Бібліогр.: 12 назв.

UDC 004.738

Microservice architecture of the big data stream processing system / D.A. Nefodov, S.G. Udovenko, L.E. Chala . Management Information System and Devices. 2022. № 178. P. 50-64.

The issue of using microservice architecture in big data stream processing systems is considered. Advantages and disadvantages of existing big data analysis architectures are studied. A version of the microservice system for big data processing using the Kafka distributed event streaming platform, the Docker program development and launch platform, the Postgres object-relational database management system, and the FastAPI web platform for creating applications is proposed. Architectural patterns have been developed that can simplify the development of large data applications using microservices, and application concepts using the resulting patterns. Simulation results are presented, which show that a microservice architecture with distributed load on several replicas can provide better scalability and speed compared to a monolithic architecture.

Tab. 1. Fig. 13. Ref.: 12 items.

УДК 004.02:005.2

Модифікація методу оцінювання трудовитрат ІТ-проєкту з розробки мобільного додатку / В.В. Ярмак. АСУ і прилади автоматики. 2022. № 178. С. 64-69.

Розглянуто питання вдосконалення існуючих методів оцінювання трудовитрат ІТ-проєктів з розробки мобільних додатків. Встановлено основні інтуїтивні та параметричні методи оцінювання трудовитрат ІТ-проєктів, визначено їх основні недоліки. Як похідні методи оцінювання трудовитрат обрано інтуїтивні методи "Maximum Size or Less", "Big/Small/Uncertain" та "Ordering Rule". З врахуванням їх недоліків та переваг розроблено модифікований метод секторів. Визначено основні відмінності розробленого методу від похідних методів оцінювання.

Табл. 2. Іл. 4. Бібліогр.: 4 назв.

UDC 004.02:005.2

Modification of the estimating labor costs method of an IT project for the development of a mobile application / V.V. Yarmak. Management Information System and Devices. 2022. № 178. P. 64-69.

The issue of improving the existing methods of labor costs estimating of IT projects for the development of mobile applications is considered. The main intuitive and parametric methods of labor costs estimating of IT projects are established, and their main shortcomings are identified. The intuitive methods "Maximum Size or Less", "Big/Small/Uncertain" and "Ordering Rule" were chosen as derivative methods of labor costs estimating. Taking into account their disadvantages and advantages, a modified method of sectors was developed. The main differences between the developed method and the derivative estimating methods are determined.

Tab. 2. Fig. 4. Ref.: 4 items.

УДК 004.89

Процесний підхід до побудови моделі контексту пояснень в інтелектуальній інформаційній системі / В.О. Лещинський. АСУ і прилади автоматики. . 2022. № 178. С. 70-76.

Виконано структуризацію контексту пояснень в інтелектуальній інформаційній системі з урахування об'єктного та темпорального аспектів. Показано, що суттєві для пояснення знання визначають умови та обмеження у об'єктному та темпоральному аспектах щодо процесу прийняття рішень в інтелектуальній системі. Запропоновано артефактну модель контексту пояснень, яка містить опис артефактів предметної області та процесів обробки цих артефактів, що дає можливість персоналізувати пояснення з урахуванням їх актуальності на основі використання комбінації явних та неявних знань, які складають умови та обмеження щодо процесу прийняття рішення в інтелектуальній системі. Розроблено процесно-орієнтований метод побудови артефактної моделі контексту пояснень, який містить етапи формування множини артефактів, процесів обробки артефактів та залежностей, що лежать в основі цих процесів. Метод дає можливість сформувати опис контексту з використанням не лише явних, але й неявних знань.

Табл. 1. Іл. 2. Бібліогр.: 12 назв.

UDC 004.89

A process approach to building a model of the context of explanations in an intelligent information system / V.O. Leshchynskyi. Management Information System and Devices. 2022. № 178. P. 70-76.

