НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ISSN 1727-4907

П Р О Б Л Е М И ПРОГРАМУВАННЯ

НАУКОВИЙ ЖУРНАЛ

P R O B L E M S IN PROGRAMMING

SCIENTIFIC JOURNAL

2020

№ 4

Теми випуску:

- ***Інструментальні засоби та середовища програмування***
- ***Моделі та засоби систем баз даних і знань***
- ***Моделі та методи машинного навчання***
- ***Теоретичні і методологічні основи програмування***

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

П Р О Б Л Е М И ПРОГРАМУВАННЯ

науковий журнал

Головний редактор
Андон Пилип Іларіонович
академік НАН України,
директор Інституту програмних систем
НАН України
✉ Інститут програмних систем
НАН України
проспект Академіка Глушкова, 40, корп. 5
03187, Київ-187
☎ Тел.+380 (44) 526 5507
E-mail: andon@isofts.kiev.ua
http://www.pp.isofts.kiev.ua

Адреса для кореспонденції

✉ Інститут програмних систем
НАН України
Проспект Академіка Глушкова, 40
03187, Київ-187

☎ Тел.: +380 (44) 526 5065
Факс: +380 (44) 526 6263
E-mail: iss@isofts.kiev.ua

Затверджено до друку вченою радою Інституту програмних систем НАН України.
Протокол № 10 від 05.11.2020 р.

Редактор *В.П. Замула*
Комп'ютерна верстка *В.П. Замула*

Підписано до друку 05.11.2020. Формат 60x84/8. Папір офс. Ум. друк. арк. 10,98.
Обл.-вид. арк. 10,41. Тираж 120 прим. Ціна договірна. Замовл.

Віддруковано ВД «Академперіодика» НАН України
вул. Терещенківська, 4, м. Київ, 01004
Свідоцтво суб'єкта видавничої справи ДК № 544 від 27.07.2001

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 4 жовтень – грудень 2020

Заснований у березні 1999 р.

ЗМІСТ

Свідоцтво про державну реєстрацію КВ № 7490 від 01.07.2003

Науковий журнал “Проблеми програмування” занесений до переліку наукових фахових видань України, в яких можуть публікуватися основні результати дисертаційних робіт.

ISSN 1727-4907

NATIONAL ACADEMY OF SCIENCES OF UKRAINE
INSTITUTE OF SOFTWARE SYSTEMS

PROBLEMS IN PROGRAMMING

scientific journal

№ 4 ***October – December*** ***2020***

Founded in March, 1999

CONTENTS

УДК 612.51.001.57+519.6 https://doi.org/10.15407/pp2020.04.003

Р.Д. Григорян, О.І. Юрчак, А.Г. Дегода, Т.В. Людовик

ПРОГРАМНА ТЕХНОЛОГІЯ ДЛЯ ПІДТРИМКИ ПРОЦЕДУР НАЛАШТУВАННЯ КІЛЬКІСНИХ МОДЕЛЕЙ ГЕМОДИНАМІКИ ЛЮДИНИ

Математичне моделювання та спеціалізовані програмні симулятори (СПС), які засновані на кількісних моделях, є сучасними інструментами дослідження, що розширюють можливості вивчення фізіології людини. Однак більшість моделей з зосередженими параметрами (МЗП) серцево-судинної системи (ССС) не включають складних механізмів, що забезпечують загальний контроль кровообігу людини. Щоб заповнити цю прогалину, ми запропонували три спеціальні моделі та моделювальну концепцію для їх функціональної інтеграції. Для забезпечення ефективних процедур налаштування МЗП пропонується спеціальне програмне забезпечення, що містить автономні модулі для розв'язування рівнянь кожної моделі в відомих умовах вхідних навантажень. Враховуючи три основні блоки моделей ССС, програма забезпечує кількісні процедури їх налаштування. Завдяки СПС розробник моделі може вручну встановлювати значення 92 параметрів 23 відсіків ССС та константи чутливості кожного фізіологічного механізму. Спеціальні інструменти налаштування дозволяють розробнику моделі імітувати певну кількість тестів і будувати графіки гемодинамічних реакцій на обраний тест.

Ключові слова: серцево-судинна система, моделювання, програмні технології, симулятор.

Вступ

Кількісні математичні моделі і спеціалізовані програмні комплекси для комп'ютерних симуляцій фізіологічних функцій і систем людини давно вже використовуються в учбових процесах та в дослідженнях [1, 2]. В останньому випадку одним з важливих аспектів симуляцій є отримання додаткової інформації про об'єкт, що моделюється. Така інформація дає новий поштовх удосконаленню існуючих концепцій біології та медицини. Саме в цьому напрямку йдуть наші розробки [3–6]. За останні роки основним об'єктом моделювання була серцево-судинна система (ССС) з урахуванням усіх відомих фізіологічних механізмів, що так чи інакше впливають на її стан та модулюють гемодинаміку [6–9].

Розроблення комп'ютерного симулятора (спеціальної автономної програми) ССС є комплексною науково-технічною проблемою, яка містить шість розділів: 1) вибір і обґрунтування фізіологічної концепції; 2) створення інформаційної моделі обраної концепції; 3) створення математичних моделей обраної концепції; 4) розроблення алгоритмів та програм для чисельного вирішення систем рівнянь моделі; 5) розроблення інтерфейсу для ефективної процедури налаштування значень констант моделей; 6) розроблення інтерфейсу користувача для забезпечення тестування симулятора та для інтерактивних взаємодій користувача з симулятором з метою проведення комп'ютерних імітаційних експериментів та досліджень. Три етапи розробок вже пройдено та опубліковано [7–9].

Публікація має за мету ознайомити читачів з проблемами, які були вирішені при створенні спеціалізованого інтерфейсу для налаштування базових математичних моделей.

Корисна інформація про об'єкт моделювання

Як транспортна система ССС людини – це загальний ефектор, за допомогою якого організм вирішує основні проблеми життєзабезпечення клітин в умовах широких змін темпів обміну речовин (метаболізму). Для забезпечення фізіологічних значень току крові мають бути спеціальні ендогенні механізми (їх звуть *регуляторами гемодинаміки*), які адаптують насосну функцію серця, тиск крові в артеріях та провідність регіональних судин до актуальних потреб клітин. Певна кількість таких механізмів фізіологам відома. Питання у тому, чи є кожний механізм, збережений

еволюцією, обов'язково самостійним і корисним фізіологічним регулятором. Відомо, що декілька з таких механізмів іноді грають роль провокаторів певних патологій (наприклад, артеріальної гіпертонії [10, 11]). Проблема в тому, що ні існуючі методи реєстрації біометричних даних людини, ні інвазійні фізіологічні експерименти на тваринах не дозволяють однозначно визначити призначення окремого ендогенного механізму. Є підстави вважати, що окремі біохімічні або фізіологічні механізми є скоріше лише такими еволюційними нагромадженнями, користь від яких умовна. Математичне моделювання та спеціальні комп'ютерні симулятори можуть вносити свою лепту в з'ясування цих проблем. Але майже усі відомі моделі (наприклад, [1, 3–5]) описують лише частину механізмів, що модулюють гемодинаміку. Для заповнення цієї прогалини нами розробляється комплексний підхід до моделювання гемодинаміки людини. Вже розроблено три автономні моделі. Модель [7] описує гемодинаміку в ССС, яка не має впливів ззовні та враховує лише саморегуляцію серця та судин в умовах стабільного обсягу крові в ССС. Модель [8] враховує вплив основних короткострокових механізмів на гемодинаміку. Модель [9] описує адитивні довгострокові гемодинамічні ефекти основних нейрогуморальних механізмів. Ці три моделі, що містять різні аспекти кровообігу людини з урахуванням певних екзогенних або ендогенних дозованих впливів на стан ССС, формують основу для створення спеціалізованого програмного симулятора (СПС). Але на цьому шляху є дві проблеми – кінцева та етапна. Кінцева проблема полягає у тому, щоб забезпечити користувача зручним інтерфейсом, за допомогою якого можна обирати та реалізовувати сценарії комп'ютерних симуляцій реагування ССС на ендогенні або екзогенні подразники, а також оформляти результати в зручному для аналізу вигляді. Етапна проблема полягає у тому, що кожна модель має багато параметрів, числові значення яких невідомі. Для їх з'ясування треба проводити багато допоміжних розрахунків. Фактично йдеться про налаштування оптимального набору констант моделей. Саме для ефективного вирішення цієї проблеми потрібно створити певні допоміжні програми.

Концепція моделювання та необхідність налаштувань моделей

Є базова модель ССС, значення параметрів та змінних якої відповідають середньому чоловіку зростом 175 см і масою тіла 75 кг. Загальний об'єм крові розраховується як 7 % від маси тіла. Кожна ділянка ССС розташована на певній дистанції (у сидячому положенні вони зменшуються на довжину стегон) від стоп. Ці показники служать для розрахунків гідростатичних тисків крові у різних позах людини.

Базою для розрахунків локальних характеристик гемодинаміки в кожній з 21 ділянки судин служать середні значення тисків крові в дузі аорти $P_A(t)$, в порожнистих венах $P_V(t)$, середній сталий потік крові $Q(t)$ та загальний периферичний опір судин $R(t)$ при $t = 0$:

$$P_A(0) \approx P_{Ad}(0) + (P_{Ac}(0) - P_{Ad}(0))/3.$$

де P_{Ad} і P_{Ac} це діастолічний та систолічний пики артеріального тиску. В нормі прийнято $R(0) = 1$, $P_A(0) = 94$ мм рт. ст., $Q(0) = 90$ мл/сек, тому

$$P_V(0) = P_A(0) - R(0) \cdot Q(0) = 4\,.$$

Початкові значення тисків у кожній ділянці артерій або вен встановлено з урахуванням відомих даних літератури про регіональні розподіли потоків та об'ємів крові в стані спокою. Далі правила встановлення констант моделей не формалізовано, вони є результатом досвіду створення багатьох версій кількісних моделей гемодинаміки здорової людини.

Фактично, цей досвід підказує певні процедури індивідуалізації базової моделі ССС. В нашому випадку персональні налаштування стосуються майже 250 характеристик модельованої ССС; в даній статті цей аспект не розглядатимемо.

Концептуальна схема комплексної моделі гемодинаміки побудована таким

чином, що передбачаються дослідження як саморегуляторних характеристик некерованої ССС (вона є об'єктом регулювання), так і гемодинамічних ефектів вмикання певного набору регуляторів (рис. 1).

Рис. 1. Схема комплексної моделі гемодинаміки з урахуванням динамічних ефектів багатьох фізіологічних механізмів (регуляторів). За відсутності регуляторів функціонує некерована ССС (об'єкт регулювання)

Базова модель об'єкта регулювання [6] нещодавно була модифікована таким чином, що вже можна симулювати вплив пози людини на гемодинаміку. Ця додаткова опція з'явилася завдяки включенню в формулу розрахунків току крові $q_{ij}(t)$ гідростатичного тиску крові. Отже, між суміжними ділянками судин $q_{ij}(t)$ розраховується як

$$q_{ij}(t) = (P_i(t) - P_j(t) + 0,735 \cdot \Delta h_{ij}) / R_{ij}(t),$$

де P – тиск крові, R – гідравлічний опір, а Δh_{ij} – різниця висот ділянок. Δh_{ij} залежить від пози людини: в лежачому положенні $\Delta h_{ij} = 0$, в сидячому або стоячому – задано фіксовані числа.

Блок-схема моделі ССС показана на рис. 2.

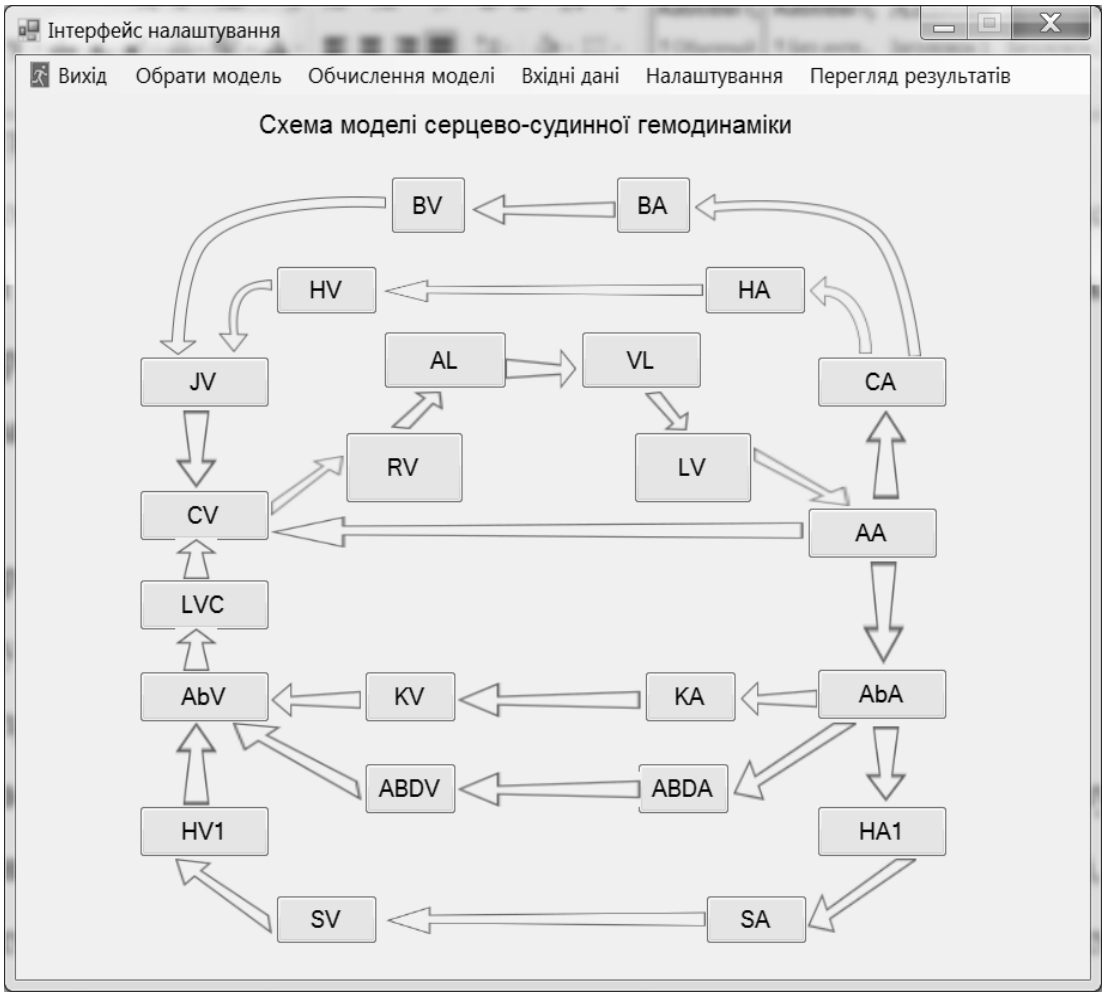

Рис. 2. Схема представлення ССС в моделі у вигляді двох шлуночків серця (RA, RV), 21 ділянки судин із зосередженими параметрами та з'єднаннями між ними

Відомості про програму

Програмний комплекс забезпечення налаштування моделі ССС розроблений з використанням інтегрального середовища розробок (IDE) Microsoft Visual Studio 2019 Community на базі об'єктно-орієнтованої мови програмування C# (Сі Шарп) і платформи .NET.

Програмний комплекс розділений на три частини:

- інтерфейсна частина;
- розрахунковий блок;
- засоби збереження та візуалізації отриманих результатів.

Інтерфейсна частина. Основним вікном інтерфейсної частини, яке також містить меню керування роботою програміста з моделлю (рис. 2), є вікно, в якому зображено блок-схему моделі ССС .

Інтерфейс забезпечує:

- обрання типу моделі (нерегульована або регульована);
- введення та зміну вхідних значень параметрів для кожного фізіологічного вузла схеми моделі ССС (рис. 2);
- запуск розрахунку моделі;
- введення та зміну вхідних значень параметрів для кожного фізіологічного вузла схеми моделі ССС із зазначенням пози модельованої людини (лежить, сидить, стоїть);
- додаткові налаштування (експозиція, крок і т. д.);
- збереження та перегляд результатів розрахунку моделі.

Передбачено можливість запам'ятовування введених даних у рамках сеансу роботи. Результати можна записувати в файл для використання в наступних сеансах роботи.

Розрахунковий блок є універсальним для обох типів моделі (нерегульованої та регульованої) і реалізує обчислення в залежності від обраного типу моделі, що задається через інтерфейс. Розрахунок ведеться згідно з фізіологічними вузлами, зазначеними в блок-схемі ССС з окремим набором вхідних даних і параметрів для кожного вузла.

Розрахунковий блок реалізований в окремому незалежному класі і включає множину методів, що реалізують обчислення за формулами, які лежать в основі моделі. Це забезпечує гнучкість в подальших модифікаціях моделі.

Результати обчислень запам'ятовуються в динамічних структурах даних, призначених для зберігання великої кількості обчислених параметрів моделі (мільйони), що дозволяє ефективно використовувати оперативну пам'ять.

Збереження отриманих результатів. Забезпечено можливість збереження отриманих результатів розрахунку моделі в текстовому файлі для подальшого аналізу. У зв'язку з великим обсягом обчислених даних передбачена їх фільтрація для збереження в файлі. Є можливість записувати потрібну кількість значень даних або загалом, або вибірково (діалогове вікно "Параметри для запису в файл").

Перегляд отриманих результатів. Основна форма перегляду отриманих результатів – це графіки. Допоміжна форма – запис числових значень змінних у файл – використовувалася для пошуку похибок у логіці або реалізації програмних модулей.

Передбачено п’ять вікон видачі графіків за допомогою меню "Перегляд результатів": "Центральна гемодинаміка", "Тиск та ЧСС", "Нервова діяльність", "Сумарні об’єми", "Кровотоки". В кожному вікні певна множина графіків, що візуалізують значення обчислених параметрів. Графіки будуються по 30 точкам на зазначеній шкалі часу з автоматичним масштабуванням по осі значень параметрів. У шапці вікна графіків вказується тип моделі та поза модельованої людини.

Проблеми налаштування моделей

Слід зазначити, що головна проблема налаштування базової моделі полягає у тому, що в кожній ділянці судин її природна нелінійна залежність тиску крові $P_i(t)$ від кількості крові $V_i(t)$, жорсткості $D_i(t)$ та ненапруженого об'єму $U_i(t)$ апроксимується за допомогою трьох лінійних ділянок вигляду:

$$P_i(t)=\begin{cases}(V_i(t)-U_i(t))\cdot 5D_i(t), V_i(t)<U_i(t)\\ (V_i(t)-U_i(t))\cdot D_i(t),\\ U_i(t)\le V_i(t)\le 1,4U_i(t)\\ 0,4D_i(t)+(V_i(t)-1,4U_i(t))\cdot 3D_i(t),\\ V_i(t)>1,4U_i(t).\end{cases}$$

Отже, в початковому стані кожна ділянка ССС має три автономні параметри $V_i(t)$, $U_i(t)$, $D_i(t)$ та додатковий параметр $R_i(t)$, який відповідає гідравлічному опору з'єднання суміжних ділянок. Процедура налаштування базової моделі для лежачої людини має мету знайти такий набір наведених параметрів, при якому $P_i(t)$ та потоки крові між ділянками будуть максимально наближеними до даних літератури. Окремо налаштовуються значення параметрів моделей насосної функції правого та лівого шлуночків серця. Загальний вигляд інтерфейсу налаштування в режимі, який забезпечує введення початкових значень параметрів моделі нерегульованої ССС, показано на рис. 3.

За допомогою екранної форми рис. 3 виконуються також додаткові перевірки обраних параметрів судин. Індикатором адекватності (або неадекватності) обраних характеристик ССС є графіки, що відображають реакції гемодинаміки на встановленій набір параметрів. Відповідно до логіки розрахункового блоку програми та завдяки модельованим механізмам саморегуляції після запуску програми на розрахунки й до кінця заданого періоду експозиції має місце перерозподіл об'ємів крові між ділянками доти, поки в ССС не буде сталий режим гемодинаміки. Із завершенням розрахунків їх результати зберігаються в формах цифрових масивів та графіків. Останні згруповані у чотири групи та відображають більшість характеристик, за допомогою яких фізіолог-дослідник будує свої висновки відповідно до станів конкретних регуляторів.

Оскільки такі висновки мають цінність лише у порівнянні із аналогічними реакціями некерованої моделі, слід розглядати та аналізувати дані моделювання для різних версій моделі. На рис. 4–9 показано приклади симуляцій гемодинаміки людини на різних версіях моделі.

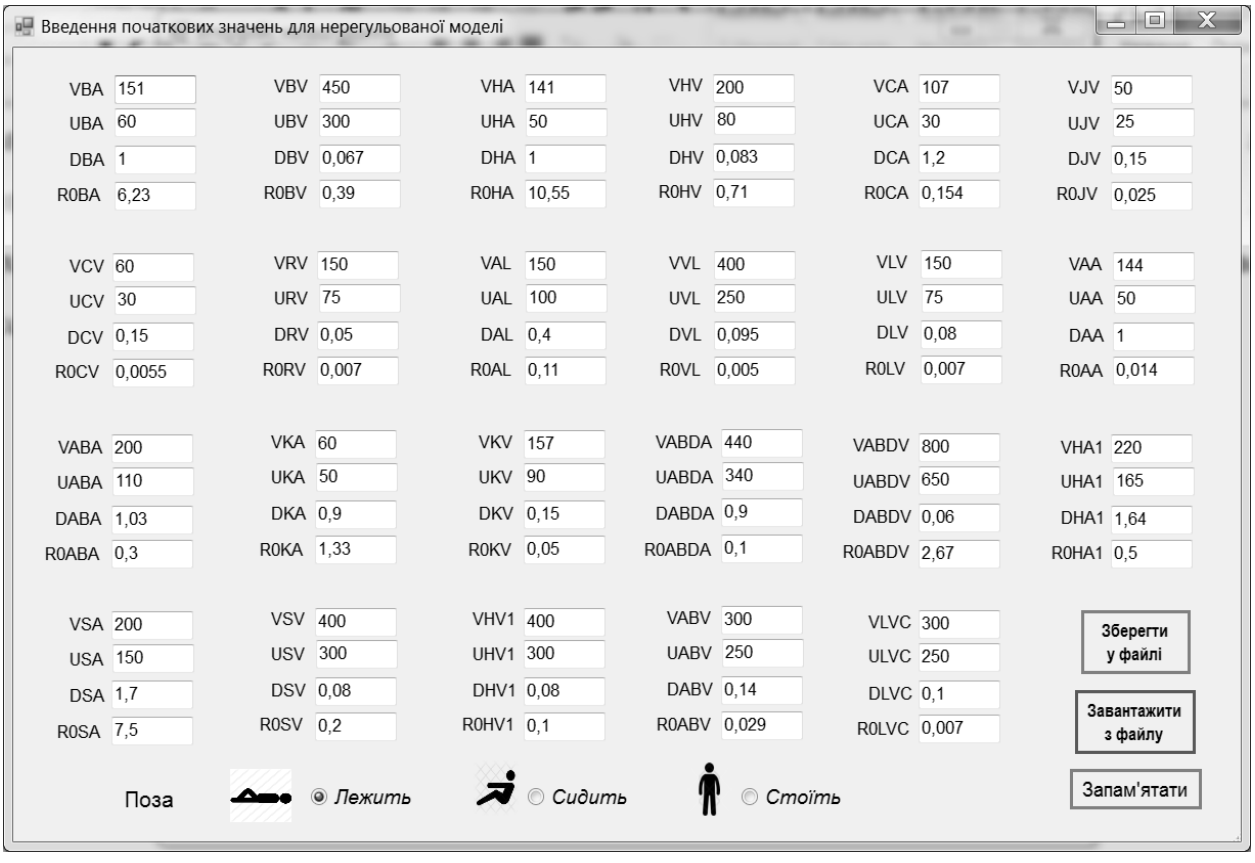

Рис. 3. Загальний вигляд інтерфейсу налаштування в режимі, який забезпечує введення початкових значень параметрів моделі нерегульованої ССС

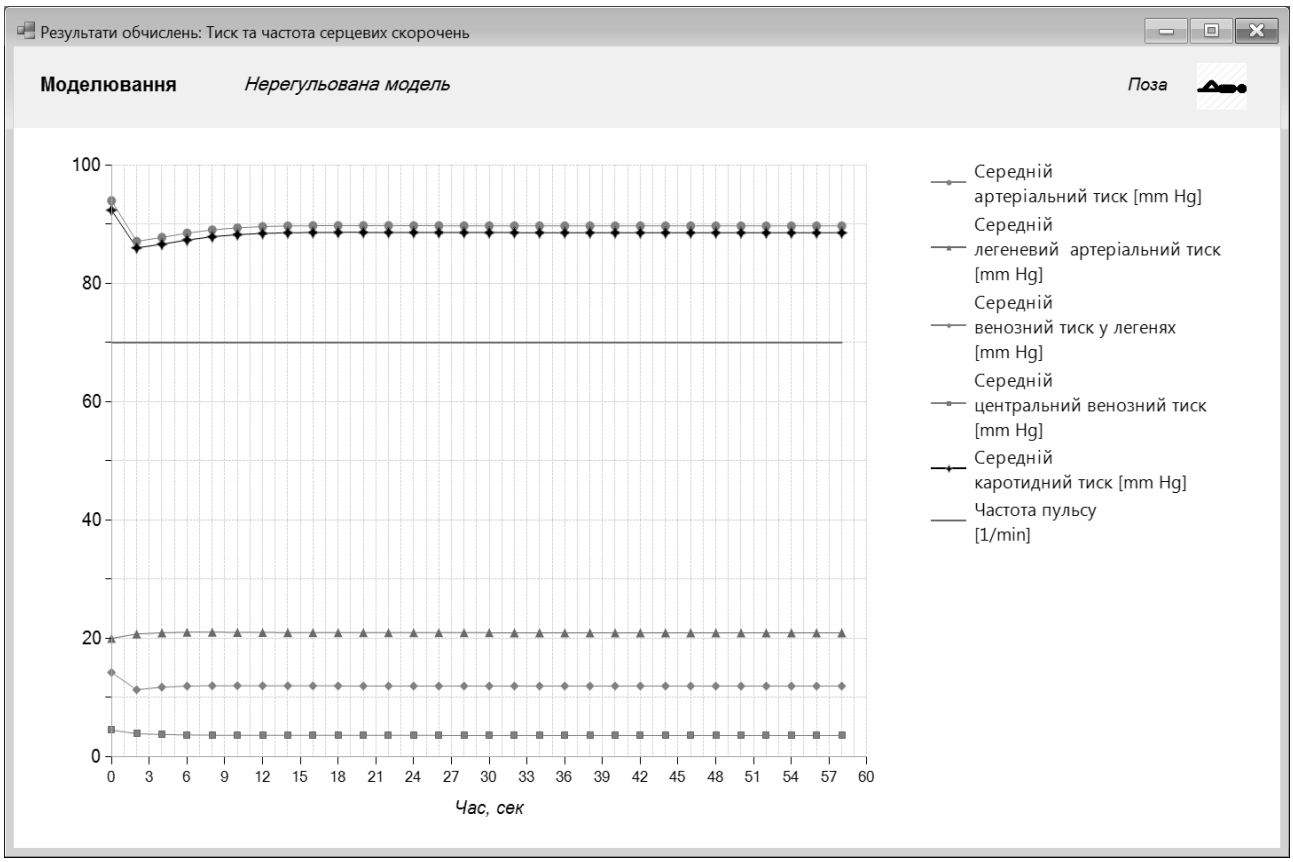

Рис. 4. Симуляція реакцій моделі некерованої гемодинаміки на запуск розрахунків (людина лежить)

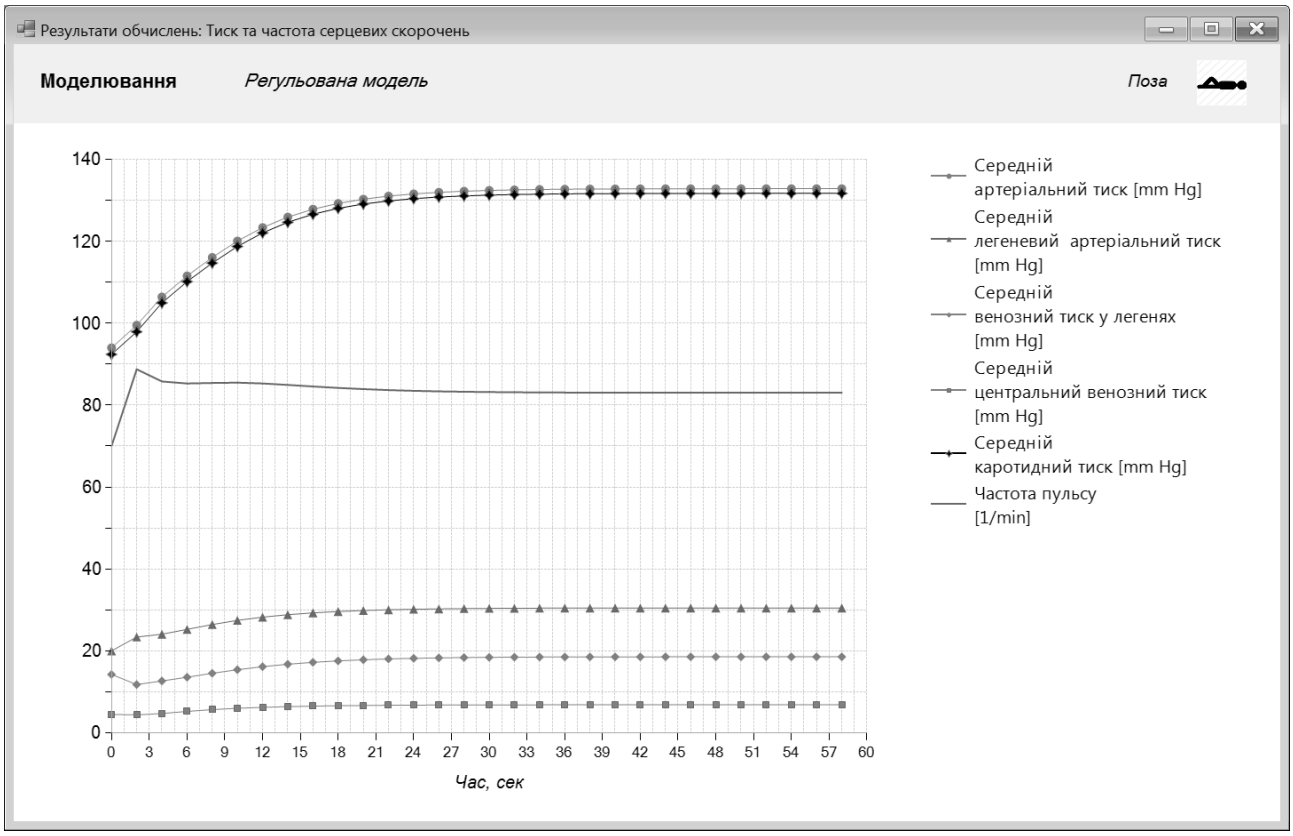

Рис. 5. Симуляція реакцій моделі, керованої за допомогою короткострокових рефлексів гемодинаміки на запуск розрахунків (людина лежить)

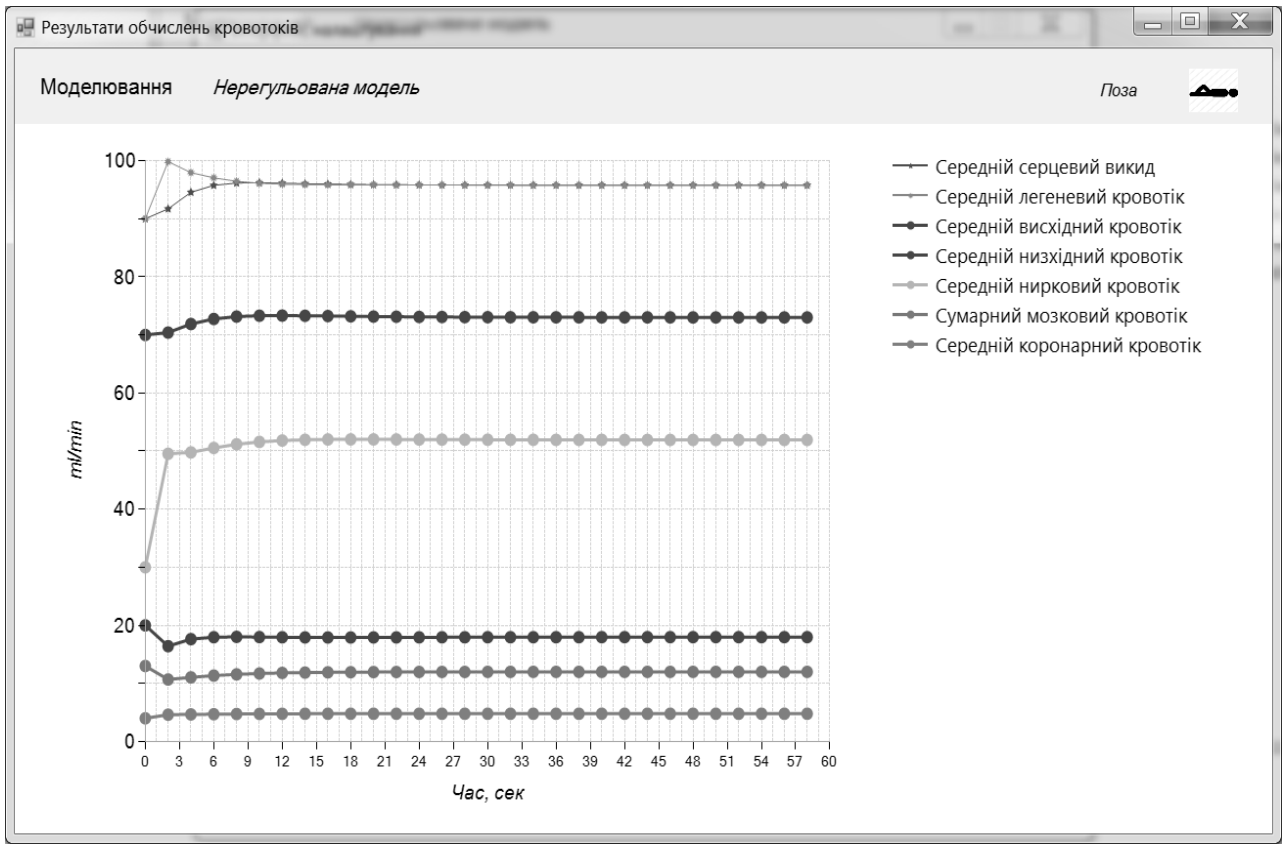

Рис. 6. Симуляція реакцій моделі некерованої гемодинаміки на запуск розрахунків (людина лежить). (Вікно “Кровотоки”)

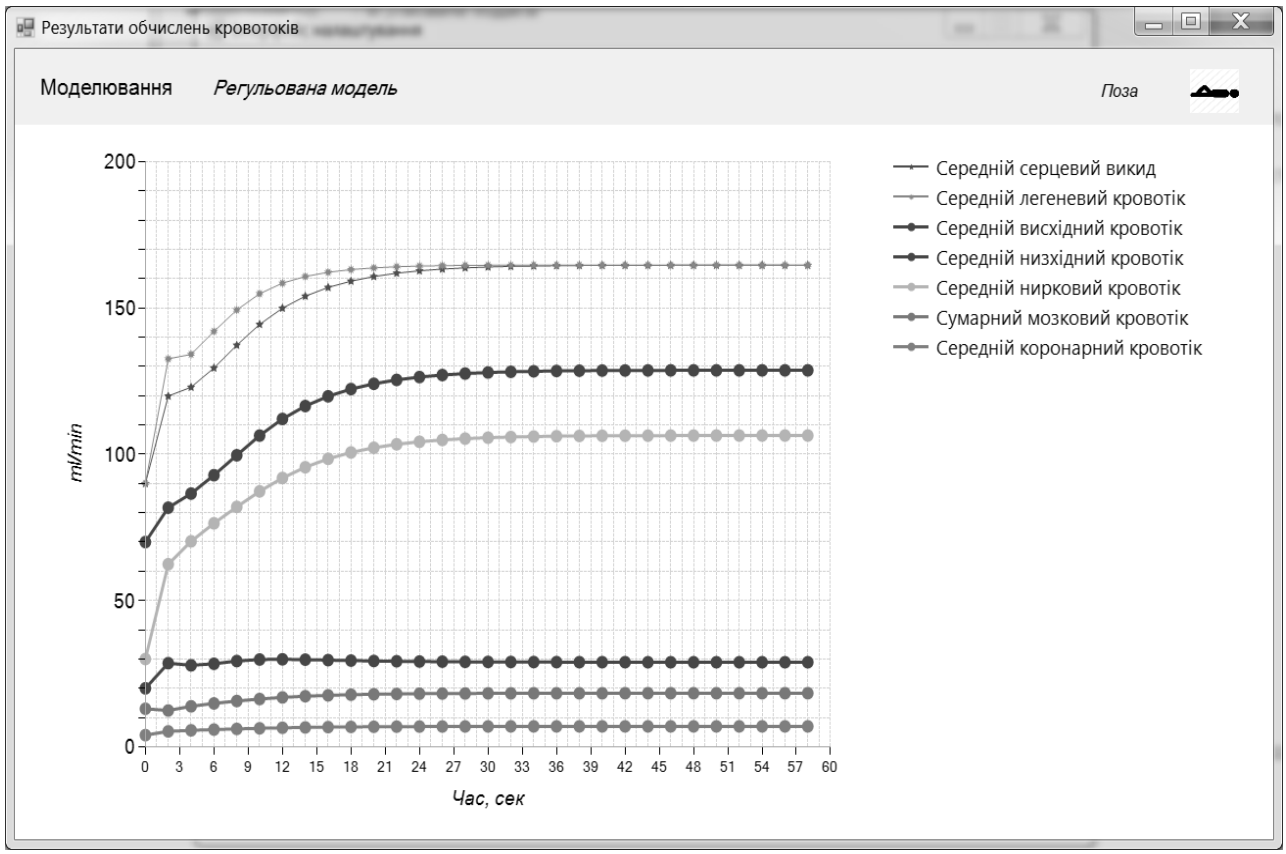

Рис. 7. Симуляція реакцій моделі, керованої за допомогою короткострокових рефлексів гемодинаміки на запуск розрахунків (людина лежить)’’

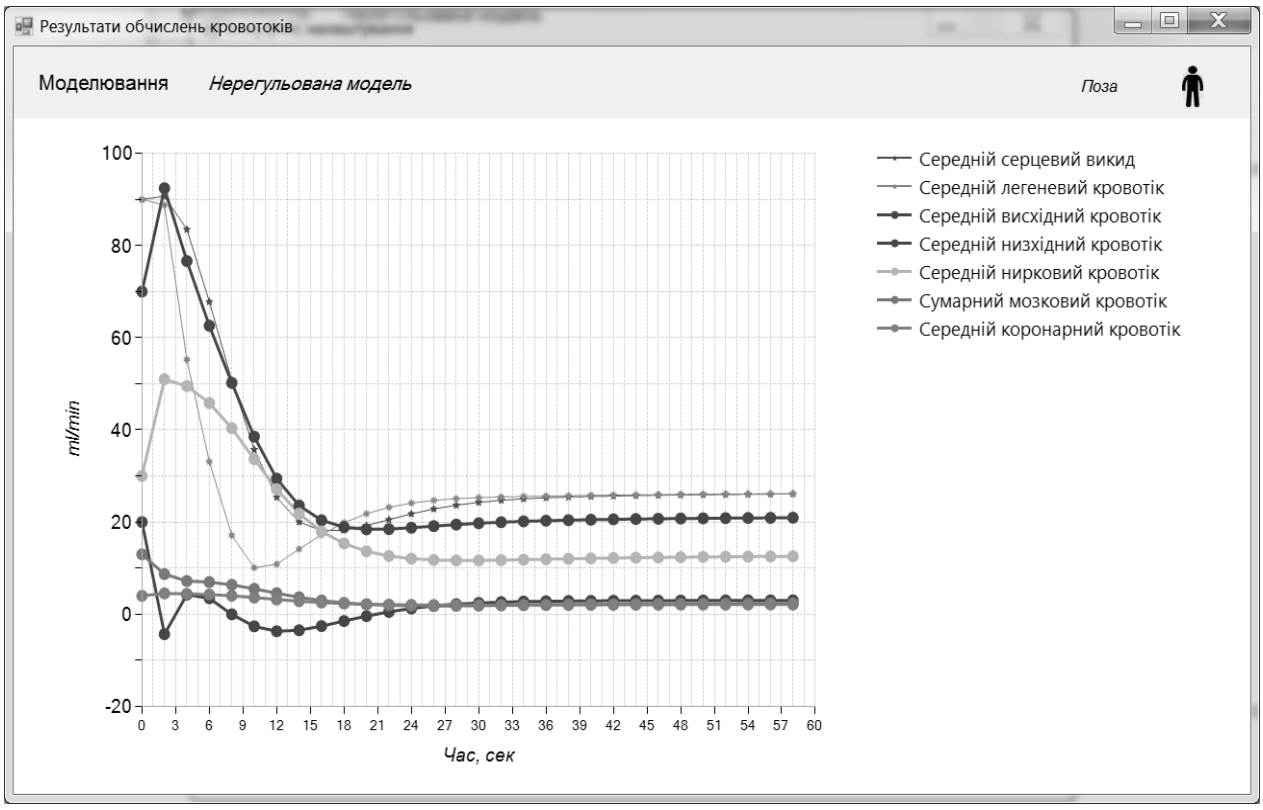

Рис. 8. Симуляція реакцій моделі некерованої гемодинаміки на запуск розрахунків (людина стоїть). (Вікно “Кровотоки”)

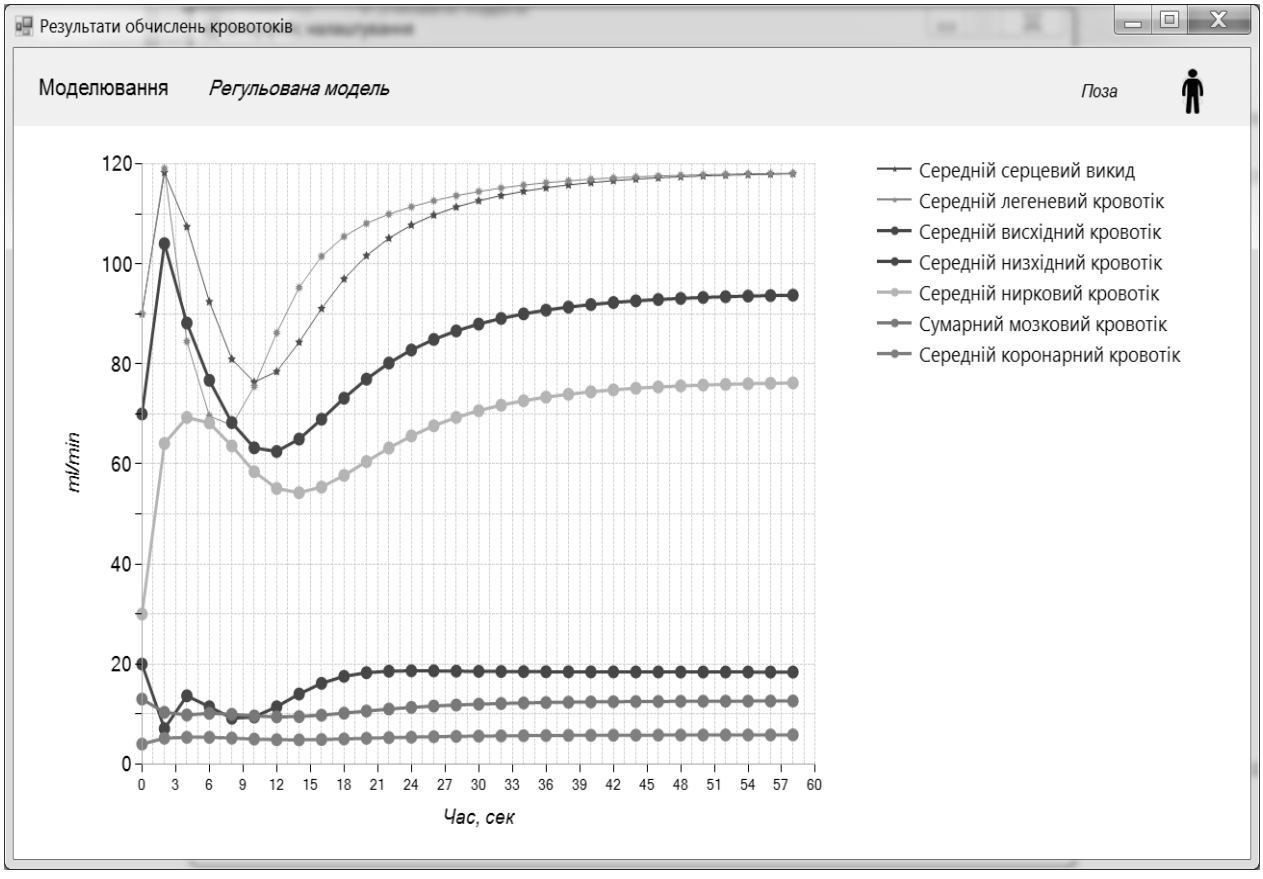

Рис. 9. Симуляція реакцій моделі, керованої за допомогою короткострокових рефлексів гемодинаміки на запуск розрахунків (людина стоїть). (Вікно “Кровотоки”)

Насправді симулятор надає можливість візуалізації набагато більшої кількості змінних центральної та регіональної гемодинаміки. Але в межах однієї статті не можна детально розглянути усі варіанти. Лише зазначимо, що розроблено спеціальну екранну форму, (рис. 10) за допомогою якої користувач може імітувати натуральні неоднорідності симпатичної іннервації у різних регіонах судин.

Висновок

Розроблено програмні засоби та технологію, які дозволяють прискорити процес кількісного налаштування комплексу спеціалізованих математичних моделей, що описують гемодинаміку здорової людини з урахуванням майже усіх відомих фізіологічних механізмів ендогенного впливу на серце, судини та загальний об'єм крові. Наведено приклади комп'ютерних симуляцій у разі конкретних наборів констант моделей та продемонстровано окремі екранні форми, за допомогою яких можна встановлювати та змінювати будь-які можливі значення основних фізіологічних констант моделей людини. На базі технології та екранних форм буде розроблено спеціальний інтерфейс для фізіолога-дослідника.