The structuring of the context of explanations in the intellectual information system was carried out, considering the object and temporal aspects. It is shown that the knowledge essential for explanation determines the conditions and limitations in object and temporal aspects regarding the decision-making process in the intellectual system. An artifactual model of the context of explanations is proposed, which contains a description of the artifacts of the subject area and the processes of processing these artifacts, which makes it possible to personalize explanations considering their relevance based on the use of a combination of explicit and implicit knowledge, which make up conditions and restrictions regarding the decision-making process in an intelligent system. A process-oriented method of building an artifact model of the context of explanations has been developed, which includes the stages of forming a set of artifacts, artifact processing processes, and the dependencies underlying these processes. The method makes it possible to form a description of the context using explicit and implicit knowledge.

Tab. 1. Fig. 2. Ref.: 12 items.

УВАГА!

Починаючи з випуску № 179, правила оформлення статей для збірника «АСУ і прилади автоматики» змінюються!

ПРАВИЛА ОФОРМЛЕННЯ СТАТЕЙ

для всеукраїнського міжвідомчого науково-технічного збірника

"Автоматизовані системи управління і прилади автоматики"

1. Загальні вимоги

До розгляду приймаються раніше не опубліковані статті українською та англійською мовами. Статті англійською мовою подаються разом з українськомовним варіантом. Статті, перекладені англійською за допомогою комп'ютерного перекладача та не відредаговані належним чином, не розглядаються.

Наукова стаття, яка подається до розгляду, має бути структурована та містити всі основні частини, характерні для наукової статті:

- постановка проблеми у загальному вигляді та її зв'язок з важливими науковими та практичними задачами;
- аналіз останніх досліджень та публікацій, у яких розпочато вирішення даної проблеми та на які спирається автор, виділення невирішених раніше частин загальної проблеми;
- формулювання цілей статті (постановка задачі);
- подання основного матеріалу досліджень з повним обґрунтуванням отриманих результатів;
- висновки даного дослідження та перспективи подальших досліджень у даному напрямку;
- перелік посилань (References).

2. Вимоги до структури рукопису

Структурно матеріали статті поділяються на такі елементи:

- УДК;
- прізвища та ініціали авторів статті;
- заголовок статті;
- анотація до статті;
- основний текст статті;
- перелік посилань;
- дата надходження статті до редколегії збірника;
- відомості про авторів статті;
- реферати українською та англійською мовами.

Бажаний порядок та зміст розділів основного тексту статті:

а) розділ 1 "Вступ", в якому визначається проблема у загальному вигляді та її зв'язок з важливими науковими та практичними задачами;

б) розділ 2 "Аналіз літературних джерел та визначення проблеми дослідження", в якому наводяться результати аналізу останніх досліджень та публікацій, де розпочато вирішення даної проблеми та на які спирається автор, виділяються невирішені раніше частини загальної проблеми дослідження та конкретизується головна проблема дослідження у даній статті;

в) розділ 3 "Мета і задачі дослідження", в якому наводяться описи мети дослідження та задач дослідження, вирішення яких дозволяє досягнути визначену раніше мету дослідження;

г) розділ 4 "Матеріали і методи дослідження", в якому наводяться описи формального апарату та раніше проведених експериментальних досліджень, які будуть використані у подальшому тексті статті;

д) розділ 5 "Результаті дослідження", в якому структуровано наводяться результати вирішення сформульованих у розділі 4 окремих задач дослідження (теоретичних та експериментальних);

е) розділ 6 "Обговорення результатів дослідження", в якому наводяться: опис особливостей отриманих результатів дослідження та їх відмінності від результатів попередніх досл-

іджень у відповідній галузі; опис переваг отриманих результатів перед існуючими; опис недоліків і обмежень, які утруднюють використання отриманих результатів дослідження; опис подальших перспектив проведення досліджень за цим напрямом;

ж) розділ 7 "Висновки", в якому наводяться стислі описи отриманих результатів вирішення окремих задач дослідження та загальний висновок про досягнення поставленої у розділі 3 мети дослідження.

Заголовки окремих розділів основного тексту статті можуть змінюватися відповідно до змісту конкретної статті.