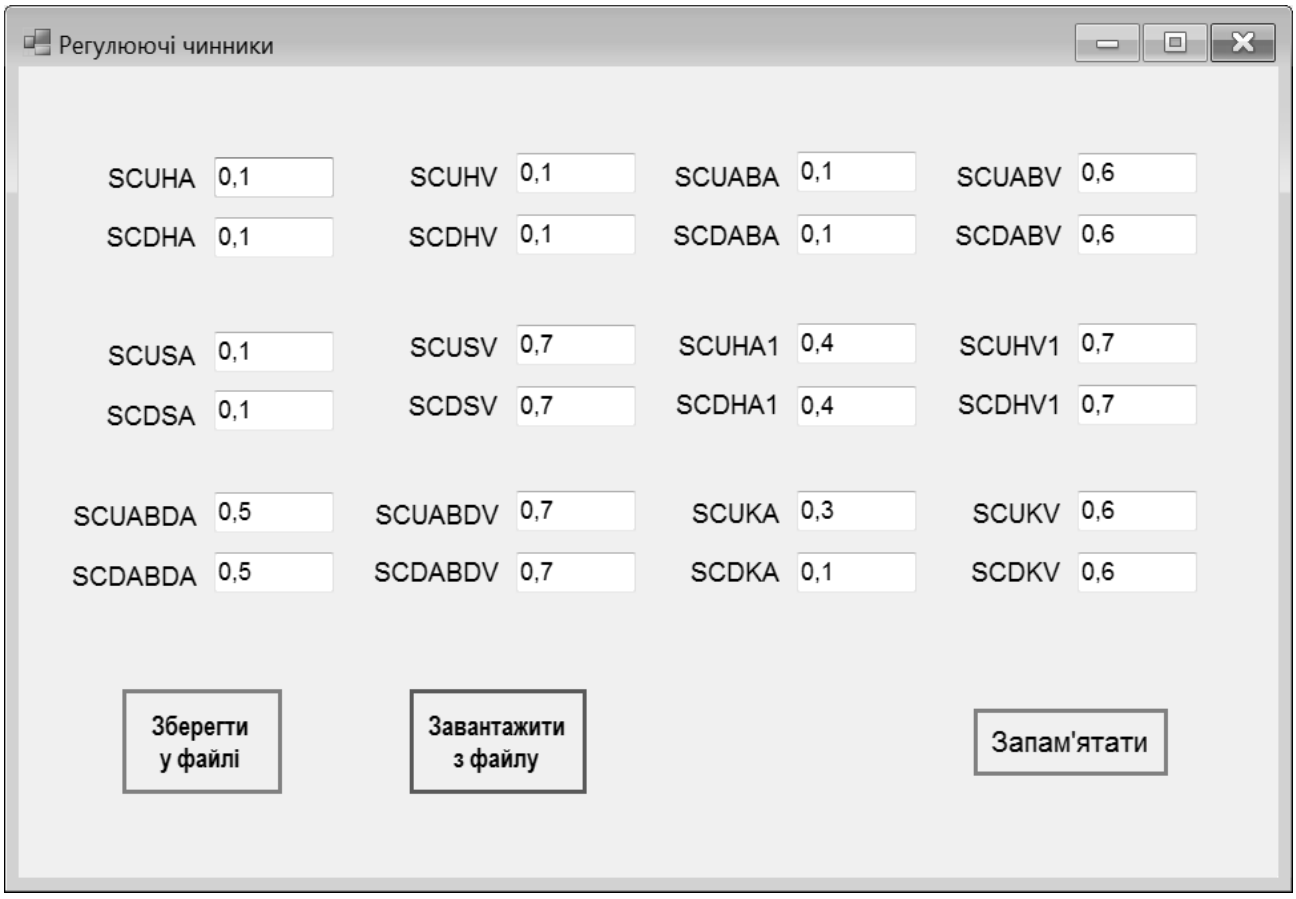

Рис. 10. Екранна форма для встановлення коефіцієнтів моделі, за допомогою яких в моделях враховується регіональна неоднорідність щільності симпатичної іннервації судин

Література

1. Larrabide I., Blanco P.J., Urquiza S.A., Dari E.A., Ve´nere M.J., de Souza e Silva N.A., Feijo´ R.A. HeMoLab – Hemodynamics Modelling Laboratory: An application for modelling the human cardiovascular system. *Computers in Biology and Medicine*. 2012. V. 42. P. 993–1004.
2. Fresiello L., Ferrari G., Di Molfetta A., Zieliński K., Tzallas A., Jacobs S., et al. A cardiovascular simulator tailored for training and clinical uses. *J. Biomed. Inform.* 2015. V. 57. P. 100–112.
3. Григорян Р.Д. Проблемно-ориентированные компьютерные симуляторы для решения теоретических и прикладных задач физиологии человека. *Проблеми програмування*. 2017. № 3. С. 161-171.
4. Григорян Р.Д., Аксенова Т.В., Дегода А.Г., Компьютерный симулятор механизмов поддержания баланса энергии в клетках человека. *Кибернетика и вычислительная техника*. 2017. № 2 (188). С. 65–73.
5. Григорян Р.Д., Лиссов П.Н., Аксенова Т.В., Мороз А.Г. Специализированный программно-моделирующий комплекс «PHYSIOLRESP». *Проблеми програмування*. 2009. № 2. С. 67–82.
6. Григорян Р.Д., Лиссов П.Н. Программный имитатор сердечно-сосудистой системы человека на основе ее математической модели. *Проблеми програмування*. 2004. № 4. С. 100–111.
7. Григорян Р.Д., Дегода А.Г., Джурінський Є.А., Харсун В.С. Симулятор пульсирующего сердца. *Проблеми програмування*. 2017. № 4. С. 98–108 (Rus.).
8. Григорян Р.Д., Дегода А.Г., Харсун В.С., Джурінський Є.А. Симулятор механизмов срочной регуляции гемодинамики человека. *Проблеми програмування*. 2019. № 1. С. 90–98.
9. Григорян Р.Д., Дегода А.Г., Харсун В.С., Джурінський Є.А. Симулятор механизмов долговременной регуляции гемодинамики человека. *Проблеми програмування*. 2019. № 4. С. 111–120.
10. Grygoryan R.D. The optimal circulation: cells' contribution to arterial pressure. 2017. Nova Science. N.Y. 298 p.
11. Grygoryan R.D. The unknown aspects of arterial pressure. *Znanstvena misel journal*. 2019. № 33. P. 19–23.

References

1. Larrabide I., Blanco P.J., Urquiza S.A., Dari E.A., Ve´nere M.J., de Souza e Silva N.A., Feijo´ R.A. HeMoLab – Hemodynamics Modelling Laboratory: An application for modelling the human cardiovascular system. *Computers in Biology and Medicine*. 2012. V. 42. P. 993–1004.
2. Fresiello L., Ferrari G., Di Molfetta A., Zieliński K., Tzallas A., Jacobs S., et al. A cardiovascular simulator tailored for training and clinical uses. *J Biomed Inform.* 2015. V. 57. P. 100–112.
3. Grygoryan R.D. Problem-oriented computer simulators for solving of theoretical and applied tasks of human physiology. *Problems of programming*. 2017. N 3. P. 102–111.
4. Grygoryan R.D., Aksenova T.V., Degoda A.G. A computer simulator of mechanisms providing energy balance in human cells. *Cybernetics and computing technologies*. 2017. N 2 (188). P. 65–73. (Rus).
5. Grygoryan R.D., Lissov P.N., Aksenova T.V., Moroz A.G. The specialized software-modeling complex "PhysiolResp". *Problems of programming.* 2009. V. 2. P.140–150 (Rus).
6. Grygoryan R.D., Lissov P.N. A software-simulator of human cardiovascular system based on its mathematical model. *Problems of programming*. 2004. N 4. C. 100–111 (Rus).
7. Grygoryan R.D., Degoda A.G., Dzhurinsky Y.A., Kharsun V.S. A simulator of pulsatile heart. *Problems of programming*. 2017. N 4. C. 98–108 (Rus.).
8. Grygoryan R.D., Degoda A.G., Kharsun V.S., Dzhurinsky Y.A. A simulator of mechanisms of acute control of human hemodynamics. *Problems of programming.* 2019. V.1. P. 90–98. (Rus.)
9. Grygoryan R.D., Degoda A.G., Dzhurinsky Y.A. A simulator of mechanisms of long-term control of human hemodynamics. *Problems of programming*. 2019. V. 4. P. 111–120.
10. Grygoryan R.D. The optimal circulation: cells' contribution to arterial pressure. 2017. Nova Science. N.Y. 298 p.
11. Grygoryan R.D. The unknown aspects of arterial pressure. *Znanstvena misel journal*. 2019. V. 33. P. 19–23.

Одержано 09.11.2020

Про авторів:

Григорян Рафік Давидович,
завідуючий відділу, д-р. біол. наук.
Кількість наукових публікацій в українських виданнях – 146.
Кількість наукових публікацій в зарубіжних виданнях – 45.
Індекс Гірша – 10.
http://orcid.org/0000-0001-8762-733X.
Юрчак Оксана Іванівна,
провідний інженер-програміст.
Кількість наукових публікацій в українських виданнях – 13.
Індекс Гірша – 0.
https://orcid.org/0000-0003-3941-1555.

Дегода Анна Григорівна,
старший науковий співробітник,
кандидат фізико-математичних наук.
Кількість наукових публікацій в українських виданнях – 14.
Кількість наукових публікацій в зарубіжних виданнях – 1.
Індекс Гірша – 3.
http://orcid.org/0000-0001-6364-5568.

Людовик Тетяна Владленівна
старший науковий співробітник,
кандидат технічних наук.
Кількість наукових публікацій в українських виданнях – 29.
Кількість наукових публікацій в зарубіжних виданнях – 16.
Індекс Гірша – 5.
https://orcid.org/0000-0003-0209-2001.

Місце роботи авторів:

Інститут програмних систем
НАН України, 03187, Київ,
проспект Академіка Глушкова, 40.

E-mail: rgrygoryan@gmail.com,
daravatan@gmail.com,
anna@silverlinecrm.com,
tetyana.lyudovyk@gmail.com

УДК 004.4'24 https://doi.org/10.15407/pp2020.04.014

А.Ю. Дорошенко, О.С. Новак

ДО ПИТАННЯ ОПТИМІЗАЦІЇ ХМАРНИХ ОБЧИСЛЕНЬ З УРАХУВАННЯМ ЇХ ВАРТОСТІ

Запропоновано підхід для архітектурних налаштувань паралельних обчислень на хмарній платформі, що дозволяє в напівавтоматичному режимі здійснити оптимізацію паралельної програми з цільовою функцією мінімуму вартості обчислень. Для розв'язання оптимізаційної задачі використано лінійне програмування, що дозволяє підбирати значення архітектурних параметрів конфігурації програми, що істотно впливають на вартість обчислень. Проведено аналітичне випробування розробленого підходу на моделі хмарного мультипроцесорного кластеру, що показує можливість суттєвого скорочення вартості хмарних обчислень внаслідок проведених оптимізацій.

Ключові слова: хмарні платформи, паралельні обчислення, методи оптимізації, лінійне програмування, вартість обчислень.

Вступ

Хмарні обчислення на сьогодні стали однією з найпопулярніших розподілених обчислювальних парадигм. Вони виступають моделлю для забезпечення зручного доступу до мережі та надійного пулу настроюваних обчислювальних ресурсів [1, 2].

Постачальники хмарних обчислень пропонують широкий спектр послуг на різних рівнях: інфраструктурному, платформному та сервісному (відповідно, IaaS, PaaS та SaaS). Еластичність є важливою характеристикою, яка відрізняє хмарні обчислення від інших розподілених обчислювальних моделей. Функція еластичності дозволяє хмарним платформам ефективно обробляти різні навантаження, не порушуючи нормального стану програми. Вона означає не тільки те, що процеси підготовки та розподілу ресурсів можуть бути реалізовані автоматично та динамічно, але також і можливість гнучкої оцінки економічних показників роботи хмарних систем, таких як вартість обчислень [3].

Користувачі платять за хмарний сервіс, виходячи з потужності та часу використання послуг на базі інфраструктури постачальника. Оскільки хмарні сервіси стають все більш і більш популярними, це привертає величезну увагу лідерів постачальників хмарної інфраструктури, таких як Microsoft, Google і Amazon, спонукаючи їх до роботи із все більш централізованими центрами обробки даних. Сучасний хмарний центр обробки даних оснащений значною кількістю обчислювального обладнання, тому навіть розміщення віртуальних контейнерів по реальним серверам вже саме по собі являє собою дуже складну оптимізаційну задачу [4]. Крім того, все це обладнання споживає значну кількість електроенергії, що спричиняє різке зростання експлуатаційних витрат на хмарні дата-центри, які звичайно перекладаються на споживачів хмарних сервісів. Велике енергоспоживання дата-центрів веде також до збільшення викидів вуглекислого газу (CO_2) та погіршення якості навколишнього середовища. Таким чином, зниження енергоспоживання хмарними дата-центрами і, залом, оптимізація хмарних обчислень є актуальною задачею для розробників хмарних сервісів.

У попередніх роботах авторів [5–6] розроблявся метод самоналаштування (автотюнінгу) паралельних програм на цільову платформу, націлений на досягнення максимальної швидкодії конкретної прикладної програми на конкретну архітектуру обчислювальної системи.

У цій роботі запропоновано узагальнений підхід для самоналаштування паралельних обчислень на хмарній платформі, що дозволяє в напівавтоматичному режимі здійснити оптимізацію потоку паралельних завдань з цільовою функцією мінімуму вартості обчислень. Як буде показано далі, для розв'язання оптимізаційної задачі може бути використано лінійне програмування, що дозволяє у напівавто-

матичному режимі підбирати значення параметрів конфігурації програми які істотно впливають на вартість обчислень.

1. Проблема оптимізації вартості обчислень

Оптимізація вартості обчислень – складна задача, яка вимагає від експерта глибоких знань як про предметну область, так і про реальну систему, на якій задача буде виконуватись. А враховуючи те, що сучасна система може складатися з сотень серверів і досить широкого набору програмних засобів – дуже швидко кількість необхідних знань починає перевищувати реальний обсяг даних, яким може оперувати людина.

Навіть якщо таку систему було колись налаштовано – майже завжди цього не достатньо. Хоча в світі існують системи, які безперервно і майже без оновлень працювали десятками років – це рідше виключення, аніж правило. Швидкі темпи розвитку програмних та апаратних засобів призводять до того, що вже через короткий час може з'явитись необхідність оновити систему, щоб вона працювала і за меншу ціну. А крім того, відбувається розширення можливостей і збільшення навантаження на систему. І тоді система, оптимізована під існуюче колись навантаження, просто перестає справлятися з поточним навантаженням, що призводить до необхідності щоразу оптимізувати систему – вже під нові умови її роботи. Таким чином, для досягнення максимальної ефективності оптимізація системи повинна бути не одиничною задачею, а *постійним процесом* який має тривати впродовж всього циклу розвитку та супроводу системи.

Для вирішення проблеми постійного процесу вдосконалення налаштування системи на вхідний потік задач на сьогодні просувається ідея безсерверних (serverless) обчислень [7], тобто, ідея такої хмарної архітектури, яка передбачає використання абстрагованих обчислювальних потужностей без явного зазначення кількості та характеристик обладнання, що використовується. Такий підхід просувається як «срібляна куля», що здатна вирішити усі проблеми. Проте виявляється, що у реальному житті вона не працює, окрім відносно невеликих проектів, де ціна інфраструктури не йде у порівняння з ціною роботи експертів. І як тільки постає питання оптимізації ціни хмарної інфраструктури – постає питання як це зробити найкраще.

Окрім найбільш очевидного «менше обчислень – менша ціна» існує ще велика кількість проміжних варіантів того, як можна масштабувати систему, від використання можливостей тільки апаратних платформ (що частіше і відбувається у реальній практиці користувачів) до зменшення алгоритмічної складності програм, що реалізують функціональні специфікації прикладної задачі. Як приклади можна привести горизонтальне масштабування невеликих контейнерів ECS/EKS у Amazon (аналогічна технологія, але з іншою назвою, є і в інших постачальників хмарної інфраструктури), а також використання так званих точкових екземплярів (spot instances) чи переніс частини обчислень у хмарні функції (Azure functions чи Amazon Lambda) [7]. І хоча кожен з цих кроків може потребувати значної додаткової роботи, на великих масштабах економія від таких оптимізацій може буди досить значною. Але у цьому випадку складності додає той факт, що зазвичай відсутня можливість тонкого налаштування кожного з контейнерів, тому вибирати доводиться з набору типових конфігурацій. Наприклад, для алгоритму що потребує 4vCPU та 8 RAM, є можливість використовувати EC2 в AWS розміром a1.xlarge з ціною 57$/місяць, але вже при необхідності в 5 vCPU доведеться використовувати a1.2xlarge з надлишковою потужністю (бо це вже 8 vCPU та 16 RAM) і вартістю в 104$/місяць [8]. А враховуючи, що зазвичай потужність vCPU пропорційна розміру контейнера, ручний підбір конфігурації, яка за мінімальну вартість зможе обробляти необхідну кількість даних, – надто складний і тривалий процес.

Виходячи з вищенаведеного, можна зробити висновок, що для оптимізації системи недостатньо тільки оптимізувати алгоритм вирішення прикладної задачі. Для максимальної ефективності потрібно правильно підбирати інфраструктуру та оптимізувати алгоритм саме під цю інфраструк-

ктуру. І тут ми отримуємо класичну проблему – для того щоб оптимізувати алгоритм, нам спочатку потрібно зрозуміти, де і як він буде виконуватись, а для того щоб оптимізувати інфраструктуру, потрібно знати скільки саме ресурсів нам буде необхідно.

Саме для вирішення цієї проблеми ми можемо ефективно використовувати метод автотюнінгу. Як механізми оптимізації з необхідним параметром такі тюнери можуть модифікувати вихідний код алгоритму використовуючи правила переписування/підстановки [9]. А вже маючи інформацію про правила заміни які може робити система автотюнінгу (і як вони впливають на ефективність системи відносно іншої конфігурації) – виникає можливість підібрати оптимальну конфігурацію. За допомогою подальшого вдосконалення правил автотюнера можна покращувати результат самоналаштування.

2. Пропонований підхід

Налаштування мультисервісної системи на її оптимальну за вартістю роботу – надто складна наукова та інженерна проблема, щоб її вирішувати у всій її складності. Тому тут найчастіше застосовують евристичні методи, що на жаль не вирішують проблему і вимагають високої кваліфікації персоналу.

Натомість при спрощеному поданні системи у вигляді незалежних ланок (у випадку, коли це взагалі можливо) (див. рис. 1) вирішення цієї задачі можна звести до розв'язання модифікованої класичної задачі про рюкзаки за умови, що треба мінімізувати загальну вартість рюкзаків, необхідних для збору усіх речей. Це класична NP-повна задача з добре відомими підходами до вирішення. Зважаючи на відносно невелику кількість наявних варіантів, у даному випадку для вирішення цієї задачі є можливість навіть не будувати специфічний солвер (програма-розв’язувач задач), а використати лінійне програмування з використанням існуючого солвера. Хоча такі програмні продукти загалом і поступаються специфічним алгоритмам, але завдяки своїй універсальності вони допомагають досить швидко якщо і не вирішити задачу повністю, то хоча б отримати наближений результат [10,11]. У нашому випадку було використано саме python-mip солвер з використанням coin-or [12]. Він використовує підхід гілок і границь щоб отримати гарантовано оптимальне рішення для заданої моделі.

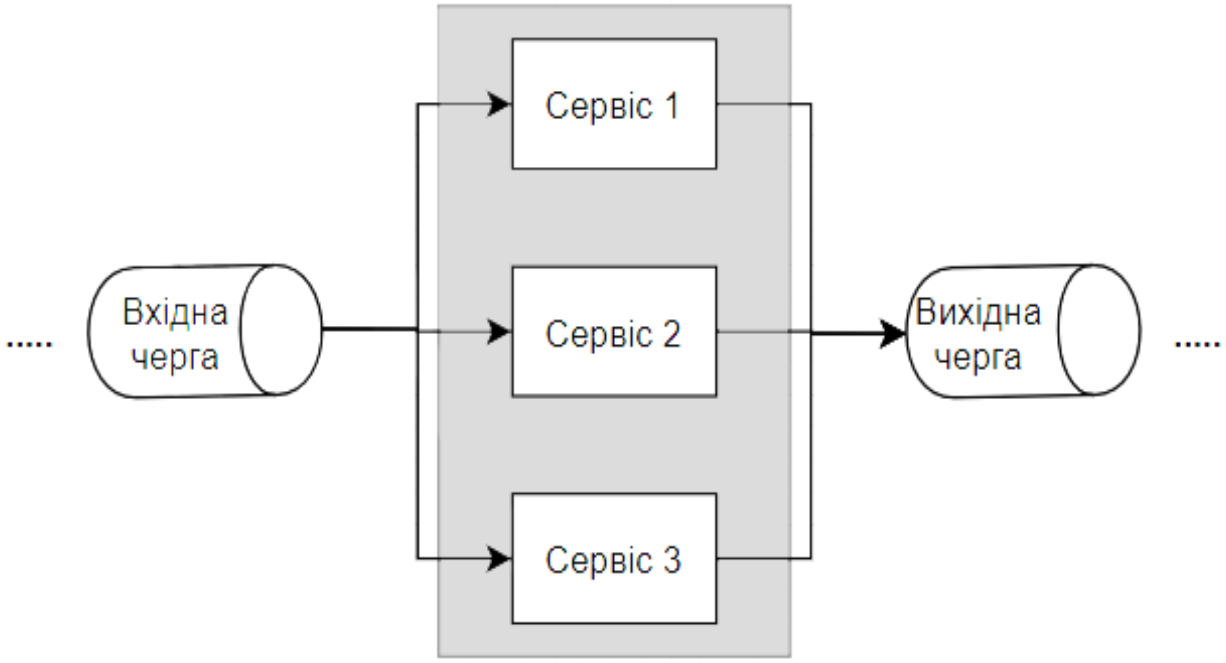

Рис. 1. Ланка архітектури для обробки даних мультисервісної платформи, здатна до горизонтального масштабування

Тобто, в нашому випадку задача зводиться до мінімізації виразу

$$\sum_{c \in C} S_c \times P_c$$

при обмеженні, що

$$\sum_{a \in A, c \in C} S_{(a,c)} \times T_{(a,c)} \leq L$$

де A – набір алгоритмів, C – набір можливих конфігурацій інфраструктурної одиниці, T – швидкодія, P – набір вартостей сервісів, L – мінімально необхідна швидкодія, S – кількість інфраструктурних одиниць.

Тепер, маючи набір можливих конфігурацій інфраструктури та декілька оптимізованих для запуску алгоритмів, можемо знайти оптимальну конфігурацію для навантаження системи.

3. Експеримент

Як експеримент розглянута ланка типового рішення, яка представляє набір з декількох сервісів поміж декількох черг. Розбивши велику задачу на декілька таких ланок ми отримуємо досить просту для моніторингу та масштабування систему, яка в свою чергу може витримувати значне навантаження. Тобто, навіть маючи десятки-сотні сервісів для обробки даних, ми можемо аналізувати пропускні можливості системи за допомогою динаміки кількості даних, що проходять через черги. Це особливо корисно коли потрібно порівняти загальну пропускну спроможність до та після якихось змін у налаштування окремих сервісів. А оскільки ланка сервіс-черга зазвичай не є вузьким місцем системи, то ми отримуємо можливість горизонтального масштабування системи без збільшення її складності.

В табл. 1 наведено список усіх доступних для експерименту контейнерів: де **ram**, **cpu** і **network –** величини що характеризують *відносну* кількість доступних контейнеру ресурсів. Як вже було сказано, у нас відсутня можливість точно налаштовувати розміри усіх характеристик контейнера, тому для порівняння ми використовуємо коефіцієнти відносно найменшого з доступних контейнерів. Тобто, cpu = 2 означає, що vCPU у цьому контейнері має приблизно у 2 рази більшу потужність, ніж у контейнері мінімального розміру. Саме по цій причині всі подальші розрахунки наведені в умовних одиницях.

Таблиця 1. Можливі конфігурації контейнерів.

ціна	назва	ram	cpu	network
1	d_0	1	0.8	1.2
2	d_1	2	1.8	2.2
3	d_2	3	2.8	3.2
4	d_3	4	3.8	4.2
5	d_4	5	4.8	5.2
1	r_0	1.2	0.72	1.2
2	r_1	2.3	2.72	2.2
3	r_2	3.4	4.72	3.2
1	c_0	0.9	0.96	1.2
2	c_1	2.9	2.06	2.2
3	c_2	4.9	3.16	3.2
1	n_0	0.9	0.72	1.68
2	n_1	1.8	1.62	3.68
3	n_2	2.7	2.52	5.68

Окремо слід уточнити причини використаного позначення контейнерів. Хоча й кількість варіантів розміру контейнера обмежена, але зазвичай компанії надають досить велику кількість можливих конфігурацій (у тому числі і специфічних для досить вузького кола задач). Тому склалася така практика, що формат назви являє собою поєднання префіксу, що характеризує тип контейнера, і суфіксу, що характеризує його розмір (як у вищенаведеному прикладі з AWS EC2 a1.xlarge та a1.2xlarge). Саме такий підхід до позначень використаний і тут. В нашому випадку маємо типи d (default), r (ram), c (cpu) та n (network) та декілька розмірів для кожного з них. Тобто, r_0 означає мінімальний розмір контейнера із збільшеною кількістю операційної пам’яті.

Тоді, маючи 2 алгоритми (a1 та a2) що вирішують нашу задачу різними способами (та з різним споживанням ресурсів) ми можемо запустити солвер і знайти оптимальну конфігурацію для навантаження 2500 одиниць (табл. 2). Як можна побачити – найдешевше необхідного результату можливо досягти використовуючи другий алгоритм на 833 контейнерах типу c_0 (мінімальній розмір, збільшена потужність cpu) та 1 контейнеру d_0 (звичайний контейнер мінімального розміру).

Зважаючи, що окрім оптимізації інфраструктури ми ще й використовуємо автотюнінг для оптимізації (прискорення) обчислювальних алгоритмів, ми не можемо виключати, що в нас ще є можливість оновити тюнер таким чином, щоб покращити один з параметрів ціною деякого погіршення іншого.

Тому є сенс додатково розглянути ще й можливість модифікацій характеристик алгоритму, коли погіршення значення одного параметру відбувається для покращення іншого. Тоді можна враховувати можливість автотюнера модифікувати алгоритм (використовуючи правила підстановки) при пошуку оптимальної конфігурації контейнерів. У цьому випадку, наша модель буде модифікована як

$$a_n = a_n^{'} + \Delta a_n ,$$

$$a_n^{'} = \left(cpu_n, ram_n, network_n\right),\ n \in N ,$$

$$\Delta a_n^{'} = \left(\Delta cpu_n,\ \Delta ram_n,\ \Delta network_n\right),\ n \in N ,$$

$$\sum_{e \in \Delta a_n} |e| \le D ,$$

$$\sum_{e \in \Delta a_n} e = 0 ,$$

де D – максимальна можлива величина модифікації (табл. 3).

Таблиця 2. Доступні конфігурації алгоритму

ціна	алг	ram	cpu	net	d_0	d_1	d_2	d_3	d_4	r_0	r_1	r_2	c_0	c_1	c_2	n_0	n_1	n_2
0	a1	0.1	0.2	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
834	a2	0.3	0.3	0.4	1	0	0	0	0	0	0	0	833	0	0	0	0	0

Таблиця 3. Приклад найвигіднішої модифікації

ціна	алг	ram	cpu	net	d_0	d_1	d_2	d_3	d_4	r_0	r_1	r_2	c_0	c_1	c_2	n_0	n_1	n_2
0	a1	0.2	0.1	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a2	0.1	0.1	0.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a3	0.2	0.2	0.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a4	0.1	0.3	0.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a5	0.2	0.4	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a6	0.2	0.3	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a7	0.4	0.2	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a8	0.3	0.2	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	a9	0.4	0.3	0.3	0	0	0	0	0	0	0	0	0	1	1	0	0	0
699	a10	0.3	0.4	0.3	0	0	0	0	0	0	0	233	0	0	0	0	0	0

Як можна побачити, для досягнення мінімальної ціни нам необхідно одночасно використовувати декілька версій алгоритму, які будуть оптимізовані по різних параметрах. Тобто, використавши алгоритм a9 (зі зниженим на 25 % навантаженням на мережевий канал, та збільшенням необхідної пам'яті на 20 %) розгорнутий на двох контейнерах типу c_1 та c_2 одночасно з a10 (зі зниженим на 25 % навантаженням на мережевий канал та збільшеним на 20 % навантаженням на процесор) на 233 контейнерах типу r_2 ми зможемо досягти зменшення загальної вартості на 15.5 %. Як наслідок, можна зробити висновок про наявність вузького місця такої конфігурації і його локалізації. Наприклад, в даному випадку це необхідна ширина мережевого каналу.

Ще одним цікавим моментом є використання пріоритетів характеристик для подальшої оптимізації. Це дозволяє оцінити очікуваний результат від оптимізацій різного типу. По перше, це дозволяє в першу чергу оптимізувати ті частини системи, де результат оптимізацій буде найбільш помітним. По друге – це дозволяє оцінити доречність подальших оптимізацій взагалі. У цьому випадку наші обмеження на зміни будуть модифіковані до наступних:

$$\sum_{e \in \Delta a_n} e \geq D .$$

Тобто, при D = 0.1, ми отримаємо (табл. 4), що відповідає зменшенню загальної вартості на 18.2 % при умові зменшення навантаження мережу на 25 %.

Висновки

В роботі запропоновано підхід до архітектурних налаштувань паралельних обчислень на хмарній платформі, що дозволяє в напівавтоматичному режимі здійснити оптимізацію паралельної програми з цільовою функцією мінімуму вартості обчислень. Для розв’язання оптимізаційної задачі використано лінійне програмування і доступний програмний солвер, який за допомогою методу гілок і границь в напівавтоматичному режимі підбирає значення архітектурних параметрів конфігурації програми, що істотно впливають на вартість обчислень.

Таким чином узагальнено попередній метод автотюнінгу, що розроблявся авторами раніше, та поширено його на випадок комплексу сервісів, що виконуються на хмарній платформі.

Проведено аналітичне випробування розробленої системи на моделі хмарного мультипроцесорного кластеру, що показує можливість суттєвого скорочення вартості хмарних обчислень внаслідок проведених оптимізацій.

Таблиця 4. Приклад найвигіднішого напрямку оптимізації

ціна	алг	ram	cpu	net	d_0	d_1	d_2	d_3	d_4	r_0	r_1	r_2	c_0	c_1	c_2	n_0	n_1	n_2
0	a1	0.1	0.1	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a2	0.1	0.2	0.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a3	0.2	0.3	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	a4	0.3	0.2	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
682	a5	0.3	0.3	0.4	0	0	0	0	0	0	0	341	0	0	0	0	0	0

Література

1. Duan Yucong, Fu Guohua, Zhou Nianjun, Sun Xiaobing, Narendra Nanjangud, Hu Bo (2015). "Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends". 2015 IEEE 8th International Conference on Cloud Computing. IEEE. P. 621–628. doi:10.1109/CLOUD.2015.88
2. Singh Jatinder, Powles Julia, Pasquier, Thomas, Bacon Jean (July 2015). "Data Flow Management and Compliance in Cloud Computing". *IEEE Cloud Computing*. 2(4). P. 24–32. doi:10.1109/MCC.2015.69
3. Zelenyuk V. (2013). "A scale elasticity measure for directional distance function and its dual: Theory and DEA estimation". *European Journal of Operational Research*. 228(3). P. 592–600. doi:10.1016/j.ejor.2013.01.012
4. Azizi S., Shojafar M., Abawajy J. and Buyya R. "GRVMP: A Greedy Randomized Algorithm for Virtual Machine Placement in Cloud Data Centers," in *IEEE Systems Journal*, doi: 10.1109/JSYST.2020.3002721.
5. Anatoliy Doroshenko, Pavlo Ivanenko, Oleksandr Novak, and Olena Yatsenko, A Mixed Method of Parallel Software Auto-Tuning Using Statistical Modeling and Machine Learning in: 14th International Conference, ICTERI 2018, Kyiv, Ukraine, May 14–16, 2018 (Vadim Ermolayev, Mari Carmen Suárez-Figueroa, Vitaliy Yakovyna, Heinrich C. Mayr, Mykola Nikitchenko, Aleksander Spivakovsky (Eds.)), Revised Selected Papers, Series: Communications in Computer and Information Science, Springer, Vol. 1007. 2019. https://doi.org/10.1007/978-3-030-13929-2_6.
6. Дорошенко А.Ю., Іваненко П.А., Новак О.С., Старушик А.М. Автотьюнінг паралельних програм з використанням системи аналізу даних IBM Watson Analytics. *Проблеми програмування*. 2018. № 1. С. 46–54.
7. Neil Savage. Going serverless. Communications of the ACM. 2018. Vol. 61(2). P. 15–16. doi:10.1145/3171583
8. https://calculator.aws/#/createCalculator
9. Андон Ф.И., Дорошенко А.Е., Жереб К.А., Шевченко Р.С., Яценко Е.А. Методы алгебраического программирования. *Формальные методы разработки параллельных программ.* Киев: Наукова думка. 2017. 440 с.
10. Gleixner Ambros, Hendel Gregor, Gamrath Gerald, Achterberg Tobias, Bastubbe Michael, Berthold Timo, Christophel Philipp M., Jarck Kati, Koch Thorsten, Linderoth Jeff, L'ubbecke Marco, Mittelmann Hans D., Ozyurt Derya, Ralphs Ted K., Salvagnin Domenico and Shinano Yuji. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. 2020 Mathematical Programming Computation. http://www.optimization-online.org/DB_FILE/2019/07/7285.pdf
11. Rimmi Anand, Divya Aggarwal & Vijay Kumar (2017). A comparative analysisof optimization solvers. *Journal of Statistics and Management Systems.* 20:4. P. 623–635. DOI:10.1080/09720510.2017.1395182
12. A Comparative Analysis of Optimization Solvers. Available from: https://www.researchgate.net/publication/314750497_A_Comparative_Analysis_of_Optimization_Solvers [accessed Nov 14 2020].

References

1. Duan Yucong, Fu Guohua, Zhou Nianjun, Sun Xiaobing, Narendra Nanjangud, Hu Bo (2015). "Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends". 2015 IEEE 8th International Conference on Cloud Computing. IEEE. P. 621–628. doi:10.1109/CLOUD.2015.88
2. Singh Jatinder, Powles Julia, Pasquier, Thomas, Bacon Jean (July 2015). "Data Flow Management and Compliance in Cloud Computing". IEEE Cloud Compu¬ting. 2(4). P. 24–32. doi:10.1109/MCC.2015.69
3. Zelenyuk V. (2013). "A scale elasticity measure for directional distance function and its dual: Theory and DEA estimation". European Journal of Operational Research. 228(3). P. 592–600. doi:10.1016/j.ejor.2013.01.012
4. Azizi S., Shojafar M., Abawajy J. and Buyya R. "GRVMP: A Greedy Randomized Algorithm for Virtual Machine Placement in Cloud Data Centers," in IEEE Systems Journal, doi: 10.1109/JSYST.2020.3002721.
5. Anatoliy Doroshenko, Pavlo Ivanenko, Oleksandr Novak, and Olena Yatsenko, A Mixed Method of Parallel Software Auto-Tuning Using Statistical Modeling and Machine Learning in: 14th International Conference, ICTERI 2018, Kyiv, Ukraine,

May 14–16, 2018 (Vadim Er¬molayev, Mari Carmen Suárez-Figueroa, Vitaliy Yakovyna, Heinrich C. Mayr, Mykola Nikitchenko, Aleksander Spivakovsky (Eds.)), Revised Selected Papers, Series: Communications in Computer and Information Science, Springer, Vol. 1007, 2019. https://doi.org/10.1007/978-3-030-13929-2_6

6. Doroshenko A., Ivanenko P., Novak O., Starushyk O. Autotuning of parallel programs using the data analysis system IBM Watson Analytics. *Problems of Programming*. 2018. N 1. P. 46–54.
7. Neil Savage. Going serverless. Communications of the ACM. 2018. Vol. 61(2). P. 15–16. doi:10.1145/3171583
8. https://calculator.aws/#/createCalculator
9. Andon P.I. et al. (2017). Methods of algebraic programming. Formal methods of parallel program development. Kyiv: Naukova dumka. (in Russian)
10. Gleixner Ambros, Hendel Gregor, Gamrath Gerald, Achterberg Tobias, Bastubbe Michael, Berthold Timo, Christophel Philipp M., Jarck Kati, Koch Thorsten, Linderoth Jeff, L'ubbecke Marco, Mittelmann Hans D., Ozyurt Derya, Ralphs Ted K., Salvagnin Domenico and Shinano Yuji. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. 2020 Mathematical Programming Computation. http://www.optimization-online.org/DB_FILE/2019/07/7285.pdf
11. Rimmi Anand, Divya Aggarwal & Vijay Kumar (2017). A comparative analysisof optimization solvers. *Journal of Statistics and Management Systems.* 20:4. P. 623–635. DOI:10.1080/09720510.2017.1395182
12. A Comparative Analysis of Optimization Solvers. Available from: https://www.researchgate.net/publication/314750497_A_Comparative_Analysis_of_Optimization_Solvers [accessed Nov 14 2020].

Одержано 15.11.2020

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу
теорії комп'ютерних обчислень Інституту
програмних систем НАН України,
професор кафедри автоматики і
управління в технічних системах НТУУ
"КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в
українських виданнях – понад 180.
Кількість наукових публікацій в
зарубіжних виданнях – понад 100.
Індекс Гірша – 6.
http://orcid.org/0000-0002-8435-1451,

Новак Олександр Сергійович,
молодший науковий співробітник.
Кількість наукових публікацій в
українських виданнях – 11.
Кількість наукових публікацій в
зарубіжних виданнях – 1.
https://orcid.org/0000-0002-1665-7360.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.

E-mail: doroshenkoanatoliy2@gmail.com

УДК 004.94 https://doi.org/10.15407/pp2020.04.022

О. Захарова

ОСНОВНІ АСПЕКТИ СЕМАНТИЧНОГО АНОТУВАННЯ ВЕЛИКИХ ДАНИХ

Семантичні анотації, у силу своєї структурованості – невід'ємне складове ефективного вирішення задач великих даних. Але, сама проблема визначення семантичних анотацій є досить не тривіальною. Ручне анотування є не прийнятним для великих даних з огляду на їх розмір та різнорідність, а також трудомісткість та вартісність самого процесу, задача повністю автоматичного анотування для великих даних поки що не має вирішення. Тобто вирішення задачі семантичного анотування вимагає сучасних змішаних підходів, які б забезпечували вирішення основних задач анотування: виявлення та витягнення сутностей та відношень з контенту будь-якого типу та визначення семантичних анотацій за основі існуючих джерел знань (словників, онтологій, тощо). Отримані анотації повинні бути точними та забезпечувати подальшу можливість вирішення прикладних задач з анотованими даними. Слід зазначити, що контенти великих даних є дуже різноманітними, як наслідок, дуже різняться їх властивості, що підлягають анотуванню. Це вимагає різних метаданих для опису даних та обумовлює наявність великої кількості різних стандартів метаданих для даних різних типів чи форматів представлення. Але, для ефективного вирішення задачі анотування треба мати узагальнену характеристику типів метаданих, в межах якої розглядати їх специфіку. Визначення загальної класифікації метаданих, спільних аспектів та підходів до семантичного анотування контенту великих даних за їх допомогою і є метою даної роботи.

Ключові слова: великі дані, анотування великих даних, класифікація метаданих, семантичні анотації, процес анотування, кероване машинне навчання, некероване машинне навчання, витягнення сутностей, витягнення відношень, домени анотування, онтологічна модель анотування, засоби онтологічного анотування, ручне анотування, автоматичне анотування, напівавтоматичне семантичне анотування, анотатор, аспекти семантичного анотування

Вступ

У поточному стані концентрації даних, існує дивовижний ресурс інформації всіляких типів, що може використовуватись з різними цілями, наприклад, для вивчення музичних інструментів або програмування, та застосунками в різних прикладних галузях. Але, існує інший шар інформації, що доступна та передається через блоги, твіти, статті, журнали. Наприклад, веб, що містить інформацію різних типів та форматів, включаючи текст, рисунки, відео та аудіо, є комунікаційним середовищем, яке дозволяє людині зрозуміти контент і контекст, а також встановити їх відношення/ зв'язати один з іншим засоби масової інформації (ЗМІ). Незважаючи на те, що комп'ютери чудово передають цю інформацію зацікавленим користувачам, системи погано розуміють саму мову представлення інформації. Тому, вирішення задач, що використовують контент великих даних, та задач самих великих даних вимагає структурованого семантичного опису цих даних.

Одним з найбільш розповсюджених підходів до семантизації є визначення семантичних анотацій. Анотації – це назви, атрибути, описи, коментарі, приєднані до документу або частини документу будь-якого формату. Вони надають додаткову інформацію (метадані) про цей, існуючий фрагмент даних. Чіткого, формального визначення терміну «семантична анотація» насьогодні не існує. Відповідно до визначення Ontotex (2016) [1] "семантичне анотування є процесом приєднання додаткової семантичної інформації (метаданих) до різних концептів контенту (сутностей)». Даний тип метаданих забезпечує інформацію як про клас сутності, так і про її екземпляри. Слід зазначити, що семантичні анотації можуть бути застосовані до будь-якого типу контенту: веб-сторінок, звичайних документів, полів у базі даних тощо. Здобуття знань може здійснюватися на основі визначення більш складних залежностей – аналізу відношень між сутностями, опису події чи ситуації тощо. Семантичне анотування забезпечує підтримку розширеного пошуку (на основі концептів), міркування про веб-ресурси та інфор-

маційну візуалізацію на основі онтологій. Окрім цього, анотації використовуються для конвертації синтаксичних структур у структури знань. Іншими словами, семантичне анотування полягає у тому, щоб генерувати специфічні метадані та схеми використання, що забезпечує нові методи доступу до інформації та розширює існуючі.

Семантичні анотації можуть використовуватись для вирішення цілої низки задач великих даних, включаючи інформаційний пошук на основі семантик, категоризацію та композицію документів, перехід від неструктурованого контенту до релевантних знань, візуалізацію інформації на основі онтологій. Якщо документ (чи будь-яка частина деякого контенту, наприклад, відео) є семантично анотованим, він стає джерелом інформації, яку простіше інтерпретувати, комбінувати та повторно використовувати автоматизованим чином. Великим даним притаманна різнорідність джерел та різноманітність форматів їх представлення. Зрозуміло, що міркуючи про набори метаданих, що описують великі дані певних типів, треба враховувати, що кожний з них має власні характеристики. Метою даної роботи є визначити спільні аспекти та підходи до семантичного анотування контенту великих даних за допомогою метаданих.

Домени та моделі семантичного анотування

Можна класифікувати 4 моделі семантичної анотації [2]: теги, атрибути, відношення та онтології. Теги займають нижчий рівень та відповідають найпростішій формі анотування з точки зору користувача; водночас, як онтології находяться на верхньому рівні та відповідають найскладнішій формі з точки зору користувача.

Теги: елемент анотація тегу є призначеним ресурсу ключовим словом або виразом, що описує певну властивість ресурсу. Прикладами тегів можуть бути назви місць, де зроблене фото, імена осіб на фото або тема статті.

Атрибути: елемент анотація атрибуту є парою двох елементів: назва атрибута та його значення. Назва атрибуту означає властивість анотованого ресурсу (наприклад, “Країна”, “день народження”), а значення атрибуту – відповідне значення (наприклад, “Туніс”, “1909”).

Відношення: елемент анотація відношення є парою двох компонент: назва відношення та пов’язаний ресурс. Анотований ресурс зв’язаний з відношенням його назвою. Іншими словами, модель анотації відношення є розширенням моделі анотації атрибуту до домену ресурсу, дозволяючи користувачеві з’єднувати ці ресурси. Наприклад, посилання в одній науковій праці на інший документ є анотуванням відношення, що визначає зв’язок між цими документами.

Онтології: онтологічна модель описує метадані, які співставляють ресурс або його частину з деякими описами його властивостей та характеристик відповідно до формальної концептуальної моделі (онтології). Онтології є корисними для витягнення знань про домен та специфікації загальноприйнятого розуміння домену (що може повторно використовуватися і бути спільним для спільнот та застосунків). Побудова онтології може бути реалізована визначенням концептів, екземплярів концептів, властивостей концептів та екземплярів, обмежень на ці властивості, відношень між концептами та відношень між екземплярами. Користувач, що використовує онтологічну модель анотації, може описати та з’єднати існуючі ресурси шляхом структурування ресурсів (концептів або екземплярів) та визначення обмежень між відношеннями та властивостями.

Використання онтологій як словників для визначення метаданих

Метадані можна класифікувати як залежні та незалежні від контексту [3]. Серед метаданих, що залежать від контексту, можна виділити прямі явні (що згадуються у контенті) та неявні (що виводяться з контексту), та не прямі (зовнішні метадані – посилання на контекст через URL у контенті). Така класифікація (рис. 1) є дуже суттєвою для подальшого визначення процесу анотування.

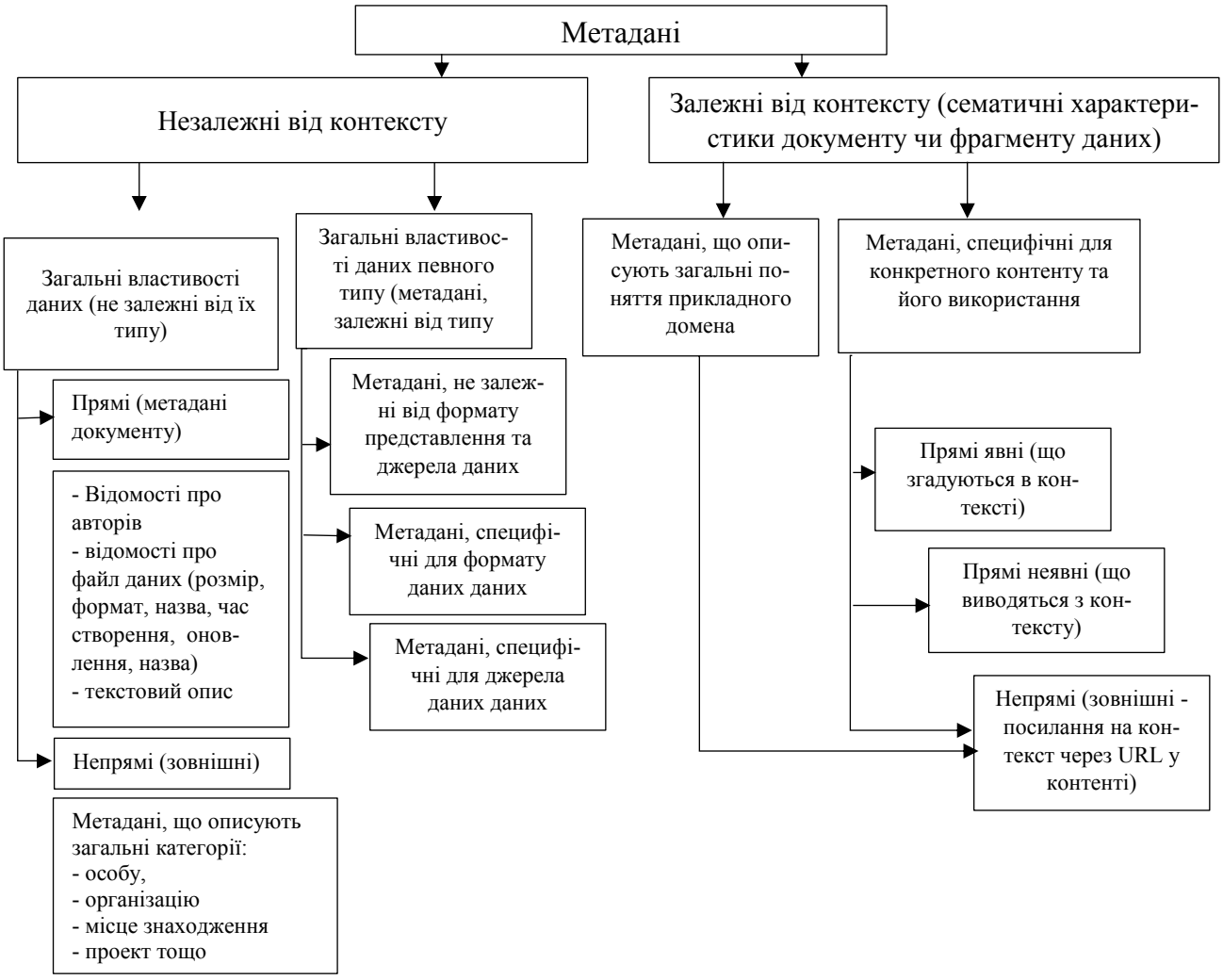

Рис. 1. Класифікація метаданих великих даних

Для створення семантичних метаданих можуть використовуватися стандарти метаданих, тезауриси та контрольовані словники (наприклад, MeSH [4], TGN [5] тощо), а також полегшені або повноцінні онтології. Стандарти, контрольовані словники та тезауруси, на відміну від повноцінної онтології, не є повністю формальними, наприклад, відношення між термінами, які вони включають, не мають явних семантик. Як правило, вони використовуються для забезпечення погодженої термінології у конкретних доменах.

Слід зазначити, що відповідно до наведеної класифікації метаданих можна виділити два головних напрямки анотування великих даних: анотування загальних характеристик документа та анотування прикладного контенту. Метадані для анотування загальних властивостей об'єктів великих даних є контекстно-незалежними і для їх визначення розроблено чимало спеціальних стандартів. Більшість з них орієнтовано на анотування даних певних типів: відео, зображень, аудіо тощо, та враховують їх специфічні характеристики. Також існують стандарти, що дозволяють описувати загальні характеристики документів будь-якого типу. Наприклад, добре відомий стандарт Dublin Core, який широко використовується для визначення характеристик електронних документів. Цей стандарт специфікує множину наперед визначених характеристик документів, таких як автор, дата створення, опис, формат тощо, які можуть бути застосовані для документу будь-якого формату.

Інша група – більш специфічні онтології, що призначені для вирішення

певних задач/підзадач опису даних. Наприклад, полегшена онтологія FOAF (Friend of a Friend) [6], яка має за мету створення анотованої мережі домашніх сторінок людей, груп, компаній тощо. Відповідно, онтологія містить такі концепти як Агент, Особа, Організація, Група, Проект, Документ, Зображення тощо, та де-які базові характеристики, що описують екземпляри цих класів. Подібне призначення мають й онтології OntoWeb [7], KnowledgeWeb [8] та онтологія опису публікацій, які описують осіб, організації, проекти, публікації тощо. Всі перелічені групи онтологій дозволяють описувати певні загальні характеристики документів в цілому, або документів певних форматів або призначення. Вони дозволяють автоматично визначити велику кількість технічних характеристик документа та деяку загальну його семантику, але не охоплюють деталей його контенту (можливо, лише деякі ключові слова та описи природньою мовою в таких характеристиках, як тема або опис). Звісно, це сприяє вирішенню певних задач, але справжній семантичний опис, який викриває сутність та особливості контенту даних, вимагає визначення різних типів контекстно-залежних метаданих, що не можливо без застосування прикладних онтологій домену. Це можуть бути загальні прикладні онтології, що призначені для анотування документів або вирішення проблем у деякому широкому домені. Наприклад, онтології Esperonto Cultural Tour [9] та Fund Finder дозволяють анотувати документи в прикладних доменах культура та фінансування, відповідно. Такі онтології, як правило, визначають загальні характеристики для обраної прикладної області. А можуть бути більш специфічні прикладні онтології, які дозволяють точніше визначити семантику даних, але їх застосування у повністю автоматизованому режимі є проблематичним з огляду на те, що є досить специфічним для конкретного об'єкта даних та задачі, що вирішується.

В цілому всі підходи анотування, що використовують такі додаткові джерела знань, як словники та онтології, відносяться до групи методів, що засновані на онтологіях. А процес створення метаданих з використанням онтологій як словників називається онтологічним анотуванням.

Використання онтологій дозволяє зв'язати фрагменти даних з формальними концептами, що забезпечує підтримку інтероперабельності та гарантує автоматичні міркування на базі структури, яка лежить в основі цих онтологій.

Анотації на основі онтологій, зазвичай, містять три типи інформації [2]:

1) *екземпляри концепта* пов'язують частину документа з одним або декількома концептами в онтології. Так, наприклад, «Інформація про рейс» представляє екземпляр сутності *Рейс* та має ім'я *AA7615_Feb08_2003*, хоча екземпляри концепту не завжди мають ім'я.

2) *значення атрибутів* пов'язують екземпляр концепта з частиною документа, що є значенням одного з його атрибутів. Так, «*Американські авіалінії*» може бути значенням атрибуту *companyName*.

3) *екземпляри відношень* зв'язують два екземпляри концептів конкретного домену. Наприклад, рейс *AA7615_Feb08_ 2003* та місцезнаходження *Мадрид* можуть бути зв'язані відношенням *departurePlace*.

Слід зазначити, що існує два типи відношень між концептами: зв'язки між концептами однієї онтології, або різних онтологій, таким чином, створюючи відображення між джерелами знань. Такі відображення є основою інтеграції множини онтологій, що створюють всебічну базу знань для семантичного опису даних. Визначення зв'язків між поняттями з різних джерел є надзвичайно корисним для взаємодії неоднорідних даних, але досить складним завданням. Системи анотування можуть використовувати спеціальні бібліотеки сервісів, що відповідають за відображення онтологій, що дозволяють використовувати вже опубліковані онтологічні відображення. Так, наприклад, спеціальні веб-сервіси NCBO [10] надають доступ до мільйонів онтологічних відображень, опублікованих у BioPortal. Це дозволяє веб-сервісу NCBO Анотатор

автоматично «тегувати» текст за допомогою термінів з онтологій BioPortal, а вебсервісу NCBO індексування ресурсів надавати доступ до онтологічного індексу публічних онлайн-ресурсів даних. Сукупність таких сервісів, разом зі спеціальними віджетами NCBO, (рис. 2) забезпечує можливість реалізації процесів візуалізації онтологій, анотування та інтеграції даних у галузі біомедицини.

Повноцінні онтології також дозволяють перевіряти обмеження на допустимі значення та їх відношення у відповідних анотаціях. Значення елементів анотацій, зазвичай, є посиланнями на екземпляри в онтології.

Рис. 2. Сервіси NCBO для анотування на основі BioPortal онтологій

Основні аспекти семантичного анотування великих даних

Концепти, що представляють контент будь-якої прикладної області, можна розділити на дві основні категорії: концепти, що представляють процеси, та концепти, які описують фізичні об'єкти, що приймають участь в цих процесах.

До основних категорій анотування відносять сутності (екземпляри фізичних об'єктів), події (екземпляри концептів, що представляють процеси) та відношення між сутностями та/або подіями.

Таким чином, при анотуванні будь-якого контенту, необхідно, перш за все, виділити події та сутності, для яких доцільно створити метадані, визначити ці концепти або їх атрибути, а потім, індексувати їх, класифікувати шляхом визначення зв'язків з онтологією прикладного домену та визначити зв'язки цих сутностей у базі даних семантичного графу.

Як основні аспекти анотування можна виділити [11]:

- *Ідентифікація тексту*. Текст витягується з будь-яких інформаційних джерел, в тому числі не текстових – відео, аудіо, pdf – файли тощо.
- *Розділення тексту на процеси та фізичні сутності.* Розпізнавання іменованих об'єктів.
- *Витягування основних сутностей та їх ідентифікація* (визначення типу сутності, її зв'язку з визначенням прикладного домену, наприклад, URI на об'єкт прикладної онтології). На цьому етапі необхідно знайти та класифікувати елементи тексту за попередньо визначеними категоріями.
- *Класифікація та ідентифікація відношень між визначеними сутностями, аргументів відношень, та визначення їх зв'язків із зовнішніми чи внутрішніми знаннями домену*.
- *Індексація та зберігання семантизованих даних в базі даних семантичного графа*. (Усі розпізнані та збагачені машинно-читаємими метаданими дані зберігаються у базі даних семантичного графа для подальших посилань та використання.)

На рис. 3 показано перелічені аспекти на прикладі анотування тексту, а саме: витягнення основних сутностей (Рим та Римська імперія) з текстового фрагменту, їх ідентифікація, встановлення зв'язків з прикладним доменом, та визначення їх місця у семантичному графі.

Рис. 3. Приклад анотування текстового фрагменту даних

Методології процесу семантичного анотування

Процес анотування має базуватися на наявному загальному теоретичному апараті: методах машинного навчання, статистичного навчання, обробки текстів природньою мовою тощо. Результати анотування великих даних мають бути придатними та уможливлювати вирішення конкретних прикладних задач з цими великими даними, як, наприклад, семантичний пошук чи доступ до даних. Сам процес анотування має вирішувати задачі виявлення та витягування концептів та відношень. Задача анотування полягає в описі визначених сутностей та відношень відповідно до онтології.

У роботі [12] наведена досить вдала загальна трирівнева архітектура процесу семантичного анотування, що визначає його основні задачі (рис. 4).

Анотування може здійснюватися в ручному, автоматичному чи напівавтоматичному режимі [1].

Ручне анотування – це методологія, яка перетворює інформаційні ресурси у взаємопов'язані структури знань шляхом додавання метаданих до деякого рівня документу. Процес ручної анотації є вартісним та трудомістким, і часто не враховує існування різних точок зору на джерела даних, які можуть представлятися різними онтологіями. Насьогодні, з'являється чимало спеціальних засобів для полегшення процесу ручного анотування.

Так, Protégé [13] дозволяє використовувати для анотування обраного фрагменту тексту екземпляри класів онтологій. Інший приклад, визначення еквівалентних об'єктів у різних місцях документу чи документів, шляхом співвідношення цих об'єктів з однією сутністю реального світу. Таке анотування еквівалентностей особливо важливе у медичних застосунках. Такий інструмент, як Semantator [14], дозволяє визначати інший важливий тип анотацій, а саме, дозволяє користувачам обрати два екземпляри та створити відношення між ними, або додати їх до списку кандидатів для побудови відношень. Після чого можна обрати будь-які властивості об'єкта онтології та визначити зміст цього нового відношення.

Очевидно, що, коли мова йде про великі дані, застосування методів ручного анотування стає неможливим, оскільки робота експерта вимагає багато часу та зусиль. Анотація, щоб забезпечити інтенсивні знання, та для розуміння контенту, має бути точною, але вона водночас має бути максимально автоматичною. Тобто, ідеальним рішенням було б автоматичне семантичне анотування, але задача повністю автоматичного створення семантичних анотацій також не має вирішення. Найефективніші семантичні метадані, створені за допомогою інструментів автоматичного анотування чи тегування, будуються на базі різних алгоритмів машинного навчання, які потребують навчальних виборок.

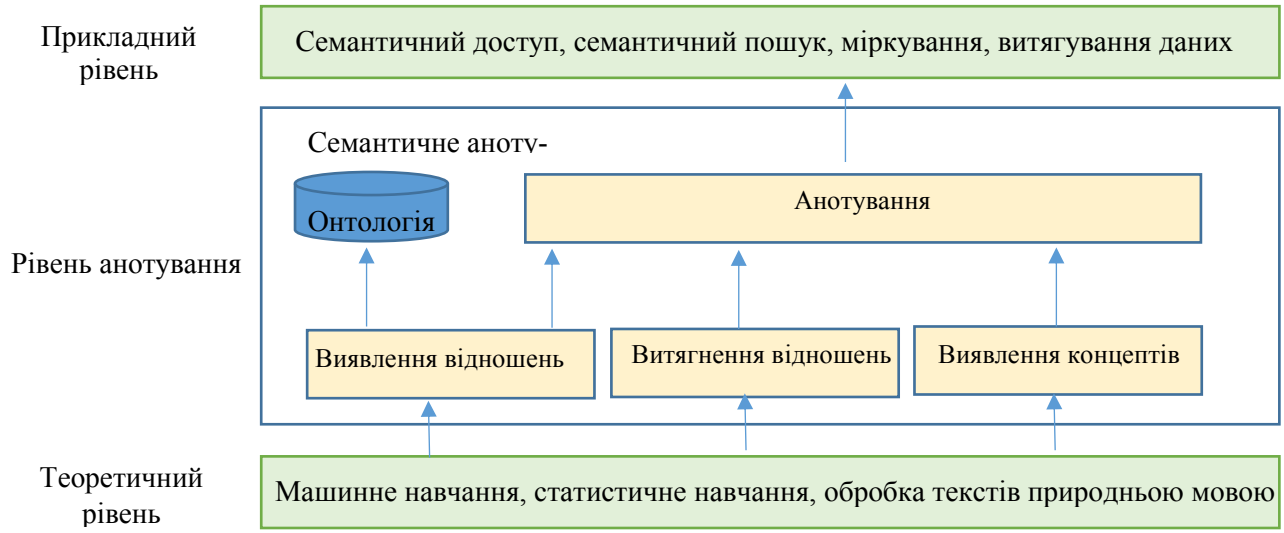

Рис. 4. Загальна архітектура процесу семантичного анотування

Серед методів, що використовуються для автоматичного анотування, можна виділити.

Методи керованого машинного навчання, що складаються з двох етапів: анотування та навчання. На етапі анотування треба визначити сутності та семантичні відношення між цими сутностями для заданого контенту. Задача етапу навчання полягає у вивченні моделі чи моделей, які використовуються на етапі анотування. Для вивчення моделей, вхідні дані часто розглядаються як послідовність деяких одиниць, наприклад, текстовий документ можна розглядати як послідовність слів або рядків тексту. Даний метод потребує розмічених даних.

Метод некерованого машинного навчання намагається створити метадані без розмічених даних. При цьому, для отримання даних з Інтернету можуть бути використані узагальнені зразки.

Слід зазначити, що в силу різноманітності та різнорідності великих даних не можливо досягти ефективного анотування при використанні якоїсь певної однієї категорії методів. Дані з різними характеристиками вимагають різних підходів для вирішення задачі. Так, для веб-сторінок, що побудовані на основі шаблонів та генерують дані з баз даних, можуть бути ефективними методи на основі правил, але вони є не прийнятними для повнотекстових фрагментів даних. Припущення, що документи мають схожу структуру або подібний текст, які досить активно використовуються існуючими підходами машинного навчання, здаються не реалістичними, враховуючи гетерогенну природу веб, та не можуть застосовуватися у випадку великих даних та сучасних типів контентів. Анотування великих даних вимагає використання комбінацій різноманіття методів та підходів для вирішення кожної з задач анотування, та динамічного прийняття рішень щодо використання тих чи інших методів та забезпечення можливості їх використання. Так, у [12] автори наводять де-які методи та моделі, що можуть використовуватися для виявлення та витягнення сутностей та відношень при автоматичному семантичному анотуванні за допомогою підходів керованого машинного навчання, а саме: витягнення сутностей на основі правил, класифікації, послідовного розмічення даних, альтернативних умовних випадкових полів, не лінійні випадкові поля Маркова тощо. Слід зазначити, що кожна з перелічених категорій містить досить широкий спектр методів та моделей (рис. 5). З більш детальним їх описом можна ознайомитись у [12].

Насьогодні, найбільш реалістичними (та використовуваними у сучасних системах) є підходи напівавтоматичного або змішаного анотування, що комбінують процеси ручного та автоматичного анотування. Вони вимагають втручання людини на певних рівнях процесу. Зазвичай, процес, що легко анотується, анотується

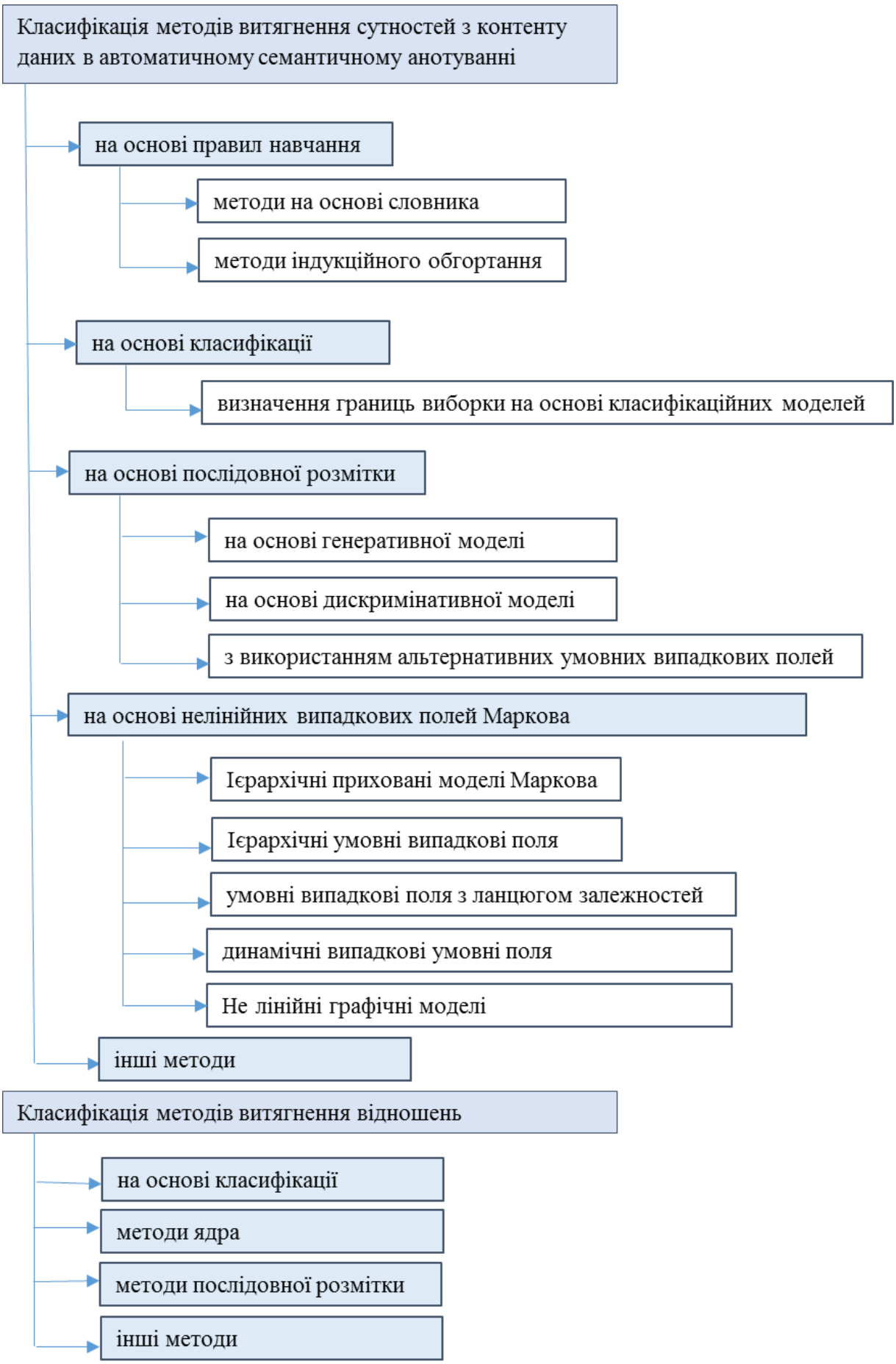

Рис. 5. Категорії методів анотування

автоматично, але існують деякі складні контенти, де втручання людини є необхідним для надання більш змістовної анотації. Ця категорія систем анотування вирізняється архітектурою, методами та засобами витягнення інформації, обсягами ручної праці, продуктивністю, організацією зберігання тощо. Прикладами інструментів напівавтоматичного анотування є GATE [15], NCBO анотатор та cTAKES [16]. NCBO анотатор та cTAKES використовують напівавтоматичне анотування додатково до Semantator.

Фокусом даної роботи є використання онтологій для різних підходів, етапів та категорій процесу анотування, хоча моделі семантичного анотування включають й простіші форми.

Засоби анотування на основі онтологій. Більшість інструментів анотування (анотаторів) на основі онтологій з'явились разом з виникненням семантичного вебу.

Метою їх розробки було забезпечення вставки розмітки на основі онтологій до веб сторінок та подальшої її підтримки. Анотатори, спочатку, були задумані як засоби для полегшення процесу ручного анотування, тобто ручного додавання анотації на веб сторінки. Згодом більшість з них перетворилися на більш повноцінне середовище, яке використовує методи вилучення інформації (IE) та машинного навчання (ML) для забезпечення напівавтоматичного процесу анотування документів. Але засоби, що полегшують процес ручного визначення анотацій, також не втрачають своєї актуальності.

Так, OntoMat-Annotizer [17, 18] та SHOE Knowledge Annotator [19] є прикладами досить широко використовуваних інструментів ручного анотування. OntoMat-Annotizer – розширюваний Java застосунок, який дозволяє створювати OWL анотації. Він включає браузер онтології для дослідження концептів та екземплярів онтології та HTML браузер для відображення документів та їх анотованих частин.

Дозволяє перетягувати частини тексту до анотацій, що створюються. Користувач може визначати екземпляри концептів, з атрибутами та екземплярами зв'язків. OntoMat-Annotizer завантажує OWL онтології.

Анотації, створені за його допомогою, зберігаються в OWL як окремі файли або як вбудовані в анотовані HTML документи та можуть використовуватися широким спектром застосунків.

SHOE Knowledge Annotator призначений для створення ручних анотацій в HTML сторінках за допомогою мови SHOE. Створені анотації можуть посилатися на концепти та відношення однієї або декількох онтологій, що реалізовані в SHOE. Але процес анотування, що забезпечується переліченими засобами, є повністю ручним, і тому використання наведених інструментів для анотування великих даних не є доцільним.

Більш розвиненим є плагін-вкладка редактора онтологій Protege ONTO-H, що дозволяє створювати анотації RTF документів. Даний плагін інтегрований до редактора Protege та має можливість використовувати багато його властивостей, наприклад, браузер онтологій. Окрім функцій ручного анотування (drag&drop), даний редактор забезпечує можливість анотування частин тексту за допомогою розпізнавання іменованих сутностей, анотацій, які вже існують з тими самими іменами або іменами-синонімами і т. і., тобто надає можливість «керованого», а не повністю ручного анотування.

Розширюваний Java-застосунок MnM [20] інтегрує веб-браузер і переглядач онтології та призначений як для ручного, так й для напівавтоматичного та автоматичного анотування. Він має можливість завантажувати онтології, які зберігаються на сервері WebOnto або у файлах, або в URL-адресах на будь-якій мові онтологій: RDF (S), OWL та OCML. Анотації, які створені за допомогою цього інструменту, можна використовувати для заповнення існуючих онтологій або

приєднання до існуючого документа (в XML форматі, де тег-імена – це назви концептів, їх атрибутів та зв'язків).

Для автоматичного анотування MnM використовує механізми витягнення даних для виявлення в документі екземплярів концептів. Ці механізми навчальними на наборах анотованих текстових або html-документів, та готовий, натренований модуль генерує правила для витягнення інформації з інших документів, виявлення екземплярів концептів, значень атрибутів, екземплярів зв'язків. Потім користувачі можуть, за необхідності, відредагувати анотації, що автоматично додані модулем. MnM зберігає екземпляри в різних форматах (OCML, RDF, OWL, XML), а анотації, що ним генеруються, можуть використовуватися в різних середовищах.

Застосунок UBOT AeroSWARM [21], що розроблений як частина UBOT (UML Based Ontology Toolset) проекта, автоматично генерує RDF анотації з текстових документів. Анотатор AeroSWARM доступний в двох версіях: як веб-форма та як окремий застосунок. У веб-версії користувач відсилає текстовий файл, а AeroSWARM повертає RDF анотації для цього тексту, які створюються відповідно до OWL версій OpenCyc [22], SUMO [23] та AeroSWARM. Функція автоматичного анотування AeroSWARM підтримується системою обміну тексту AeroText, яка аналізує тест природньою мовою та витягує з нього елементи, які відповідають онтології, що використовується. Правила витягнення, які використовуються за замовченням системою AeroText, можуть бути зміненими. AeroSWARM генерує екземпляри понять (власні іменники, загальні іменники, кількісні значення валют тощо), значення атрибутів та екземпляри властивостей (наприклад, особа належить організації тощо). Оскільки AeroSWARM забезпечує анотації в RDF форматі, вони можуть використовуватись будь-яким інструментом, що підтримує RDF. Таким чином, AeroSWARM може використовуватися як сервіс автоматичного анотування для забезпечення RDF анотацій в он-лайн режимі.

Висновки

Проведені дослідження були спрямовані на виявлення загальних характеристик як самих великих даних, так і процесів їх семантизації. Це дозволило визначити узагальнену класифікацію метаданих великих даних, що є, насьогодні, найпоширенішим інструментом визначення семантичних описів великих даних, та основні аспекти та категорії процесу анотування для контенту великих даних взагалі, не прив'язуючись до специфіки конкретних типів чи форматів. Онтології є потужним та ефективним засобом семантизації. Тому, при визначені основних аспектів процесу анотування за основу приймаються онтологічні підходи. Ефективність використання онтологій обумовлюється не лише їх розвиненими властивостями семантичного опису прикладного домена, а й можливостями, які надають онтологічні мови та відповідний апарат міркування щодо встановлення подібності сутностей, екземплярів, визначення ступеня їх відповідності, класифікації сутностей та відношень контента відповідно до таксономії онтології тощо. Методологія анотування обов'язково повинна базуватися на існуючому теоретичному апараті, охоплювати вирішення задач анотування контенту та визначення семантичних анотацій, що, в свою чергу, забезпечуватиме розв'язування прикладних задач з даним контентом, як, наприклад, семантичний пошук, витягування даних, міркування тощо. Дослідження існуючих підходів дозволив визначити основні групи методів (теоретичного апарату), що є найбільш ефективними сьогодні, та проаналізувати наявні та використовувані засоби для ручного та автоматичного створення анотацій.

Література

1. https://www.ontotext.com/services/semantic-data-modeling/
2. Thabet Slimani, Taif University, Taif, Saudia Arabia, "Semantic Annotation: The Mainstay of Semantic Web". *International Journal of Computer Applications Technology and*

Research. 2013. Vol. 2. Issue 6. P. 763–770. ISSN: 2319–8656
3. http://oa.upm.es/5638/2/IJMSO_Corcho_FinalVersionPrintedInJournal.pdf
4. http://www.nlm.nih.gov/mesh/meshhome.html
5. http://www.getty.edu/research/tools/vocabulary/tgn/index.html
6. http://www.foaf-project.org/
7. http://www.ontoweb.org/
8. http://knowledgeweb.semanticweb.org/
9. http://www.esperonto.net
10. https://pubmed.ncbi.nlm.nih.gov/23734708/
11. Phesto Enock Mwakyusa. Semantic Annotation and Big Data Techniques for Patent Information Processing. Master's Thesis in Information Technology, October 10, 2017.
12. Tang, Jie, Duo Zhang, Limin Yao, and Yi Li, "Automatic Semantic Annotation Using Machine Learning". IGI Global 1:1. doi: 10.4018/978-1-60566-028-8.ch006, 2009.
13. https://studme.org/235608/informatika/proektirovanie_ontologiy_srede_protege
14. Song D., Chute C.G., Tao C. 2011. Semantator: a semi-automatic semantic annotation tool for clinical narratives. In 10th International SemanticWeb Conference (ISWC2011).
15. Cunningham H., Maynard D., Bontcheva K., Tablan V. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL '02), Philadelphia.
16. Savova G.K., Masanz J.J., Ogren P.V., Zheng J., Sohn S., Kipper-Schuler K.C., Chute C.G. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association.* 2010. 17(5). P. 507–513.
17. http://annotation.semanticweb.org/
18. Handschuh S., Staab S. and Maedche A. (2001) 'CREAM – creating relational metadata with a componentbased, ontology-driven annotation framework', in Gil, Y., Musen, M. and Shavlik, J. (Eds.): First International Conference on Knowledge Capture (KCAP'01), ACM Press, Victoria, Canadá, 1-58113-380-4. New York. P. 76–83.
19. http://www.cs.umd.edu/projects/plus/SHOE/KnowledgeAnnotator.html.
20. Vargas-Vera M., Motta E., Domingue J., Lanzoni M., Stutt A. and Ciravegna F. (2002) 'MnM: ontology driven semi-automatic and automatic support for semantic markup', in Gómez-Pérez, A. and Benjamins, V.R. (Eds.): 13th International Conference on Knowledge Engineering and Management (EKAW 2002), Springer Verlag. P. 379–391.
21. Kogut P. and Holmes W. (2001) 'AeroDAML: applying information extraction to genérate daml annotation from web pages', in Handschuh S., Dieng R. and Staab S. (Eds): KCAP'01 Workshop on Semantic Markup and Annotation, Victoria, Canadá.
22. http://www.cyc.com/2003/04/01/cyc
23. http://reliant.teknowledge.com/DAML/SUMO.owl

References

1. https://www.ontotext.com/services/semantic-data-modeling/
2. Thabet Slimani, Taif University, Taif, Saudia Arabia, "Semantic Annotation: The Mainstay of Semantic Web". *International Journal of Computer Applications Technology and Research.* 2013. Vol. 2. Issue 6. P. 763–770. ISSN: 2319–8656
3. http://oa.upm.es/5638/2/IJMSO_Corcho_FinalVersionPrintedInJournal.pdf
4. http://www.nlm.nih.gov/mesh/meshhome.html
5. http://www.getty.edu/research/tools/vocabulary/tgn/index.html
6. http://www.foaf-project.org/
7. http://www.ontoweb.org/
8. http://knowledgeweb.semanticweb.org/
9. http://www.esperonto.net
10. https://pubmed.ncbi.nlm.nih.gov/23734708/
11. Phesto Enock Mwakyusa. Semantic Annotation and Big Data Techniques for Patent Information Processing. Master's Thesis in Information Technology, October 10, 2017.
12. Tang, Jie, Duo Zhang, Limin Yao, and Yi Li, "Automatic Semantic Annotation Using Machine Learning". IGI Global 1:1. doi: 10.4018/978-1-60566-028-8.ch006, 2009.
13. https://studme.org/235608/informatika/proektirovanie_ontologiy_srede_protege
14. Song D., Chute C.G., Tao C. 2011. Semantator: a semi-automatic semantic annotation tool for clinical narratives. In 10th

International SemanticWeb Conference (ISWC2011).
15. Cunningham H., Maynard D., Bontcheva K., Tablan V. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL ’02), Philadelphia.
16. Savova G.K., Masanz J.J., Ogren P.V., Zheng J., Sohn S., Kipper-Schuler K.C., Chute C.G. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association.* 2010. 17(5). P. 507–513.
17. http://annotation.semanticweb.org/
18. Handschuh S., Staab S. and Maedche A. (2001) 'CREAM – creating relational metadata with a componentbased, ontology-driven annotation framework', in Gil, Y., Musen, M. and Shavlik, J. (Eds.): First International Conference on Knowledge Capture (KCAP'01), ACM Press, Victoria, Canadá, 1-58113-380-4. New York. P. 76–83.
19. http://www.cs.umd.edu/projects/plus/SHOE/KnowledgeAnnotator.html.
20. Vargas-Vera M., Motta E., Domingue J., Lanzoni M., Stutt A. and Ciravegna F. (2002) 'MnM: ontology driven semi-automatic and automatic support for semantic markup', in Gómez-Pérez, A. and Benjamins, V.R. (Eds.): 13th International Conference on Knowledge Engineering and Management (EKAW 2002), Springer Verlag P. 379–391.
21. Kogut P. and Holmes W. (2001) 'AeroDAML: applying information extraction to genérate daml annotation from web pages', in Handschuh S., Dieng R. and Staab S. (Eds): KCAP'01 Workshop on Semantic Markup and Annotation, Victoria, Canadá.
22. http://www.cyc.com/2003/04/01/cyc
23. http://reliant.teknowledge.com/DAML/SUMO.owl

Одержано 04.11.2020

Про автора:

Захарова Ольга Вікторівна,
кандидат технічних наук,
старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 29.
http://orcid.org/0000-0002-9579-2973.

Місце роботи автора:

Інститут програмних систем
НАН України,
проспект Академіка Глушкова, 40.
Тел.: 526 5139.

E-mail: ozakharova68@gmail.com.

УДК 004.89 https://doi.org/10.15407/pp2020.04.034

О.П. Жежерун, М.С. Рєпкін

КЛАСИФІКАЦІЙНА СИСТЕМА З ПІДБОРУ ПЕРСОНАЛУ, БАЗОВАНА НА АНАЛІЗАТОРІ УКРАЇНСЬКОЇ МОВИ

У статті розглядається класифікаційна системах, яка базується на аналізі природньої мови. В багатьох таких системах використовуються нейронні мережі, проте вони потребують даних для навчання, які не завжди наявні. Автори пропонують використання онтологій в подібних системах аналізу природньої мови. В якості прикладу представлено класифікаційну систему, яка допомагає сформувати список найкращих кандидатів під час підбору персоналу. Представлено огляд методів побудови онтологій та мовних аналізаторів, доречних для класифікаційних систем, і побудовано систему у вигляді бази знань. Здійснена підтримка української та англійської мов у класифікаційній системі. Описані можливості розширення системи.

Ключові слова: класифікаційна система, база знань, онтологія, Protégé, аналіз природньої мови, аналіз української мови.

Вступ

Сьогодні у світі розробки програмного забезпечення та бізнесу зростає тенденція із зменшення присутності людини у повсякденних завданнях, особливо це стосується завдань монотонних або легко алгоритмізованих. Одна з причин цієї тенденції – значний зріст обсягу робіт такого типу, які людям обробити стає все важче. Разом із такою проблемою прогрес приносить і часткові вирішення у вигляді зростання обчислювальної потужності сучасних комп'ютерів. Це дає інженерам можливості використання таких алгоритмів, які виконуватимуть монотонні завдання за розумний час [1].

У ході цієї роботи, яка є продовженням статті «Класифікаційна система з підбору персоналу» [2], було розглянуто одне з таких завдань та запропоновано метод його вирішення. Це завдання вибору кандидатів серед великого списку претендентів на одну з посад у компанії. Щодня рекрутери переглядають десятки чи сотні резюме та обирають найбільш релевантні. Ефективність такого пошуку не є високою, тому було вирішено автоматизувати саме цей процес. Як основу системи було побудовано онтологію, яка описує ієрархію та відношення навичок, навчальних закладів та компаній для найбільш ефективного та обґрунтованого вибору кандидатів.

Онтології для побудови баз знань, засоби реалізації та застосування

Поняття онтології з'явилось ще за часів античності, більше ніж дві тисячі років до появи інформаційних технологій. У ті часи поняття онтології з'явилось у межах філософської науки. Онтологія у філософії – це вчення про буття, у якому з'ясовуються фундаментальні проблеми існування [3].

В інженерії "онтологічні" методи зазвичай застосовуються для побудови моделей процесів. Інженерна модель процесу і є онтологією. Точніше, онтологіє описує цю модель. Такі описи моделей є формальними, тобто зроблені спеціальною для цього мовою, конструкції якої завжди інтерпретуються точно та однозначно. Це дозволяє, наприклад, перевірити чи не існує в цьому процесі логічних протиріч [4]. Для зручної побудови необхідної онтології було обрано інструмент Protégé. Цей додаток надає зручний інтерфейс та містить в собі багато допоміжного інструментарію: збереження побудованої онтології у восьми різних форматах, вісім найпопулярніших різонерів та автоматична генерація Java коду на основі онтології.

Для побудови онтології система Protégé оперує наступними поняттями:

- Class – головний елемент будь-якої онтології. За їх допомогою описується ієрархія предметної області.

- Individual – екземпляр класу.
- Data property – властивості класів. Зазвичай є простими типами даних, наприклад integer/string.
- Object property – зв'язки класів між собою.

Одним з необхідних інструментів, який надає Protégé, є різонер. Це програмний застосунок, який аналізує побудовану ієрархію та зв'язки між її сегментами. Він є універсальним, тому не обов'язково будувати власний різонер під свою онтологію. Проте, якщо заданого функціоналу та точності не вистачає, то, звичайно, доведеться дописувати свої модулі до існуючого рішення [5]. Ось їх список:

- ELK
- FaCT
- HermiT
- Mastro DL-Lite Reasoner
- Ontop
- Pellet
- Pellet(Incremental)
- Jcel

Алгоритми обробки та аналізу тексту

1. Стемінг. В інформаційному пошуку такий алгоритм як стемінг відомий дуже давно. Він є базою для цієї гілки програмної інженерії. Цей алгоритм дозволяє знаходити основу слова, при цьому вона може не збігатися з коренем слова, тому цей алгоритм є евристикою. Проте він є основним алгоритмом компанії Google. Морфологія на основі стемінгу має декілька переваг. Наприклад, завдяки зменшенню обсягу інформації зростає швидкість аналізу. Найголовнішою перевагою цього алгоритму є той факт, що навіть за відсутності словника основ слів ми отримуємо морфологічну базу необмеженого розміру. При цьому морфологія ніколи не скаже, що такого слова немає у словнику [6].

Серед недоліків – невисока точність методу та неможливість синтезу на базі без основ. Без основи ми не зможемо зрозуміти чи є слово дією, яку необхідно виконати над певним об'єктом, чи це об'єкт цієї дії і т. д.

У результатах роботи алгоритму стемінгу ми бачимо, що у всіх випадках просто відкидається закінчення слова: в англійській мові закінчення, що характеризує множину об'єкту – "s", дієслово в минулому часі – "ed". Для коректної роботи алгоритму треба оновлювати список таких закінчень в залежності від мови, яку ви аналізуєте.

2. Лематизація. Як зазначено вище, стемінг – це кит, на якому стоїть інформаційний пошук. Лематизація – це другий кит. Цей алгоритм схожий на стемінг за своєю метою, яка полягає у зменшенні кількості інформації та приведення до однієї основи [7]. На відміну від стемінгу, який не гарантує, що результатом його роботи буде корінь слова, лематизація гарантує виокремлення кореня, або трансформацію слова до кореня. Для роботи цього методу недостатньо зберігати закінчення та приставки слів у вибраній мові. Цей алгоритм вимагає збереження всього словника мови, тому що, наприклад, до слів у множині може не тільки додаватись відповідне закінчення, а ще повністю змінюватись слово. Теж саме можна сказати про дієслова у різних часах.

3. POS-tagging (Part Of Speech tagging) – алгоритм класифікації слів за частинами мови, який також є основним алгоритмом інформаційного пошуку. Сам по собі алгоритм є низького рівня, за допомогою якого можна побудувати свою пошукову систему. Алгоритм не має певної послідовності дій, адже в кожній мові існують свої правила морфології, але найчастіша імплементація – це виокремлення закінчення слова, його аналіз і класифікація всього слова [8].

Також використовується аналіз порядку слів для мов із фіксованим порядком у реченні. У роботі використано бібліотеку nltk для англійскьої та pullenti для української, які класифікують наступні типи слів:

- CC – координуючі сполучники
- DT – коми, крапки…
- FW – іноземне слово
- JJ – прикметник
- JJR – порівняльна форма прикметника
- JJS – вища порівняльна форма
- NN – іменник

- VB – дієслово (інфінітив)
- VBD – дієслово в минулому часі
- та інші…

Огляд побудованої онтології

Онтологія, яка була побудована у редакторі Protégé показана на рис. 1.

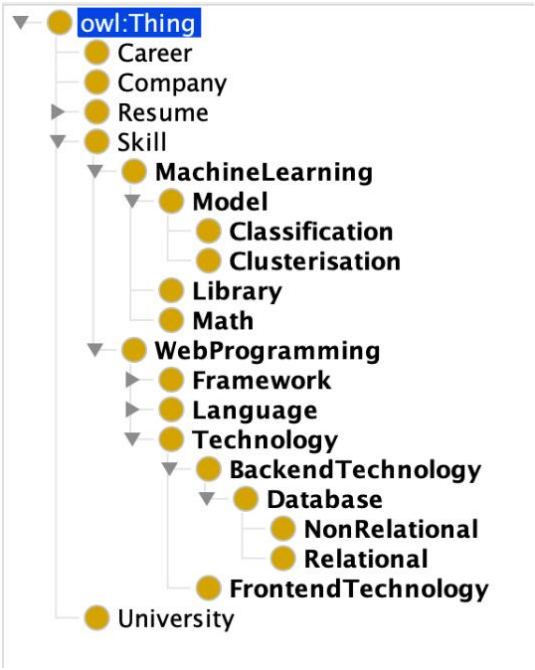

Рис. 1. Побудована онтологія

Далі опишемо тільки основні класи онтології.

Career. Клас використовується для опису вакансій в компанії та є підкласом головного класу “Thing”, тобто є класом найвищого рівня в цій онтології. Має object property “requireSkill”, що описує необхідні навички для посади. Та має такі data properties: “career_id”, “career_name”.

Resume. Є абстрактним класом та зберігає дані про користувачів, які лишили свої резюме на обрану вакансію. Має наступні data properties: “resume_id” – унікальний номер резюме, “resume_file_name” – шлях, де збережено файл із резюме, “resume_link” – всі покликання в файлі з резюме, “candidate_terms” – усі можливі терміни, які зустрілись в резюме, але вони ще не наявні в базі знань, щоб адміністратор зміг занести їх в один з класів онтології. Має object property – “resumeHasSkill”, що зберігає посилання на всі навички, що було знайдено у файлі з резюме.

Company. В цьому класі зберігаються усі IT компанії за 2020 рік . Має наступні data property – “company_id” – унікальний номер компанії, “company_name” – назва компанії.

University. В цьому класі зібрані всі університети України . Має data property – “university_id” – унікальний номер університету, “university_name” – назва університету.

Skill. Є абстрактним класом, він визначає можливості користувачів та необхідні навички для здобуття визначеної роботи. Має наступні data properties – “skill_id” – визначає унікальний номер навички та “skill_name” – ім’я навички, що буде показуватись користувачу.

Огляд розробленого аналізатора текста

Побудований аналізатор текста працює з файлами у формати “PDF”. Коли кандидат надсилає своє резюме, система зберігає його на диск в директорію, яка відповідає обраній кандидатом вакансії. Після цього аналізатор починає свою роботу. Далі будуть описані основні етапи роботи аналізатора з прикладами.

Етап 1. Прийняття резюме та попередня обробка (рис. 2).

Цей етап передбачає виконання наступних кроків.

1. Відкриття файлу.
2. Видалення всіх несуттєвих символів, які не містять інформації. Наприклад, серед них є “\n”, “\r”, “|” та інші.
3. Пошук та видалення в тексті всіх адрес електронної пошти та телефонних номерів українського стандарту. Пошук здійснюється за допомогою регулярних виразів. Все, знаходиться на цьому етапі, зберігається в онтології в класі “Resume” в data property “resume_link” та “resume_email”.
4. Переведення тексту до нижнього регістру.
5. Видалення “стоп слів”, які не містять інформації. Список цих слів було взято з бібліотеки “get_stop_words” (рис. 3).

Після виконання усіх зазначених кроків першого етапу надіслане резюме має вигляд (рис. 4).

Максим Рєпкін

ryepkin.maksym98@gmail.com | https://github.com/maxflexx

Навички

· Python, C++

· Знання базових алгоритмів комп'ютерного зіру(Canny edge detection, contour curvature, segmentation, boundary tracing, erosion)

· Лінійна алгебра

· Теорія ймовірностей

· Алгоритми та структури даних

· Розуміння алгоритмів машинного навчання

Власні проекти

В рамках курсової роботи було розроблено систему запису на вибіркові дисципліни за допомогою технології Laravel та HTML.

https://github.com/maxflexx/kursova_php

Реалізував алгоритм, який допомагає знаходити схожі картинки(однакові, картинки з додатковими ефектами, однакові об'єкти з інших кутів), використовуючи ssim/

https://github.com/maxflexx/image-similarity

Рис. 2. Приклад надісланого резюме

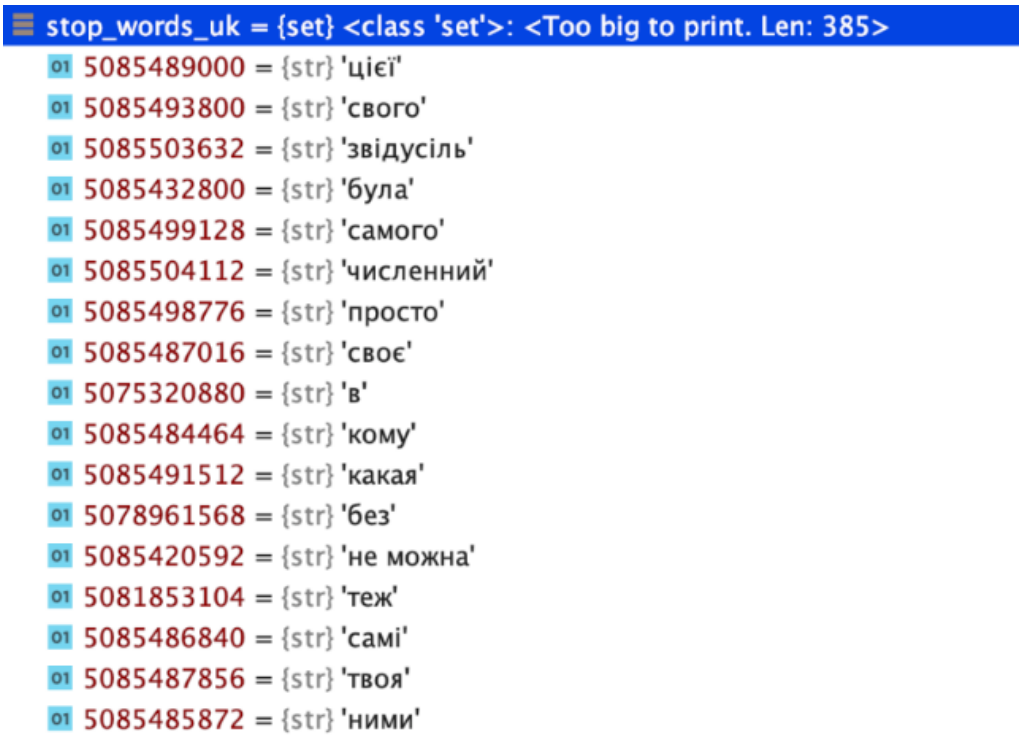

Рис. 3. Приклад «стоп-слів»

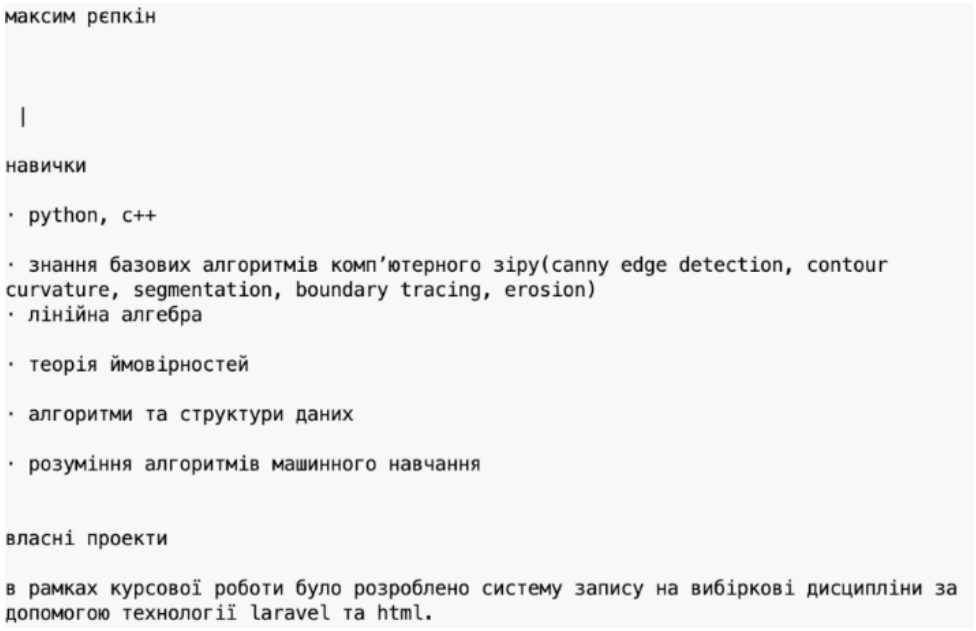

Рис. 4. Резюме після першого етапу

Етап 2. На цьому етапі здійснюється пошук по термінах, які вже наявні в онтології. Під час цього процесу виділяються терміни таких типів.

1. Компанії, в яких кандидат працював раніше. База IT компаній була взята з сайту dou.ua і нараховує 7400 позицій.
2. Університет, в якому навчався кандидат. База університетів була взята з сайту vstup.info.
3. Навички, якими володіє кандидат. Цю базу було створено вручну, враховуючи всі популярні мови програмування, фреймворки та інші технології.

Після виконання зазначених кроків резюме має вигляд (рис. 5).

Етап 3. На цьому етапі здійснюється пошук усіх можливих термінів. Для виконання цього завдання здійснюється лематизація та POS-tagging. Можливі терміни мають задовольняти такі умови.

1. Бути іменником, прикметником або дієсловом.
2. Якщо це термін з кількох слів, то між словами не повинно бути розділових знаків та пропусків рядків.

Результат виконання цього етапу виглядає так (рис. 6).