Розділи основного тексту статті, перелік посилань, дата надходження статті до редколегії збірника та відомості про авторів статті відокремлюються один від одного одним порожнім рядком.

3. Вимоги до оформлення рукопису

До розгляду приймаються матеріали статей обсягом не менше 5 повних сторінок (з урахуванням рисунків і таблиць).

Матеріали статті повинні бути набраними у редакторі MS Word. Припустимі формати файлу з матеріалами статті - .doc або .docx.

Формат сторінки - А4 (210х297 мм). Поля знизу, зверху, справа, зліва - 3 см.

Основний текст статті набирається шрифтом Times New Roman, кегль 11, інтервал - 1,1, абзацний відступ 8 мм, інтервали перед і після - 0 мм, вирівнювання по ширині.

Для УДК - шрифт Times New Roman, кегль 11, інтервал - 1,1, абзацний відступ 8 мм, інтервал перед - 0 мм, інтервал після - 6 мм, вирівнювання по ширині.

Для прізвищ та ініціалів авторів статті - шрифт Times New Roman, кегль 11, інтервал - 1,1, абзацний відступ 8 мм, інтервали перед і після - 6 мм, вирівнювання по ширині.

Для заголовка статті - шрифт Times New Roman, кегль 11, напівжирний, інтервал - 1,1, абзацний відступ 8 мм, інтервали перед і після - 6 мм, вирівнювання по ширині.

Для анотації - шрифт Times New Roman, кегль 10, інтервал - 1,1, відступ зліва - 0,8 см, абзацний відступ 8 мм, інтервал перед - 6 мм, інтервал після - 0 мм, вирівнювання по ширині.

Для заголовків таблиць - шрифт Times New Roman, кегль 10, інтервал - 1,1, абзацного відступу немає, інтервали перед і після - 0 мм, слово "Таблиця" та її номер з вирівнюванням вправо, назву таблиці (якщо вона є) з вирівнюванням по центру.

Для підрисункових підписів - шрифт Times New Roman, кегль 10, інтервал - 1,1, абзацного відступу немає, інтервали перед і після - 0 мм, вирівнювання по центру.

Для переліку посилань та відомостей про авторів - шрифт Times New Roman, кегль 9, інтервал - 1,1, абзацний відступ 8 мм, інтервали перед і після - 0 мм, вирівнювання по ширині.

Для рефератів - шрифт Times New Roman, кегль 10, інтервал - 1,1, абзацний відступ 8 мм, інтервали перед і після - 0 мм, вирівнювання по ширині.

Формули набираються у редакторі формул Microsoft Equation або MathType, розташовуються у центрі робочого поля, нумерація - з правої сторони поля. Для цього необхідно весь рядок розташувати справа, а потім вирівняти формулу табуляціями так, щоб вона розташовувалася по центру. Відступ зверху і знизу по 6 пунктів. Нумерація формул всередині кожної статті наскрізна.

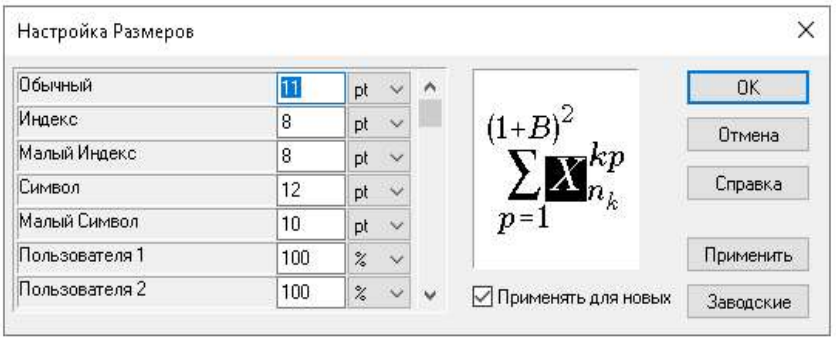

Рис. 1. Параметри настроювання розмірів редактора формул MathType

Формули, а також їх складові, які присутні у тексті, набираються з такими параметрами (див. рис. 1).