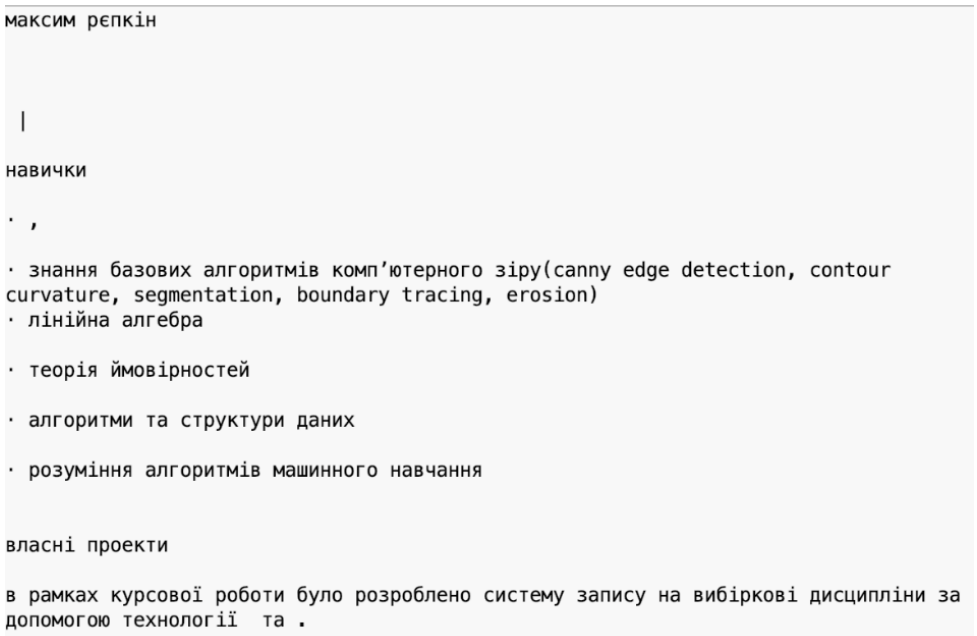

Рис. 5. Резюме після другого етапу

```
00 = {str} 'максим'
01 = {str} 'навичка'
02 = {str} 'знання базовий алгоритм компютерний зір'
03 = {str} 'лінійний алгебра'
04 = {str} 'теорія ймовірність'
05 = {str} 'алгоритм структура даний'
06 = {str} 'розуміння алгоритм машинний навчання'
07 = {str} 'власний проект'
08 = {str} 'рамка курсовий робота розробити система запис вибірковий дисципліна'
09 = {str} 'допомога технологія'
```

Рис. 6. Результат третього етапу

Всі можливі терміни зберігаються в онтології в класі “CandidateTerms”, щоб адміністратор мав змогу продивитись ці терміни та прийняти рішення щодо того, чи дійсно вони є термінами, які необхідно додати.

Швидкість аналізу резюме складає приблизно 1MB / секунда. Це достатньо швидко, зважаючи на те, що загалом розмір резюме не перевищує 5MB.

Класифікаційна система здатна аналізувати дві мови: українську і англійську. Для англійської всі кроки не відрізняються, проте використовується бібліотека “nltk”, а не “pullenti”. Система побудована таким чином, що додавання нових мов у аналізатор буде відбуватись без переписування або редагування вже наявного коду, що робить її легко розширюваною.

Можливості розширення

Є декілька шляхів розширення представленої системи:

1. Розширення онтології, щоб робити більш широку класифікацію.
2. Удосконалення системи аналізу природньої мови. Додавання нових мов та оптимізація здійснених операцій, щоб підвищити швидкість аналізу.
3. Додавання можливості горизонтального розширення, що дозволить запускати кластери машин, які будуть аналізувати резюме та додавати їх до онтології.
4. Додавання можливості робити висновки щодо університетів, наприклад, “Випускники університету N більше розуміються на технологій K”.
5. Додавання можливості робити висновки щодо компаній. Так можна скласти нову онтологію: Компанія-Технологія.

Висновки

Після аналізу можливостей онтології можна зробити висновок, що така технологія є доречною у вирішенні розглянутої проблеми, завдяки своїм класифікаційним можливостям та можливостям логічного виводу. Також з цього аналізу можна зробити висновок, що онтологія має великі переваги над нейронними мережами в контексті розглянутої задачі, бо нам не потрібно заново тренувати модель кожного разу, коли змінюються дані. На великих об’ємах даних це є критичним.

Література

1. Glybovets A., Glybovets M., Polyakov M. Intelligent networks. NaUKMA. Dnipropetrovsk. 2014. 462 p.
2. Жежерун О.П., Рєпкін М.С. Класифікаційна система з підбору персоналу. Наукові записки НаУКМА. 2019.
3. Шинкарук В. І. Філософський енциклопедичний словник. Київ: Інститут філософії імені Григорія Сковороди: Абрис. 2002. 742 с.
4. Лапшин В. А. Онтологии в информационных системах. Современный подход. Москва. 2009.
5. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies [Електронний ресурс].

6. Lovins Julie Beth. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*. 1968. T. 11.
7. Manning Christopher D., Raghavan Prabhakar, Schütze Hinrich. Introduction to Information Retreival. Cambridge University Press.
8. DeRose Steven J. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 1988.

References

1. Glybovets A., Glybovets M., Polyakov M. Intelligent networks. NaUKMA. Dnipropetrovsk. 2014. 462 p.
2. Zhezherun O., Repkin M. Classification system for personnel selection. *Scientific Notes of NaUKMA*. 2019.
3. Shynkaruk V. Philosophical encyclopedic dictionary. Kyiv: Grigory Skovoroda Institute of Philosophy: Abris. 2002. 742 p.
4. Lapshin V. Ontologies in information systems. Modern approach. Moscow. 2009.
5. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies [Електронний ресурс].
6. Lovins Julie Beth. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*. 1968. T. 11.
7. Manning Christopher D., Raghavan Prabhakar, Schütze Hinrich. Introduction to Information Retreival. Cambridge University Press.
8. DeRose Steven J. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 1988.

Одержано 18.10.2020

Про авторів:

Жежерун Олександр Петрович,
кандидат фізико-математичних наук,
завідувач кафедри мультимедійних систем факультету інформатики.
Кількість наукових публікацій в українських виданнях – більше 80.
Індекс Гірша – 4.

Рєпкін Максим Сергійович,
студент 1 року навчання магістерської програми «Інженерія програмного забезпечення» факультету інформатики.
Кількість наукових публікацій в українських виданнях – 1.

Місце роботи авторів:

Національний університет
«Києво-Могилянська академія»
04655, Київ, вул. Г. Сковороди, 2.

E-mail: zhezherun@ukma.edu.ua,
ryepkin.maksym98@gmail.com

UDC 004.62 https://doi.org/10.15407/pp2020.04.041

I.S. Chystiakova

IMPLEMENTATION OF MAPPINGS BETWEEN THE DESCRIPTION LOGIC AND THE BINARY RELATIONAL DATA MODEL ON THE RDF LEVEL

This paper is dedicated to the data integration problem. In article the task of practical implementation of mappings between description logic and a binary relational data model is discussed. This method was formulated earlier at a theoretical level. A practical technique to test mapping engines using RDF is provided in the current paper. The mappings DL ALC and its main extensions to the RDF triplets are described in the publication. The mapping of the DL axioms into an RDF triplet also is considered in the publication. The main difficulties in describing DL-to-RDF transformations are given in the corresponding section. The paper also provides an overview of existing methods that relate to the use of RDF when mapping RDB to ontology and vice versa.

Key words: binary relational data model, description logic, mapping, RDF, DL, RM^2, ALC, OWL.

Introduction

The research series [1–7] is dedicated to the analysis and solution of the problem of creating a mapping mechanism between the description logic (DL) and the relational data model (RDM). It took place as a part of the complex problem of data integration, the analysis of which can be found in [7]. The mentioned series provides an overview of the current existing approaches to address the problem of mapping. According to the result of analysis a taxonomy of research on the subject was created. This result also revealed a number of disadvantages of the existing approaches to establish one-to-one correspondences between the description logic and the relational data model. Based on this **a binary relational data model (RM^2)** was proposed as an integrating model for the creation of mappings. Complete and detailed description of it can be found in [1]. Information about the interaction between RM^2 and the classical relational data model can be found in [2].

The mechanism for mapping the ALC description logic and its main extensions to RM^2 was developed and described in [6], as well as classical RDM to RM^2. In the publication [6] you can get acquainted with the argumentation of the following statement: description logic can be considered as an independent data model. It also describes in detail why DL ALC is used in the developed approach, justifies the choice of ALC extensions and outlines the way they are mapped in RM^2.

Until now a significant drawback of this approach has been the lack of any practical testing of the proposed results. The description of mappings using RM^2 is purely theoretical. A real practical check can make significant changes both in the structure of the approach itself and in its main individual components, e.g. to complement or restrict the operations of a binary relational algebra (RA^2), which is a constituent part of RM^2. In the current paper a method for checking mappings between the description logic and the binary relational data model using RDF graphs is proposed.

Section 1 is dedicated to the analysis of the number of practical works on the implementation of mappings using RDF. Section 2 formulates the problem of practical approbation of the approach to the description of mappings between DL and RDM. Section 3 outlines a method for mapping the description logic ALC and its extensions to RDF using OWL 2. Conclusions can be found in the section 4.

Related work

The publications [1, 6] provide an overview of the current approaches to address the problem of mapping. According to the result of analysis a taxonomy of research on the subject was created. It is shown in Fig. 1.

The results given in [1, 6] will not be duplicated in the current paper. On the contrary, this section is dedicated to works that were not included in the above survey. The publications that will be overviewed in this part of the article are intended to complement the existing taxonomy. They will be classified and will take their place in the hierarchy of the body of research on the establishment of correspondences between ontologies and relational databases. This overview focuses on those methods that concern the use of RDF at RDB-to-ontology mapping or ontology-to-RDB mapping. There is a need to allocate a place in the corresponding column of the taxonomy for the researches that will be considered in the current section. Also, it is necessary to formulate a number of intermediate conclusions that are necessary for setting the task of a current work.

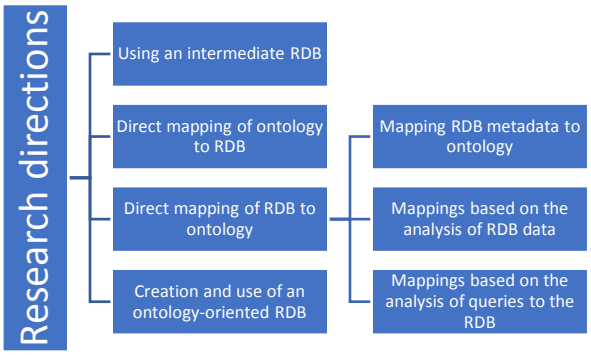

Fig. 1. The approaches to describe mappings between DL and RDB research taxonomy

The first step is to consider the fundamental work [8]. It belongs to the following section of the classification: *direct mapping of RDB in ontology*, the direction is *mappings taking into account the analysis of RDB data*. It should be noted the following features of this study:

1. *A clear distinction between the concepts of "mapping RDB into ontology" and "transformation of RDB into ontology"*. If in the first case interaction between the existing ontology and the database is supposed, then in the other case it is supposed to create a completely new ontology based on the data and the RDM structure. The authors of the approach attribute it to the second case.

2. *Using RDF in the transformation rules.* A large amount of literature provides a set of rules that map RDB constructs directly into the ontology construct, without using an intermediate model [8]. Such an ontology is built as a result of a direct transformation from a semantically less developed database. It suffers from a number of serious shortcomings, which are mentioned by the authors of study. The difference of this approach from the others is that an independent attempt to transform the RDB structure and data into an OWL structure that is built with the help of RDF and RDFS using XSD is made. The database data is converted into regular RDF triples.

In fact, this is one of the first serious studies in the field of establishing interoperability between ontologies and RDB using RDF. The authors of the methodology do not use the R2R ML language, which has the official W3C specification. However, in the section dedicated to the source analysis the existence of various approaches that use both RDF-OWL constructs and those using R2R ML is mentioned. The R2R ML itself will be discussed below.

This approach has a serious drawback: there is no mention of how the operations of relational algebra are mapped. In their continued research [9], the authors of the technique tried to eliminate it. Based on the rules for transforming the structural part of the database and its data, the rules for mapping the relational algebra operators (expressed in SQL) into equivalent SPARQL queries were made. If we consider this approach in a complex of two works [8, 9], then it can be retrained and referred to the taxonomy section as the *direct mapping of RDB in ontology*, the direction is *mappings, taking into account the analysis of queries to the RDB.*

Generally, this approach does not stand out from the general mass of other methods in its category. It has the same drawbacks: the one-sidedness of the mechanism (although the authors note that the description of the mapping rules in the opposite direction remains in the field of future research), the absence of a formal approach, the separation of the structural and manipula-

tive parts of the RDB within a unified approach, as well as the silence that ontologies and RDB obeys two different open and closed world hypotheses. It remains unclear how the interaction between these two very different concepts will be carried out.

It should be mentioned that in studies dedicated to the mappings subject matter, a number of works that present their specification of mapping languages have appeared. For example, in [10, 11] the RDB2OWL Mapping Language specification is declared, where the specification and tools for mapping implementation are presented. However, despite the deep study of the topic and the presence of serious practical testing, RDB2OWL ML operates exclusively with the structural part of RDM. The approach pays attention to the aspect of integrity constraints but does not address the relational algebra operations converting.

Analyzing the works dedicated to the practical implementation of mappings, there was revealed the following tendency: many successful techniques increasingly prefer to work not directly with OWL, but using RDF and RDFS, in conjunction with XML data types. There are the following reasons for this. Over the years the W3C has released two official specifications for mapping mechanisms between OWL, RDF, and RDB. One of them [12] describes the mapping mechanism of the OWL ontology into an RDF triplet. The document provides an exhaustive list of rules for transforming each OWL construct into a set of RDF triplets. It also contains a description of the mechanism for transforming RDF triplets into the corresponding OWL constructs, with the necessary list of transformation rules. Another document [13] declares the R2R ML (RDB to RDF Mapping Language) language specification, which describes the mechanism for mapping a relational database to the set of RDF triplets. Thus, RDF is established by default as the intermediate stage in mappings between DL and RDM.

This is argued as follows.

1. Looking at the semantic web pie's stack architecture, it is seen that OWL and RDF are adjacent layers. That is, a very important task is to show the way of interaction between two parts of the same concept.

2. Accessing data from the “deep web”. This term refers to data that is very difficult to index with standard search engines. These include, for example, unstructured documents (pictures, scanned copies), semi-structured (CSV, PDF files), structured data sources (RDB, XML databases, NoSQL databases, LDAP directories). However, to ensure the sustainability of the applications that were developed along with the data they exploit, and to leverage the properties engineered into RDBs for decades (scalability, ACID properties, security, performance optimization), the data should remain hosted and delivered by the legacy RDBs. This situation creates a need for RDB-to-RDF methods that can access relational data and convert it into RDF triplets.

3. Linked data. Linking open data to other related pieces of data increases its value. Driven by recommendations proposed by Tim Berners-Lee [14] the Linking Open Data community project aims at extending today`s web by publishing various open data sets in the RDF model and setting RDF links between data sources. This is done in order to enable developers of new programs and applications to use existing data in a new capacity, i.e. create added value by repurposing datasets, using the data in some new way, possibly beyond what data providers may have initially expected.

4. Integration of heterogeneous sources. This point has been discussed several times earlier. However, it is worth reminding the main aspects. In the modern web, there is an acute problem of using not just data, but also their semantics. Relational schemas usually convey no or poor semantics. This means that it is necessary to define somehow the semantics of the data stored in RDB in an explicit machine-readable form. Using RDF as a format for representing relational data appears as a powerful and promising method to achieve such data integration, in which RDB-to-RDF methods will pay a key role.

The last three points were taken from a fundamental review [15] of methods and tools for converting RDB to RDF. The paper

describes similar studies that were conducted earlier. It notes that no RDB-to-RDF research has been conducted since the publication of R2R ML in 2012. As a result, none of the articles cited in the overview didn`t review the R2R ML compliant tools.

The authors in [15] proposed the following classification of RDB-to-RDF research areas:

- description of mappings (mapping type, expression);
- implementation of mappings (when and how data is converted to RDF);
- data retrieval method (query-based methods, related data).

According to this classification, 17 approaches are ordered in the review. There is also a separate detailed analysis of the R2R ML language. The conclusions to the work indicate that it is a promising language, which, however, may not be applicable to the entire wide range of RDB-to-RDF mapping needs, leaving room for future research.

In this regard, in current paper for a practical test of the implementation of mappings between DL and RDM, the task was formulated: to describe a way to check mappings between the description logic and a binary relational data model using RDF graphs. A detailed description of the problem statement is presented in the next section.

Problem statement

Before setting the task, it is necessary to give a brief description of the proposed theoretical approach for mappings creation between description logic and the relational data model. The results of the description of mappings DL to RM^2 can be found in [1]. Here is a summary of their essence.

The mappings DL to RM^2 can be split into the following components:

1. *To build a conceptual information model of DL and to transform it into RM^2*. One of the main tasks of the conceptual information model of any subject area is to define the basic concepts and to describe their properties and relationships. The ER language is one of the most used for this purpose. It assumes that a conceptual information structure is described using concepts such as entity, attribute and relationship. The conceptual information model of description logic with a detailed description of its components can be found in [6]. It also contains the RM^2 scheme, which corresponds to the given ER-model.

2. *To map DL ALC into RM^2*. Any description logic consists of two conceptual parts: syntax and semantics. The latter is specified through the interpretation concept. Interpretation is a pair $I = (\Delta, \bullet^I)$, where Δ is a non-empty set called the domain of interpretation and $\bullet^I$ is an interpretation function, which assigns to each atomic concept A a set $A^I \subseteq \Delta$ and to each atomic role R a binary relation $R^I \subseteq \Delta \times \Delta$. In turn, RDM operates with the set-theoretical concepts of intension and extension. The establishment of such correspondences between the components of the DL syntax and the RM^2 intension, in which the semantics of the DL expression will be equal to the extension of the corresponding RM^2 expression will be called ***the mapping***. The theoretical representation of formulas for converting the DL ALC syntax into RM^2 can be found in [6].

3. *To map DL ALC extensions into RM^2*. There are many different DL dialects. They represent a basis of ALC logic extended with one or more operations. For example, DL SHOIQ denotes the presence of all ALC syntax operators, and also includes operations of number restrictions, nominals, and there is also a role hierarchy, transitive and inverse roles. More information about how to create an ALC extensions by adding a new operation to it can be found here [16]. Also, a detailed description of which extensions were covered by the theoretical research with their mapping in RM^2 can be found in [3–5].

Based on the analysis in the previous section, the following idea arose to test the mapping mechanism between DL and RDM. It is known that the mathematical basis of any ontology describing language is description logic. Thus, all constructors of concepts and roles that are present in the foundational DL are reflected in the toolbox of the corresponding language. OWL 2 is no exception. It also has the official W3C specification. Based on

this fact to set the approbation problem of the theoretical part of the description of mappings between DL and RDM the following idea is proposed. Description logic statements expressed in OWL 2 are mapped to RDF triplets using OWL-to-RDF conversion rules on the one hand, and RDB expressions to RDF triplets are mapped using R2R ML on the other hand. The resulting graphs are compared by equivalence criteria.

The idea of such an implementation is schematically shown in Fig. 2.

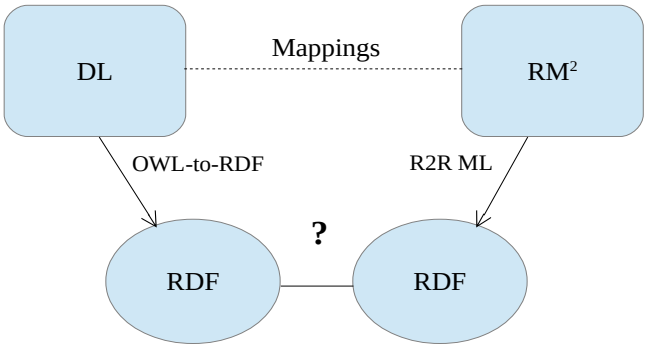

Fig. 2. The mapping method between DL and RM^2 checking scheme

The idea of proof is not new. The global meaning is to transform a statement in a new theory into statements of some other existing theory. The next step is to prove the statement obtained as a result of the transformation within the framework of those methods and proofs of the established concept. If in the existing theory this statement is true, then in the area being proved the correlated expression is also true.

In our case, there is some statement of description logic that is mapped into a statement of a binary relational data model. Such a statement is represented in an OWL 2-expression from the DL side. Further such an expression is transformed into RDF triplets, forming an RDF graph, using the official W3C rules for mapping OWL-RDF. The statement is formulated in terms of RDB from the RM^2 side. Such a statement is then transformed into RDF triplets using the R2R ML language [13]. The resulting triples constitute an RDF graph. Thus, as a result of such transformations, two RDF graphs are obtained. They are proposed to be compared. If they are equivalent, then the DL-to-RM^2 mapping formula is true.

Here some points should be mentioned. As known [17], OWL 2 is based on the SROIQ description logic. Thus, in the documentation on OWL-to-RDF mapping [12], all issues related to both the basic syntax of DL ALC and the main extensions (concepts and roles hierarchy, nominals, number restrictions, inverse roles, DL axiomatics, as well as some of the roles restrictions) are worked out in detail. However, the scope of mapping OWL 2 to RDF is limited only by those operations that are present in DL SROIQ. The issue of mapping some of the role constructors in RDF, for which the theoretical part of DL-to-RM^2 mappings has been worked out, remains open.

The question of converting RDM to RDF is not so simple. Obviously, R2R ML allows you to transform the RDB structure and integrity constraints into RDF triplets. However, the way how to map the manipulative part of RDM without using the SPARQL query language have not yet been found. A mapping method of the operations of relational algebra is currently being investigated.

The key question of the approbation problem is to prove the equivalence of the resulting graphs as a result of pairwise mapping of statements DL and RM^2. In the course of research, the conclusion that an RDF graph is a special case of an ordinary graph was formulated. This means that the question of equivalence is leaded to proving their isomorphism. It was found that such a problem has already been investigated in [17]. It analyzes an RDF graph as a special case. Also all isomorphism criteria for the general case were studied. On the basis of these criteria three necessary and sufficient conditions for the equivalence of RDF graphs are formulated. Let's list them:

1. *Equal number of vertices*. Both graphs must contain the same number of vertices. Otherwise they are not isomorphic.

2. *Equivalence of vertices*. In a pairwise comparison, each vertex of one graph must have an equivalent in the other graph. Otherwise, such graphs are not isomorphic.

3. *Equivalence edges*. In a pairwise comparison, each edge of one graph must have an equivalent in the other graph. Otherwise, such graphs are not isomorphic.

The last question that needs to be worked out within the task is reducing the graph to a self-isomorphic. As a result of mappings, at the RDF level, a situation may arise when the vertex of one graph will semantically correspond to a subgraph from the graph with which the comparison is made. On the RDF level as a result of mappings may arise such a situation: the vertex of one graph will semantically correspond to a subgraph from the graph with which the comparison is made. Such a subgraph can consist of several vertices connected by edges. This situation should be assumed as a result of such a fact: when a statement is mapped to RDF, a large number of anonymous (empty) nodes arise, which, nevertheless, have their own semantic purpose. Thus, the question of reducing an RDF graph to a self-isomorphic remains open.

Mapping DL to RDF

It is known [18] that all modern description logics are based on the simplest version of DL ALC. This means that it is fully included in the DL SROIQ. Therefore, OWL 2 uses all the functionality of ALC.

The syntax for this logic is defined as follows:

$$\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C,$$

where A – atomic concept, R – atomic role, C, D – concept.

The concept of DL semantics does not play a significant role in the context of mapping to RDF, so in this article there is no focus on this point.

The OWL 2 ontology and the mapped graph are related as follows:

$$G = T(O),$$

where G – graph, O – ontology, T – the mapping function.

Before proceeding to the description of the mappings, a number of designations should be given. To describe OWL 2 constructs the OWL Abstract Syntax will be used. To describe RDF expressions, the standard triples and serialization to N3 notation will be used.

Table 1 shows the notation for the main namespaces.

The notation T (SEQ y_1, …, y_n) shows the translation of a sequence of the OWL objects from a structural specification into an RDF collection. A few words should be said about this way of organizing resources. An anonymous node, which belongs to the rdf: list class, and two types of predicates act as a subject to create a collection. The predicates are as follows:

- rdf: first – the first element of the collection (head);
- rdf: rest – the link to sub-collection (tail).

Table 1. Namespace notation

Prefix name	Expansion
@prefix rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
@prefix rdfs:	http://www.w3.org/2000/01/rdf-schema#
@prefix owl:	http://www.w3.org/2002/07/owl#
@prefix xsd:	http://www.w3.org/2001/XMLSchema#

The head of the collection points to its first element (an RDF triple object). The tail contains the remaining items, which are also organized into a collection (a sub-collection of the main collection). It looks the same way – a node (subject) and two predicates (head and tail), one of which points to the object, and the second points to the remaining elements. The tail that contains the last element of the collection points to the built-in resource rdf: nil.

_: x will denote an anonymous RDF triplet node.

So, let's describe the mappings of the description logic ALC to RDF using OWL 2. Table 2 shows the mapping rules for all components of the ALC syntax.

Table 2. DL ALC to RDF mapping rules

ALC	OWL 2	RDF
⊤	owl:Thing	-
⊥	owl:Nothing	-
C, D	Declaration (Class (C)), Declaration (Class (D))	T(C) rdf:type owl:Class, T(D) rdf:type owl:Class
R	Declaration (ObjectProperty(R)) Declaration (DatatypeProperty(R))	T(R) rdf:type owl:ObjectProperty, T(R) rdf:type owl:DatatypeProperty
¬C	ObjectComplementOf(C)	_:x rdf:type owl:Class _:x owl:complementOf T(C)
C⊓D	ObjectIntersectionOf(C, D)	_:x rdf:type owl:Class _:x owl:intersectionOf T(SEQ C, D)
C⊔D	ObjectUnionOf(C, D)	_:x rdf:type owl:Class _:x owl:unionOf T(SEQ C, D)
∃R.C	ObjectSomeValuesFrom(R C)	_:x rdf:type owl:Restriction _:x owl:onProperty T(R) _x: owl:someValuesFrom T(C)
∀R.C	ObjectAllValuesFrom(R C)	_:x rdf:type owl:Restriction _:x owl:onProperty T(R) _x: owl:allValuesFrom T(C)

The concepts ⊤ and ⊥ are represented in OWL by the special classes – owl:Thing and owl:Nothing. When constructing RDF triplets, these classes are used in the same way as within OWL itself.

The images of RDF triplets and their serialization to N3 notation are shown below.

Definition of concepts C and D

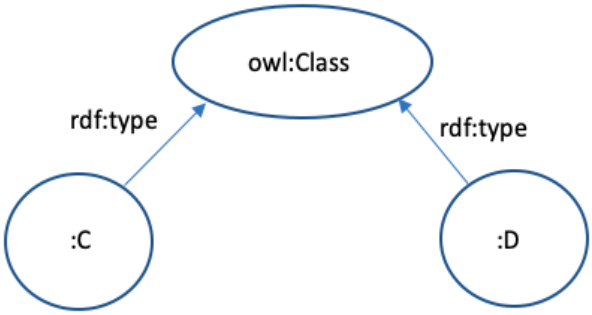

Fig. 3. RDF triplets of C and D definition image

```
@prefix : <http://example.com/Ch.owl#>
        :C rdf:type owl:Class.
        :D rdf:type owl:Class.
```

Definition of the role R

Fig. 4. RDF triplets of the object property R definition image

Fig. 5. RDF triplets of the datatype property R definition image

Since the role R can describe both an object property and a datatype property, two RDF triplets for this element were defined. N3 serialization format for two triples is also present.

```
@prefix : <http://example.com/Ch.owl#>
```

:R rdf:type owl:ObjectProperty.
@prefix : <http://example.com/Ch.owl#>
:R rdf:type owl:DatatypeProperty.

Complement of concept C

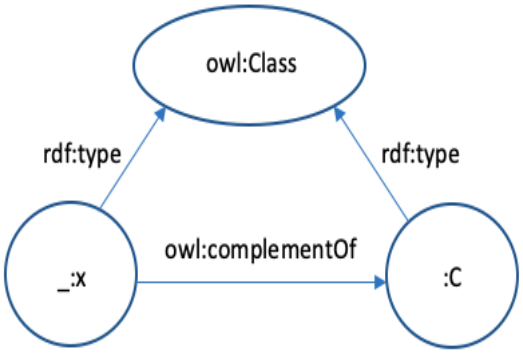

Fig. 6. RDF triplets of the complement of concept C image

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Class;
owl:complementOf :C.

Concept intersection C ⊓ D

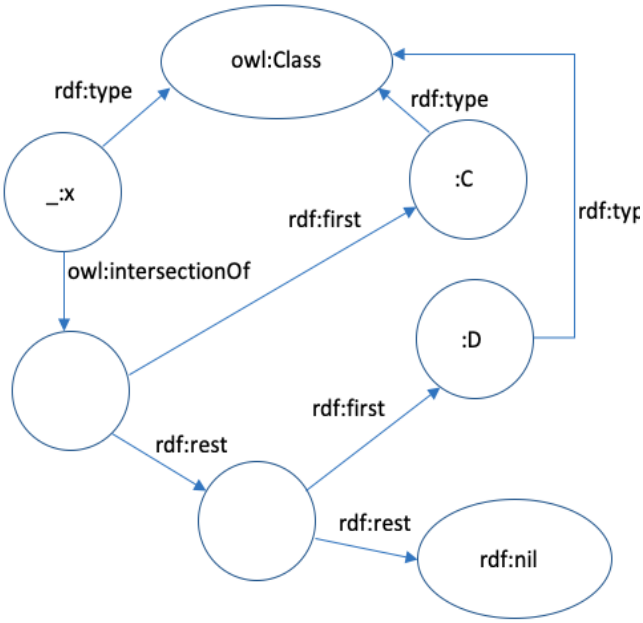

Fig. 7. RDF triplets of the concept intersection image

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Class;
_:x owl:intersectionOf: (:C :D).

Concept union C ⊔ D

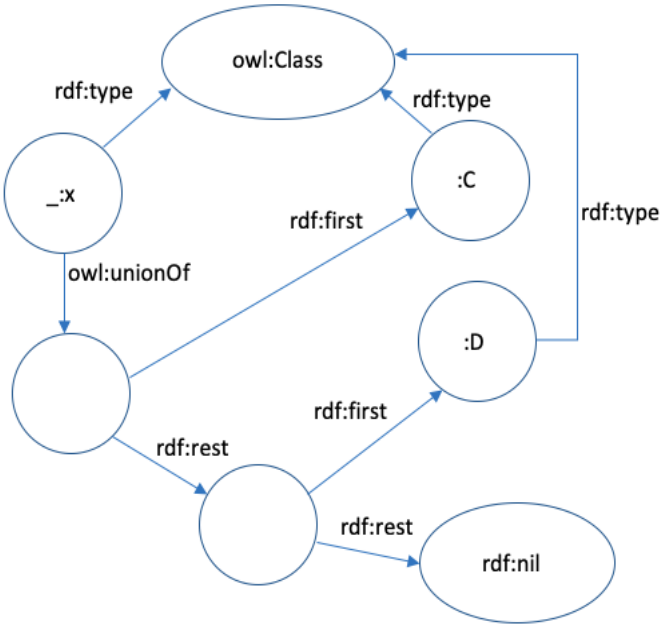

Fig. 8. RDF triplets of the concept union image

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Class;
_:x owl:unionOf: (:C :D).

Existential quantification ∃R.C

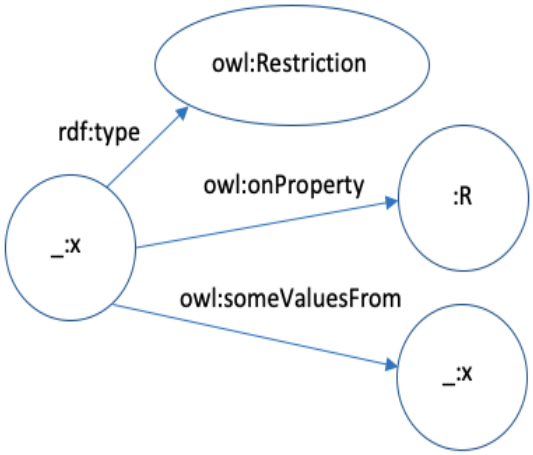

Fig. 9. RDF triplets of the existential quantification image

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Restriction;
owl:onProperty :R;
owl:someValuesFrom:C.

Value restriction ∀R.C

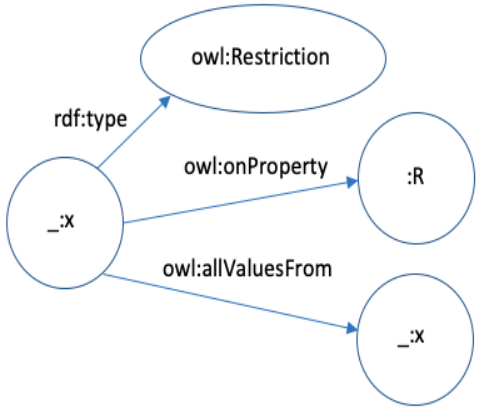

Fig. 10. RDF triplets of the value restriction image

```
@prefix : http://example.com/Ch.owl#
        _:x rdf:type owl:Restriction;
            owl:onProperty :R;
            owl:allValuesFrom :C.
```

In cases of mapping the existential quantification and value restriction concepts, where the role R reflects a datatype property, T (DR) instead of the node :C is used, where DR is the data range.

Number restrictions, nominals

The following constructors are called number restrictions. If R is a role, C is a concept, and n ≥ 0 is a natural number, then:

- (≤nR) и (≥nR) – at-least and at-last number restrictions;
- (≤nR.C) и (≥nR.C) – qualified number restrictions.

In the OWL there is an owl:cardinality constraint [19]. It describes a class of all individuals that have exactly N semantically distinct values (individuals or data values) for the property concerned, where N is the value of the cardinality constraint. This construct is in fact redundant as it can always be replaced by a pair of matching owl:minCardinality and owl:maxCardinality constraints with the same value. It is included as a convenient shorthand for the user. Table 3 shows the mapping rules for number restrictions and nominal.

Table 3. Number restrictions and nominal to RDF mapping rules

Extensions	OWL 2	RDF
1	2	3
=nR	ObjectExactCardinality(n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty T(R). _:x owl:cardinality «n»^^xsd:nonNegativeInteger.
=nR.C	ObjectExactCardinality(n R C)	_:x rdf:type owl:Restriction. _:x owl:onProperty T(R). _:x owl:cardinality «n»^^xsd:nonNegativeInteger. _:x owl:onClass T(C)
≤nR	ObjectMinCardinality(n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty T(R). _:x owl:minCardinality «n»^^xsd:nonNegativeInteger.
≤nR.C	ObjectMinCardinality(n R C)	_:x rdf:type owl:Restriction. _:x owl:onProperty T(R). _:x owl:minQualifiedCardinality «n»^^xsd:nonNegativeInteger. _:x owl:onClass T(C)
≥nR	ObjectMaxCardinality(n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty T(R). _:x owl:maxCardinality «n»^^xsd:nonNegativeInteger.

1	2	3
≥nR.C	Object-MaxCardi-nality(n R C)	_:x rdf:type owl:Restriction. _:x owl:onProperty T(R). _:x owl:maxQualifiedCardinality «n»^^xsd:nonNegativeInteger. _:x owl:onClass T(C)
{a}	Objec-tOneOf(a)	_:x rdf:type owl:Class. _:x owl:oneOf T(SEQ a).

Number restriction =nR

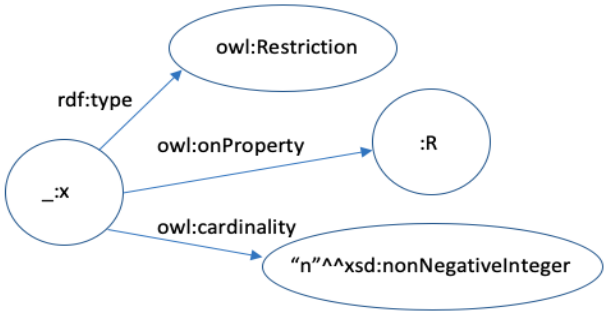

Fig. 11. RDF triplet of the number restriction =nR

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Restriction;
owl:onProperty :R;
owl:cardinality
«n»^^xsd:nonNegativeInteger.

Number restriction =nR.C

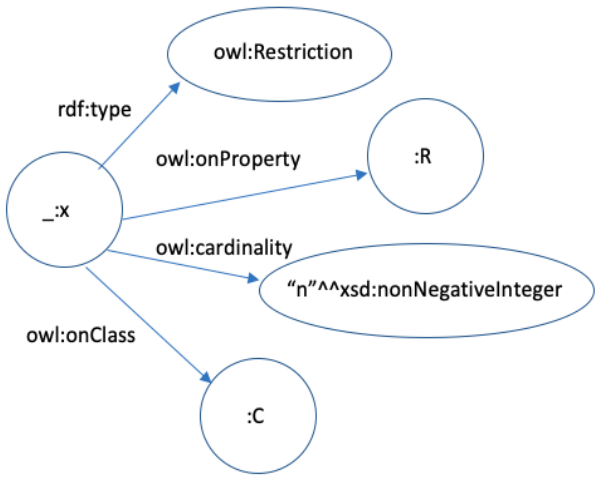

Fig. 12. RDF triplet of the number restriction =nR.C

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Restriction;
owl:onProperty :R;
owl:cardinality
«n»^^xsd:nonNegativeInteger.
owl:onClass :C.

At-least number restriction ≤nR

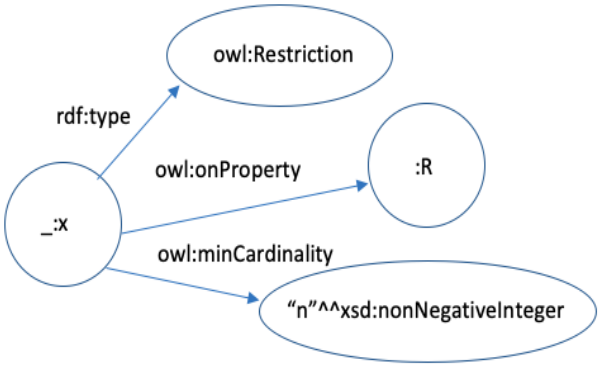

Fig. 13. RDF triplet of the at-least number restriction

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Restriction;
owl:onProperty :R;
owl:minCardinality
«n»^^xsd:nonNegativeInteger.

Qualified number restriction ≤nR.C

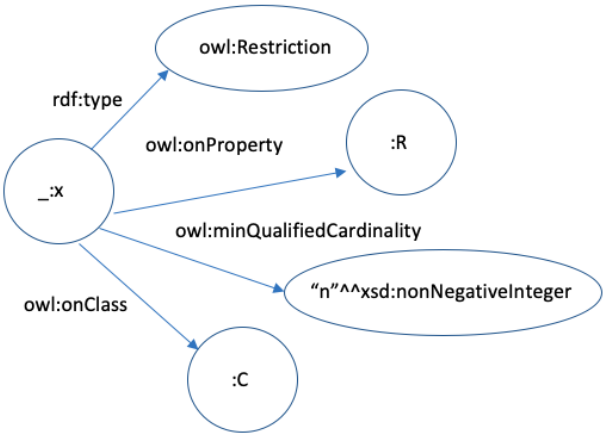

Fig. 14. RDF triplet of the qualified number restriction <nR.C

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Restriction;
owl:onProperty :R;

owl:minQualifiedCardinality «n»^^xsd:nonNegativeInteger.
owl:onClass :C.

At-last number restriction ≥nR

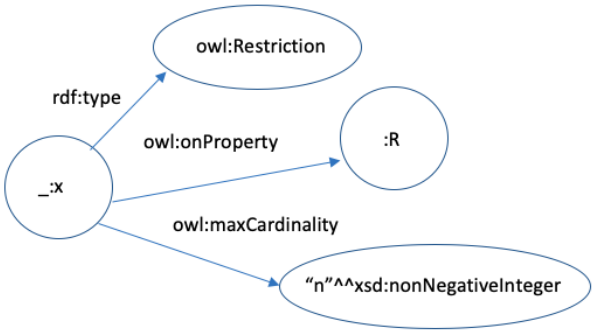

Fig. 15. RDF triplet of the at-last number restriction

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Restriction;
owl:onProperty :R;
owl:maxCardinality «n» xsd:nonNegativeInteger.

Qualified number restriction ≥nR.C

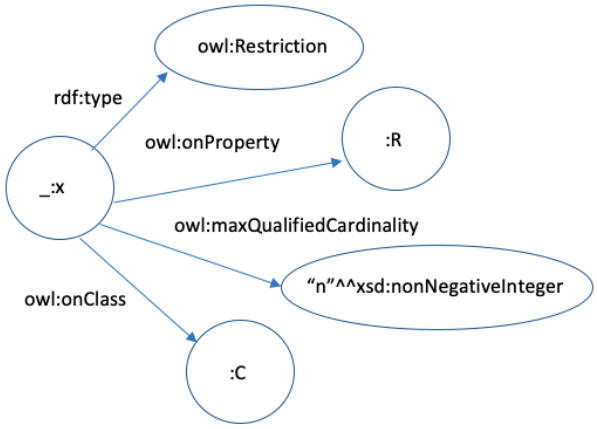

Fig. 16. RDF triplet of the qualified number restriction >nR.C

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Restriction;
owl:onProperty :R;
owl:maxQualifiedCardinality «n» xsd:nonNegativeInteger.
owl:onClass :C.

Nominal {a}

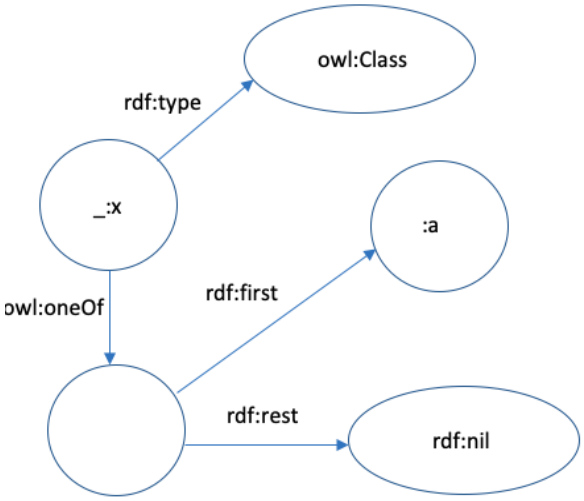

Fig. 17. RDF triplet of the nominal

@prefix : <http://example.com/Ch.owl#>
_:x rdf:type owl:Class;
owl:oneOf(:a).

Role constructors

If R and S are roles, and C is a concept, then the following expressions are also roles: R^- (inverse role), ¬R (complement), R ⊓ S (intersection), R ⊔ S (union), R ° S (composition), R^+ (transitive closure), R* (reflexive-transitive closure), id (C) (role identity).

In OWL 2 through all the role constructors only inverse role is present. This means that the mapping rules exist only for this operation. It looks like this in the Table 4.

Table 4. Inverse role to RDF mapping rule

Constructors	OWL 2	RDF
R^-	InverseObjectProperties (R^- R)	T(R^-) owl:inverseOf T(R)

Fig. 18. RDF triplet for inverse role

@prefix : <http://example.com/Ch.owl#>
:R- owl:inverseOf :R

The issue of mapping the remaining role operators to RDF remains open.

DL axiomatics

The DL axioms include the following rules:

- concept nesting C ⊑ D;
- concept equivalence C ≡ D;
- role nesting R ⊑ S;
- role equivalence R ≡ S;
- concept individual equivalence a = b.

Mapping rules are represented in table 5.

Table 5. DL axiomatics to RDF mapping rules

Axiom	OWL 2	RDF
C ≡ D	Equivalent-Classes (C D)	T(R) owl:equivalentClass T(D)
C ⊑ D	SubClassOf(C D)	T(R) rdfs:subClassOf T(D)
R ⊑ S	SubProjectProp-ertyOf(R S) SubDataProper-tyOf(R S)	T(R) rdfs:subPropertyOf T(S) T(R) rdfs:subPropertyOf T(S)
R ≡ S	Equivalen-tObjectProper-ties(R S) Equivalent-DataProperties(R S)	T(R) owl:equivalentPrope rty T(S) T(R) owl:equivalentPrope rty T(S)
a = b	SameIndividual (a b)	T(a) owl:sameAs T(b)

Concept equivalence

Fig. 19. RDF triplet for concept equivalence

@prefix : <http://example.com/Ch.owl#>
:C owl:equivalentClass :D.

Concept nesting

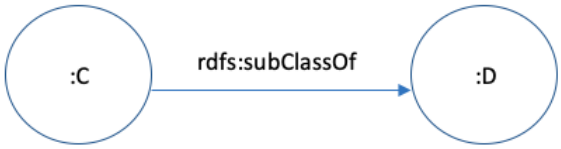

Fig. 20. RDF triplet for concept nesting

@prefix : <http://example.com/Ch.owl#>
:C rdfs:subClassOf :D.

Role equivalence

Fig. 21. RDF triplet for role equivalence

@prefix : <http://example.com/Ch.owl#>
:R owl:equivalentProperty :S

Role nesting

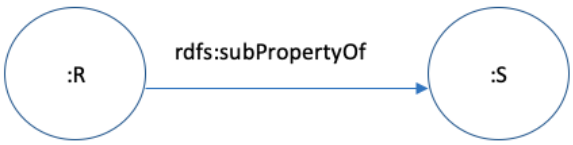

Fig. 22. RDF triplet for role nesting

@prefix : <http://example.com/Ch.owl#>
:R rdfs:subPropertyOf :S

Concept individual equivalence

Fig. 23. RDF triplet for concept individual equivalence

@prefix : <http://example.com/Ch.owl#>
:a owl:sameAs :b

Conclusion

A method for checking mappings of description logic to the binary relational data model using transformations to RDF is described in the article. The description of the approach is given. Bottlenecks and potential

problems are identified. Mappings for the DL axiomatics, as well as for all those constructors of concepts and roles that are implemented in the OWL 2, based on the W3C OWL 2-to-RDF mapping rules are provided in the publication. The issue of mapping several of role constructors to RDF remains open. Mapping a binary relational data model to RDF is in the field for further research.

References

1. Andon P., Reznichenko V., Chystiakova I. Mapping of Description Logic to the Relational Data Model. *Cybernetics and Systems Analysis*. 2017. 53 (6). P. 963–978.
2. Reznichenko V., Chystiakova I. Binary Relational Data Model. *Problems in Programming*. 2017. Vol. 2 (4). P. 96–105.
3. Chystiakova I. Integration of the description logics axiomatic into relational data model. *Problems in Programming*. 2017. Vol. 1(3). P. 51–58.
4. Chystiakova I. Integration of the description logics with extensions into relational data model. *Problems in Programming*. 2016. N 4. P. 58–65.
5. Reznichenko V., Chystiakova I. Integration of the family of extended description logics with relational data model. *Problems in Programming*. 2016. N 2–3. P. 38–47.
6. Reznichenko V., Chystiakova I. Mapping of the Description Logics ALC into the Binary Relational Data Structure. *Problems in Programming*. 2015. N 4. P. 13–30.
7. Chystiakova I. Ontology-oriented data integration on the Semantic Web. *Problems in Programming*. 2014. N 2–3. P. 188–196.
8. Hazber M.A.G., LI R., GU X., XU G. Integration Mapping Rules: Transforming Relational Database to Semantic Web Ontology. *Applied Mathematics & Information Sciences*. 2016. Vol. 3(10). P. 881–901.
9. Hazber M., LI B., XU G., Mosleh M., GU X., LI Y. An Approach for Generation of SPARQL Query from SQL Algebra based Transformation Rules of RDB to Ontology. *Journal of Software. San Bernardino*. 2018. CA. USA. 2018. Vol. 13(11). P. 573–599.
10. Cerans K., Bumans G. RDB2OWL: a RDB-to-RDF/OWL Mapping specification Language. Proceedings of the 2011 conference on Databases and Information Systems VI: Selected Papers from the Ninth International Baltic Conference, DB&IS 2010. Riga, Latvia, 5–7 july 2010. P. 139–152.
11. Cerans K., Bumans G. RDB2OWL: A language and tool for database to ontology mapping. Proceedings of the 27th International Conference on Advanced Information Systems Engineering (CAiSE 2015). Stokholm, Sweeden, 8–12 june 2015. Vol. 1367. P. 81–88.
12. OWL 2 Web Ontology Language. Mapping to RDF Graphs (Second Edition). [Online] December 2012. Available from: https://www.w3.org/TR/owl2-mapping-to-rdf/#Translation_of_Axioms_without_Annotations. [Accessed: 20 february 2020].
13. R2RML: RDB to RDF Mapping Language. [Online] September 2012. Available from: https://www.w3.org/TR/r2rml/. [Accessed: 20 february 2020].
14. Berners-Lee T. Linked data, in design issues of the WWW. [Online] 2006. Available from: https://www.w3.org/DesignIssues/LinkedData.html. [Accessed: 20 february 2020].
15. Michel F., Montagnat J., Zucker C.F. A survey of RDB to RDF translation approaches and tools. [Research Report] I3S. 2014. [Online]. 2014. Available from: https://hal.archives-ouvertes.fr/hal-00903568/file/Rapport_Rech_I3S_v2_-_Michel_et_al_2013_-_A_survey_of_RDB_to_RDF_translation_approaches_and_tools.pdf. [Accessed 20 february 2020].
16. Description logics with axiomatics. [Online] 2017-2018. Available from: http://lpcs.math.msu.su/~zolin/dl/pdf/DL_07_SHIQ.pdf
17. Kontchakov R., Zakharyschev M. An introduction to description logics and query rewriting. Reasoning Web International Summer School. Birmingham, UK. 8 september 2014. Vol. 8714. P. 195–244.
18. Baader F. et al. The Description Logic Handbook. 2003. P. 51–55.
19. OWL Web Ontology Language Reference [Online] 2004. Available from: https://www.w3.org/TR/owl-ref/

Література

1. Andon P., Reznichenko V., Chystiakova I. Mapping of Description Logic to the Relational Data Model. *Cybernetics and Systems Analysis*. 2017. 53 (6). P. 963–978.
2. Reznichenko V., Chystiakova I. Binary Relational Data Model. *Problems in Programming*. 2017. Vol. 2 (4). P. 96–105.

3. Chystiakova I. Integration of the description logics axiomatic into relational data model. *Problems in Programming*. 2017. Vol. 1(3). P. 51–58.
4. Chystiakova I. Integration of the description logics with extensions into relational data model. *Problems in Programming*. 2016. № 4. P. 58–65.
5. Reznichenko V., Chystiakova I. Integration of the family of extended description logics with relational data model. *Problems in Programming*. 2016. № 2–3. P. 38–47.
6. Reznichenko V., Chystiakova I. Mapping of the Description Logics ALC into the Binary Relational Data Structure. *Problems in Programming*. 2015. № 4. P. 13–30.
7. Chystiakova I. Ontology-oriented data integration on the Semantic Web. *Problems in Programming*. 2014. № 2–3. P. 188–196.
8. Hazber M.A.G., LI R., GU X., XU G. Integration Mapping Rules: Transforming Relational Database to Semantic Web Ontology. *Applied Mathematics & Information Sciences*. 2016. Vol. 3(10). P. 881–901.
9. Hazber M., LI B., XU G., Mosleh M., GU X., LI Y. An Approach for Generation of SPARQL Query from SQL Algebra based Transformation Rules of RDB to Ontology. *Journal of Software. San Bernardino*. 2018. CA. USA. 2018. Vol. 13(11). P. 573–599.
10. Cerans K., Bumans G. RDB2OWL: a RDB-to-RDF/OWL Mapping specification Language. Proceedings of the 2011 conference on Databases and Information Systems VI: Selected Papers from the Ninth International Baltic Conference, DB&IS 2010. Riga, Latvia, 5-7 july 2010. P. 139–152.
11. Cerans K., Bumans G. RDB2OWL: A language and tool for database to ontology mapping. Proceedings of the 27th International Conference on Advanced Information Systems Engineering (CAiSE 2015). Stokholm, Sweeden, 8-12 june 2015. Vol. 1367. P. 81–88.
12. OWL 2 Web Ontology Language. Mapping to RDF Graphs (Second Edition). [Online] December 2012. Available from: https://www.w3.org/TR/owl2-mapping-to-rdf/#Translation_of_Axioms_without_Annotations. [Accessed: 20 february 2020].
13. R2RML: RDB to RDF Mapping Language. [Online] September 2012. Available from: https://www.w3.org/TR/r2rml/. [Accessed: 20 february 2020].
14. Berners-Lee T. Linked data, in design issues of the WWW. [Online] 2006. Available from: https://www.w3.org/DesignIssues/LinkedData.html. [Accessed: 20 february 2020].
15. Michel F., Montagnat J., Zucker C.F.. A survey of RDB to RDF translation approaches and tools. [Research Report] I3S. 2014. [Online]. 2014. Available from: https://hal.archives-ouvertes.fr/hal-00903568/file/Rapport_Rech_I3S_v2_-_Michel_et_al_2013_-_A_survey_of_RDB_to_RDF_translation_approaches_and_tools.pdf. [Accessed 20 february 2020].
16. Description logics with axiomatics. [Online] 2017-2018. Available from: http://lpcs.math.msu.su/~zolin/dl/pdf/DL_07_SHIQ.pdf
17. Kontchakov R., Zakharyschev M. An introduction to description logics and query rewriting. Reasoning Web International Summer School. Birmingham, UK. 8 september 2014. Vol. 8714. P. 195–244.
18. Baader F. et al. The Description Logic Handbook. 2003. P. 51–55.
19. OWL Web Ontology Language Reference [Online] 2004. Available from: https://www.w3.org/TR/owl-ref/

Received 18.10.2020

About the author:

Inna Chystiakova,
junior researcher at the Institute of software systems of NASU.
The number of publications in Ukrainian journals – 10.
The number of publications in foreign journals – 1.
Hirsh index is 5.
https://orcid.org/0000-0001-7946-3611.

Affiliation:

Institute of software systems of NASU
03187, Kyiv,
pr. Glushkova, 40, build 5.
Tel.: +38(066)8477784.
E-mail: inna_islyamova@ukr.net.

УДК 004.853, 004.55 https://doi.org/10.15407/pp2020.04.055

Ю.В. Рогушина, А.Я. Гладун

ЗАСТОСУВАННЯ ОНТОЛОГІЧНОГО АНАЛІЗУ ДЛЯ ОБРОБКИ МЕТАДАНИХ ПРИ ІНТЕРПРЕТАЦІЇ BIG DATA НА СЕМАНТИЧНОМУ РІВНІ

Розглядається застосування менеджменту знань для аналізу Big Data. Щоб визначати, яку саме інформацію можна отримати з Big Data, і зробити це здобуття більш ефективним, пропонується застосовувати фонові знання з онтологій предметних областей. За допомогою таких онтологій користувачі можуть формально описувати свої інформаційні потреби, задавати структуру потрібних інформаційних об'єктів та явно виділяти важливі для поточної задачі аспекти. Предметом аналізу Big Data є їх метадані, в яких відомості про семантику, як правило, представлені неструктурованим природномовним описом. Тому виникає потреба у стандартизації подання метаданих, в яких онтології визначають структуру та семантику окремих елементів.

Ключові слова: Big Data, онтологія, метадані, семантична розмітка.

Вступ

Метадані дозволяють охарактеризувати контекст, контент і структуру Big Data, а також методи керування ними. Метадані накопичуються з плином часу та документують історію Big Data. Метаданими необхідно керувати, як самими даними, оскільки вони мають бути захищені від втрати, несанкціонованого видалення, збережені або знищені, а також доступ до керування ними має бути організовано через розподіл прав доступу і виконання певних правил безпеки. Семантику Big Data відображають, як правило, неструктуровані природномовні описи, що входять до складу метаданих, але обробка такої інформації потребує значно більше зусиль порівняно з обробкою структурованої інформації. Тому ціль даної роботи – аналіз напрямків структурування метаописів Big Data з використанням існуючих стандартів.

Метадані та їх властивості

Метадані у найбільш широкому розумінні – це дані про дані. Але таке визначення надто просте й неконструктивне. Вікіпедія визначає метадані як дані з формальної системи вищого рівня, що описує задану систему даних або як структуровані дані, що характеризують певні сутності для їх ідентифікації, пошуку, оцінки та керування ними [1]. Це окремий тип інформаційних ресурсів (ІР), які потребують специфічних засобів подання, створення та обробки (ІР – це будь-яка сутність, яка спроможна передавати чи зберігати інтелектуальну інформацію або знання [2]).

Хоча спочатку метадані призначалися тільки для опису даних, проте останнім часом вони використовуються для опису найрізноманітніших інформаційних ресурсів (ІР) та об'єктів (концептуальних схем, онтологій, сервісів тощо). Вони дозволяють характеризувати життєвий цикл даних, дії та потреби різних суб'єктів обробки даних. Нині метадані дозволяють характеризувати зміст ІР, наприклад, описувати модель предметної області (ПрО) на семантичному рівні.

Розвиток інформаційних технологій став причиною істотного розширення функцій метаданих і викликав їхнє різноманіття. Зміст метаданих, їхні функції і засоби їхнього представлення визначалися тими інформаційними технологіями, що використовувалися для створення таких ІС, специфікою ПрО та тих ІР, що оброблялися цими ІС.

Розповсюдження електронних бібліотек [3], в яких зберігаються ІР різних типів, сховищ даних та знань, що впроваджують технології Semantic Web [4], викликало посилення інтересу до семантизації метаданих [5].

На сьогодні існує велика кількість визначень метаданих, що відображають

різні точки зору на цей термін та на сферу використання метаданих [6]. Метадані — це інформація, що робить дані корисними [7]. Таке визначення описує сферу застосування метаданих, але є надто загальним для практичного використання. Наприклад, для Big Data це визначає роль метаданих, але не дозволяє конкретизувати вимоги до способів їх представлення.

Метадані призначені як для комп'ютерної обробки, так і для інтерпретації людиною інформації про цифрові і нецифрові об'єкти [8]. В роботі [9] метадані визначаються як структуровані дані, що містять характеристики сутностей, які вони описують, для цілей їхньої ідентифікації, пошуку, оцінки та керування. Слід враховувати, що метадані, які використовуються для опису ресурсів Web, є, як правило, слабо структурованими, але вони відповідають погодженим моделям, що забезпечують їх операційну інтероперабельність у неоднорідному середовищі [10].

В роботі [11]. метаданими називається будь-яка дескриптивна інформація про інші джерела даних, яка сприяє організації, ідентифікації, представленню, визначенню місця розташування, забезпеченню інтероперабельності, керуванню і використанню цих даних. В роботі [12] метадані характеризують не інформаційний ресурс у цілому, а певний елемент даних, що відноситься до цього ресурсу. Такий підхід найбільш відповідає специфіці збереження Big Data у великих сховищах, тоді як ідентифікувати потрібно підмножину даних, що пертинентні конкретній задачі користувача.

Метадані можуть використовуватися для визначення семантики інформації, отже, для поліпшення її пошуку і вибірки, розуміння і використання. Наприклад, в [13] розглядається застосовуватися онтологій та тезаурусів для семантичного анотування ІР та їх елементів, що є основою для машинного навчання та здобуття знань з даних. Залежно від цілей анотування можуть застосовуватися онтології різної складності (від контрольованих словників та глосаріїв до онтологій із складними відношеннями інверсії, неперетину тощо). Dublin Core (http://www.dublincore.org/) є прикладом легкої онтології, яка широко використовується для опису характеристик електронних документів та семантизації метаданих.

Конкретний склад функцій метаданих залежить від особливостей тієї системи, що їх використовує, від характеру ІР та їх елементів, які описують ці метадані, від базових інформаційних технологій системи, від потреб її користувачів і від багатьох інших факторів.

Властивості метаданих:

1. В*ідносність* поділу ІР на дані та метадані – метадані для однієї ІС можуть розглядатися як дані в іншій, та навпаки (наприклад, онтологія, що використовується для анотування ПМ-тексту, є елементом метаданих, а та сама онтологія в репозиторії онтологій [14] є даними);

2. *Багаторівневість* опису властивостей будь-якого іншого ресурсу може здійснюватися в термінах більш абстрактної системи понять, які можуть утворювати ієрархію рівнів, яка може включати довільну кількість рівнів (наприклад, Meta Object Facility (MOF) [15] має три рівні, а Dublin Core – два);

3. *Гетерогенність* ІР та даних, що можуть описуватися метаданими: властивості, які дозволяють охарактеризувати метадані, залежать від специфіки самих даних та сфери їх використання;

4. *Відчуженість* метаданих від ІР: метадані можуть зберігатися незалежно або бути убудованими в ІР, які вони характеризують;

5. Ступінь *залежності від контента* визначається змістом самих метаданих (наприклад, дата створення і тип файлу не залежать від контенту, тоді як анотація тексту визначається контентом);

6. Ступінь *залежності від ПрО* визначається цілями створення метаописів, які можуть бути спеціалізованими або універсальними;

7. Ступінь *структурованості;*

8. Рівень *гранулярности опису ресурсів* визначає, які саме елементи ІР описуються метаданими;

9. Ступінь *динамічності* визначається тим, за яких умов та як часто можуть змінюватися метадані;

10. Ступінь *формалізованості* визначається тим, які засоби використовуються для представлення метаданих. Для представлення метаданих (ПМ, ПМ з обмеженим словником, формальні мови – наприклад, OWL [16]).

Існує багато інших властивостей метаданих, які можуть враховуватися в різних дослідженнях (наприклад, засоби представлення, способи збереження та наявність явного подання), але вони не є принциповими для опису Big Data і тому не розглядаються у даній роботі.

Недоліки систем метаданих [17] – це низька оперативність відновлення інформації; неузгоджене введення змін у метадані, що призводить до суперечливості та дублювання; недостатня автоматизація системи ведення метаданих на основі керування контентом; орієнтованість на роботу з одним типом об'єктів (ІР та їх елементів, які описують метадані); відсутність єдиної моделі метаданих для всіх типів об'єктів; відсутність спільного розуміння одиниці опису метаданих – екземпляра метаданих, який описується сукупністю параметрів, що не перетинається з іншими сукупностями, що описуються іншими метаданими; неповнота набору об'єктів метаданих, які зазвичай не містять відомості про засоби обробки та збереження даних.

Неструктуровані дані

Неструктуровані дані (НСД) – це інформація, яка не має попередньо визначеної моделі даних або не організована заздалегідь [18]. Якщо певні елементи метаданих не мають формалізованої структури, то для здобуття з них потрібної інформації необхідно застосовувати методи, що орієнтовані на аналіз НСД. Саме НСД потенційно мають найбільшу цінність як джерела нових знань, і чим більше таких даних доступні для аналізу, тим точніше результати. Більш детально властивості НСД та засоби їх обробки проаналізовано в [19].

Природномовна інформація – набори слів природної мови (ПМ) довільної довжини, поєднані за слабо формалізованими лінгвістичними правилами та представлені в електронній формі, може аналізуватися як НСД. Це обумовлюється тим, що хоча така текстова інформація містить деякі структурні елементи, але у більшості ІР такі структурні елементи не представлені явно, і тому їх здобуття потребує великого часу та зусиль.

Для аналізу НСД можна застосовувати семантичну розмітку. Найбільш корисним засобом семантичної розмітки є зв'язування елементів ІР з елементами онтології (наприклад, фрагмент ПМ-тексту пов'язується з класом або екземпляром класу онтології, а інший елемент – із значенням його властивості). Але з точки зору легкості впровадження безпосереднє застосування онтологій для семантизації ІР є недоцільним – більшість користувачів не володіють онтологічним аналізом, не знають мови подання онтологій тощо. Тому більш корисно використовувати простіші засоби семантизації, наприклад, семантичну Wiki-розмітку. Така семантична вікіфікація може виконуватися як експертами ПрО, так і технічними співробітниками.

Значний недолік цього підходу – семантична Wiki-розмітка ІР, що побудована для однієї ПрО, не може використовуватися для іншої ПрО. Тому доцільно застосовувати онтології вищого рівня, для створення яких можуть застосовуватися онлайнові енциклопедії, що побудовані на основі технологій семантичних Wiki (наприклад, портальна версія Великої української енциклопедії e-ВУЕ [20]). Семантична розмітка дозволяє також аналізувати семантичну подібність між поняттями обраної та використовувати її надалі для аналізу НСД [21].

Метадані для Big Data

Властивості метаданих, їх склад і функції істотно залежать від технологій реалізації систем, в яких вони використовуються, особливостей описуваних ними ресурсів, а також від області застосування і конкретних програм.

Певний набір даних розглядається як Big Data, якщо він володіє однією або декількома характеристиками, так званими характеристиками «5V»: *об'єм; швидкість; різноманіття; достовірність; цінність* [22]. Метадані, які характеризують Big Data, можуть містити інформацію про

джерело даних; про автора і дату створення документа; кількість записів у наборі даних; опис цих даних тощо. В обробці Big Data аналіз метаданих має ключове значення, тому що метадані містять інформацію не тільки про походження даних [23, 24], але й про їх зміст.

Метадані для Big Data [25] – це структурована або напівструктурована інформація, яка дозволяє створювати, керувати і використовувати Big Data у різний час і у різних сферах діяльності, а також робити відбір таких наборів Big Data, що релевантні задачі, яку необхідно вирішити [26]. Для опису метаданих використовуються різні природні та штучні мови. Природні мови є найбільш багатими і виразними в порівнянні з іншими засобами подання метаданих. Вони призначені не для комп'ютерної обробки, а для людей, і не забезпечують однозначності і строгості інтерпретації метаданих, і тому такі описи аналізуються як НСД.

Штучні мови, які використовуються для опису метаданих, – це мови опису даних СУБД, концептуального моделювання, опису онтологій, бізнес-процесів; мови подання онтологій OWL, RDF; мови розмітки тощо.

Стандартизація метаданих

Стандартизація метаданих – основа інтероперабельності та повторного використання як самих метаданих, так і тих IP, що характеризують ці метадані. Тому міжнародні організації зі стандартизації приділяють велику увагу розробці форматів метаданих, які призначені для формального опису різних типів IP та інформаційних об'єктів (IO). Такі стандарти включають в себе набір властивостей, що дозволяють характеризувати конкретний IO. Такі стандарти можуть бути залучені (з різною ефективністю) для опису Big Data. Нині в Україні три міжнародні стандарти, що стосуються метаданих, (ISO 15489-1:2016 [27], ISO 15836-1:2017 [28], ISO 15836-2:2019 [29]) прийнято як національні стандарти методом підтвердження [30, 31].

Стандарт *ISO 15489-1:2016 Information and documentation – Records management — Part 1: Concepts and principles (Інформація і документація. Керування документами. Частина 1: Поняття і принципи)* визначає основні поняття і принципи керування документами і інформацією. Цей стандарт може бути застосований для відображення основних властивостей Big Data: 1) автентичності; 2) достовірності; 3) цілісності; 4) придатності їх до обробки). В стандарті описано інформаційні поля, що входять в структуру метаданих. Для Big Data ці поля дозволяють відобразити наступну інформацію: опис контенту Big Data – це структура даних (форма, формат, зв'язки між блоками Big Data); середовище створення; взаємозв'язок з іншими блоками Big Data (шардинг, реплікація) і метаданими; ідентифікатори та іншу інформацію, що потрібна для видобутку і подання даних; дії і події, що пов'язані з цими Big Data (дата, час дій, зміна метаданих тощо). Big Data, які не супроводжуються такими метаданими, не можуть використовуватися повноцінно.

Стандарт *ISO 15836-1:2017 Information and documentation — The Dublin Core metadata element set — Part 1: Core elements* (Інформація та документація. Набір елементів метаданих «Дублінське ядро». Частина 1: Основні елементи) описує 15 елементів Dublin Core, які використовують для опису ресурсів. В цьому стандарті під ресурсом розуміють будь-який об'єкт, який можна ідентифікувати (наприклад, у сфері комп'ютерних наук ресурсами виступають окремі документи, тексти, аудіо- та відео-файли, Web-сторінки, бази даних тощо). Big Data та їх метадані теж відповідають такому визначенню і можуть розглядатися як ресурси. 15-елементне «ядро», зазначене в цьому стандарті, є частиною більшого набору словників метаданих та технічних специфікацій, що підтримуються Дублінською ініціативою метаданих (Dublin Core Metadata Initiative, DCMI) [32]. Основні елементи можуть використовуватися в поєднанні з термінами метаданих з інших сумісних словників у контексті профілів застосунків, як зазначено в абстрактній моделі DCMI [DCAM]. В табл. 1 приведена специфікація 15 елементів метаданих Dublin Core.

Таблиця 1. Специфікація 15 елементів метаданих Dublin Core

Назва елемента	**Мітка елемента**	**Визначення**	**Коментар**
title	**Заголовок**	**Назва ресурсу**	
creator	Автор	Сутність, відповідальна за створення контенту ресурсу	Людина, організація або сервіс; зазвичай збігається з ім'ям людини, назвою організації або сервісу
subject	Тема	Тема контенту ресурсу	Як правило, подається ключовими словами, фразами або кодами класифікації. Рекомендується вибирати значення з певного словника. Просторова або часова приналежність ресурсу повинна описуватися елементом coverage
description	Опис	Опис контенту ресурсу	Опис контенту ресурсу може включати зміст, анотацію, графічну презентацію або короткий текстовий опис ресурсу
publisher	Видавець	Сутність, що робить ресурс доступним	Людина, організація або сервіс; зазвичай збігається з ім'ям людини, назвою організації або сервісу
contributor	Учасник	Сутність, що бере участь у створенні контенту ресурсу	Людина, організація або сервіс; зазвичай збігається з ім'ям людини, назвою організації або сервісу
Date	Дата	Дата події в життєвому циклі ресурсу	Може використовуватися для подання інформації про час з будь-яким рівнем точності
type	Тип	Вид або категорія контенту ресурсу	Рекомендується вибирати значення з певного словника, такого як DCMI Type Vocabulary. Фізичне або цифрове подання ресурсу визначається елементом format
format	Формат	Фізичне або цифрове подання ресурсу, вимір	Вимірювання може бути, наприклад, розміром або тривалістю
identifier	Ідентифікатор	Конкретне посилання на ресурс в цьому контексті	Рекомендується визначати ресурс за допомогою рядка або числа, що задовольняє формальній системі ідентифікації
source	Джерело	Посилання на ресурс, на основі якого складено цей ресурс	Цей ресурс може складатися з "Джерела" частково або повністю. Рекомендується визначати "Джерело" за допомогою рядка або числа, що задовольняє формальній системі ідентифікації
coverage	Охоплення	Простір або границі, з якими пов'язано вміст ресурсу	Як правило, географічне положення (назва місця або координати), часовий період (назва періоду, дата, набір дат) або підвідомча область (така як адміністративна область)
language	Мова	Національна мова вмісту	Рекомендується вибирати значення з певного словника, такого як RFC 4646
relation	Зв’язування	Посилання на зв'язаний ресурс	Рекомендується визначати "зв'язування" за допомогою рядка або числа, що задовольняє формальній системі ідентифікації
rights	Правова інформація	Правова інформація, пов'язана з ресурсом	Зазвичай "Правова інформація" містить правові угоди щодо ресурсу, включаючи інформацію про права на інтелектуальну власність

Міжнародний стандарт *ISO 15836-2:2019 Information and documentation – The Dublin Core metadata element set – Part 2: DCMI Properties and classes* (Інформація та документація. Набір елементів метаданих «Дублінське ядро». Частина 2: DCMI властивості і класи) є розширенням і доповненням першої частини цього стандарту ISO 15836-1. Розширення полягає у тому, що він надає програмістам загальну універсальну мову для створення та аналізу метаданих. Така універсальна мова забезпечує розширений опис елементів метаданих, використовуючи їх оновлені властивості та класи. Стандарт ISO 15836-2 збільшує початковий набір з 15 основних властивостей до 40 властивостей і 20 класів для підвищення точності і виразності описів у стандарті Dublin Core. Основна увага цього стандарту зосереджена на опису загальних властивостях елементів метаданих, що необхідні для базової інтероперабельності між різними мовами програмування та предметними областями їх застосування.

Такий набір властивостей і класів подається як словник RDF і може використовуватися для зв'язаних даних (Linked Data). Кожна властивість і клас ідентифікується глобальним ідентифікатором для використання в даних RDF. Розробники метаданих, що не належать до RDF, можуть використовувати словник у XML, JSON, UML та реляційних БД, не застосовуючи глобальний ідентифікатор і специфічні для RDF аспекти визначень термінів.

Значення URI можуть бути використані для створення посилань зі значень елементів на відповідні ресурси Web. URI – це уніфіковані локатори ресурсів (URL-адреси) або постійні ідентифікатори, такі як уніфіковані імена ресурсів (URN). Стандарт Dublin Core визначає лише посилання другого типу. У стандарті подані імена властивостей, які можуть бути префіксами для використання як ідентифікатори або цитуватися як повні URI, використовуючи простір імен PURL за замовчуванням.

Таким чином, важливим досягненням базового набору елементів Dublin Core є те, що його розширена семантика дає можливість опису будь-яких Web-ресурсів. Однак існують і негативні наслідки цієї позитивної характеристики.

1. Розширення семантики припускає різні інтерпретації (найбільш складними в інтерпретації є пари "relation – source", "creator – contributor", "type – format".

2. Для опису конкретних категорій ресурсів глобальний рівень є недостатнім: він не відображає важливі характеристики ресурсу. Це стосується основних ПМ-об'єктів опису в репозиторіях – статей, матеріалів конференцій, книг, дисертацій.

Тому можуть вводитися більш детальні елементи опису ресурсів з використанням: розширеного набору термів Dublin Core, які нам надає стандарт ISO 15836 Part 2: "DCMI Properties and classes "(ISO 15836-2: 2019); інших форматів метаданих, таких як MODS (Metadata Object Description Schema) на базі спрощеного набору елементів формату MARC, ETD-MS для опису дисертацій, Data Cite Metadata Schema та інших; власних наборів метаданих, які формуються на основі розширеного формату з додаванням специфічних елементів.

Для забезпечення уніфікації значень і потрібного рівня деталізації метаданих, отримуваних по OAI-PMH у форматі базового DC, репозиторії-агрегатори застосовують набір рекомендацій щодо обов'язкового використання деяких полів; уніфікації використання полів (наприклад, для статей рекомендується записувати назву журналу в поле dc: source); уніфікації формулювань значень полів, важливих для пошуку та щодо заповнення полів з можливостями структурування.

Тенденції розвитку структур метаданих йдуть у напрямку більшого різноманіття і диференціації елементів. Це пов'язано з підвищенням ролі репозиторіїв в структурі відкритої науки, з розміщенням наукових публікацій, підготовлених за підтримки фондів, у репозиторії як альтернативі публікацій в журналах відкритого доступу.

З огляду на ці тенденції, ми можемо виділяти у своїх внутрішніх структурах метаданих окремі елементи, щоб згодом передавати їх в деталізованих обмінних форматах.

Метадані та типові інформаційні об'єкти

Як показав аналіз сучасних систем метаданих, вони дозволяють описувати не тільки ІР у цілому, але й типові для певної ПрО інформаційні об'єкти, які описуються у цих ІР та є їх елементами. Типові інформаційні об'єкти (ТІО) характеризуються набором семантичних властивостей, які можуть бути описані в метаданих кожного екземпляра. ТІО можуть описувати як ІО (документи, елементи БД, мультимедійну інформацію), так і об'єкт реального світу (персоналій, організації, географічні об'єкти тощо). Доцільність створення ТІО визначається специфікою ПрО конкретної ІС: якщо в системі обробляється певна кількість елементів із подібним набором властивостей та характеристик, тоді доцільно виділити для них окремий ТІО.

Відповідно до концепції ТІО [33], які дозволяють класифікувати інформацію про різноманітні ІО зі складною структурою на семантичному рівні, значення деяких елементів метаданих Dublin Core можуть бути віднесені до певних ТІО (табл. 2), що надалі визначає правила їх аналізу та обробки. Крім того, деякі з них можуть розглядатися як ТІО – поняття ПрО, що відповідають класам та екземплярам класів онтології ПрО, тоді як інші є ПМ-описами. Визначити ТІО елементів дозволяє аналіз коментарів, що надаються у стандарті.

Таблиця 2. ТІО елементів метаданих Dublin Core

Назва	**ТІО**
title	Поняття ПрО
creator	Персоналія, Організація, Сервіс
subject	Поняття ПрО
description	ПМ-опис, НСД
publisher	Персоналія, Організація, Сервіс
contributor	Персоналія, Організація, Сервіс
Date	Структуровані дані, Дата
type	Поняття з онтології "Ресурси"
format	ТІО (поняття з онтології "Типи даних")
identifier	Посилання
source	Посилання
coverage	Поняття з онтології "Географічні об'єкти"
language	Поняття з онтології "Мови"
relation	Посилання
rights	ПМ-текст, НСД

Структура та відношення між ТІО можуть відображатися різними засобами подання знань. Наприклад, в онтологіях ТІО відповідають класи, а їх характеристикам – властивості екземплярів класів. В семантичних Wiki-ресурсах для подання ТІО використовуються шаблони, що містять категорії та набір семантичних властивостей ТІО (рис. 1).

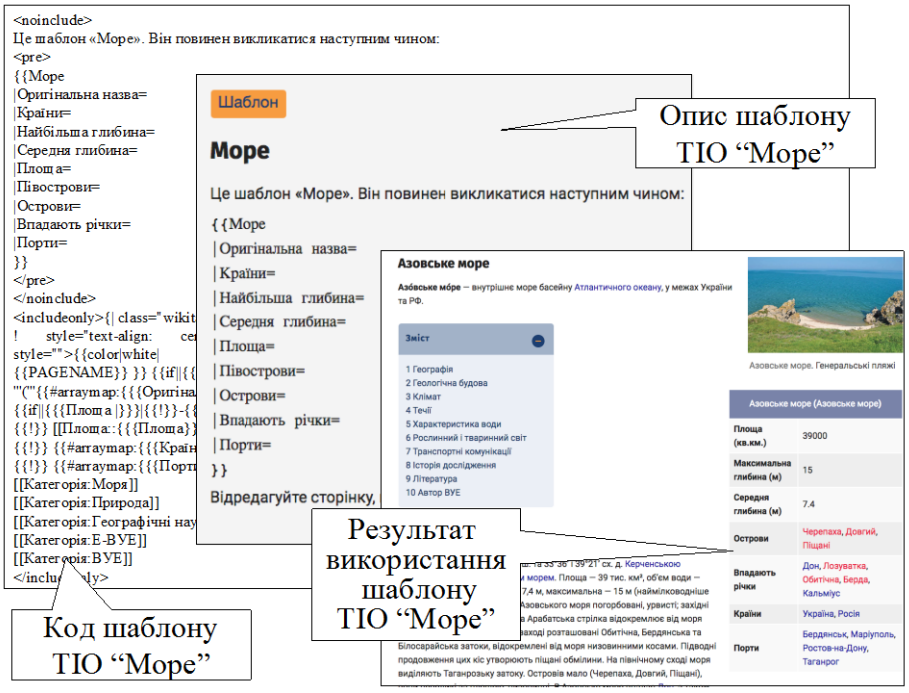

Рис. 1. Використання шаблонів в Semantic MediaWiki для подання ТІО

Використання Data Mining для аналізу метаданих Big Data

На сьогодні створено багато методів, що забезпечують здобуття знань з різних типів ІР – структурованих, частково структурованих та неструктурованих [34]. Аналіз таких методів показує, що внесення структурних елементів у дані значно зменшує простір рішень та зменшує час обробки.

Досить часто основою для створення ТІО є застосування різних напрямків Data Mining для здобуття знань з метаданих цих ТІО для більш ефективної роботи ІС. Особливо це актуально для Big Data, тому що саме аналіз метаданих такої інформації є основою для створення наборів Big Data, що можуть використовуватися як дані для машинного навчання (категоризації та кластеризації). В такому випадку властивості ТІО є параметрами вибірки даних, значення яких аналізуються методами Data Mining [35] , і тому коректне створення ТІО є визначальним фактором обробки Big Data в цілому.

Data Mining – це процес, спрямований на виявлення нових значущих кореляцій, шаблонів і тенденцій у результаті аналізу великого обсягу збережених даних з використанням методик розпізнавання зразків та застосування статистичних і математичних методів. Особливо ефективними методи Data Mining стали із розвитком та накопиченням Big Data. Можна казати, що Data Mining – це процес автоматизованого здобуття з наявних інформаційних ресурсів нових знань, які неявним чином присутніми в оброблюваній інформації.

Результати *Data Mining* у значній мірі залежать від тих даних, які вони обробляють: від їх повноти, актуальності, релевантності поставленій задачі та якості, та від знань, на основі яких обираються ці дані. Тому в тому випадку, якщо побудова набору даних базується на аналізі їх метаданих, саме склад та якість метаданих значним чином визначають якість тих знань, що можна здобути з ІР.

Інструменти Data Mining дозволяють знаходити нові закономірності у даних самостійно й також самостійно будувати гіпотези про взаємозв'язки між їх елементами. Оскільки саме формулювання гіпотези щодо залежностей є найскладнішим завданням, то перевага Data Mining у порівнянні з іншими методами аналізу є очевидною. Але для їх ефективного використання ці результати мають бути пов’язані з відповідним поняттєвим апаратом, який формалізується засобами подання знань, наприклад, за допомогою онтологій [36]. У багатьох випадках такий зв’язок встановлюється через семантичні метадані – ті елементи метаданих, що пов’язані з певним поданням знань, наприклад, з елементами онтології відповідної ПрО. Знання, що здобуваються таким чином з даних, дозволяють у свою чергу вдосконалити онтологію ПрО, яка надалі використовуватиметься для створення метаданих. Таким чином, створення метаданих та їх використання для вдосконалення онтологій є циклічним процесом, який підтримує більш ефективне збереження та використання даних.

Найпоширеніші сфери використання Data Mining пов’язані із вирішенням задач класифікації, кластеризації та прогнозування. Слід відмітити, що Data Mining характеризує не стільки конкретну інформаційну технологію, скільки процес пошуку закономірностей (кореляцій, тенденцій, взаємозв'язків) за допомогою математичних і статистичних алгоритмів, наприклад, регресійного й кореляційного аналізу тощо.

Найбільш розповсюджена задача, що вирішується за допомогою Data Mining, – це задача *класифікації*: вирішення задачі класифікації дозволяє виявити ознаки, що характеризують групи об'єктів досліджуваного набору даних – класи, за якими новий об'єкт можна віднести до того чи іншого класу. Ця задача безпосередньо пов’язана з онтологічним аналізом і дозволяє віднести екземпляри до відповідних класів. Для вирішення задачі класифікації можуть використовуватися методи: найближчого сусіда (Nearest Neighbor); k-найближчого сусіда (k-Nearest Neighbor); Байєсівські мережі (Bayesian Networks); індукція дерев рішень; нейронні мережі (neural networks).

Задачу *кластеризації* можна розглядати як логічне продовження ідеї класифікації і полягає в розподілі множини об'єктів на групи (кластери), при цьому в кожному кластері зібрані об'єкти, які схожі за параметрами. Варто зауважити, що на відміну від класифікації, кількість кластерів і їхніх характеристик визначають у процесі побудови кластерів, виходячи зі ступеня близькості поєднуваних об'єктів по сукупності параметрів. В онтологічному аналізі ця задача виникає на попередньому етапі та дозволяє побудувати набір базових класів онтології й встановити між ними ієрархічні відношення.

Задача *асоціації* – задача пошуку асоціативних правил (визначення взаємозв'язків), що полягає у визначенні наборів об'єктів, які часто зустрічаються серед множини подібних наборів. Відмінність асоціації від двох попередніх задач Data Mining: пошук закономірностей здійснюється не на основі властивостей аналізованого об'єкта, а між декількома подіями, що відбуваються одночасно.

Інші розповсюджені задачі Data Mining – задачі прогнозування, асоціації, визначення відхилень тощо – також можуть застосовуватися для вдосконалення онтологій шляхом обробки даних відповідних ПрО, доступних через Web.

Якщо дані, що обробляються в Data Mining, є ресурсами Web, то це вносить багато додаткових вимог до методів аналізу. Тому у Data Mining виокремлюють такий напрямок, як Web Mining. Системи Web Mining дозволяють знаходити закономірності в інформаційних ресурсах Web, застосовуючи технологію Data Mining для аналізу неструктурованої, неоднорідної, розподіленої і значної за обсягом інформації, яка знаходиться на Web-вузлах. У Web Mining можна виділити такі напрямки, як Web Content Mining і Web Usage Mining, Opinion Mining. В Web Mining можна виділити наступні етапи:

- *вхідний етап* (input stage) – отримання "сирих" даних із джерел (логи серверів, тексти електронних документів);
- *етап попередньої обробки* (preprocessing stage) – дані представляються у формі, необхідній для успішної побудови тієї чи іншої моделі;
- *етап моделювання* (pattern discovery stage);
- *етап аналізу моделі* (pattern analysis stage) – інтерпретація отриманих результатів.

Конкретні процедури кожного етапу залежать від поставленого завдання. У зв'язку із цим виділяють різні категорії Web Mining [37]: аналіз використання Web-ресурсів (Web Usage Mining); отримання Web-структур (Web Structure Mining); здобуття Web-контенту (Web Content Mining.

Значна частина даних – це ПМ-тексти. Саме в таких даних зазвичай міститься найбільш корисна інформація. Тому аналіз таких даних в Data Mining також виокремлюють в спеціальний підрозділ – Text Mining [38]. Технологія Text Mining містить процеси добування знань і високоякісної інформації з ПМ-масивів. Це звичайно відбувається за допомогою виявлення шаблонів і тенденцій за допомогою статистичних та лінгвістичних методів.

Значно підвищити ефективність Data Mining в усіх його напрямках дозволяє застосування фонових знань ПрО. Це дозволяє не шукати заново вже відомі користувачам закономірності та семантично збагатити зв'язки між параметрами (властивостями об'єктів, що аналізуються) за рахунок наявних знань щодо відношень між ними.

Одним з актуальних напрямків застосування фонових знань в Data Mining є аналіз Big Data та їх метаданих. Це обумовлено надзвичайно великими обсягами самих даних та їх динамічністю, що призводить до динамічності тих метаданих, що їх описують. Тому важливими вимогами до методів їх аналізу є швидкодія та наявність евристик, що дозволяють значно скоротити час аналізу. Наприклад, знання щодо відношення "клас-підклас" між параметрами метаданих дозволяє вдосконалити навчальну вибірку.

Це обумовлює необхідність отримання таких фонових знань, яке складається з наступних підзадач:

1) пошук ІР, що пертинентні задачі користувача;

2) здобуття з цих ІР необхідних фонових знань;

3) використання отриманих знань для аналізу даних.

У випадку аналізу Big Data ці задачі конкретизуються наступним чином:

1.1. Вибір сховища Big Data, в якому здійснюється пошук;

1.2. Пошук або створення онтології ПрО, що містить фонові знання щодо задачі користувача;

1.3. Аналіз метаданих Big Data з метою вибору набору даних, що пертинентні задачі користувача, з використанням фонових знань обраної онтології ПрО;

1.4. Генерація потрібного набору даних (підмножини Big Data за визначеними умовами) з використанням знань онтології;

2) Здобуття з онтології ПрО тих термінів та відношень між ними, які потрібні для більш ефективного аналізу великого обсягу інформації (наприклад, для зменшення кількості параметрів даних або для зменшення кількості записів за більш точними умовами відповідності задачі);

3) Використання отриманих знань для аналізу отриманого набору даних та для інтерпретації отриманого результату.

Таким чином, онтології дозволяють як аналізувати семантично метадані, що описують Big Data (наприклад, заміняти терміни в описі задачі на синоніми або на семантично подібні поняття, звужувати або розширювати запит), так і аналізувати самі дані (наприклад, використовуючи обмеження на можливі значення параметрів або виводячи з одних даних інші).

Семантичні Wiki-ресурси як джерело фонових знань для аналізу метаданих Big Data

Дослідження методів отримання фонових знань, які характеризують ПрО Big Data, є актуальним напрямком наукових досліджень, що спрямовані на обробку таких даних. Це обумовлено тим, що, як правило, для наборів Big Data не пропонуються пертинентні онтології тими особами або організаціями, що створюють та зберігають такі набори даних. У більшості випадків використання онтологічного аналізу для Big Data обмежується вибором онтології для визначення структури та змісту метаданих, яка не є специфічною для певної ПрО. Але використання знань ПрО може значно підвищити ефективність обробки.

Висока часова складність, на яку впливає великий розмір простору ознак у Big Data, викликає проблеми в використанні традиційних методів штучного інтелекту до такої інформації. Доцільно для їх оптимізації застосовувати наявні знання щодо ПрО, до якої відносяться як самі Big Data, так і задача, для вирішення якої здійснюється аналіз цих Big Data. Це дозволяє не здобувати ці знання повторно та використовувати їх для логічного виведення та встановлення відношень між елементами метаданих Big Data. Ефективність такого підходу визначається пертинентністю вибору бази знань та засобами подання самих знань. На сьогодні найбільш поширеним рішенням для подання розподілених знань з точки зору сумісного та повторного використання є онтології. Але побудова та пошук онтологій, що є пертинентними конкретній задачі, є складною проблемою. Значно простіше генерувати онтологічні структури за семантизованими Wiki-ресурсами. Такі онтології мають обмежену виразну здатність, але вони можуть створюватися автоматизовано за тим набором Wiki-сторінок, які обирає користувач. Крім того, такий підхід дозволяє відфільтровувати тільки ту інформацію, яка потрібна для вирішення задачі, що значно обмежує обсяг побудованої онтології та зменшує час на її використання.

Пошук пертинентної онтології неможливо повністю автоматизувати, хоча співставлення метаданих Big Data з метаописами онтологій в репозиторії дозволяє виконати попередній відбір. Проблема ускладнюється тим, що значна частина спеціалістів, що працюють з Big Data та їх метаданими, не мають достатнього досвіду у роботі з онтологіями. Тому доцільно застосовувати як джерело фонових знань такі ІР, що задовольняють наступним умовам:

1) досить прості для розуміння їх змісту та обсягу;

2) досяжні через Web;

3) зберігаються у відкритих форматах;

4) дозволяють автоматизовано генерувати онтології з фіксованим набором понять.

Таким вимогам відповідають семантично розмічені Wiki-ресурси. Виразні можливості Semantic MediaWiki [39] – семантичного розширення MediaWiki [40] – дозволяє явно фіксувати зміст відношень між Wiki-сторінками, які відповідають класам онтології.

Для того, щоб використовувати такий Wiki-ресурс як джерело фонових знань в аналізі Big Data, доцільно застосовувати Wiki-онтологію цього IP, яка є формалізованою моделлю знань ресурсу та дозволяє фіксувати характеристики його елементів, їх зв'язків, властивостей та відношень у формі, придатній для автоматичного оброблення, логічного виведення та аналізу. Wiki-онтологія – це окремий випадок онтології ПрО [41], виразні можливості якої обмежені відповідно до виразності Wiki та її семантичного розширення та не припускають застосування характеристик для об'єктних властивостей та властивостей даних. Використання цієї моделі для семантичної розмітки (як назви категорій та семантичних властивостей) забезпечує побудову уніфікованого набору ієрархічно пов'язаних категорій, шаблонів типових інформаційних об'єктів, їх семантичних властивостей та запитів, що їх використовують.

Важливою особливістю семантизованих Wiki-ресурсів є можливість генерації Wiki-онтології не для всієї сукупності сторінок, а тільки для певної підмножини, обраної користувачем явно переліком сторінок або за допомогою семантичного запиту (рис. 2). Параметрами такого запиту є категорії та умови щодо значень семантичних властивостей сторінок.

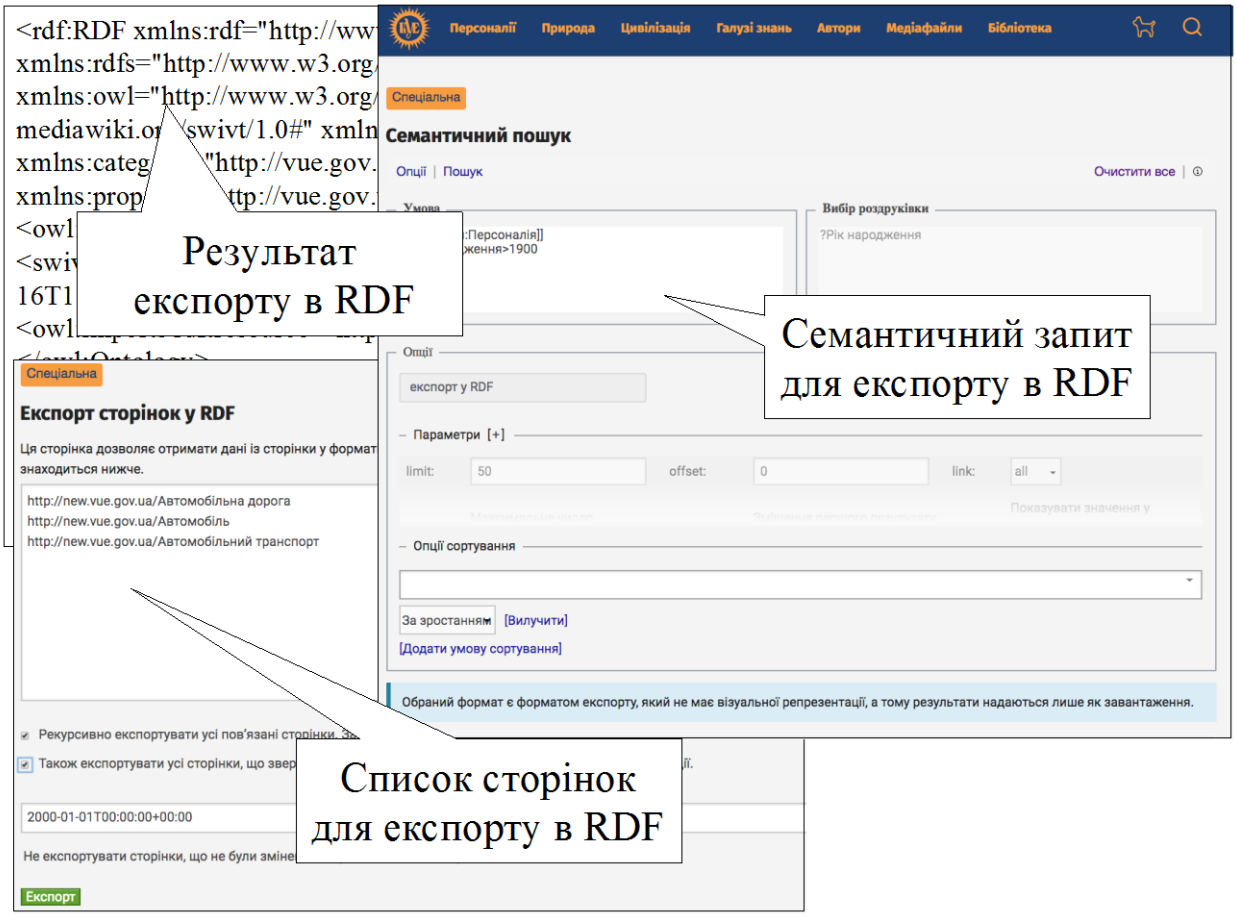

Рис. 2. Засоби Semantic MediaWiki для експорту інформації в RDF-форматі

Висновки

Для можливості інтеграції даних із внутрішніх та зовнішніх джерел та покращення керування Big Data, їх оцінювання та інтерпретації для виконання прикладних задач штучного інтелекту ми використали семантичні технології та онтології. Метадані є основними джерелами інформації про Big Data на протязі всього їх життєвого циклу. Для того, щоб правильно відбирати набори даних з Big Data, необхідно навчитись автоматично видобувати знання з їх метаданих за допомогою семантичних технологій. Доцільно застосовувати для цього такі джерела фонових знань як щодо цих метаданих, так і щодо ПрО, для якої потрібно аналізувати дані, як онтології та тезауруси.

Для семантичного аналізу метаданих ми використовуємо природномовні анотації, які входять до складу метаданих. Семантична обробка інформації метаданих дозволяє отримати від них неявні знання про самі дані. Аналіз текстів метаданих безпосередньо пов'язана із семантикою та певними логічними правилами, тому без метаданих та методів їх аналізу було б практично неможливо обійтися. Запропоновані нами методи аналізу природномовних анотацій є найбільш адекватним засобом співставлення семантики метаданих Big Data з тими задачами, для рішення яких вони можуть застосовуватися. На сьогоднішній день відсутні загальноприйняті, універсальні стандарти про метадані, а найбільш часто використовується універсальний стандарт опису метаданих Dublin Core.

Ми запропонували використовувати технології Wiki та їх семантичне розширення як джерело фонових знань щодо ПрО задачі користувача. Ці знання можуть також бути використані при оцінюванні семантичної близькості термінів домену для структурування елементів метаданих Big Data.

Новизна досліджень, які запропоновані у цій роботі, полягає у новому підході до інтеграції та структуруванні даних в інтелектуальних системах, який базується на семантичному аналізі та інтерпретації структурованих, частково структурованих та неструктурованих метаданих, які описують Big Data, та формуванні на їх основі пертинентного задачі користувача набору даних із застосуванням онтології предметної області.