Кожна таблиця виконується та розташовується в тексті одразу після посилання на неї. Усі таблиці у статті обов'язково нумеруються, незважаючи на їх кількість.

Таблиця відокремлюється від попереднього та наступного тексту (таблиці, рисунка тощо) одним порожнім рядком.

Дані всієї таблиці набираються шрифтом розміром 10 пунктів, розміщуються по центру; у випадках, коли необхідно показати розрядність - вирівнювання за знаком. Товщина сітки таблиці 1 пункт. Приклад оформлення таблиці наведено на рис. 2.

Таблиця 1

Множина описів сутностей функціональної задачі

ID	Найменування
1	Academic_load
2	Academic
3	Department
4	Individual_plan
5	Academic_section

Рис. 2. Приклад оформлення таблиці у тексті статті

Бажано таблицю зі сторінки на сторінку не переносити. Якщо таблиця не може розміститься на сторінці, її поділяють на частини. У кожній частині таблиці повторюють її головку та боковик або замінюють їх відповідно номерами колонок або рядків, нумеруючи їх арабськими цифрами на першій частині таблиці. Слово "Таблиця" подається лише над першою її частиною. Над наступними її частинами праворуч друкується: "Продовження таблиці", а на останній - "Кінець таблиці", в усіх випадках вказується номер таблиці.

Кожен рисунок виконується та розташовується в тексті одразу після посилання на нього. Усі рисунки у статті обов'язково нумеруються, незважаючи на їх кількість. Необхідно вставляти рисунки у текст як графічні об'єкти (файли з розширенням .bmp, .jpg, .tiff чи .png, якість не менше 300 dpi), об'єкти MS Word або MS Visio.

Рисунок відокремлюється від попереднього та наступного тексту (таблиці, рисунка тощо) одним порожнім рядком.

Кожен рисунок повинен мати підрисунковий підпис, в якому вказується номер та, у випадку необхідності, назва рисунка. Якщо рисунок займає менше 50% ширини робочого поля, то можна зробити обтікання рисунку текстом, розташувавши його ліворуч або праворуч від робочого поля. Приклад рисунка з підписом наведений на рис. 3.

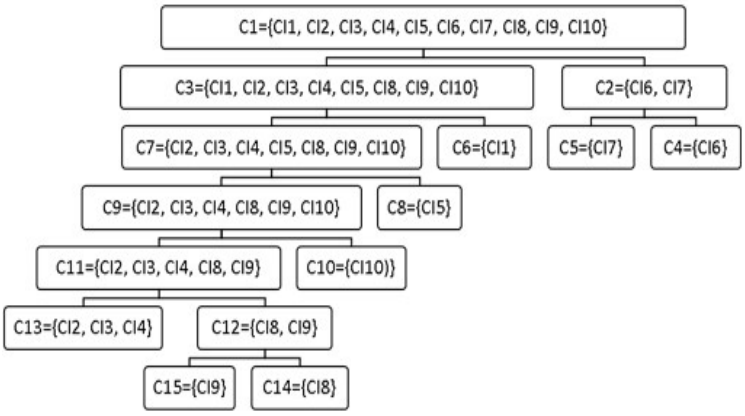

Рис. 3. Приклад виконання рисунку та підрисункового підпису

Посилання на літературні та електронні джерела у тексті статті позначаються у квадратних дужках [1]. До переліку посилань включаються тільки ті роботи, на які посилається автор статті. Посилання на неопубліковані роботи не допускаються.

Для оформлення переліку посилань слід використовувати один з таких шаблонів:

а) шаблон IEEE (автоматичне оформлення за шаблоном IEEE https://www.citethisforme.com/ieee/source-type);

б) положення ДСТУ 8302:2015 "Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання" та ДСТУ 3582:2013 "Інформація та документація. Бібліографічний опис. Скорочення слів і словосполучень українською мовою. Загальні вимоги та правила".

Кожен з цих шаблонів слід використовувати для оформлення усіх елементів переліку посилань. Використання двох шаблонів для оформлення одного й того ж переліку посилань неприпустимо.