Література

1. Метадані. https://uk.wikipedia.org/wiki/Метадані
2. Dublin Core Metadata Initiative. DCMI TYPE Vocabulary. http://dublincore.org/documents/demitype-vocabulary
3. Резніченко В А., Захарова О В., Захарова Е.Г. Електронні бібліотеки: інформаційні ресурси та сервіси. *Проблеми програмування*. 2005. № 4. С. 60–72.
4. Berners-Lee T., Hendler J., Lassila O. The semantic web. *Scientific american*. 2001. 284(5). P. 34–43.
5. Dunsire G., Willer M. Standard library metadata models and structures for the Semantic Web. *Library hi tech news*. 2011.
6. Когаловский М. Р. Метаданные, их свойства, функции, классификация и средства представления. Труды 14-й Всероссийской научной конференции «*Электронные библиотеки: перспективные методы и технологии, электронные коллекции*» – RCDL-2012. 2012. http://ceur-ws.org/Vol-934/paper3.pdf
7. Grotschel M., Lugger J. Scientific Information System and Metadata. Konrad-Zuse-Zentrum fur Informationstechnik. Berlin. http://www.zib.de/groetschel/pubnew/paper/groetschelluegger1999.pdf
8. Halshofer B., Klas W. A Survey of Techniques for Achieving Metadata Interoperability. ACM Computing Surveys. 2010. Vol. 42. N 2. Article 7.
9. Taylor C. An Introduction to Metadata. The University of Queensland, Australia. http://www.libraty.uq.edu.au/papers/ctmeta4.html
10. Lagose C. Metadata for the Web. Cornell University. CS 431 - March 2. 2005.
11. Feng L., Brussee R., Blanken H., Veenstra M. Languages for Metadata. In: Multimedia Retrieval. Data-Centric Systems and

Applications, Springer, 23–51. http://www.springerlink.com/ content/m276p88003533q86/.
12. Jeusfeld M.A. Metadata. In: Encyclopedia of Database Systems, Springer. 2009. P. 1723–1724. http ://www. springerlink.com/content/ h241167167r35055/.
13. Corcho O. Ontology based document annotation: trends and open research problems. *Intern. Journal of Metadata, Semantics and Ontologies*. 2006. Vol. 1. Is. 1. http://www.dia.fi.upm.es/~ocorcho/documents/IJMSO2006_Corcho.pdf .
14. Гладун А.Я., Рогушина Ю.В. Репозитории онтологий как средство повторного использования знаний для распознавания информационных объектов. *Онтология проектирования*. 2013. № 1 (7). С. 35–50.
15. Overbeek J. F. Meta Object Facility (MOF): investigation of the state of the art. 2006. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.4092&rep=rep1&type=pdf.
16. OWL Web Ontology Language. Overview. W3C Recommendation: W3C, 2009. – http://www.w3.org/TR/owl-features/.
17. Кобелев А. Е., Вязилов Е. Д. Сучасні підходи по створенню метаданих. *Сучасні проблеми дистанційного зондування Землі з космосу*. 2010. 7(4). С. 194–203. http://d33.infospace.ru/d33_conf/sb2010t4/194-203.pdf.
18. Unstructured_data. – https://en.wikipedia.org/wiki/Unstructured_data.
19. Рогушина Ю. В. Засоби та методи аналізу неструктурованих даних. *Проблеми програмування*. 2019. № 1. С. 57–77. http://pp.isofts.kiev.ua/ojs1/article/view/348/346.
20. Андон П.І., Рогушина Ю.В., Резніченко В.А., Киридон А.М., Арістова А.В., Тищенко А.О. Досвід використання семантичних технологій для створення інтелектуальних ВЕБ-енциклопедій (на прикладі розробки порталу Е-ВУЕ). *Проблеми програмування*. 2020. № 2–3. С. 246–258.
21. Rogushina J. Use of Semantic Similarity Estimates for Unstructured Data Analysis CEUR Vol-2577, Selected Papers of the XIX International Scientific and Practical Conference "*Information Technologies and Security*" (ITS 2019). Kyiv. 2019. P. 246–258. http://ceur-ws.org/Vol-2577/paper20.pdf.
22. Demchenko Y., De Laat C., Membrey P. Defining architecture components of the Big Data Ecosystem. In 2014 International Conference on Collaboration Technologies and Systems (CTS). 2014. P. 104–112.
23. Smith K., Seligman L., Rosenthal A., Kurcz C., Greer M., Macheret C., Eckstein A. "Big Metadata" The Need for Principled Metadata Management in Big Data Ecosystems. Proceedings of Workshop on Data analytics in the Cloud. 2014. P. 1–4).
24. Dey A., Chinchwadkar G., Fekete A., Ramachandran K. Metadata-as-a-service. 31st IEEE International Conference on Data Engineering Workshops. 2015. P. 6–9.
25. Chen M., Mao S., Liu Y. Big data: A survey. Mobile networks and applications. 2014. 19(2). P. 171–209.
26. Rogushina J., Gladun A., Pryima S. Use of Ontologies for Metadata Records Analysis in Big Data. Selected Papers of the XVIII International Scientific and Practical Conference "*Information Technologies and Security*" (ITS 2018). CEUR Vol-2318. http://ceur-ws.org/Vol-2318/paper5.pdf.
27. ISO 15489-1:2016 Information and documentation – Records management – Part 1: Concepts and principles.
28. ISO 15836-1:2017 Information and documentation – The Dublin Core metadata element set – Part 1: Core elements.
29. ISO 15836-2:2019 Information and documentation – The Dublin Core metadata element set – Part 2: DCMI Properties and classes.
30. ДСТУ ISO 15489-1:2018 Інформація та документація. Керування записами. Частина 1. *Поняття та принципи* (ISO 15489-1:2016, IDT).
31. ДСТУ ISO 15836-1:2018 Інформація та документація. Набір елементів метаданих Дублінського ядра. Частина 1. *Основні елементи* (ISO 15836-1:2017, IDT).
32. Weibel S.L., Koch T. The Dublin core metadata initiative. *D-lib magazine*. 2000. 6(12). P. 1082–9873.
33. Рогушина Ю.В. Використання тезаурусів для пошуку складних інформаційних об’єктів у Web на основі онтологій. *Проблеми програмування*. 2019. № 4. С. 11–27.
34. Гладун А.Я., Рогушина Ю.В. Семантичні технології: принципи та практики. – К.:ТОВ "ВД "АДЕФ-Україна". 2016. 308 с. http://eprints.isofts.kiev.ua/669/.

Гладун А.Я., Рогушина Ю.В. Data Mining: пошук знань в даних. К.:ТОВ "ВД "АДЕФ-Україна". 2016. 452 с.

36. Nigro H.O. ed. Data Mining with Ontologies: Implementations, Findings, and Frameworks: Implementations, Findings, and Frameworks. IGI Global. 2007. 289 p.
37. Kosala R., Blockeel H. Web mining research: A survey. *ACM Sigkdd Explorations Newsletter*. 2000. 2(1). P. 1–15. https://arxiv.org/pdf/cs/0011033.pdf
38. Berry M. W., Castellanos M. Survey of text mining. Survey of Text Mining:Clustering, Classification, and Retrieval. *Computing Reviews*. 2007. 45(9). P.548.
39. Krötzsch M., Vrandečić D., Völkel M. Semantic MediaWiki. International Semantic Web Conference. 2006. P. 935–942. https://link.springer.com/content/pdf/10.1007/11926078_68.pdf.
40. MediaWiki. URL: https://www.mediawiki.org/wiki/MediaWiki.
41. Rogushina J. Analysis of Automated Matching of the Semantic Wiki Resources with Elements of Domain Ontologies. *International Journal of Mathematical Sciences and Computing (IJMSC)*. 2017. Vol. 3. N 3. P. 50–58. URL: http://www.mecs-press.org/ijmsc/ijmsc-v3-n3/IJMSC-V3-N3-5.pdf.

References

1. Metadata. – https://uk.wikipedia.org/wiki/Метадані
2. Dublin Core Metadata Initiative. DCMI TYPE Vocabulary.– http://dublincore.org/documents/demitype-vocabulary. (in Ukrainian)
3. Reznichenko V.A., Zakharova O.V., Zakharova E.G. Electronic libraries: information resources and services. *Problems in programming*. 2005. № 4. P.60–72. (in Ukrainian)
4. Berners-Lee T., Hendler J., Lassila O. The semantic web. *Scientific american.* 2001. 284(5). P. 34–43.
5. Dunsire G., Willer M. Standard library metadata models and structures for the Semantic Web. Library hi tech news. 2011.
6. Kogalovsky M.R. Metadata, their properties, functions, classification and presentation means. Proc. of the 14th All-Russian Scientific Conference "*Digital Libraries: Promising Methods and Technologies, Electronic Collections*" – RCDL-2012, 2012. http:ceur-ws.org/Vol-934/paper3.pdf. (in Russian)
7. Grotschel M., Lugger J. Scientific Informa¬tion System and Metadata. Konrad-Zuse-Zentrum fur Informationstechnik. Berlin. http://www.zib.de/groetschel/pubnew/paper/groetschelluegger1999.pdf
8. Halshofer B., Klas W. A Survey of Techni¬ques for Achieving Metadata Interoperability. ACM Computing Surveys. 2010. Vol. 42. No. 2. Article 7.
9. Taylor C. An Introduction to Metadata. The University of Queensland, Australia. http://www.libraty.uq.edu.au/papers/ctmeta4.html
10. Lagose C. Metadata for the Web. Cornell University. CS 431 - March 2. 2005.
11. Feng L., Brussee R., Blanken H., Veenstra M. Languages for Metadata. In: Multimedia Retrieval. Data-Centric Systems and Applications, Springer, 23–51. http://www.springerlink.com/content/m276p88003533q86/.
12. Jeusfeld M.A. Metadata. In: Encyclopedia of Database Systems, Springer. 2009. P. 1723–1724. http ://www. springerlink.com/content/h241167167r35055/.
13. Corcho O. Ontology based document annotation: trends and open research problems. Intern. Journal of Metadata, Semantics and Ontologies. 2006. Vol. 1. Is. 1. http://www.dia.fi.upm.es/~ocorcho/documents/IJMSO2006_Corcho.pdf .
14. Gladun A., Rogushina J. Repositories of ontologies as a means of knowledge reuse for recognition of information objects. *Ontology of design.* 2013. N 1 (7). P. 35–50. (in Russian)
15. Overbeek J. F. Meta Object Facility (MOF): investigation of the state of the art. 2006. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.4092&rep=rep1&type=pdf.
16. OWL Web Ontology Language. Overview. W3C Recommendation: W3C, 2009. – http://www.w3.org/TR/owl-features/.
17. Kobelev A.E., Vyazilov E.D. Modern approaches to metadata creating. Modern problems of remote sensing of the Earth from space. 2010. 7 (4). P. 194–203. http://d33.infospace.ru/d33_conf/sb2010t4/194-203.pdf. (in Ukrainian)
18. Unstructured_data. – https://en.wikipedia.org/wiki/Unstructured_data.

19. ROGUSHINA J. (2019) Means and methods of unstructured data analysis. // Problems in programming, N 1, P. 57–77. http://pp.isofts.kiev.ua/ojs1/article/view/348/346. (in Ukrainian)
20. Andon P., Rogushina J., Grishanova I., Reznichenko V., Kyrydon A., Aristova A., Tyschenko A. (2020) Experience of the semantic technologies use for intelligent Web encyclopedia creation (on example of the Great Ukrainian Encyclopedia portal). Problems in programming, N 2-3. P. 246–258. (in Ukrainian)
21. Rogushina J. Use of Semantic Similarity Estimates for Unstructured Data Analysis CEUR Vol-2577, Selected Papers of the XIX International Scientific and Practical Conference "Information Technologies and Security" (ITS 2019). Kyiv. 2019. P. 246–258. http://ceur-ws.org/Vol-2577/paper20.pdf.
22. Demchenko Y., De Laat C., Membrey P. Defining architecture components of the Big Data Ecosystem. In 2014 International Conference on Collaboration Technologies and Systems (CTS). 2014. P. 104–112.
23. Smith K., Seligman L., Rosenthal A., Kurcz C., Greer M., Macheret C., Eckstein A. "Big Metadata" The Need for Principled Metadata Management in Big Data Ecosystems. Proceedings of Workshop on Data analytics in the Cloud. 2014. P. 1–4).
24. Dey A., Chinchwadkar G., Fekete A., Ramachandran K. Metadata-as-a-service. 31st IEEE International Conference on Data Engineering Workshops. 2015. P. 6–9.
25. Chen M., Mao S., Liu Y. Big data: A survey. Mobile networks and applications. 2014. 19(2). P. 171–209.
26. Rogushina J., Gladun A., Pryima S. Use of Ontologies for Metadata Records Analysis in Big Data. Selected Papers of the XVIII International Scientific and Practical Conference "Information Technologies and Security" (ITS 2018). CEUR Vol-2318. http://ceur-ws.org/Vol-2318/paper5.pdf.
27. ISO 15489-1:2016 Information and documentation – Records management – Part 1: Concepts and principles.
28. ISO 15836-1:2017 Information and documentation – The Dublin Core metadata element set – Part 1: Core elements.
29. ISO 15836-2:2019 Information and documentation – The Dublin Core metadata element set – Part 2: DCMI Properties and classes.
30. DSTU ISO 15489-1: 2018 Information and documentation. Records management. Part 1. Concepts and principles (ISO 15489-1: 2016, IDT). (in Ukrainian)
31. DSTU ISO 15836-1: 2018 Information and documentation. Dublin Core Metadata Element Set. Part 1. Basic elements (ISO 15836-1: 2017, IDT). (in Ukrainian)
32. Weibel S.L., Koch T. The Dublin core metadata initiative. *D-lib magazine*. 2000. 6(12). P. 1082–9873.
33. Rogushina J. The use of thesauri to search for complex Web information objects based on ontologies. *Problems of programming*. 2019. № 4, P. 11–27. (in Ukrainian)
34. Gladun A., Rogushina J. Semantic technologies: principles and practices. 2016. Kyiv. ADEF-Ukraine. 308 p. (in Ukrainian)
35. Gladun A., Rogushina J. Data Mining: search for knowledge in data. 2016. Kyiv. ADEF-Ukraine. 452 p. (in Ukrainian)
36. Nigro H.O. ed. Data Mining with Ontologies: Implementations, Findings, and Frameworks: Implementations, Findings, and Frameworks. IGI Global. 2007. 289 p.
37. Kosala R., Blockeel H. Web mining research: A survey. ACM Sigkdd Explorations Newsletter. 2000. 2(1). P. 1–15. https://arxiv.org/pdf/cs/0011033.pdf
38. Berry M. W., Castellanos M. Survey of text mining. Survey of Text Mining:Clustering, Classification, and Retrieval. Computing Reviews. 2007. 45(9). P. 548.
39. Krötzsch M., Vrandečić D., Völkel M. Semantic MediaWiki. International Semantic Web Conference. 2006. P. 935–942. https://link.springer.com/content/pdf/10.1007/11926078_68.pdf.
40. MediaWiki. URL: https://www.mediawiki.org/wiki/MediaWiki.
41. Rogushina J. Analysis of Automated Matching of the Semantic Wiki Resources with Elements of Domain Ontologies. International Journal of Mathematical Sciences and Computing (IJMSC). 2017. Vol. 3. N 3. P. 50–58. URL: http://www.mecs-press.org/ijmsc/ijmsc-v3-n3/IJMSC-V3-N3-5.pdf.

Одержано 23.10.2020

Про авторів:

Рогушина Юлія Віталіївна,
Кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість наукових публікацій в
українських виданнях – 130.
Кількість наукових публікацій в
зарубіжних виданнях – 28.
http://orcid.org/0000-0001-7958-2557,

Гладун Анатолій Ясонович,
кандидат технічних наук, доцент,
старший науковий співробітник відділу комплексних досліджень інформаційних технологій.
Кількість наукових публікацій в
українських виданнях – 67.
Кількість наукових публікацій в
зарубіжних виданнях – 53.
https://orcid.org/0000-0002-4133-8169.

Місце роботи авторів:

Інститут програмних систем
НАН України, 03181, Київ-187,
проспект Академіка Глушкова, 40.

E-mail: ladamandraka2010@gmail.com.

Міжнародний науково-навчальний центр інформаційних технологій та систем НАН та МОН України,
03680, Київ, Україна,
проспект Академіка Глушкова, 40.

Тел.: +38(044) 526-2549.
E-mail: glanat@yahoo.com

УДК 004.04 https://doi.org/10.15407/pp2020.04.071

О.Р. Жежерун, О.Р. Смиш

АВТОМАТИЗАЦІЯ РОЗВ'ЯЗУВАННЯ ЗАДАЧ З ПЛАНІМЕТРІЇ, ЗАПИСАНИХ ПРИРОДНОЮ УКРАЇНСЬКОЮ МОВОЮ

У роботі досліджено й описано створення системи для розв'язування задач з планіметрії за допомогою сучасних можливостей обробки природної української мови та розробленої сукупності алгоритмів опрацювання тексту задачі. Розробка базується на аналізі текстів планіметричних задач та аналізі доступних засобів обробки живої української мови, що наразі наявні. Результатом роботи є кінцевий програмний продукт, написаний мовою Python, що дає змогу вирішувати прості завдання з планіметрії.
Ключові слова: обробка природньої мови, токенізація, лематизація, розмічування частин мови, сегментація тексту, видобування інформації, розмічений корпус.

Вступ

Мова – це той інструмент, за допомогою якого люди спілкуються та розуміють одне одного. Саме ці мови, які використовує людство у повсякденному житті між собою є природною, тобто такою, що виникла природнім шляхом серед людей. Проте, коли потрібно задати команди комп'ютеру, використовують формальну (штучну) мову – мову програмування. Мова програмування є тим ключем, що дає змогу командами (наборами інструкцій) створити зв'язок між людьми та комп'ютерами.

Природна мова має вагому, досі невирішену проблему, через що не годиться для взаємодії з комп'ютером, здебільшого через свої синтаксичні, смислові, відмінкові та референційні неоднозначності.

Обробка природної мови (англ. Natural language processing або NLP) – галузь в лінгвістиці, комп'ютерних науках, інформаційній інженерії та штучному інтелекті, яка спрямована на комп'ютерний аналіз та обробку природної (людської) мови. Загалом, метою NLP є надати можливість уніфікувати природну мову для розуміння її комп'ютером. Звісно, наразі машини не здатні розуміти українську мову, так само як розуміють її люди, проте вже нині вони мають досить великі можливості. Але, варто зазначити, що найбільші можливості все ж доступні лише для англійської мови. У цій роботі розглядається обробка природної української мови.

Розглянемо деякі ланки, що передбачає NLP, які будуть використовуватися в цій роботі.

Сегментація – це поділ тексту на певні значущі одиниці, такі як слова, речення, абзаци. Поділ на слова в українській мові, як і в інших багатьох мовах світу, що певною мірою використовують кирилицю чи латиницю, не є складним завданням, оскільки такі мови для поділу на слова застосовують пробіли, тобто пусті пропуски між словами, або знаки пунктуації. Дещо складніша ситуація з поділом на речення. Хоч речення в українській мові треба починати з великої літери та закінчувати крапкою (чи іншим символом пунктуації, що позначає закінчення речення, як-от знак оклику), все ж є слова, які граматично правильно писати з великої букви, наприклад імена людей. Також існують скорочення слів, де потрібно поставити крапку [1].

Розмічування частин мови (англ. Part-of-speech tagging) – це встановлення кожному слову з тексту відповідного теґу, який вказує, яка це частина мови зважаючи на визначення слова та на контекст у словосполученні, реченні чи абзаці. Розмічування частин мови ускладнюється тим, що в природній українській мові одне й те ж слово у різних реченнях може відповідати різним частинам мови, а отже й мати інше значення.

Стемінг (англ.Stemming) – процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення чи суфікс.

Лематизація (англ. Lemmatization) – це процес отримання базової словникової форми слова – леми. На відміну від стемінгу лематизація використовує у процесі словник та морфологічний аналіз для цього.

Видобування інформації – це процес вилучення з неструктурованого або малоструктурованого тексту структурованої інформації, такої як сутності, зв'язки між ними, атрибути [2]. Як галузь в NLP, видобування інформації набуває все більшої зацікавленості, оскільки гостро постає необхідність у структуруванні інформації. Варто також додати, що під кожну задачу, надбудовується додаткова логіка на готовий інструмент роботи з текстом, оскільки охопити всі аспекти різних задач наразі не є можливим.

Різновиди геометричних задач

Геометрія, як наука, поділяється на певні галузі такі, як планіметрія, стереометрія, тригонометрія та інші. У цій роботі розглянуто планіметричні задачі, оскільки їхні умови легше піддаються опису в текстовій формі (можливо обійтись без зображення фігур та без тригонометричних рівнянь), також вивчення геометрії розпочинають саме з планіметрії, про що свідчать програми шкільної освіти. Слід провести аналіз видів задач та можливостей їхнього подання.

Існують різні типи задач, які відрізняються між собою, як складністю обрахунку, так і структурою побудови умови, питання (невідоме, що потрібно знайти) і можливостей відповіді на неї.

Наразі планіметричні задачі учні починають вивчати починаючи з 7 класу школи в Україні. Зважаючи на це, доцільно ознайомитися з підручниками школярів за 7 клас та з'ясувати структуру задач, які там використано. Використовуючи термінологію ЗНО [3], у цій роботі будемо розглядати завдання відкритого типу з короткою відповіддю. Далі розглянемо структуру та побудову таких задач, вивівши їхні закономірності шляхом статистичного аналізу, для цього взявши задачі потрібного типу зі шкільних підручників. Для демонстрації наведено приклад задачі відкритого типу з короткою відповіддю, що ілюструють більшість задач із підручника: «У трикутнику ABC відомо, що ∠A = 30°, ∠B = 45°, CM – висота, AC = 10 см. Знайдіть відрізок BM.». Надалі, саме цю задачу використано як приклад, на основі якого простежуватиметься хід обробки та розв'язку всіх задач.

Отже, з тексту задачі видно, що завдання подається здебільшого двома реченнями (рідше одним, ще рідше більш як два), де перше вказує, що дано (умова), а друге пояснює, що потрібно знайти. Як можна побачити, в підручниках також використані спеціальні математичні символи (знак кута, знак градуса та інші), що скорочують написання обсягу тексту, але, як буде видно далі в роботі, ускладнює роботу опрацювання такого тексту аналізатором. Усі іменування змінних (сутностей) подаються латинськими літерами, що очевидно та міжнародно прийнято. Спеціального слова, яке ідентифікувало би початок подання інформації умови (дано) немає, тому речення починаються прямо з подачі цієї умови. Проте, речення, що пояснює шукане, зазвичай починається зі слова «знайдіть», рідше «обчисліть» та «визначте», ще рідше зустрічаються питання, що починаються на «чому дорівнює». Розв'язком на такі типи задач зазвичай є одне число, наприклад, шуканий градус кута («Знайдіть ∠AMC.») чи довжина сторони («Знайдіть гіпотенузу AB.»), рідше зустрічається сукупність («Знайдіть бічні сторони трикутника.»).

Для проведення кількісного аналізу, яке зможе показати частоту повторень слів, що так само допоможе у визначенні на що саме варто сконцентрувати увагу, насамперед потрібно звести весь список зібраних задач до певного однакового виду. Тому для цього варто виконати лематизацію, а символи замінити на прописне слово. Процес методу реалізації лематизації є одним із ключових етапів попередньої обробки тексту, бо дає змогу вилучити закінчення й повертає основну чи словникову форму слова, яка й називається лемою. Виконавши процес лематизації поданого вище прикладу задачі, отримано таке речення: «у трикутник ABC відомо, що кут

А дорівнювати 30 градус, кут В дорівнювати 45 градус, СМ — висота, АС дорівнювати 10 см. знайти відрізок ВМ.». Очевидно, що метод зняв відмінкові форми зі слів, що допомагає при подальшому кількісному аналізі задач. Проведено лематизацію вибраних 76 задач, взятих з підручника. Фрагмент результату показано в таблиці.

Таблиця. Фрагмент результату кількісного аналізу частоти повторень слів у 76 задачах з планіметрії

№	Слово	Повторюваність
1	дорівнює	137
2	см	119
3	трикутник	95
4	кут	82
5	знайти	80
6	градус	54
7	і	45
8	сторона	44
9	висота	32
10	у	29
11	abc	27
12	основа	26
13	що	22
14	прямокутний	20

З таблиці видно, що на першому місці за повторюваністю є слово «дорівнює», потім одиниця виміру відрізків «см». На третьому місці розташоване слово «трикутник», з чого можна зробити висновок, що з більшості задач, вибраних випадково зі шкільних підручників, найбільше завдань, які стосуються саме трикутника (чотирнадцяте місце теж свідчить про трикутник, а саме на його різновид). Це очікувано, оскільки трикутник – це найменший за кількістю кутів багатокутник, а також трикутник вивчають більш поглиблено ніж інші фігури, через те, що саме трикутник лежить в основі тригонометрії, де вивчають взаємозв'язки між сторонами й кутами трикутників. Також цікаво, що на п'ятому місці слово «знайти», що доводить те, що шукане в завданні маркується цим словом. Одинадцяте місце, що зайняло «abc», вказує на іменування трикутника, тобто «АВС» та рідше кут з вершиною «В». Зазвичай, в задачах вказано назву фігури, оскільки це дозволяє легше описати умову, як-от з якого кута проведено відрізок.

Різновиди геометричних задач іншими мовами

Для подальшого створення універсального алгоритму опрацювання тексту, варто розглянути подання задач з планіметрії іншими мовами. Це також дасть змогу програмі працювати навіть змінюючи природну мову на іншу.

Розглянемо для початку східнослов'янські мови, оскільки українська мова входить до їхнього складу. До східнослов'янських мов ще входять білоруська та російська мови. Почнемо з білоруської. Варто зазначити, що оригінального підручника, написаного білоруською мовою не вдалось знайти. Знайдені два підручники, що використані для аналізу, є перекладом підручників з російської мови.

Оглянемо одну стандартну планіметричну задачу білоруською мовою з підручника з геометрії за 7 клас. Дано задачу: «У раўнабедраным трохвугольніку адна старана роўна 5 см, другая – 10 см. Знайдзіце перыметр трохвугольніка.» [4]. Можна помітити, що задача має таку ж структуру, як і задачі українською мовою. Оскільки, спершу йде опис того, що дано, а потім, у наступному реченні, зі словом «Знайдзіце», йде пояснення того, що потрібно знайти. Більшість задач подають опис одним складним реченням та одним простим реченням іде пояснення шуканого.

Отже, всі подібності, що існують між українською та білоруською мовами дають змогу у майбутньому переформатувати й використовувати створену програму з білоруською природною мовою.

Російська мова, входячи спільно з українською та білоруською до східнослов'янської підгрупи слов'янських груп мов, має багато однакових ознак, як граматичних, так і пунктуаційних, через що мо-

жна припустити можливість для використання створеного алгоритму роботи з текстом. Розглянемо типову планіметричну задачу, взяту з підручника з геометрії за 8 клас. Задача: «В равнобедренном треугольнике ABC с основанием AC проведена биссектриса AD. Найдите углы этого треугольника, если ∠ADB = 110°.» [5]. Очевидно, що структура тексту задачі подібна до тексту українських задач. Ідентично, перше речення пояснює умову задачі, друге ж речення описує, що потрібно знайти. Можна зробити висновок, що для алгоритму програми не буде важно перейти на російську мову, змінивши лише лексику.

Для повноти дослідження, варто також оглянути й задачі англійською мовою. Англійська є частиною германської групи, що входить в індоєвропейську сім'ю мов. Англійська та українська побудова тексту задач значно відрізняється, що є очевидним. Все ж розглянемо кілька задач для розуміння побудови, що хоч трохи подібні до структури тексту задач українською мовою. «Triangle ABC has side lengths of AB = 10, BC = 24, and AC = 26. Find the three angles of the triangle.». Такий тип задачі не розповсюджений, хоча й досить наближений до українського варіанту, оскільки є два речення, де в першому вказано умову задачі, а в другому те, що потрібно знайти, яке починається словом «Find» (можна перекласти як «Знайдіть»). Варто зазначити, що все ж більшість задач англійською мовою використовують різний опис для пояснення шуканого. Знайдено такі варіанти у книжці для тих, хто вчиться у коледжі: «find the measures of ∠B and ∠C», «How long is each leg?», «find the lengths of the two legs» [6]. Вищенаведені приклади опису шуканого цілком можуть бути вирішені в розробленому алгоритмі програми цієї роботи.

Аналіз готових рішень для розв'язку поставленої задачі

Розглядаючи проблему створення системи для розв'язування задач з геометрії, критичним аспектом у вирішенні залишається першочергово вибір готового рішення, яке змогло б задовольнити всім потребам в обробці природної української мови. Наразі список таких рішень досить малий, порівняно, до прикладу, з англійською мовою.

Це пояснюється й тим, що робота з NLP часто залежить від спеціального текстового корпусу. Текстовий корпус – це структурована та ретельно підібрана колекція текстів певною мовою. Найбільш вагомими й інформативними корпусами вважаються розмічені, оскільки вони несуть у собі морфологічну прописану до слів інформацію, як рід, число, відмінок та інше. Очевидно, що таких корпусів досить мало для будь-якої мови, тим більше для української, враховуючи те, що розмічення зазвичай відбувається в ручну командою людей-науковців.

Далі розглянемо наявні програмні рішення, що працюють з обробкою саме української живої мови.

Великий електронний словник української мови (ВЕСУМ) – це електронний зведений словник, що містить слова української мови з парадигмами відмінювання. Також, окрім граматичної інформації, словник пропонує заміни слів-покручів, надає розрізнення омонімів з відмінними парадигмами, позначки для рідковживаних слів тощо [7].

Морфологічний аналізатор та генератор для української та російської мов Pymorphy2. Аналізатор може переводити слово до нормальної форми, тобто надавати лему слова, переводити слово до потрібної форми та надавати граматичну інформацію про слово. Підтримка української мови у цьому аналізаторі є не основною, а експериментальною [8]. Можливості Pymorphy2 досить обмежені функціонально, що не є достатнім для цієї роботи. Цей створений аналізатор базується на словнику ВЕСУМ.

Розглянемо модель UDPipe, що навчена на золотому стандарті. «Золотий морфосинтаксовий стандарт» – це текстовий корпус, спеціально розроблений для універсальних залежностей, що розмічено повністю вручну у кілька шарів [9]. УЗ (UD) скорочення від «універсальні залежності» (Universal Dependencies). Це міжнародний проект, випущений у 2014 році спеціально для того, щоб описати синтак-

сичні зв'язки у природних мовах однією спільною метамовою, спільним набором понять. Основним поняттям синтаксичної теорії, на якій базується проект, є залежність, яка прописується для кожного слова (і не тільки) у реченні. Залежність – це зв’язок між двома словами у реченні, де одне з них є підрядним (залежник), а друге (голова) – керує залежником. Цю залежність можна проілюструвати графічно, поєднавши голову та залежник за допомогою стрілки, яка йде з голови до залежника. UDPipe – це здатний до навчання пайплайн для токенізації, маркування, лематизації та парсингу залежностей CoNLL-U файлів [10]. CoNLL-U формат – це перевірена та надійна версія формату CoNLL-X, анотації в якому кодуються у простий текстовий файл. Так, в їхні можливості входить: повернення леми слова; визначення частини мови; морфологічний розбір слова; номер до голови слова, що буде або номером голови, або ж нулем; зв’язок у реченні, який пов’язує слово з головою (якщо слово є головою, то його іменовано коренем у реченні).

Зрозуміло, що найбільше переваг і можливостей присутні в моделі UDPipe, що працює з розміченим корпусом. З боку швидкості, зручності й якості роботи він є найкращим інструментом. Звісно, у ньому є свої недоліки, як некоректне тегування чи розподіл на слова або речення, проте наразі це найоптимальніше рішення для роботи з обробкою української. Тому вирішено використовувати саме цей засіб.

Побудова класів та методів для опису планіметрії

Обираючи мову програмування, як інструмент для створення застосунку, вибір зроблений на користь Python. Оскільки це мова загального призначення, швидко набирає популярність останніми роками, одна з найвикористовуваніших мов для машинного навчання, існує багато готових математичних бібліотек та пакетів. А також через те, що UDPipe підтримує Python.

Визначивши у другому розділі роботи, що трикутник найчастіше зустрічається в задачах з планіметрії, вирішено приділити увагу саме цій фігурі.

Доцільно створити ієрархію класів, де головним буде клас «Багатокутник» («Polygon»), від якого унаслідуватимуться всі інші багатокутні опуклі фігури. Хоча зосередженість цієї роботи є на трикутнику, але опис вищого класу дасть змогу у майбутньому з легкістю додавати нові фігури, не змінюючи та не перероблюючи створену архітектуру. У цьому класі описано функцію для знаходження периметра, додаючи в циклі значення сторін багатокутника. Для подальшого створення класів, ще варто описати функції, що повертатимуть кількість відомих значень: так, для сторін, функція повертатиме значення, яке вказує на кількість відомих сторін, а для кутів, функція повертатиме значення, яке вказує на кількість відомих кутів. Відомі кути чи сторони, це ті, що вже вказані в умові чи знайдені у процесі розв’язання задачі. Це знадобиться у випадку, коли, наприклад, у трикутнику відомі два його кути, тоді знаючи цю інформацію, можна буде з легкістю знайти третій невідомий кут.

Клас «Трикутник» («Triangle») наслідуватиме клас «Багатокутник». Спираючись на таксономію онтології планіметрії, різновиди трикутника описуватимуться в окремих призначених класах, проте деякі властивості, притаманні для всіх трикутників, можливо запрограмувати й у цьому класі. Наприклад, властивість — змінну, що зберігатиме суму градусів кутів трикутника, що дорівнює 180. Існують задачі, в яких не вказано назви трикутника, тоді, для подальшого розв’язання задачі, варто самостійно надати фігурі назву, наприклад «ABC».

Далі, розглянемо функції, що створені для класу «Трикутник». Функція обрахунку площі трикутника, в основі якої закладена формула Герона, що дає змогу визначити площу трикутника за довжинами його сторін. Функція обрахунку сторони, яка шукає довжину сторони: за трьома кутами та однією відомою стороною; за периметром та двома відомими сторонами; за відомою площею та висо-

тою, що проведена до цієї сторони; за властивістю медіани, що проведена до цієї сторони. Функція обрахунку кутів, що шукає значення кута: за трьома сторонами, через арккосинус; за двома відомими кутами, через їхнє віднімання від суми кутів трикутника; за допомогою відношення сторони до синуса протилежного кута; за властивістю бісектриси, що проведена з цього кута; за властивістю медіани, що проведена до сторони.

Коли в задачі є висота, медіана чи бісектриса, тобто відрізки, що ділять основний трикутник, варто розуміти, що подальше розв'язання задачі має відбуватися з оглядом на утворені фігури, як на окремі трикутники. Очевидно, що для розв'язування таких типів задач потрібно весь час переходити з однієї фігури до іншої, а в деяких випадках це може бути потрібно й кілька разів. Отже, виникає проблема, яка полягає у тому, щоб синхронізувати отримувану інформацію з одного трикутника в інший і навпаки, поки не буде знайдено шукане. Для цього створено функцію у класі «Трикутник», що відповідає за синхронізацію даних між утвореними меншими трикутниками. Задача цієї функції у тому, щоб спершу присвоїти утвореним фігурам уже відомі значення та передавати нові значення під час знаходження кутів чи сторін у малих трикутниках основному трикутнику. Виклик функції здійснюється під час того, як проходить встановлення медіани, бісектриси чи висоти. Синхронізація відбувається кілька разів за все розв'язування задачі.

Встановлення медіани, бісектриси та висоти відбувається трьома окремими функціями, в яких є свої унікальні особливості, зважаючи на властивості цих відрізків у трикутнику. Виклик таких функцій здійснюється тоді, коли встановлена їхня наявність після опрацювання тексту.

Клас «Прямокутний Трикутник» наслідує вищий клас «Трикутник». У цьому класі створена змінна, що вказує на різновид трикутника, тобто зберігає значення «прямокутний». Також у класі прописано назви сторін трикутника, тобто перша сторона має назву «гіпотенуза», а інші дві – «катет». Одному з кутів трикутника присвоєно значення «90», цьому ж куту присвоєно назву «прямий». Іншим двом кутам присвоєно назву «гострий». Додатково додано функцію, що обчислює площу трикутника. Обрахунок здійснюється за формулою половини добутку катетів трикутника.

Клас «Рівнобедрений Трикутник» наслідує вищий клас «Трикутник». У класі створено змінну, що вказує на різновид трикутника, тобто зберігає значення «рівнобедрений». Також в цьому класі прописано назви сторін трикутника, тобто перша та третя сторона має назву «бічний», а друга сторона має назву «основа».

Клас «Рівносторонній Трикутник» наслідує вищий клас «Трикутник». У класі додано змінну, що вказує на різновид трикутника, тобто зберігає значення «рівносторонній». Усім кутам трикутника присвоєно значення «60».

Клас «Знайти» поєднує у собі кілька функцій. Функції для виводу інформації, в які входять: вивід кута за його іменем; пошук кута, де відбувається пошук шуканого кута за його іменем з усіх наявних кутів; вивід всіх кутів фігури; вивід сторони за його іменем; вивід сторони за його назвою; пошук сторони, де відбувається пошук шуканої сторони за іменем з усіх наявних сторін; перевірка числа шуканого (однина чи множина).

Проте, основний алгоритм роботи відбувається в конструкторі об'єкта класу. Він пов'язаний з синтаксовим аналізатором та буде розглянуто в наступному розділі.

Використання морфосинтаксового аналізатора в задачах планіметрії

Перед опрацюванням «сирого» тексту задачі потрібно спершу підготувати цей текст. Оскільки аналізатор, що використано в цій роботі, не здатен боротися з багатьма помилками. Тому першим кроком має бути очищення тексту від цих помилок. Помилки можуть бути синтаксичні, орфографічні, лексичні, тавтологічні тощо.

Наступним кроком є заміна математичних символів на прописний аналог. Аналізатор навчено на розмічених текстах, переважно художнього стилю. Це спричиняє не завжди коректне опрацювання текстів математичних задач. Тому варто автоматично замінювати такі символи: «°» на « градусів», «=» на «дорівнює», «∠» на «кут », «||» на «паралельна» та інші. На прикладі вибраної задачі отримано такий проміжний результат: «У трикутнику ABC відомо, що кут A дорівнює 30 градусів, кут B дорівнює 45 градусів, CM — висота, AC дорівнює 10 см. Знайдіть відрізок BM.»

Останнім кроком є заміна різноманітного формулювання для шуканого одним уніфікованим аналогом, оскільки визначення шуканого спиратиметься надалі на це слово. Відбувається автоматична заміна таких варіантів, як: «Обчисліть», «Визначте» та «Чому дорівнюють» на «Знайдіть». У випадку вибраної демонстраційної задачі такої заміни не потрібно.

Як можна помітити, майже всі заміни не є лемою цього ж слова. Це зроблено для того, щоб подальший аналіз тексту коректніше проставив залежності між словами, а на це впливає форма слова.

Налаштування аналізатора

Для використання UDPipe можна скористатись уже готовим прикладним програмним інтерфейсом. Формат тексту повинен бути UTF-8, а розмір одного запиту не повинен перевищувати одного мегабайта. Обробка тексту відбувається через файл.

Одна з проблем, яка виникає, коли в задачі задані числові значення з комою (не цілі числа), є досить суттєвою. Аналізатор створений так, щоб числа було поділено. Бо, саме синтаксично, це є окремими токенами. Оскільки, до прикладу, рахунок матчу 10:0 та позначення часу 15:30 (ще можуть записувати як 15.30) є різними токенами. Таке розбиття з одного числа, створить три окремі токени, де першим токеном є ціла частина, другим є знак коми, а третім дробовою частиною числа. Це призведе до некоректного створення залежностей між словами, що унеможливить подальшу обробку тексту. Для виправлення цього створено попередню перевірку ще не проаналізованого тексту на наявність чисел з комою. За допомогою можливостей Python та розробленого регулярного виразу відбувається пошук усіх чисел з комою, подальший їхній запис у масив, а потім заміна в тексті цих чисел на «00». Після виконання своєї роботи аналізатором, числа вписуються назад на своє місце з масиву. Оскільки аналізатор розпізнає число «00» як один числовий токен, то подальше повернення справжнього числа з масиву аж ніяк не впливає на результат, а проведений аналіз відповідає очікуваному.

Варто додати, що майже всі задачі містять у тексті або знак «=», або ж слово «дорівнює» (можлива й інша форма цього слова). Проте знак під час обробки буде замінено на слово, а надалі після лематизації отримано всюди одне уніфіковане слово «дорівнювати». Оскільки, більшість задач подають дані в умові за допомогою цього слова, доречно на основі нього створити пошук, який зможе відшукати в тексті всі значення.

Пошук базується на двох умовах. Спершу, перевірка чи токен перед словом «дорівнювати» є кутом, тобто одна велика або три великі латинські букви. Здійснюється за допомогою регулярного виразу: «^[A-Z]{1}$|^[A-Z]{3}$». Якщо умова виконана, то значення кута з його назвою заносяться у відповідні змінні. Наступна умова перевіряє чи перед словом «дорівнювати» є сторона і здійснюється за допомогою регулярного виразу: «^[A-Z]{2}$».

Додано й третє розгалуження, в разі, якщо перші дві умови не виконуються. Часто трапляється, що дані подано не до конкретної змінної, або ж в задачі не надано фігурі назви чи вказані дані описують не кут чи сторону. Прикладом можуть бути такі варіанти: «основа рівнобедреного трикутника дорівнює 12», «площа квадрата дорівнює 8», «більша сторона прямокутного трикутника дорівнює 5» тощо. Тоді пошук відбувається так: іде пошук слова вліво, доки не знайдеться слово, яке аналізатором визначено як підмет, це й буде змінною. Також ще відбувається пошук

означення до цього підмета, який зазвичай може стояти зразу перед ним.

Цей пошук здійснюється за допомогою обробленого тексту після синтаксового аналізатора, коли слова поділені на токени, проведена лематизація і визначені зв'язки між ними. Підмет у стовпці, що описує зв'язок, позначено як «nsubj», означення через стовпець частин мови як «ADJ».

Коли під час пошуку знайдено підмет, далі йде порівняння цього токена з переліком визначених слів. Якщо слово збігається із заготовленим можливим підметом, значення присвоюється відповідній змінній.

Пошук в умові задачі того, що потрібно знайти, відбувається в конструкторі об'єкта класу «Find». Обробка здійснюється за допомогою аналізатора. Орієнтиром, на який спирається пошук, є слово «знайти».

Алгоритм роботи такий: відбувається проходження всього тексту після слова «знайти», оминаючи токени, у яких визначено частину мови, як «PUNCT» (знаки пунктуації), «CCONJ» (різновиди сполучників), «VERB» (дієслова), «NUM» (числа), «SCONJ» (різновиди сполучників), «ADP» (прийменники), «DET» (детермінант), «ADV» (прислівниково-числівникові займенники). Це потрібно для відкидання інформації, яка не несе в собі значущі, важливі для обробки, значення.

Проходження припиняється, якщо на шляху трапляються такі токени, як: «.», «?», «якщо».

Після відбирання токенів, потрібно порівняти їх із тими, які заздалегідь визначені як важливі. Створено чотири масиви, які поділені за своїм смислом: масив різновиду шуканого (прикметники, як «найбільший»); масив, що означає сторону (іменники, як «відрізок» чи «відстань»); масив, що означає кут. Ще один масив, де зберігаються назви фігур. Також, може бути знайдено ім'я шуканого за допомогою регулярного виразу: «^[A-Z]{1,3}$». Іще перевірка, якщо токен дорівнює словам «площа» чи «периметр».

Насамперед потрібно для обробки задачі пройтися по лемах токенів, щоб знайти назву фігури. Для цієї перевірки створено масив, з назвами усіх багатокутних плоских фігур, а також масив з варіантами написання їхнього виду. Після знаходження токена назви фігури, перед ним завжди стоятиме токен з його видом.

Після збігу назви фігури з токеном, створюється екземпляр класу відповідної фігури чи класу відповідного виду цієї фігури.

Механізм розв'язування планіметричної задачі на прикладі

Увівши задачу: «У трикутнику ABC відомо, що ∠A = 30°, ∠B = 45°, CM – висота, AC = 10 см. Знайдіть відрізок BM.» у програму, спершу відбудеться заміна символів на їхні прописні слова-аналоги. Після цього кроку відбувається перевірка програмою на числа з комою. Наступний крок є одним з найосновніших, бо саме тут відбувається перевірка тексту задачі аналізатором. Саме тут текст аналізується, поділяється на токени, відбувається лематизація, проставляння залежностей між словами. Далі робота виконується лиш з цим підготовленим текстом.

Спершу визначається фігура, її вид (може бути відсутнє), її іменування (може бути відсутнє). Визначивши фігуру, відбувається виклик відповідного класу, що їй належить. Варто зазначити, якщо в задачі вказано вид фігури, до прикладу «прямокутний трикутник», відповідний і більш конкретний клас буде викликано. Отже, в задачі з прикладу, буде визначено фігуру «трикутник» та іменування фігури «ABC».

Далі відбувається пошук в тексті слів, таких як «бісектриса», «медіана» і «висота». Знайшовши таке слово, викликається відповідний метод в основному класі «Triangle». У прикладі є висота, тому цьому відрізку присвоюється ім'я «CM» з умови. У методі визначається: кут, з якого проведено висоту; сторони, що утворюють кут; до якої сторони проведена висота; відрізки, на які ділить висота, трикутники на які ділить висота.

За допомогою створених методів, які опрацьовують дані навколо слова «до-

рівнювати», у задачі визначено спершу значення кутів: куту «А» присвоєно значення «30», куту «В» присвоєно значення «45»; потім визначено значення відрізків: стороні «АС» присвоєно значення «10».

Тепер відбувається виклик класу «Знайти». Результатом пошуку шуканого є два масиви, де перший масив [«side», «name»] вказує на проставлені мітки знайденим токенам, а другий масив показує самі ж токени [«відрізок», «ВМ»]. Порядок елементів у першому та другому масивах відповідає один одному.

За допомогою методу для знаходження кутів, визначається невідомий кут «С».

Враховуючи, що в задачі є висота, яка ділить основний трикутник, утворені менші трикутники досліджуються окремо й розглядаються як повноцінні. Зважаючи на властивості висоти, менші трикутники належать до класу «Прямокутний Трикутник». Далі у трикутнику «САМ» знаходиться довжина відрізка «СМ» за допомогою відомих значень сторони та синуса кута. Відбувається виклик методу, що відповідальний за синхронізацію нових отриманих даних з трикутників. І в результаті у трикутнику «ВСМ» за допомогою значень сторони та тангенса кута отримано шуканий відрізок «ВМ».

Висновки

Отже, в роботі проведено аналіз наявних на сьогодні можливостей з обробки живої української мови.

Проведено аналіз задач з геометрії, який показує закономірності у формулюванні й структурі текстового подання задач.

Створена архітектура є гнучкою й це вказує на можливість додавання як опису нових фігур та їхніх властивостей, так і на створення додаткової логіки в застосунку.

Варто вказати на можливість переформатування програми для використання її з іншою природною мовою, до прикладу англійською, білоруською чи російською.

Література

1. Reynar J. C. A Maximum Entropy Approach to Identifying Sentence Boundaries. Reynar Jeffrey – Philadelphia, Pennsylvania, USA.
2. Sarawagi S. Information Extraction / Sarawagi Sunita – Mumbai. 2008.
3. ЗНО з математики: особливості тесту 2020 року [ЗНО з математики: особливості тесту 2020 року [Електронний ресурс]. 2019. Режим доступу до ресурсу: https://osvita.ua/test/training/5017/.
4. Казакоў В.У. ГЕАМЕТРЫЯ. Мінск: Народная асвета. 2017.
5. Геометрия. 7–9 классы [Л. С. Атанасян, В. Ф. Бутузов, С. Б. Кадомцев та ін.]. Москва: Просвещение. 2010. 384 с. (20).
6. Alexander D.C., Koeberlein G.M. Elementary Geometry for College Students. Belmont. (5).
7. Великий електронний словник української мови (ВЕСУМ) [Електронний ресурс]. 2017. Режим доступу до ресурсу: https://github.com/brown-uk/dict_uk/blob/master/doc/announcement.md.
8. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages. Ekaterinburg. 2015.
9. Золотий морфосинтаксовий стандарт [Електронний ресурс]. Режим доступу до ресурсу: https://mova.institute/золотий_стандарт.
10. Straka M. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe / Straka Milan Vancouver, 2017.

References

1. Reynar J. C. A Maximum Entropy Approach to Identifying Sentence Boundaries. Reynar Jeffrey – Philadelphia, Pennsylvania, USA.
2. Sarawagi S. Information Extraction / Sarawagi Sunita – Mumbai. 2008.
3. ZNO z matematyky: osoblyvosti testu 2020 roku. (2019). Retrieved June 12, 2020, from https://osvita.ua/test/training/5017/
4. Kazakow V.U. Geometryja. Minsk: Narodnaja asveta. 2017.
5. Geometriya: 7–9-e klassy: uchebnik dlya obshcheobrazovatelnykh uchrezhdeniy. 2010.

6. Alexander D.C., Koeberlein G.M. Elementary Geometry for College Students. Belmont. (5).
7. Velykyi elektronnyi slovnyk ukrainskoi movy (VESUM). (2017, November 30). GitHub Retrieved June 12, 2020, from https://github.com/brown-uk/dict_uk/blob/master/doc/announcement.md
8. Korobov M. Morphological analyzer and generator for Russian and Ukrainian languages. Communications in Computer and Information Science. 2015. P. 320–332. https://doi.org/10.1007/978-3-319-26123-2_31
9. Zolotyi morfosyntaksovyi standart. Laboratoriia ukrainskoi. Retrieved June 12. 2020, from https://mova.institute/золотий_стандарт.
10. Straka M. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe / Straka Milan Vancouver. 2017. https://doi.org/10.18653/v1/k17-3009

Одержано 23.10.2020

Про авторів:

Жежерун Олександр Петрович,
кандидат фізико-математичних наук, старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 80.
Індекс Гірша – 4.
http://orcid.org/ 0000-0002-4034-6730,

Смиш Олег Русланович,
аспірант.
https://orcid.org/0000-0002-8074-9745.

Місце роботи авторів:

Національний університет «Києво-Могилянська академія», завідувач кафедри мультимедійних систем.
04655, Київ,
вул. Г. Сковороди 2.
Тел.: (044) 425-77-53
E-mail: zhezherun@ukma.edu.ua

Інститут програмних систем НАН України,
03187, Київ,
проспект Академіка Глушкова, 40.
Тел.: (044) 526-33-19
E-mail: o.smysh@ukma.edu.ua

УДК 004.85 https://doi.org/10.15407/pp2020.04.081

А.П. Жиркова, О.П. Ігнатенко

АНАЛІЗ МЕТОДІВ МАШИННОГО НАВЧАННЯ В ЗАДАЧІ КЛАСИФІКАЦІЇ ДОКУМЕНТІВ

Публікація досліджує методи класифікації документів за наявністю печатки. Для цього, по-перше, проаналізовано вже існуючі методи вирішення поставленої проблеми; по-друге, запропоновано модель згорткової нейронної мережі для класифікації документів; по-третє, відображено залежність коректності роботи нейронної мережі від кількості вхідних даних, на яких навчається модель. В результаті отримано нейронну мережу, що класифікує документи за наявністю печатки з точністю трохи більше ніж 88 %.

Ключові слова: машинне навчання, класифікація, згорткові нейронні мережі, послідовна модель.

Вступ

Дана робота досліджує актуальну проблему класифікації відсканованих документів за наявністю печатки. Ця проблема виникає у багатьох областях діяльності, пов’язаних з документообігом, оскільки поточна нормативна база України орієнтується, в основному, на паперові документи. Особливо важливим напрямком застосування є класифікація документів у системі публічних закупівель “Прозорро”, в якій у 2019 р. успішно відбулось 1,238 млн. публічних закупівель з очікуваною вартістю 581,3 млрд грн. Кількість активних організаторів закупівель склала 28 850, кількість активних учасників – 159 980 (дані за 2019 рік). З огляду на успішність даного проекту планується подальше розширення сфери застосування системи Прозорро на інші напрямки.

Розпізнавання паттернів на відсканованих документах, зокрема визначення наявності печатки, підпису, реквізитів та інших шаблонізованих частин є критично важливим для автоматичної перевірки коректності завантажених документів. Кількість щоденних тендерів змушує шукати алгоритмічні шляхи розв’язання проблеми, і тут природнім напрямком пошуку є методи машинного навчання. Машинне навчання – вже не новий, але дуже популярний напрям для досліджень та розробки, орієнтований на роботу з різними видами даних, розуміння їх структури та взаємозв’язків.

Обсяг даних, представлений у вигляді зображень, значно виріс з розвитком технологій та популяризацією фотографій як способу поділитися певною інформацією. Присутність камери на всіх мобільних пристроях, що випускаються останнім часом, та поліпшення якості фотографій, є значним рушієм для поширення використання даних такого виду. А отже, з’являються і методи обробки подібної інформації, де центральним поняттям є термін “computer vision”, який на українську перекладається як “комп’ютерний зір” та означає процес обробки графічних даних, який має на меті, наприклад, розпізнавання об’єктів певного класу.

Можливість сканувати або фотографувати документи дозволяє зберігати їх у вигляді зображень, а отже, і застосовувати до них відповідні методи обробки. Наявність або відсутність печаток на зображенні є типовим представником задачі класифікації. При складанні документів виникають помилки різного типу, які важко відстежити та навіть при дуже уважному перегляді можна пропустити. Тоді виникає потреба обробляти документи в автоматизованому режимі, щоб відслідковувати помилки та мінімізувати їхню присутність у документах. Використання машинного навчання для розв'язання задач цього класу є актуальним способом вирішення подібних проблем. В даній роботі

пропонується метод вирішення описаної задачі за допомогою навченої нейронної мережі. Розробка моделі згорткової нейронної мережі для коректної класифікації документів за наявністю або відсутністю печаток на ньому потребує вирішення наступних завдань, а саме: проведення аналізу існуючих методів вирішення поставленої проблеми, проведення збору та обробки даних, побудови нейронної мережі для класифікації документів, дослідження залежності точності моделі від кількості даних для її навчання.

Огляд літератури

Опишемо існуючі методи, які використовуються для розпізнавання печаток на документах. Двоетапний підхід до вилучення візуальних об'єктів з паперових документів, про який розповідається у [1], серед інших задач вирішує і поставлену у даній роботі. Двоетапний підхід працює наступним чином: спочатку застосовується певний алгоритм для розпізнавання об'єкту на вхідному зображенні, після чого застосовується інший метод, заснований на добуванні з зображення низько-рівневих характеристик (з англ. "features") – таким чином перевіряється правильність роботи попереднього етапу. Отже, перший етап – каскадне навчання та розпізнавання на основі класифікатора AdaBoost. На другому етапі проводиться оцінка низько-рівневих характеристик зображення за допомогою різних алгоритмів машинного навчання. Зображення представляється даними, отриманими з нього за допомогою різних функцій, однією з яких є статистика першого порядку, що означає представлення зображення у вигляді таких характеристик як середня інтенсивність пікселів, дисперсія, асиметрія, центральний момент та ентропія. Наступним методом представлення зображення є його опис за допомогою статистики довжини сірого, дані про яку надають інформацію щодо текстури зображення. Також серед таких методів є гістограма напрямлених градієнтів, яка допомагає розрізняти об'єкти різних типів, та локальні бінарні патерни, які є універсальними дескрипторами текстури. На другому етапі до усіх цих представлень застосовуються алгоритми машинного навчання, такі як метод k-найближчих сусідів (k=1), наївний Байєс, метод опорних векторів, бінарне дерево рішень та інші. Використання двоетапного підходу до вилучення візуальних об'єктів з паперових документів у випадку розпізнавання печаток дало середню точність 53.3 %.

Наступний підхід до виявлення печаток у документах, представлений у [2], використовує поєднання деяких простих характеристик зображення. Алгоритми машинного навчання (такі як метод k-найближчих сусідів, метод опорних векторів, випадкові ліси), що використовуються для виявлення печаток, обробляють інформацію про зображення, в якому закодували початкову модель RGB у модель, що представляє зображення як поєднання Y, Cb, Cr, де кожна з компонент є сумою значень RGB, перемножених на сталі коефіцієнти, після чого зображення бінаризується. В результаті навчання та валідації зображень, виявилося, що середня точність передбачень становить близько 70 %.

Також існує підхід до розпізнавання печаток, який в цілому фокусується на розпізнаванні геометричних форм, притаманних їм. Для цього використовується перетворення Хафа, оскільки його метою є виявлення кругів та квадратів. Також тут застосовується алгоритм згладжування, який прибирає шуми. Після усіх перетворень метод опорних векторів класифікує документи за наявністю/відсутністю печаток. За словами авторів, вони досягли 92 % точності роботи алгоритму [3]. Але тут варто зауважити, що даний підхід фокусується на розпізнаванні печаток, представлених у формі кругу чи квадрату, що є лише підмножиною усіх можливих

форм печаток. А тому результати дослідження є ідеалізованими і не відповідають реальним практичним задачам.

Наступним варіантом вирішення задачі розпізнавання печаток у документах може бути підхід, викладений у роботі [4]. Він полягає у наступному: відскановані зображення розбиваються за кольорами, які в свою чергу розділяються на кандидатів на печатку, після чого з зображення виділяються деякі його характеристики, які допомагають визначитися, чи об'єкт є печаткою, чи ні. Даний метод добре працює, коли документ є кольоровим і колір тексту відрізняється від кольору печатки, в інакшому випадку він не дає задовільних результатів.

Одним з методів, запропонованих останнім часом, є використання FCN (Fully Convolutional Neural Network). Автори представляють підхід, який розпізнає печатки на картинках документів, під назвою D-StaR [5]. Для кращої роботи нейронної мережі вони використовують переднавчену модель VGG-Net, вихідний результат роботи якої є вхідними даними для FCN. Такий підхід використовується, оскільки при його реалізації автори зіштовхнулися з нестачею даних для навчання і валідації (в доступі було 400 картинок), до того ж поділ на навчальну та тренувальну вибірки зроблений як 90 % і 10 % від усього обсягу даних. Тобто, валідація роботи методу проходила на зовсім малій вибірці, що варте зауваження. В результаті роботи D-StaR точність становить 87 %, але при розпізнаванні печаток, які накладаються на текст, вона падає до 74 %. Також можна відзначити роботу [6], де методи навчання застосовуються до розпізнавання традиційних монгольських печаток. Ключова ідея полягає у поєднанні аналізу головних компонент (PCA) та рекурентних нейронних мереж. Автори декларують високу точність, хоча можливо це пояснюється невеликими розмірами датасету.

Цікавим напрямком розвитку є логічне продовження ідеї перевірки документів, яке полягає у розпізнаванні підпису. Цій нетривіальній задачі присвячена робота [7], де пропонується спочатку "очищувати" сліди печатки (як правило підпис і печатка мають перекриватись, забезпечуючи додатковий захист документу) за допомогою генеративних нейромереж (GAN), а потім розпізнавання підпису виконується з використанням тих же згорткових мереж.

Постановка задачі

При роботі з документами деякі компанії стикаються з проблемами, які необхідно автоматизувати, оскільки їх вирішення співробітниками займає багато часу та не звільняє від помилок, які важко контролювати. Прикладами таких задач є розпізнавання печаток (чи їхню відсутність) у документах, розпізнавання підписів, виявлення слів, написаних однією мовою, у документі, написаному іншою.

Всі ці задачі можливо вирішити засобами машинного навчання.

Наразі проведено роботу з розпізнавання печаток у документах. Для цього оброблено документи, вивантаженні з сайту Prozorro [8], які є дозволеними для їх подальшого використання та розповсюдження. Тут варто враховувати, що документи представлені не тільки в форматі зображень (з розширенням ".png", ".jpg" або ".jpeg"), а також у вигляді документів PDF. Тому при формуванні навчальної вибірки (а також при подальшій класифікації документів, що мають печатку, або не мають її взагалі) слід перетворити усі документи, представленні у форматі PDF, на зображення.

Для навчання обраної моделі машинного навчання, на вхід подаються зображення документів, які вона класифікує як 0, якщо на зображенні немає печатки, та як 1, якщо має хоча б одну (рис. 1).

12.2. ПІДРЯДНИК несе персональну відповідальність за наявність ліцензій та інших дозвільних документів, необхідних для виконання робіт, визначених нормами чинного законодавства.
12.3. ПІДРЯДНИК є платником податку на загальних підставах.

13. ЮРИДИЧНІ АДРЕСИ СТОРІН.

ЗАМОВНИК:
Маріупольська загальноосвітня санаторна школа-інтернат І-ІІ ступенів №11 Донецької обласної ради
Юридична адреса: 87518, Донецька область, м. Маріуполь, вул. Бахчиванджи, 29
р/р
Держказначейська служба України м.Київ
ЄДРПОУ

Директор

М.П.

ПІДРЯДНИК:
ТОВ «АУДИТ-ПРОМ»
87400, Донецька область, Мангушський район смт. Мангуш, вул. Котанова, б.100
р/р
в МФ ПАТ «Приватбанк»
ЄДРПОУ
Витяг з реєстра платників ПДВ
№ від
Тел. +380.

Директор ТОВ «АУДИТ - ПРОМ»

М.П.

4. Доплати і нарахування

1. Компенсація витрат на відрядження експертів на засідання ДКЗ (без ПДВ):
2. Разом витрат:
3. ПДВ (20%):
4. ВСЬОГО:
5. Надбавка за виконання робіт, не передбачених планом (%):
6. Всього по договору:

грн. кон. (. грн. коп.), у т.ч. ПДВ – грн. коп.

Начальник управління
Начальник відділу
Головний економіст

*Примітка:** Коефіцієнт повних затрат враховує витрати, пов'язані з організаційно-господарською діяльністю ДКЗ. Величина коефіцієнту повних витрат визначається як відношення загальних витрат ДКЗ до вартості експертизи. (Порядок визначення вартості робіт по проведенню державної експертизи та оцінки запасів корисних копалин, затверджений наказом ДКЗ України від 02.08.1996р. №29 та зареєстрований в Міністерстві юстиції України 05.11.1996 за № 652/1677).
Загальні витрати складаються з фонду оплати праці працівників ДКЗ, податків і обов'язкових платежів, витрат на комунальні послуги, зв'язок та інші послуги, використання та обслуговування техніки, організаційне та інформаційне забезпечення експертних робіт і визначаються як правило один раз на рік за фактичними економічними даними минулого року і тими, що плануються у поточному році. Згідно з виконаними підрахунками коефіцієнт повних витрат на 2019 р. дорівнює 7,7 (додаток до наказу ДКЗ України від 28.02.2019 р. № 108).

Рис. 1. Приклад документів з печатками (зліва) та без печаток (справа)

Опис методу вирішення задачі

Розв'язувати задачу розпізнавання печаток у документах було вирішено за допомогою нейронних мереж. В ході виконання роботи було застосовано декілька різних моделей, в результаті чого підтвердилося припущення, що згорткові нейронні мережі є найбільш ефективним методом роботи з зображеннями (звичайні послідовні нейронні мережі дали 59.9 % точності, а рекурентні – 59.8 %, водночас як згорткові нейронні мережі на тих самих даних мають 88.03 % точності). Тому далі більш детально розглянуто саме згорткові нейронні мережі.

Перед етапом навчання моделі необхідно сформувати тренувальну та тестову вибірки, для чого: усі документи було перетворено у формат зображень, стандартизовано їх розмір та розбито на тренувальну та тестову вибірки випадковим чином так, щоб перша містила 70 % усіх зображень, а друга – відповідно, 30 %.

Для класифікації документів використовується послідовна модель, яка у своїй структурі має три згорткових шари для обробки зображень, та використовує ще три для їхньої класифікації.

Числові значення, показані на рис. 2, є розмірністю вихідного простору відповідного шару. До усіх шарів, окрім вихідного, застосовується активаційна функція "relu", останній же використовує "sigmoid". При заміні активаційної функції у вихідному шарі (зокрема, на активаційну функцію "softmax") точність роботи моделі зменшується приблизно у два рази.

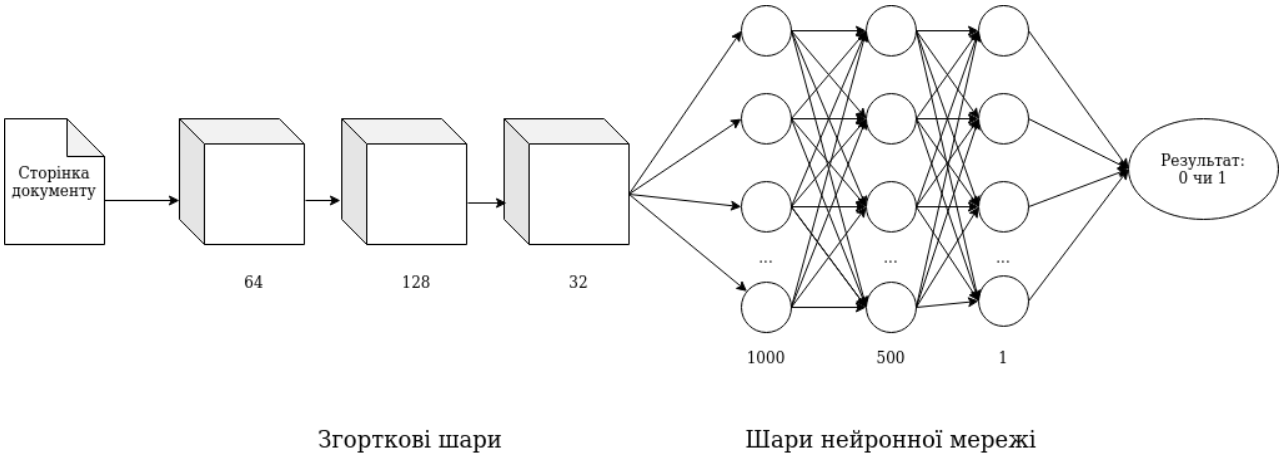

Рис. 2. Реалізація нейронної мережі для класифікації документів за наявністю або відсутністю печаток

Результати експериментів

Для порівняння правильності роботи моделі в залежності від кількості тренувальних даних, тестування проводилося на одному й тому самому наборі зображень. Це дає можливість побачити вплив більшої кількості даних на більш правильне налаштування моделі під час навчання.

Після отримання даних для навчання, набір поділяється на тренувальну та валідаційну вибірки, які використовуються для навчання моделі. Набір даних для валідації роботи моделі складає близько 25 % від загального обсягу тренувальних даних (рис. 3, 4).

Під час тестування роботи моделі, яку було навчено на 435 зображеннях, виявилося, що правильно класифіковано приблизно 81 % тестових даних. А після навчання нейронної мережі на 3216 об'єктах, доля правильних відповідей становила трохи більше 88 % (табл. 1).

Також варто зауважити, що на другій і третій ітерації точність роботи моделі на тестових даних трохи погіршилась, після чого досить різко виросла на двох останніх. Такий ефект можна приписати якості самих даних та їхньої попередньої класифікації, оскільки вона проводилася вручну, а при цьому можлива невелика похибка. Щодо самих даних, то при класифікації було виявлено велику кількість погано відсканованих документів, на яких відображалися печатки з наступних сторінок. В результаті було вирішено класифікувати такі зображення як ті, що не містять печаток, хоча їх на документі гарно видно.

При збільшенні розміру тренувальної вибірки час на навчання моделі зростає, а час роботи на тестових даних починає істотно збільшуватися тільки на останніх ітераціях, що зображено у табл. 2.

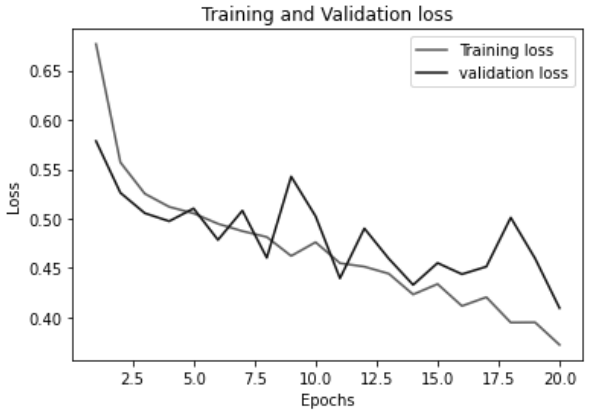

Рис. 3. Втрати в процесі тренування та валідації моделі

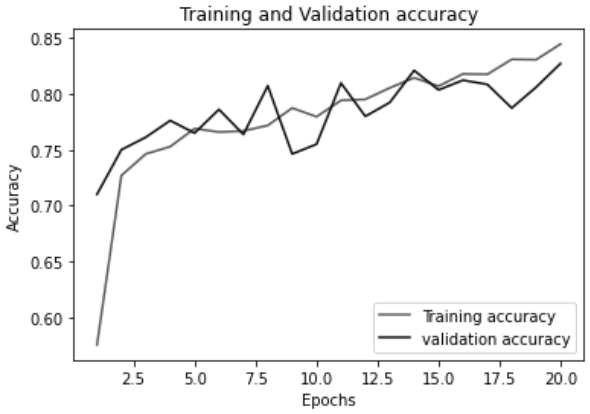

Рис. 4. Точність в процесі тренування та валідації моделі

Таблиця 1. Вплив розміру тренувального набору даних на точність роботи нейронної мережі

Розмір тренувальної вибірки	Розмір валідаційної вибірки	Розмір тестової вибірки	Точність роботи моделі
435	109	234	81.19 %
1164	291		80.76 %
1896	474		80.76 %
2596	650		86.32 %
3216	804		88.03 %

Таблиця 2. Вплив розміру тренувального набору даних на час роботи нейронної мережі

Розмір тренувальної вибірки	**Розмір валідаційної вибірки**	**Розмір тестової вибірки**	**Час, необхідний на навчання моделі (у хв)**	**Час роботи мережі на тестових даних (у сек)**
435	109	234	0,66	0,23
1164	291		1,78	0,22
1896	474		2,86	0,21
2596	650		3,86	0,27
3216	804		4,81	0,37

Висновки

Машинне навчання широко використовується для роботи з зображеннями, особливо ефективним підходом до їхньої обробки є використання згорткових нейронних мереж. Через структуру архітектури мереж даного типу вони гарно працюють з зображеннями, тому широко використовуються для роботи з цим типом даних.

Автоматизоване розпізнавання різних типів помилок у документах є актуальною темою, що потребує вирішення.

Прикладом таких помилок є відсутність печаток на документах, для розпізнавання чого в рамках даної курсової роботи було побудовано нейронну мережу, здатну класифікувати дані, що подаються на вхід у вигляді зображень, за наявністю або відсутністю печаток.

Початкова точність роботи даної моделі складала 81.19 %, при цьому для її навчання використовувалося 435 зображень (для тренування мережі) і 109 (для валідації її роботи). Наступні дві спроби навчити модель на більшій кількості даних, ніж при попередній спробі, дали трохи гірші результати, а саме точність роботи моделі на тестовому наборі даних складала 80.76 % в обох випадках. Спочатку для навчання мережі використовувалося 1164 зображення (і 291 для валідації), після чого на вхід моделі було подано 1896 зображень для навчання (і 474 для валідації). Останні два процеси навчання на більшій кількості даних (2596 для навчання і 650 для валідації в першому випадку, 3216 для навчання і 804 для валідації – в другому), показали покращення при класифікації тестового набору, що дає 86.32 % і 88.03 % правильних відповідей.

Література

1. Forczmanski P., Markiewicz A. Two-stage approach to extracting visual objects from paperdocuments. *Machine Vision and Applications*. 2016. N 27. P. 1243–1257.
2. Forczmanski P., Markiewicz A. Stamps Detection and Classification Using Simple Features Ensemble. *Mathematical Problems in Engineering*. 2015.
3. Roy P., Pal U., Lladós J. Seal Detection and Recognition: An Approach for Document Indexing [Електронний ресурс]. 10th International Conference on Document Analysis and Recognition. 2009. Режим доступу до ресурсу: https://www.researchgate.-net/publication/220861099_Seal_Detection_and_Recognition_An_Approach_for_Document_Indexing.
4. Micenkova B., van Beusekom J., Shafait F. Stamp Verification for Automated Document Authentication [Електронний ресурс]. Режим доступу до ресурсу: http://pure.au.dk/-portal/files/51730044/Barbora_Stamp_Verification_IWCF12.pdf.
5. D-StaR: A, Younas M., Afzal M., Malik та ін. Generic Method for Stamp Segmentation from Document Images [Електронний ресурс]. 2017. Режим доступу до ресурсу: https://tukl.seecs.nust.edu.pk/members/project

s/conference/D-StaR-A-Generic-Method-for-Stamp-Segmentation-from-Document-Images.pdf.
6. Gantuya P., Mungunshagai B., Suvdaa B. "Mongolian Traditional Stamp Recognition using Scalable kNN." International journal of advanced smart convergence 4.2 (2015): 170–176.
7. Engin Deniz, et al. "Offline Signature Verification on Real-World Documents." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020.
8. Офіційний портал оприлюднення інформації про публічні закупівлі України [Електронний ресурс]. Режим доступу до ресурсу: https://prozorro.gov.ua.

References

1. Forczmanski P., Markiewicz A. Two-stage approach to extracting visual objects from paperdocuments. *Machine Vision and Applications*. 2016. N 27. P. 1243–1257.
2. Forczmanski P., Markiewicz A. Stamps Detection and Classification Using Simple Features Ensemble. *Mathematical Problems in Engineering*. 2015.
3. Roy P., Pal U., Lladós J. Seal Detection and Recognition: An Approach for Document Indexing [Електронний ресурс]. 10th International Conference on Document Analysis and Recognition. 2009. Режим доступу до ресурсу: https://www.researchgate.-net/publication/220861099_Seal_Detection_and_Recognition_An_Approach_for_Document_Indexing.
4. Micenkova B., van Beusekom J., Shafait F. Stamp Verification for Automated Document Authentication [Електронний ресурс]. Режим доступу до ресурсу: http://pure.au.dk/-portal/files/51730044/Barbora_Stamp_Verification_IWCF12.pdf.
5. D-StaR: A, Younas M., Afzal M., Malik та ін. Generic Method for Stamp Segmentation from Document Images [Електронний ресурс]. 2017. Режим доступу до ресурсу: https://tukl.seecs.nust.edu.pk/members/projects/conference/D-StaR-A-Generic-Method-for-Stamp-Segmentation-from-Document-Images.pdf.
6. Gantuya P., Mungunshagai B., Suvdaa B. "Mongolian Traditional Stamp Recognition using Scalable kNN." International journal of advanced smart convergence 4.2 (2015): 170–176.
7. Engin Deniz, et al. "Offline Signature Verification on Real-World Documents." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020.
8. Official portal for publishing information on public procurement in Ukraine [Electronic resource]. Access to the resource: https://prozorro.gov.ua.

Одержано 04.11.2020

Про авторів:

Жиркова Анастасія Павлівна,
студентка Національного університету "Києво-Могилянська Академія".
Кількість наукових публікацій в українських виданнях – 1.
https://orcid.org/0000-0002-4604-1137,

Ігнатенко Олексій Петрович,
доктор фізико-математичних наук,
провідний науковий співробітник.
Кількість наукових публікацій в українських виданнях – 27.
Кількість наукових публікацій в зарубіжних виданнях – 7.
http://orcid.org/0000-0001-8692-2062.

Місце роботи авторів:

Національний університет
"Києво-Могилянська Академія",
вулиця Григорія Сковороди, 2, Київ.

Інститут програмних систем
НАН України, 03187, Київ-187,
проспект Академіка Глушкова, 40.

E-mail: nastia.nastia.zh@gmail.com,
o.ignatenko@gmail.com

УДК004.05 https://doi.org/10.15407/pp2020.04.088

О.О. Летичевський, С.О. Горбатюк, В.О. Горбатюк

АЛГЕБРАЇЧНЕ МОДЕЛЮВАННЯ В СИСТЕМАХ МІЖНАРОДНОЇ ТА МІСЦЕВОЇ ОБСЛУГОВУЮЧОЇ ЛОГІСТИКИ

Міжнародна та внутрішня обслуговуюча логістика розвивається шаленими темпами у сучасному житті, а прогнози на майбутнє для цієї галузі оптимістичні. З цих причин ми постали перед задачею контролю, оптимізації, безпечної та надійної перевірки комплексних логістичних систем з багатьма внутрішніми агентами, які працюють у середовищі, що змінюється. Дана робота має на меті показати як математичне моделювання, зокрема, алгебра поведінок, надає можливість передбачувати поведінку, стабільність логістичних середовищ та перевіряти їх властивості безпеки, та надійності. Ці процедури є модельно-орієнтованими розробками комплексних систем програмного забезпечення, таких як логістичні системи.

Ключові слова: алгебра поведінок, інсерційне моделювання, формальна верифікація, ланцюг постачання, модельно-орієнтовані розробки, властивості безпеки.

1. Загальна інформація

Стабільність функціонування, стійкість до зовнішніх загроз, виявлення та усунення вразливостей є необхідними властивостями комплексних систем міжнародної та внутрішньої обслуговуючої логістики. Такі системи програмного забезпечення є критично важливими до безпеки та вимагають розробок з модельно-орієнтованим підходом для надійності. Модельний підхід до розробки передбачає створення моделей як інструментів на кожному етапі розробки програмного забезпечення для застосування методів верифікації, тестування та валідації.

Методи математичного моделювання з такими інструментами, як алгебра поведінок, дають можливість керувати логістичними системами та забезпечувати їх безпеку та захищеність за допомогою формальної верифікації та формалізації за допомогою тестування на основі моделей. Алгебраїчні моделі логістичних систем можуть бути використані для аналізу поведінки всіх залучених агентів і доведення їх здатності виконувати свої цілі та здатності всієї системи постійно існувати та залишатися стабільною.

Формальна верифікація використовується в аналізі та верифікації бізнес-логістики вже декілька десятиріч. Формальні мови такі як: BPMN [1], UML [2], SysML [3] використовуються для опису логістичних бізнес процесів, а досить велика кількість методів такі як: VDM [4], SPIN [5] та інші застосовується для перевірки властивостей безпеки. Ускладнення специфіки та збільшення інформаційного навантаження обумовило використання більш новітні методики та відповідні математичних теорії.

У цьому дослідженні ми демонструємо, як алгебраїчний підхід, оснований на алгебрі поведінок, може бути ефективно використаний для перевірки властивостей безпеки складних систем. Як приклади ми застосовуємо її до реальної закритої логістичної системи місцевого обслуговування (фермерське господарство) та відкритої міжнародної логістичної системи (експортні поставки виробництва вищезгаданого фермерського домогосподарства).

2. Алгебра поведінок

На початку 1970-х, В.М. Глушков, директор Інституту кібернетики Національної академії наук України, заснував комп'ютерну наукову школу, яка сфокусувалась на алгебраїчних методах. Тематика дослідження варіювалась від автоматизованих доведень теорем до алгебраїчного моделювання та від наукових концепцій до промислових прикладних програм. Пізніше в 2000 Система Алгебраїчного Програмування (APS) [1] як складова частина цієї

програми досліджень розширилась за допомогою включення в себе концепції взаємодії перехідних систем у деяких алгебраїчних середовищах. Ключова ідея – інсерційне функціонування, яке визначає поведінку взаємодії перехідних систем в алгебраїчному середовищі. Ця концепція розроблена в контексті IMS [2]. Алгебра поведінок та основні специфікації протоколу використовується для формального опису моделі. Такі алгебраїчні інструменти, як теореми та абстрактні алгоритми, служать основою для формальних методів верифікації. Алгебраїчний підхід та інсерційне моделювання дозволяють довести або спростувати властивості, подавши контр-приклади для системи з довільною кількістю агентів. Використовуючи формальну модель програми на якомусь рівні абстракції, ми можемо генерувати різні сценарії поведінки агентів або груп агентів. Ці сценарії є символічними і можуть бути проілюстровані конкретними прикладами за допомогою формальних методів. Генерація символічних сценаріїв забезпечує гарне висвітлення поведінки, а отже, отримані конкретні сценарії можуть розглядатися як тестовий набір для створеного програмного забезпечення.

Інсерційне моделювання фокусується на моделях побудови та вивченні взаємодії агентів та середовищ у складних розподілених мультиагентних системах. Загальні концепції інерційного моделювання – це ієрархія середовищ та агентів, занурених у ці середовища, взаємодія цих агентів, середовищ та середовищ вищого рівня, двосторонній вплив агентів та середовищ один на одного, зміна поведінки набору агентів під час занурення в нові середовища. Середовище – це агент, який має функцію занурення. Агенти розглядаються як системи переходу атрибутів. У таких системах стани визначаються значеннями атрибутів. Агенти мають набір атрибутів, які визначають тип агента. Атрибути середовища пов'язані з глобальними атрибутами, які відомі всім агентам.

В 1997 Гілберт та Летичевський ввели поняття алгебри поведінок [3] як інструмент та невід'ємна частина інсерційного моделювання. Алгебра поведінок – це універсальна алгебра двох видів. Основний різновид – це набір поведінок, а другий – сукупність дій. Ця алгебра має дві операції, три термінальні константи та відношення наближення. Операції позначаються префіксами a.u (де «a» – дія, а «u» – поведінка) та недетермінований вибір поведінки u + v (асоціативні, комутативні та ідемпотентні операції на множині поведінок). Кінцевими константами є успішно визначене Δ, заключення 0, та дивергентна поведінка ⊥. Знаходження наближеного зв'язку ⊑ є частковим порядком у наборі поведінок з мінімальним елементом ⊥. Алгебра поведінок також наповнена двома операціями: паралельними (||) та послідовними (;) композиціями поведінок.

Одним з прикладів вираження поведінок є:

$$B0 = a1.a2.B1 + a3.B2,$$

$$B1 = a4,\ B2 = \ldots$$

Мається на увазі, що поведінка B0 може бути представлена, як послідовність дій a1 та a2 та поведінки B1, або як послідовність дії a3 та поведінки B2. Поведінка B1 закінчується після дії a4 [4].

3. Приклад формалізації системи взаємодії агентів закритої логістичної системи

Розглянемо застосування на практиці алгебри поведінок на прикладі діючої закритої логістичної системи фермерського господарства. Закрите середовище містить у собі набір агентів, що мають певні атрибути, властивості та список дій, які вони виконують за певних передумов. Контроль та взаємодія агентів даної системи здійснюється за допомогою Централізованої бази даних (сервера), яка збирає інформацію про стан кожного агента та координує їх взаємодію за допомогою команд. Централізована база даних є одним із агентів системи, функція якого є безперервне отримання, накопичення та обробка інформації від агентів. За умови виконання певної передумови Централізована база даних надсилає агенту сповіщення-команду (коротке сервісне повідомлення) про початок виконання певної дії. Активність

кожного агенту описана математичною моделлю поведінок, яка передбачає хід усіх можливих дій такого агенту. Моделювання поведінок усіх агентів дає змогу проаналізувати цілісну комплексну логістичну систему в роботі та виявити її вразливості. Список основних агентів та їх дій наведено в табл. 1.

Таблиця 1. Список агентів та їх дій

Агент	**Список дій**
Агроном	– дає команду про готовність сої для збору урожаю
Комбайн (один або більше)	– простій (або зупинка) – збір урожаю та вивантаження на ходу у вантажівку – технічне обслуговування – заправка – заміна водія при цілодобовій роботі позмінно
Бензовоз	– простій (або зупинка) – забір пального на базі – заправка транспортних засобів та механізмів у разі отримання сигналу від централізованої бази даних про низькі об’єми палива, отримані від датчиків палива цих засобів та механізмів – заміна водія при цілодобовій роботі позмінно – технічне обслуговування – заправка
Вантажівка (одна або більше)	– простій (або зупинка) – супровід комбайна під час збору урожаю та приймання сої на ходу – транспортування на склад та вивантаження – технічне обслуговування – заправка – заміна водія при цілодобовій роботі позмінно
Склад та лінія переробки	– простій (або зупинка) – приймання сої та зважування – контроль показників якості – контроль вологості та досушка, провіювання – переробка сої на соєву олію – зважування готової продукції – відвантаження готової продукції – заміна працівників при цілодобовій роботі позмінно – передача отриманої інформації на всіх етапах до ЦБД
Фура	– простій (або зупинка) – очікування завантажування на складі – завантажування та зважування – доставка кінцевому споживачеві (по Україні або за кордон)
Сервісна служба (механік)	– простій (або зупинка) – виконання технічного обслуговування – заправка – заміна водія при цілодобовій роботі позмінно
Централізована база даних (ЦБД)	– безперервний збір та обробка інформації від GPS та інших датчиків від всіх агентів на всіх етапах – передача команд та інформації відповідним агентам – формування бази даних – опрацювання отриманої інформації на всіх етапах від агенту Склад та лінія переробки, внесення до операційної бази даних – надання інформації зацікавленим сторонам та агентам

Приклади запису моделей поведінок агентів за допомогою алгебри поведінок, розглянемо поведінки декількох базових агентів:

1. Комбайн

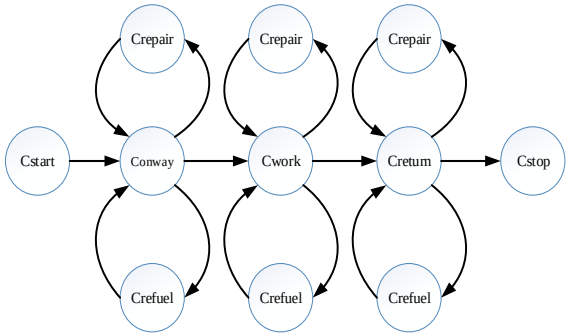

Рис. 1. Схема математичної моделі комбайну

Математична поведінкова модель:

```
C = Cstart.Conway.Cwork.Creturn.Cstop

Conway =
Cmove.Conway + Cmove.Crepair.Conway +
Cmove.Crefuel.Conway

Creturn =
Cmove.Creturn + Cmove.Crepair.Creturn
+ Cmove.Crefuel.Creturn

Cmove =
(PdX.Cmove + PdY.Cmove) + Conplace.
```

Поведінка переміщення комбайна до робочого місця Conway та назад до бази Creturn складається з поведінки циклу Cmove, що закінчується Conplace та поведінок ремонту Crepair та заправки Crefuel.

```
Cwork =
Cgath.Cwork + Cgath.Crepair.Cwork +
+ Cgath.Crefuel.Cwork

Cgath =
SdX.Cgath + Sdy.Cchdir.Cgath) + Send.
```

Поведінка роботи комбайна Cwork складається з поведінки циклу Cgath, що закінчується Send та поведінок ремонту Crepair та заправки Crefuel.

```
Crepair = Cstop.Cdamage,
```

Поведінка ремонту комбайна Crepair є дії Cstop та Cdamage.

```
Crefuel = Cstop.Cfuel.
```

Поведінка заправки комбайна Crefuel є дії Cstop та Cfuel.

2. Бензовоз

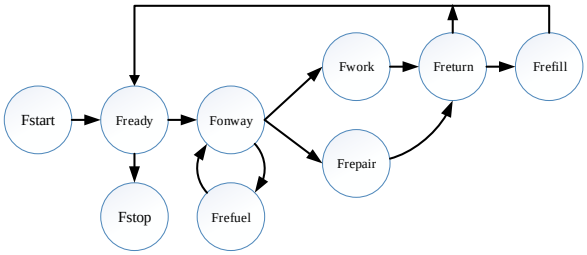

Рис. 2. Схема математичної моделі бензовозу

Математична модель поведінки:

```
F = Fstart.Fcycle.Fstop,

Fcycle = Fready.Fcycle +
Fready.Fonway.Fwork.Freturn.Frefill.F
cycle + Fready.Fonway.Freturn.Fcycle.
```

Поведінка робочого циклу бензовоза Fcycle – цикл зміни дій:

```
Fonway =
Fmove.Fonway + Fmove.Frepair +
Fmove.Frefuel.Fonway,

Freturn =
Fmove.Freturn + Fmove.Frepair.Freturn
+ Fmove.Frefuel.Freturn.
```

Поведінка переміщення бензовоза до робочого місця Fonway та назад до бази Freturn складається з поведінок ремонту Frepair, заправки Frefuel та поведінки циклу Fmove.

```
Fmove = (SdX.Fmove +
SdY.Fmove) + Fonplace.
```

Поведінка переміщення бензовоза до робочого місця Fmove є циклом зміни положення, що закінчується Fonplace або поведінками ремонту Frepair та заправки Frefuel.

```
Frepair = Fstop.Fdamage,
```

Поведінка ремонту бензовоза Frepair є дії Fstop та Fdamage.

```
Frefuel = Fstop.Ffuel.Fwork.
```

Поведінка заправки бензовоза Frefuel є дії Fstop, Ffuel та Fwork.

3. Механік

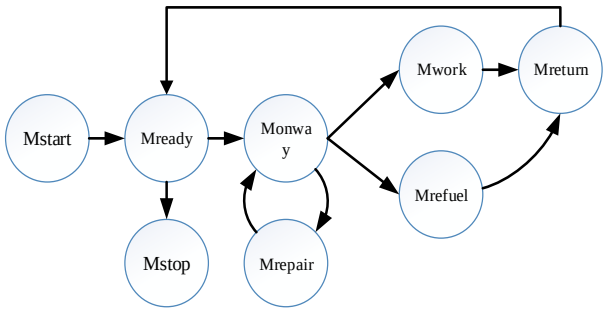

Рис. 3. Схема математичної моделі механіка.

Математична модель:

```
M = Mstart.Mcycle.Mstop

Mcycle = Mready.Mcycle + Mready.Monwa
y.Mwork.Mreturn.Mcycle + Mready.Monwa
y.Mreturn.Mcycle
```

Поведінка робочого циклу механіка Mcycle – цикл зміни дій

```
Monway =
Mmove.Monway + Mmove.Mrefuel+
+ Mmove.Mrepair.Monway

Mreturn =
Mmove.Mreturn + Mmove.Mrepair.Mreturn
+ Mmove.Mrefuel.Mreturn
```

Поведінка переміщення механіка до робочого місця Monway та назад до бази Mreturn складається з поведінок ремонту Mrepair, заправки Mrefuel та поведінки циклу Mmove

```
Mmove = (SdX.Mmove + SdY.Mmove)+
+ Monplace
```

Поведінка переміщення механіка до робочого місця Mmove – цикл зміни положення, що закінчується Monplace або поведінками ремонту Mrepair та заправки Mrefuel

```
Mrepair = Mstop.Mdamage.Mwork
```

Поведінка ремонту механіка Mrepai – дія Mstop та Mdamage

```
Mrefuel = Mstop.Mfuel
```

Поведінка заправки механіка Mrefuel – діяї Mstop, Mfuel та Mwork.

4. Вантажівка

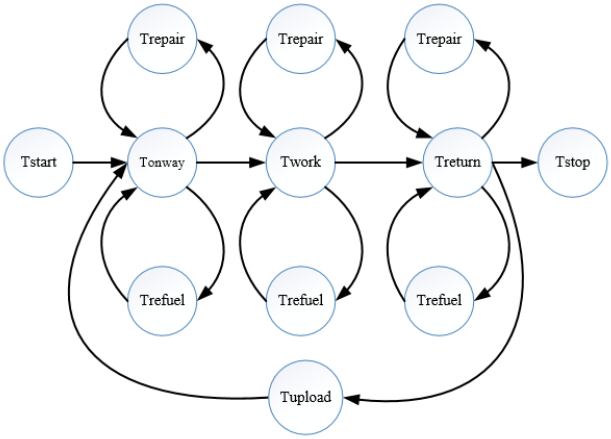

Рис. 4. Схема математичної моделі вантажівки.

Математична модель:

```
T = Tstart.Tonway.Twork.Treturn.Tstop
 Tonway =

Tmove.Tonway + Tmove.Trepair.Tonway +
 Tmove.Trefuel.Tonway

Treturn =
Tmove.Treturn + Tmove.Trepair.Treturn
 + Tmove.Trefuel.Treturn

 Tmove =  (PdX.Tmove  +  PdY.Tmove)
+ Tonplace
```

Поведінка переміщення вантажівки до робочого місця Tonway та назад до бази Treturn складається з поведінки циклу Tmove, що закінчується Tonplace та поведінок ремонту Trepair та заправки Trefuel

```
Twork = Tgath.Twork +
+ Tgath.Trepair.Twork +
```

```
+ Tgath.Trefuel.Twork +
+ Tgath.Tunload.Twork
Tgath =          (SdX.Tgath          +
Sdy.Tchdir.Tgath) + Send
```

Поведінка роботи вантажівки Twork складається з поведінки циклу Tgath, що закінчується Send та поведінок ремонту Trepair та заправки Trefuel

```
Trepair = Tstop.Tdamage
```

Поведінка ремонту вантажівки

Trepair є дії Tstop та Tdamage

```
Trefuel = Tstop.Tfuel
```

Поведінка заправки вантажівки Trefuel є дії Tstop та Tfuel

```
Tunload = Treturn.Treload.Tonway
```

Поведінка вигрузки вантажівки Tunload є дії Treturn, Treload та Tonway.

Крім поведінок ми маємо також список дій, що можуть бути представленими перед- та післяумовою. Ми наводимо як приклад дії комбайну з описом відповідної семантики. Ми не наводимо поведінку інших агентів таких як склад, фура, механіка, вантажівки, централізована база даних у зв'язку з великою кількістю формалізації (табл. 2).

Таблиця 2. Список дій та їх передумов та післяумов

Наза дії	**Передумова**	**Післяумова**	**І-ція переходу**	**Коментар**
Cstart	1 (безумовна дія)	(X==Xs) && (Y ==Ys)		Початок роботи
Cdamage	1<=i<=combineQuantity	combine(i).state= DAMAGE	send help(i)	Якщо трапилась поломка відіслати сигнал про допомогу та перейти в стадію DAMAGE
Cstop	1<=i <= combineQuantity	combine(i).state= STOP		Зупинитися та перейти в стадію STOP
Cfuel	1<=i<=combineQuantity) && (combine(i).fuelLevel <= 0.2 * combine(i).fuelVolume)	combine(i).state = FUEL	send help(i)	Якщо рівень пального менше норми відіслати сигнал про допомогу та перейти в стадію FUEL
PdX	X < Xp	X = X + deltaX		Якщо не на місці призначення, то рухатись на якусь deltaX
PdY	Y < Yp	Y = Y + deltaY		Якщо не на місці призначення, то рухатись на якусь deltaY
SdX	((dir > 0) && (X < XLe)) || ((dir < 0) && (X > XLs))	X = X + deltaX * dir		Якщо не кінець довжини ділянки, то рухатись на якусь deltaX
SdY	(((dir > 0) && (X > XLe)) || ((dir < 0) && (X < XLs))) && (Y < YWe)	Y = Y + deltaY		Якщо кінець довжини ділянки, але не кінець ширини поля, то починаємо нову смугу на полі
Cchdir	(((dir > 0) && (X > XLe)) || ((dir < 0) && (X < XLs))) && (Y < YWe)	dir = -dir		Якщо кінець довжини ділянки, але не кінець ширини поля, то змінюємо напрямок
Conplace	(X == Xp) &&(Y == Yp)	1		Якщо комбайн на місці призначення

4. Формальна верифікація моделей

Дана методика формалізації дозволяє застосувати далі формальні методи верифікації, таких як перевірка властивостей безпеки та випадки загрози для безперервної життєдіяльності системи. Надзвичайно важливо перевіряти деякі ключові фактори, які впливають на можливість виконання злагоджених дій усіх агентів системи, що в результаті і відображається в безперервній та продуктивній діяльності системи. Розглянемо основні властивості безпеки, що перевіряються в логістичній системі фермерського господарства.

1. Терміни збору врожаю. Дана властивість може бути виражена формулою у базовій логічній мові, що вибрана для формалізації дій (T <= 90 днів). При розгляді довільної кількості комбайнів та обслуговуючого транспорту така ситуація може бути при нераціональній кількості транспортних та стану техніки. Для перевірки властивості використовуються методи моделювання такі як символьне, конкретне та змішане моделювання системи. На початку враховуються такі умови, як кількість допустимих несправностей техніки (як арифметична нерівність), конкретна кількість одиниць техніки та відстані до місця роботи. Для визначених початкових даних проводиться конкретне моделювання та визначається властивість порушення термінів збору врожаю. При використанні зворотного символьного моделювання від стану, що відповідає дотримання термінів та гіпотетичних можливих кількостей одиниць, несправностей та відстаней до місця роботи, можливо визначити початкові умови при яких ця властивість не буде порушена. Слід відмітити, що дана технологія не вирішує задачу оптимізації, а лише перевіряє можливість її реалізації. Методи символьного моделювання розглянуті в [9].