Кожне посилання у переліку посилань наводиться за порядком появи цих посилань у тексті статті.

У переліку посилань бажано використовувати посилання на сучасні публікації, вік яких не перевищує п'яти років у момент подачі статті до редакції. Крім того, під час формування переліку посилань статті необхідно дотримуватися такого розподілу: самоцитування до 20 %, цитування зарубіжних публікацій не менше 50 %.

Відомості про авторів слід наводити українською та англійською мовами. У відомості про авторів слід включати: повні прізвище, ім'я та по-батькові; вчений ступінь (за наявності); вчене звання (за наявності); посаду; місто; країну; e-mail (вкрай бажано вказувати корпоративний e-mail, можна вказувати кілька e-mail, на які ви бажаєте отримувати повідомлення від редакції та читачів, які можуть зацікавитися вашою статтею); ORCID.

Реферат набирається українською та англійською мовами. Реферат повинен бути змістовним, дотримуватися логіки опису результатів у статті та давати можливість встановити її основний зміст. Реферат не повинен містити формул та рисунків. Необхідні символи в рефераті необхідно додавати через функцію вставки символів.

Реферат містить: УДК, назву статті (напівжирним шрифтом), ініціали та прізвища авторів (курсивом), текст (не менше 1800 друкованих знаків з пробілами та ключовими словами), ключові слова, кількість таблиць, рисунків та посилань у статті.

Ключові слова повинні містити до 10 слів, а не словосполучень, без використання абревіатур, в іменному відмінку, розділятися крапкою з комою.

Реферати надаються до редколегії разом із статтею у вигляді окремого файлу.

4. Правила надсилання статей та подальшої взаємодії з редакційною колегією збірника

До редколегії збірника "АСУ і прилади автоматики" слід надсилати такі матеріали:

- файл у форматі .doc або .docx з текстом статті українською мовою;
- файл у форматі .doc або .docx з текстом статті англійською мовою (якщо автори бажають опублікувати статтю у збірнику англійською мовою);
- файл у форматі .doc або .docx з текстами рефератів статті українською та англійською мовами;
- відскановану копію експертного висновку з дозволом опублікувати матеріали статті у відкритому друку. В разі потреби експертні висновки для авторів - співробітників (студентів, аспірантів тощо) ХНУРЕ можуть оформлюватися редколегією централізовано.

Матеріали статей надсилати електронною поштою - за адресою misd@nure.ua.

Кожна надіслана в редакцію стаття після проходження рецензування і при позитивному рішенні редколегії буде надрукована в найближчому випуску збірника. Для цього авторам від імені редколегії надсилається ліцензійний договір, який закріплює право першої публікації статті у збірнику "АСУ і прилади автоматики". Автори статті повинні підписати цей ліцензійний договір та завірити свої підписи печаткою організації, в якій працюють автори. Підписаний ліцензійний договір автори статті надіслають на адресу редколегії збірника.

Відповідальний випусковий В.М. Левикін
Редактор О.Є. Неумивакіна
Комп'ютерна верстка М.В. Євланов, О.Є. Неумивакіна
Дизайн обкладинки номера за участі Є. Чех

Підп. до друку 23.12.2022. Формат 60x841/8. Умов. друк. арк. .
Обл.-вид. арк. 9,6. Тираж 300 прим.
Зам. № . Ціна договірна.

Харківський національний університет радіоелектроніки (ХНУРЕ).
Україна, 61166, Харків, просп. Науки, 14.

Оригінал-макет підготовлено у редакційно-видавничому відділі ХНУРЕ.
Україна, 61166, Харків, просп. Науки, 14.

Збірник віддруковано в ТОВ «ДРУКАРНЯ МАДРИД»
61024, м. Харків, вул. Гуданова, 18
Тел.: +38(057) 7565325
Свідоцтво суб'єкта видавничої справи
Серія ДК № 4399 від 27.08.2012 р.
www.madrid.in.ua e-mail:info@madrid.in.ua