2. Достатність кількості обслуговуючих агентів (типу бензовоз чи механік) для обслуговування життєдіяльності головних агентів системи, здатність вчасно прибути до заданої точки призначення для виконання заданих Центральною базою даних дій. Дані властивості перевіряються аналогічно при висуванні деяких гіпотез при проектуванні задачі або реальної наявності техніки. Такі властивості також представляються за допомогою нерівностей (або рівностей).

3. Перевірка злагодженості та узгодженості алгоритмів дій агентів (наприклад, дії бензовоза при одночасному виклику обслуговування декількох агентів). Дана властивість може бути виражена за допомогою алгебри поведінок. Наприклад, чи встигне бензовоз заправити всі комбайни, якщо вони одночасно зупиняться, за встановлений термін.Маємо властивість безпеки (T<=2 год) та саму поведінку:

```
Cdamage.Cdamage.Cdamage.Cdamage
.X;(Trepair.Trepair.Trepair.Trepair).
```

Розглянутий приклад ілюструє поведінку чотирьох комбайнів, які мали несправність та були полагоджені за деякий час. Чи відповідає цей час встановленому-визначає конкретне або змішане моделювання. Зауважимо, що при символьному моделюванні можливо оцінити властивість для довільної кількості комбайнів.

4. Вплив непередбачуваних факторів типу погоди, рельєфу та інше. Дана властивість вивчається за допомогою встановлення можливих дій, що моделюють дані фактори та відповідними видами моделювання.

5. Дотримання умов перевезення вантажу (в нашому випадку зібраного урожаю) при передачі від одного агента іншому. При перевезенні розглядаються додаткові фактори середовища, які моделюються в процесі верифікації. Це час перевезення, вологість, температура, які беруться до уваги в моделі, як додаткові атрибути системи.

5. Верифікація і формалізації системи міжнародної логістики

Відкрита система міжнародної логістики є продовженням описаної закритої логістичної обслуговуючої системи фермерського господарства. Вихідний продукт фермерського господарства (наприклад, соя) є вхідним продуктом для відкритої міжнародної логістичної системи (переробка сої на соєву олію та її експорт). Істотною відмінністю цієї системи є теоре-

тично нескінченна кількість не пов'язаних між собою незалежних агентів кожного типу, і, як наслідок, постійна зміна середовищ існування. Аналогічно з закритою логістичною системою, де функцію збору інформації та контролю виконує Централізована база даних, в даній відкритій системі існує подібний агент – Система. Однак на відміну від закритої системи, де агенти діють злагоджено та в спільних інтересах, вимоги надійності та безпеки до відкритої системи значно вищі, так як кожен з незалежних агентів діє в своїх особистих інтересах. Тому для даних цілей доцільно застосовувати технологію розподілених децентралізованих систем (блокчейн) замість стаціонарного серверу. Список типових основних агентів, їх функції та інформацію, яку вони надають розподіленій системі, наведено в табл. 3.

Таблиця 3. Список агентів, їх функцій та інформації, яку вони надають централізованій системі

Agent	**Функції**	**Інформація, яку надає для системи**
Виробник (він же експортер) товару	– виробництво товару – пошук покупця – аналіз цін (пропозицій) та вибір оптимального покупця – відвантаження товару – пошук оптимальних логістичних агентів – портових операторів та морських перевізників	наявний об'єм товару до відвантаження – базові характеристики та якість товару (замовляється експортером) – узгоджена ціна – узгоджений покупець – узгоджені з Покупцем умови відвантаження та оплати всіх дій (хто оплачує послуги посередників та на яких етапах здійснення) – дата готовності до відвантаження – узгоджені логістичні агенти – портовий оператор та морський перевізник
Портовий оператор	– здійснення внутрішньої обслуговуючої логістики від виробника до завантаження на морське судно	– дата подачі контейнера виробнику згідно дати готовності до відвантаження, надану виробником (пізніше або в день його дати) – підтвердження здійснення та дата проміжних етапів: відвантаження у виробника, прибуття в порт, завантаження на судно та інші можливі етапи
Морський перевізник	– приймає на борт судна контейнер від Портового оператора та здійснення перевезення	– факт та дата прийняття контейнера на борт судна
Покупець товару	– пошук продавця – аналіз цін (пропозицій) та вибір оптимального продавця – оплата за товар	– бажаний об'єм товару для відвантаження – мінімальні характеристики та якість товару (підтверджується продавцем) – узгоджена ціна – узгоджений продавець – узгоджені з продавцем умови відвантаження та оплати всіх дій (хто оплачує послуги посередників та на яких етапах здійснення) – підтвердження наданих продавцем логістичних агентів – портовий оператор та морський перевізник
Банк кожного з агентів	– здійснює оплату за товар чи послуги згідно інформації, отриманої від свого агенту	– інформація про суму та дату здійснення оплати, кому з агентів та за яку роботу

Поведінка та дії агентів описуються аналогічно моделі фермерського господарства. До моделі включаються відстані та географічна інформація про точки в яких діють агенти. Як властивості які можуть бути перевірені такі:

- своєчасна доставка;
- дотримання температурного режиму;
- дотримання режиму вологості;
- здатність протистояти зовнішнім факторам.

Властивості перевіряються такими методами алгебри поведінок, як алгебраїчне зіставлення, символьне моделювання системи із довільною кількістю агентів та статичні методи доведення. Специфіка системи – використання децентралізованих систем, які є однотипними, але зберігають всі атрибути, які стосуються всього ланцюга постачання, дані про перевозку, транзакції з оплатою тощо. Кожен агент зберігає файл з інформацією про ланцюг постачання або транзакції. Ми розглядаємо два типи використання формальних методів в даній моделі.

1. Моделювання ланцюга постачання для перевірки властивостей. В даній процедурі ми перевіряємо чи відповідає вибраний ланцюг постачання вимогам перевезення. Перевіряються також властивості супротиву до зовнішніх небажаних дій – погоди, шахрайства, кібератаки.

2. Моніторинг перевезення згідно вибраної моделі безпеки. Дана процедура призначена для виявлення порушення властивостей безпеки заздалегідь при вивченні сценарію поведінки. При моніторингу разом із моделлю безпеки може бути використана модель класифікації, що створена за допомогою машинного навчання.

Висновки

У цій роботі ми дослідили можливі підходи до застосування математичного моделювання та застосування алгебри поведінок для прогнозування безпеки та надійності логістичних систем відкритого та закритого типів. Наші експериментальні результати підтвердили велику значущість використання математичного моделювання та використання алгебри поведінок для застосування в реальних логістичних системах.

Для складних комплексних систем, такі як логістичні, вкрай важливе значення має перевірка властивостей безпеки та здатності функціонувати тривалий час без втрати результативності. Дана методика є одним з вагомих інструментів при виконанні цих задач. Моделювання поведінки кожного агента системи дає змогу проаналізувати властивості всієї системи загалом.

На даний час перспектива цієї роботи вбачається у поєднанні описаної методики з розподіленими системами (зокрема блокчейн). Вже розроблена платформа-клієнт, що виступає незалежним агентом, що контролює дії усіх учасників, та найближчим часом планується експеримент з використання даного проекту при реальних експортних відправках.

Література

1. BPMN (Business Process Model and Notation). www.bpmn.org
2. UML (Unified Modelling Language). www.uml.org
3. SysML (Systems Modelling Language). www.sysml.org
4. VDM (Vienna Development Method). www.vienna.cc/e/evdm.htm
5. SPIN. http://spinroot.com/spin/whatispin.html
6. APS (Algebraic Programming System).www.apsystem.org.ua
7. Letichevsky A., Letychevskyi O., and Peschanenko V. Insertion Modeling and Its Applications. *Computer Science Journal of Moldova*. 2016. vol. 24. N. 3. P. 357–370.
8. Letichevsky A. and Gilbert D. Interaction of agents and environments. In *Recent Trends in Algebraic Development Technique.* LNCS 1827. Springer-Verlag. 1999.
9. Letychevskyi O,, Letichevsky A., Peschanenko V., Weigert T., Insertion modeling and symbolic verification of large systems,"I LNCS 9369 SDL 2015: Model-Driven Engineering for Smart Cities. Springer. 2015. P. 3–18.

References

1. 1 BPMN (Business Process Model and Notation). www.bpmn.org
2. UML (Unified Modelling Language). www.uml.org
3. SysML (Systems Modelling Language). www.sysml.org
4. VDM (Vienna Development Method). www.vienna.cc/e/evdm.htm
5. SPIN. http://spinroot.com/spin/whatispin.html
6. APS (Algebraic Programming System).www.apsystem.org.ua
7. Letichevsky A., Letychevskyi O., and Peschanenko V. Insertion Modeling and Its Applications. *Computer Science Journal of Moldova*. 2016. vol. 24. N. 3. P. 357–370.
8. Letichevsky A. and Gilbert D. Interaction of agents and environments. In *Recent Trends in Algebraic Development Technique*. LNCS 1827. Springer-Verlag. 1999.
9. Letychevskyi O,, Letichevsky A., Peschanenko V., Weigert T., Insertion modeling and symbolic verification of large systems,"I LNCS 9369 SDL 2015: Model-Driven Engineering for Smart Cities. Springer. 2015. P. 3–18.

Одержано 23.10.2020

Про авторів:

Летичевський Олександр Олександрович,
доктор фізико-математичних наук,
завідувач відділу теорії цифрових автоматів
Кількість наукових публікацій в українських виданнях – 32.
Кількість наукових публікацій в зарубіжних виданнях – 37.
Індекс Гірша – 4,

Горбатюк Сергій Олександрович,
аспірант кафедри Комп'ютерних наук та інформаційних технологій.
https://orcid.org/0000-0001-6834-4211,

Горбатюк Віктор Олександрович,
аспірант кафедри Комп'ютерних наук та інформаційних технологій.
https://orcid.org/0000-0001-7544-0260.

Місце роботи авторів:

Інститут кібернетики імені В.М. Глушкова
Національної академії наук України.
03187, м. Київ
проспект Академіка Глушкова, 40.

Тел.: (044) 526-20-08.

E-mails: gorbatiuk_sergiy@i.ua,
lit@issukraine.com,
viktor.gorbatiuk@gmail.com

UDC 004.5 https://doi.org/10.15407/pp2020.04.098

L.A. Girchenko, A.Y. Doroshenko, Y.V. Ziubrytska

MANAGEMENT OF THE COORDINATION PROCESS IN THE SOCIOTECHNICAL SYSTEM

The article considers the theoretical principles of management of the coordination process in the sociotechnical system. On the example of the selected IT company, the corporate knowledge base is researched, the problems of using the corporate knowledge base are analyzed and the directions of corporate knowledge base transformation are outlined with the help of software tools for its coordinated use and in-depth presentation of knowledge. As a result, the types of spaces in the corporate knowledge base are identified and recommendations for methods of managing them are offered.

Key words: knowledge management, corporate knowledge base, in-depth presentation of knowledge, information space, IT company, sociotechnical systems.

Introduction

A distinctive feature of the modern information society is that most employees are engaged in the production, storage, processing and sale of information, especially its highest form – knowledge. In the informatization of society, the main attention is paid to a set of measures aimed at ensuring the full use of reliable, comprehensive and timely knowledge in all human activities. The direction of knowledge management began to develop rapidly in the 90s of last century. This was due to a change in priorities in business and society, as well as the ongoing scientific and technological revolution, which is based on the use of the latest information technology in all areas of human activity. The most profitable and attractive area for investment is the development of high technology and service, which determine the rapid return on investment, reduce dependence on raw materials, provide an opportunity to capture new niches in the use of goods or services. Thus, the direction of research on the management of corporate knowledge bases in sociotechnical systems is relevant.

Important aspects considered from a practical point of view are methodological approaches and the practice of ensuring the management of the coordination process in the sociotechnical system. The research methods were general scientific methods, systematization, grouping, comparison, analogies, coefficients, structural, factorial, statistical and comparative analysis, empirical knowledge, expert assessments, design. Development tools: Confluence, XMind, Draw.io [1–3].

In recent years, there has been a growing interest in complex systems that involve both humans and computers that can be coordinated. Coordination can be seen as a process of *managing the relationships* between different activities. Coordination issues have been studied by many scholars Bond and Gasser, Huhns and Gasser, Malone and Crowston, A.V. Anisimov, F.I. Andon, A.Y. Doroshenko, S.D. Pogoriliy, G.E. Zeitlin, O.A Yatsenko and others. Thus, further progress may be possible by characterizing different types of dependencies and identifying coordination processes that can be used to manage them [4].

Coordination models and languages have attracted the attention of the scientific community in many fields, including the design of distributed and parallel computer systems. Parallel programs developed in the style of coordination have two components – a computational model and a coordination model, which are responsible for the algorithmic and behavioral aspects of computing, respectively. Coordination tools, as shown earlier, can serve not only as a software integrator, but also as a software accelerator in terms of improving the performance of parallel programs [5–7].

The purpose of this article is to outline the areas of management of the coordination process, to develop practical recommendations for the management and use of corporate

knowledge bases and to implement these recommendations in improving the effectiveness of sociotechnical systems.

1. Analysis of existing problems of using the corporate knowledge base

Any production organization is a complex sociotechnical system, which logically distinguishes material and human factors of production, and which is considered in terms of interaction of the two subsystems:

- technical and economic, which includes not only technical and technological factors, but also management knowledge, organizational structures, methods of production planning, job development, techniques and skills, the level of qualification and design of the labor process, which in turn improves economic efficiency of the organization.
- social, which includes values of the organization, attitude to the functions performed, forms of incentives for employees, management style, employee participation in decision-making, career opportunities, organizational culture, etc.

The term «sociotechnical system» was first coined in the 1960 s by Eric Trist and Fred Emery, consultants at the Tavistock Institute of Human Relations (London). The concept of sociotechnical systems, in contrast to the unilateral action of technology on man, is based on the idea of human-machine interaction. The design of technical and social conditions should be carried out in such a way that technological efficiency and humanitarian aspects do not contradict each other.

Researchers cite several characteristics of the sociotechnical system, which ensure success in the intensified competition and at the same time characterize the level of development of new managerial thinking. One of them is the organizational philosophy, which provides knowledge and understanding by the employees of the organization's goals and mission, readiness to take full responsibility for the final results of the activity. Second, the organizational structure of management provides employees with the right to participate in the management of the organization. An innovative approach to job development and the role of the executor in the decision-making process is also important. The characteristics also include innovative forms and methods of training and retraining, more flexible personnel policy aimed at guaranteeing employment. Training should be based on mastering a wide range of specializations, as well as on acquiring knowledge that enables employees to perform various functions, be competent in various aspects of work, master related professions and master the so-called professions of the future. Another important characteristic is the new criteria in assessing the economic efficiency of the use of modern technologies and capital investments in the development of production [8].

The widespread use of information technology has changed the way people work together. Coordination can take place in different types of systems. For example, specialized software was developed to support the collaboration of several authors on the same document; now it helps people display and manipulate information more effectively in meetings; helps people use the email process productively.

The continued development of new computer programs in this area is guided by a consistent theory of how people coordinate their activities without information technology, and how they can do so much more productively with computer support. It is important to understand the effects of information technology on the activities of human organizations and markets, the design of cooperative working tools and the design of distributed and parallel computer systems.

Well-established coordination is often invisible, but its absence immediately becomes visible. Examples of fuzzy coordination are when we book and pay for an apartment in a foreign country, but spend an hour waiting for the keys or cleaning finishing, or when our favorite word processor stops working on a new version of the operating system. We can become very aware of the consequences of poor coordination.

Coordination can be used in many types of systems: human, computational, biological, and so on. As for the question of dependency management in human activity, it is central to the theory of organizations, eco-

nomics, management science, sociology, social psychology, anthropology, linguistics, law and political science. In computer systems, the dependencies between different computational processes must be managed, and, as many researchers point out, certain types of interactions between computational processes resemble interactions between people [8].

One of the advantages of the definition used for coordination is that coordination offers a direction for addressing issues. If coordination is defined as dependency management, then further progress can be made by characterizing the different types of dependencies and identifying coordination processes that can be used to manage them.

For interdependence between components, coordination is important because it explicitly or implicitly affects the performance of certain activities (eg, design or redesign of components) [4].

Many coordination processes require decisions that affect the activities of the organization. For example, in sharing resources, the group must "decide" how to allocate resources; in task/subtask management, the group must "decide" how to segment the task. In all these cases, alternative ways of making group decisions generate coordination processes. For example, any group decision may be made by management (e.g., a "manager" who makes a decision), a vote, or a consensus (as a result of negotiations) [4].

An obvious way to generate new coordination processes are alternative forms of communication (synchronous versus asynchronous, paper versus electronic) for everywhere in the process where information needs to be transmitted. The coordination structure also emphasizes new aspects of the problems. For example, when we view communication as a way to manage the relationship between producer and consumer, we need to take care of how to make the information "usable." Coordination models can include parameters for things like incentives, productivity, and communication costs that vary widely across human, computing, and biological systems [4].

In order to analyze the terminology of coordination, it is important to clearly define the components of coordination. It is useful to define evaluation criteria. For example, we may define some general "goals" of the activity (for example, car production or report printing) and other measurements to assess how well these goals are being met (for example, minimizing time or cost). Some coordination processes may be faster or more accurate than others, for example, greater coordination is not always worth it. It is important to realize that there is no "right" way to determine the components of coordination in different situations. For example, we can sometimes analyze everything that happens in a production unit as one "activity", and at other times we can analyze each station on the assembly line as a separate "activity". As another example, we can give the example of muscle coordination, when different parts of the body of the same person are implicitly considered as separate "actors" performing separate "actions". When analyzing coordination in human organizations, it is often helpful to simply ask people what their goals are and evaluate their behavior in terms of these criteria. Another important example is the coordination of market operations. The goal of market participants may be to maximize their individual benefits, but the market as a coordination mechanism must be assessed in terms of how well it meets such general criteria as maximizing the needs of market participants [4].

Coordination is an activity that in itself involves certain costs. Although there are many other factors that can affect the way organizations and markets coordinate (e.g., global competition, national culture, government regulation, and interest rates). One important feature, of course, is its cost. And it seems plausible that information technology can significantly reduce the cost of certain types of coordination.

You can make a few simple predictions about the possible consequences of reducing coordination costs. It is useful to illustrate these effects by analogy with similar changes in transportation costs caused by the introduction of trains and cars:

1. The effect of reducing the "first order" of transport costs trains and cars were

achieved simply by a certain replacement of the old transportation technologies with new ones: people began to ride trains more and in carriages with horses less.

2. The effect of reducing the "second order" of transport costs was reflected in the increase in the number of used vehicles: people began to travel more, and it goes without saying that it could be done cheaper and more convenient on trains than on foot.

3. Finally, the effect of the "third order" allowed to create more structures of intensive transportation: people eventually began to live in a remote suburb and use shopping malls outside the city. These two examples of new structures undoubtedly depend on the wide availability of cheap and convenient transportation.

You can also expect several effects from the use of new information technologies to reduce coordination costs:

1. The "first order" effect of reducing the cost of coordination with information technology may be a replacement for information technology of some types of coordination of people. For example, many banks and insurance companies have replaced large numbers of human officials in their offices with automated systems. It has also long been predicted that computers will lead to the demise of middle management, as communication tasks performed by middle managers can be performed more cheaply, namely with the help of computers. This prediction has not been realized for decades, but many people believed that it would finally begin to happen en masse in the mid-1980 s and 1990 s.

2. The effect of reducing the "second order" You can increase the cost of coordination of the total amount on second-hand coordination. In some cases, this may interfere with the first-order effect. For example, in one case studied by the authors of the study, a computerized conference system was used to eliminate middle managers [4]. However, a few years later, almost as many new positions were created for corporate staff, many of whom were helped by new computer systems. This example demonstrates that the resource management task can now be applied to more complex analysis that was not possible before.

3. The "third order" effect of reducing coordination costs may encourage the transition to more "intensive coordination" of structures. In other words, coordination structures that were previously too "expensive" will now become more effective. Technologies such as e-mail and computer conferencing can help reduce the cost of these types of communications and advanced means of information exchange [4].

Reducing the cost of coordination through information technology can lead to a general shift to smaller firms and a proportional increase in the use of markets, from internal decision-making within firms to the coordination of economic activity in general. It is clear that the coordination of market operations is much more expensive than internal coordination.

IT can lead to both centralization and decentralization, depending on how they are used. This conclusion can be made by clearly defining the main factors. When IT implementation reduces information costs for decision making, it leads to greater centralization. For example, Otis Elevator used IT to centralize reporting and manage customer service functions, instead of distributing these functions to numerous remote field offices. On the other hand, when IT primarily reduces agency costs, it leads to greater decentralization. In this case, agency costs are the costs of employees who do not act in the interests of the firm. For example, when one insurance company developed a system to more effectively monitor the overall performance of its vendors, they were able to decentralize many decisions to vendors, whereas previously they were made centrally [4]. Coordination theories are also useful in the following areas:

- in the empirical study of human coordination or other biological systems (eg, field, laboratory or econometric studies);
- in the development of new technologies to support human coordination;
- when designing and experimenting with new methods of coordination of distributed and parallel computer systems;

– in formal modeling of coordination processes (for example, mathematical or computer modeling).

Thus, we can conclude that the ideas of coordination are useful, because they offer new systems, by classifying systems and analyzing how systems use them; that scientists have been paying attention to coordination since the middle of the twentieth century, and that new information technologies have an impact on both markets and organizations, in particular: the size of organizations and the degree of centralization of decision-making in them. General coordination mechanisms can manage many dependencies: both market operations, centralized and decentralized management decisions, and internal decision-making processes in an individual organization. Coordination helps address a variety of immediate practical needs, including:

– designing computer and communication tools that allow people to work together more effectively;

– using the power of several computer processors working simultaneously on common problems;

– creating more flexible and efficient ways of organizing collective human activity.

The knowledge management system in the set of applied tools and the mechanism of information support on the basis of modern information technologies is called to provide innovative development of the company, so necessary in the conditions of development of a society.

IT companies, more than others, should focus on the use of the latest technologies for more efficient organization of the external and internal environment: electronic document management, gamification, socialized corporate portal, BigData, BI-analytics, OLAP, automated controlling and more. All this is absolutely close to the understanding by the information technology professionals.

Therefore, IT companies must act as representatives of the innovation sector of the economy, creating unique mechanisms of management systems to stimulate their innovative development, including comprehensive controlling systems to assess the effectiveness of implemented tools based on selected indicators.

Effective solutions help create new products, improve old ones and increase customer satisfaction. When the support service answers questions faster and more accurately, solves problems of users, dissatisfied customers become less. This reduces the outflow of users, which increases financial performance.

Knowledge management is the process of processing, managing and using the knowledge and experience of employees (internal experts) to effectively solve problems.

Coding Sans conducted research in European IT companies (see Fig. 1). Employees were asked one question: "What is the main challenge in the field of software development do you see for yourself?" A fifth of respondents from various positions indicated that this is an exchange of knowledge. This is the second result after «capacity» – the ability to solve more problems in less time.

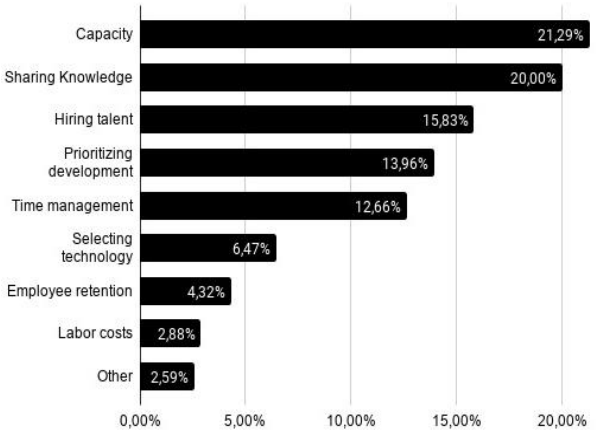

Fig. 1. Results of respondents' answers to the main challenge in the field of software development

But it is much more interesting to compare the voices between those who write the code and their managers (Fig. 2).

The column with information about managers is marked in white, and the column with information about developers is marked in dark color. The difference between the values is almost a third in relative terms. It turns out that the developer finds channel for knowledge management more important.

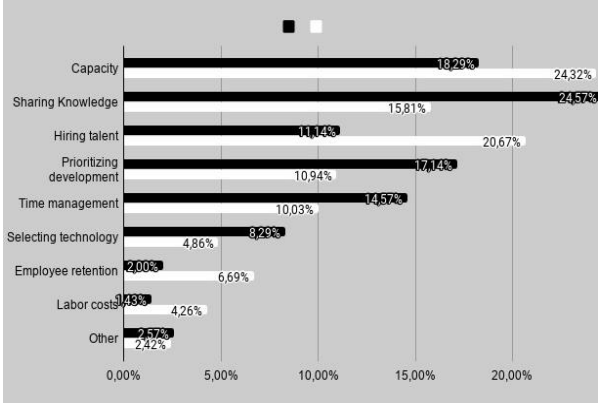

Fig. 2. Respondents' responses to the main challenge in software development compared to code writers and managers [9]

It is more important for managers to keep up with everything. Typically, they delegate tasks to subordinates: they turn to HR, hire more developers, and expect for tasks to be closed on time.

Even if the recruiter collects all the cream of the market, the problem will not go anywhere. Developers don't have a way to get information operatively in order to close tasks quickly, so the problem will arise again, but on a larger scale. This is an endless cycle from which the manager himself will not come out.

Software companies face the challenge of developing, selling, supplying and maintaining increasingly sophisticated solutions and products. This can be done most effectively only with a knowledge management system.

2. General characteristics of the company

An international technology product company with a full cycle of development in the field of Entertainment was selected for the study. It is part of a global holding company with a total staff of almost 2,000 people.

The company is an ecosystem that combines IT products, marketing and other areas. Works on the development of product innovations, the creation of competitive advantages of products and the company as a whole, accumulates expertise and speed of implementation, unites in research teams of professionals and implements bold and risky ideas.

3. Corporate knowledge base management tools

Effective management of the corporate knowledge base requires the use of modern tools. The choice of tools depends on the tasks, problems and opportunities in a particular organization. For example, for one company it is necessary to create an effective and stable communication space for interaction between departments, and for another it is necessary to start processes of continuous selection and evaluation of ideas and proposals of employees, for the third it is necessary to focus on creating special relationships with its customers. Let's consider the features of using the tools Confluence, XMind, drow.io for the collection, analysis, exchange and transfer of corporate knowledge.

Confluence – Web-based corporate wiki, which is used mainly within the organization, forming a single whole organization to achieve its goals, and is a space for teamwork, which accumulates information and opportunities for collaboration. It was developed by Atlassian Software Systems and more than 50 million companies worldwide use the company's products (the most popular are JIRA and Confluence). Confluence simplifies collaboration within the team and allows you to organize effective information management. Integration with Microsoft Office facilitates its use in a familiar environment: create content and collaborate on documents, spreadsheets and presentations on Confluence pages, edit pages directly from Microsoft Office.

Confluence has a sophisticated information retrieval system with the ability to quickly search for keywords. The information in Confluence is placed by sections or spaces that contain pages. Spaces allow you to share information between projects or teams, can have a unique data structure and appearance depending on the goals of the project. You can create a space using appearance templates for easier data organization. The basic editing tools available in the panel at the top of the window are used to edit and add information to pages. Macros are used to add files, charts, content, various reports, etc. [1].

XMind is software for creating so-called mind maps for brainstorming, which help to organize information in a visual associative form. XMind is a tool that helps capture thoughts and chart them. In other words, with this program you can detail your task and work on it more purposefully. Each element of the map can be an idea connected to other ideas through hierarchical connections. The program allows the user to capture their thoughts, build them into different charts, use these charts with other users. XMind supports intelligence maps (connection diagrams), Ishikawa diagrams (also known as causal or fishbone diagrams), tree diagrams, and tables. XMind is convenient to use for corporate knowledge management, during meetings, in task management, in time management [2].

Draw.io – a service designed for the creating of classic diagrams, relationship diagrams of logical objects, network diagrams, organizational charts of diagrams, graphs, flowcharts, electronic circuits, UML-models, business process models, interactive design prototypes and layouts, inserts in image diagrams, etc. [3].

The described tools are modern means of conceptual design of information systems that can significantly improve the productivity and coordination of joint work.

4. Practical recommendations

Let's define the transformation of corporate knowledge databases. To put documentation in defined order, firstly, it's recommended to logically separate it into product (which relates to description of the product) and operating (which relates to internal processes of company). They have different business requirements, working regulations, accounting, etc. This logical separation is offered to be done with the help of Confluence labels or another pages attributes. Also, the structure of the corporate knowledge database is developed considering the future necessity to take out part of information into external databases with minimal efforts, for example, for customers.

To such kind of information belongs general description of product/module, manuals, user guidance's, description of external API's, articles about malfunction eliminations.

Some part of information in the corporate knowledge database will be presented just for internal use, information about system's architecture, information about physical infrastructure (servers, networks, etc), technical support, administrators guidances, different operating documentation of the team (reports, statuses of projects and tasks, etc).

Recommended steps for realization during creating of new united corporate knowledge database:

1. Poll final users and formulation of requirements for internal Wikipedia.
2. Analysis current database to highlight popular and not relevant blocks.
3. Creating a plan and vision of corporate knowledge database development.
4. Creating a clear corporate knowledge database structure on Confluence and descriptions in schemas for every space.
5. Transformation of the database main page.
6. Creating a glossary with all local terminology for team synchronization.
7. Implementation of a new safety principle, which will be defined in the next clause.
8. Making the transition when working with Confluence from «everything is secret» to «everything is open» on Read (if there is any secret information – hide it on the closed page).
9. Database modification in the way that any employee of any department can find the necessary information, without previously knowing where to look for it.

After modification of all spaces they can be classified due to three types.

The first type includes space, where documentation of one team of developers is kept (see Fig. 3).

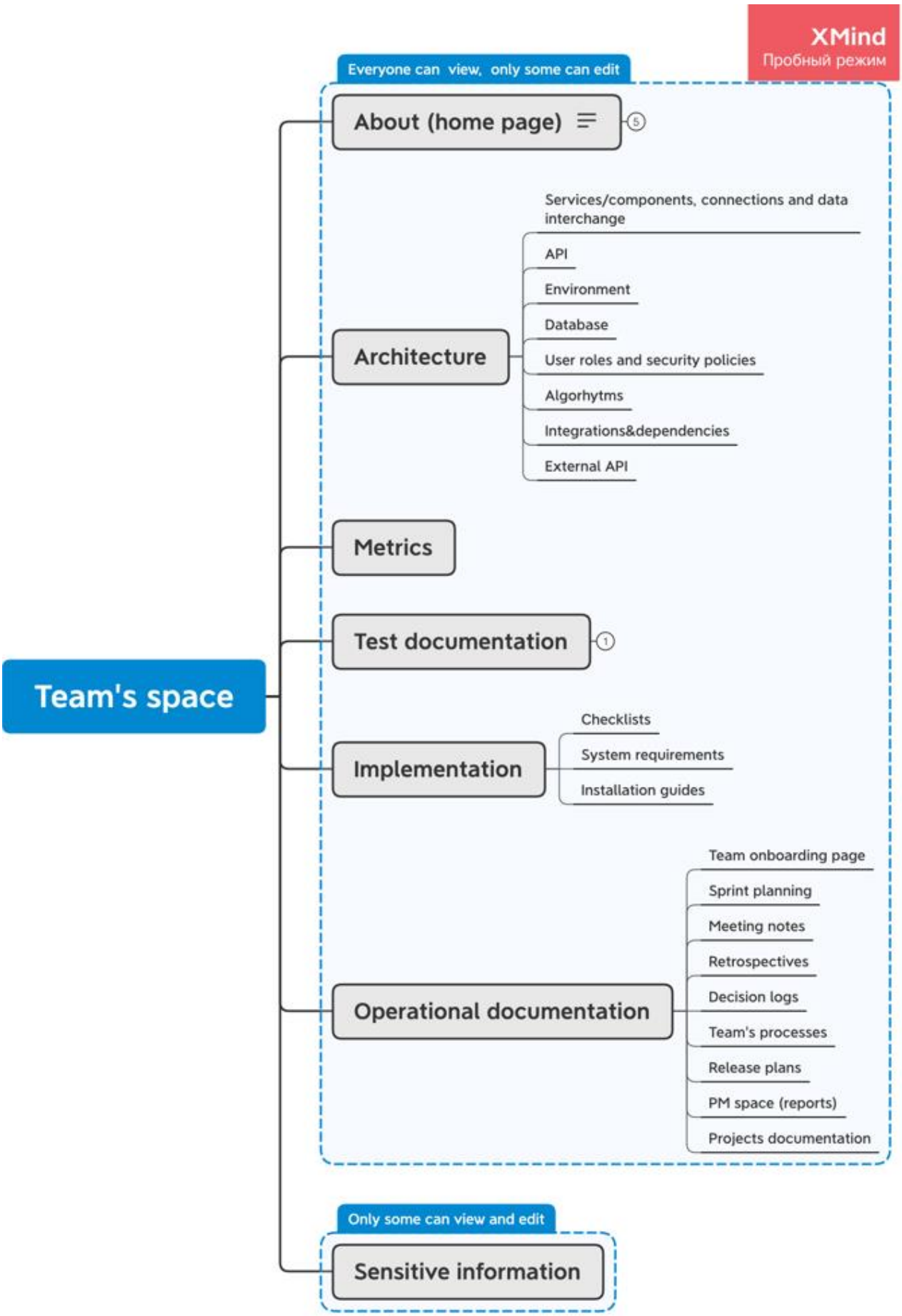

Fig. 3. Space of the first type

It is necessary to adhere to the principle of division documentation into product (all, that relates directly to the product: architecture, interaction with others, environment, API, etc) and operating which concerns directly the working process within the team).

The second type includes space, where there is processed documentation of a few teams or only one team, but including a few products. For successful use of created schemes it is recommended to use additional navigation tools: pages will be assigned team labels, then a list of all labeled pages will be automatically compiled by the content by label macro into content for one team.

Version 1: At the first level we create separate sections for each team, within these sections we repeat the distribution, presented in drawing. The documents of each team are collected together, which is convenient for the team. For a third-party user looking for information, extra levels of page hierarchy are added, and it may not be obvious which section contains the document you want.

It may turn out that the documentation for the same product is divided into different sections (see Fig. 4).

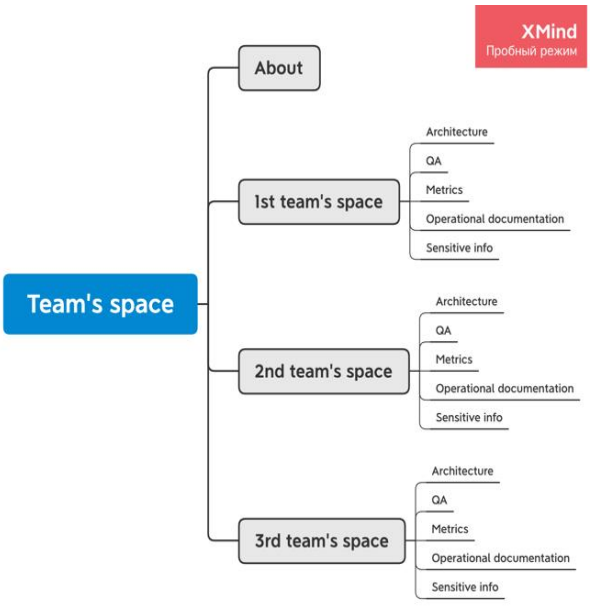

Fig. 4. Space of the second type of ver.1

Version 2: At first level we save distributions, for example, Architecture, Metrics, QA, Operating Documentation, at the next levels we divide into specific teams. Documents for the product are collected, which makes finding information more easily, positively affects its integrity and fullness. The team may feel uncomfortable when its documents are classified for different sections. To reduce this uncertainty it is suggested to use an additional navigation tool: pages will have teams labels and then a list of all labeled pages should be automatically collected by macro content by label into the content for one team (see Fig. 5).

The third type includes a space in which there is no documentation for specific products, and only reporting and other documentation is maintained (see Fig. 6).

The first page of all spaces should be as informative as possible for the user who entered this space for the first time. It should contain information about the team / products, links to the most important and frequently visited pages, it should simplify navigation in space.

Below is a list of sections that are recommended to be presented on a first page of each space:

1. For which team/product or for what purpose does this space exist. Optionally should be added, on what questions this team is responsible and in which way. Example message for one of the teams: “Hi, this is a space of a team of technical writers. We write documents for everyone, keep a glossary, maintain the relevance of documentation. If you need to add your term to the glossary, draw the architecture of the program or write a user manual, we will help. Write an application to the Technical Writers project at JIRA”.
2. The composition of the team or at least the leader and the one that is responsible for the product.
3. How to look for information in space. Links to the most commonly used product and operating documentation, necessarily including the link to the onboarding page.
4. Recently edited pages.
5. Popular documents.
6. Search.
7. Useful links.

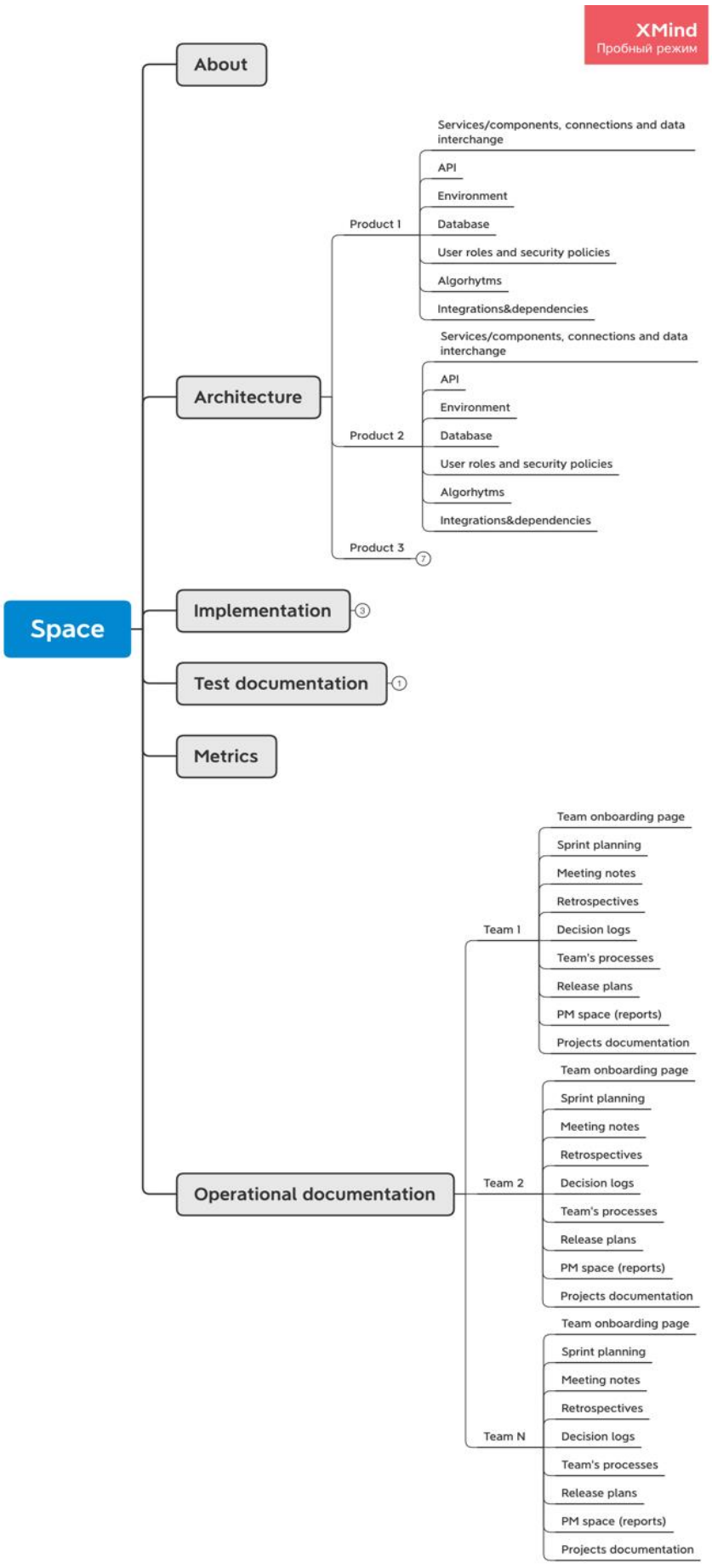

Fig. 5. Space of the second type of ver.2

Fig. 6. Space of the third type

8. You can also place a tree of space pages here, but this is optional because it is always presented in the menu on the left. By default, all Confluence spaces will be created open for viewing to all logged-in users. Anyway, each space will have a section for confidential information, access to which by default will be open only to the owner of the space and closed to all others.

Since the access permissions for Confluence pages are inherited by hierarchy, i.e. if the user is denied access to the parent page, it is automatically closed to all children, then access within the space will be easy to control: the page placed in the confidential information section or created in it will automatically disappear from public access, and to share the page, just move it from this section to any other.

Conclusions

Thus, the analysis of a large array of statistics, the study of existing software, foreign experience and the current state of knowledge management allows us to conclude that the ideas of coordination are useful because they offer new systems, by classifying systems and analyzing how present systems use them. Scientists have been paying attention to coordination since the middle of the twentieth century, and new information technologies have an impact on both markets and organizations, in particular: the size of organizations and the degree of centralization of decision-making in them. General coordination mechanisms can manage many dependencies: both market operations, centralized and decentralized management decisions, and internal decision-making processes in an individual organization. Coordination helps to address a variety of immediate practical needs. Research of the corporate knowledge base on the example of an IT company, analysis of the problems of its use and features of the use of development tools allowed to structure the spaces of the corporate knowledge base of the IT company; unify the pages of the corporate knowledge base of the IT company; create a convenient navigation of the corporate knowledge base of the IT company; introduce templates for key pages in space (about, processes, metrics, etc.) and new approaches to data storage in terms of company security. As a result of the study, the methods of renewal, updating, supplementing and avoiding duplication of information in the corporate knowledge base were improved.

References

1. https://www.atlassian.com/software/confluence
2. https://www.xmind.net/
3. https://drawio-app.com/
4. Malone T.W., Crowston K. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*. 1994. Vol. 26. N. l. P. 87–119.
5. Doroshenko A. Coordination Facilities to Enhance Concurrency of Race-Free Parallel Algorithms "High Performance Computing and Networking", Proc.Int. Conf., Amsterdam, Netherlads. *Lect. Notes in Computer Sci*. 1998. Vol. 1401. Springer Verlag. P. 950–952.
6. Doroshenko A., Thorelli L.-E., Vlassov V., Coordination Models and Facilities Could be

Parallel Software Accelerators. in *"High Performance Computing and Networking"*, Proc. 7-th Int. Conf. HPCN'99.-LNCS, Vol. 1593. P. 1219–1222.
7. Doroshenko A. Coordination Programming Abstractions for Efficient Parallel Programs. in UkrPROG'98: Proc. I Int. Scientific and Practical conf. in programming UkrPROG'98, Kyiv, Sept. 2-4.1998, Glushkov Institute of Cybernetics. P. 235–242.
8. Gary M. Olson, Thomas W. Malone, JOHN B. Smith (eds.). Coordination theory and collaboration technology. Lawrence Erlbaum Associates, Inc., 2001. 816 p.
9. https://codingsans.com/state-of-software-development-2019

Література

1. https://www.atlassian.com/software/confluence
2. https://www.xmind.net/
3. https://drawio-app.com/
4. Malone T.W., Crowston K. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*. 1994. vol. 26. No. l. P. 87–119.
5. Doroshenko A. Coordination Facilities to Enhance Concurrency of Race-Free Parallel Algorithms "High Performance Computing and Networking", Proc.Int. Conf., Amsterdam, Netherlads. *Lect. Notes in Computer Sci*. 1998. Vol. 1401. Springer Verlag. P. 950–952.
6. Doroshenko A., Thorelli L.-E., Vlassov V., Coordination Models and Facilities Could be Parallel Software Accelerators. in *"High Performance Computing and Networking"*, Proc. 7-th Int. Conf. HPCN'99.-LNCS, Vol. 1593. P. 1219–1222.
7. Doroshenko A. Coordination Programming Abstractions for Efficient Parallel Programs. in UkrPROG'98: Proc. I Int. Scientific and Practical conf. in programming UkrPROG'98, Kyiv, Sept. 2-4.1998, Glushkov Institute of Cybernetics. P. 235–242.
8. Gary M. Olson, Thomas W. Malone, JOHN B. Smith (eds.). Coordination theory and collaboration technology. Lawrence Erlbaum Associates, Inc., 2001. 816 p.
10. https://codingsans.com/state-of-software-development-2019

Received 23.10.2020

About authors:

Girchenko Liudmyla Andriivna,
Graduate student
Faculty of Computer Science and Cybernetics.
Number of scientific publications in Ukrainian publishing houses – 7.
Number of scientific publications in foreign publishing houses – 3.
http://orcid.org/0000-0002-5433-2399,

Doroshenko Anatoly Yuhymovych,
Doctor of physical and mathematical sciences,
Professor, Head of the Theory Department computer computing of the Institute software systems of the NAS of Ukraine,
Professor of Automation and management in Technical systems.
Number of scientific publications in Ukrainian publishing houses – more than 150.
Number of scientific publications in foreign publishing houses – more than 50.
h-index – 6.
http://orcid.org/0000-0002-8435-1451,

Ziubrytska Yuliia Viacheslavivna,
Graduate student
Faculty of Computer Science and Cybernetics.
http://orcid.org/0000-0001-5025-8614.

Affiliation:

Taras Shevchenko National University of Kyiv.
60 Volodymyrska Street, City of Kyiv, Ukraine, 01033

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
37, Prosp. Peremohy, Kyiv, Ukraine, 03056

E-mail: girchenkoliu@gmail.com,
doroshenkoanatoliy2@gmail.com,
yuliia.ziubrytska@gmail.com

UDC 004.415.2. (043.3) https://doi.org/10.15407/pp2020.04.110

N. Sydorov

TOWARD SOFTWARE ARTIFACTS ECOSYSTEM

In the process of developing and maintaining a software product, many things are created and used that are called software artefacts. Software artifacts are changed, reused, and change relationships in the development and maintenance processes of a software product. The complexity and variety of software artifact relationships require adequate means of description and management. They may be a software artifacts ecosystem. In the article, for the first time, a concept of a software artifact ecosystem is proposed. The concept describes a generic model of the software artifacts ecosystem, which is the Cornerstone ecosystem type and consists of three actors – the platform, the software, and the artifact. Based on the generic model, the SD model of the software artifacts ecosystem is described. The roles of actors in the ecosystem are indicated, the relationships between actors are described. The developer's activities will be more efficient, the software is understandable, and the development and maintenance is cheaper when the styles (standards) are used. As case study, based on the generic model of the software artifacts ecosystem, a declarative model of the programming style ecosystem has been developed. Three-level model of programming style artifact is proposed. The tools and processes for creating and using a programming style artifact are developed and described.

Key words: software engineering, software artifact, software ecosystem, programming, programming style, ontology.

1. Introduction

In the processes of developing and maintaining software products, many things are created and used, which are called artifacts. Artifacts can be different in form and presentation. They can be part of a software product or provide processes for its development and maintenance, be intermediate results of processes, or be part of other artifacts. Thus, there is a huge variety of software artifacts, including design plans, work products (specifications, architectural and detailed designs, code, and documentation), user stories, bug reports, tools, including for processing artifacts, but not limited to this. Various and often complex connections are established between artifacts. Artifacts change, reuse, and change links in the development and maintenance of a software product. Therefore, artifacts play an important role in the software life cycle whatever of its model and require the attention of all interested parties.

The software industry is constantly evolving and changing. Not only products and technologies are developing. Many software companies are experimenting with new business models, leading to fundamental changes in the structures of both the company and its client. Recently, many companies have been using the concept of “software ecosystem” to describe development, creating them around themselves or their products, taking into account customer connections. Ecosystems have shown themselves to be a promising management tool, an evolving software product.

The complexity and diversity of software artifact relationships require adequate description and management tools. This could be a software artifacts ecosystem. Such an ecosystem points to more detailed level than a software ecosystem, but at this level most of the approaches, methods and tools that are used in a software ecosystem can be used.

In the article, for the first time, a model of a software artifacts ecosystem is proposed. Its application is shown on the case of a programming style ecosystem. Within the conception framework, a generalized model of the software artifacts ecosystem is described. The ecosystem belongs to the Cornstoun ecosystem type and consists of three actors – platform, software and artifact. The roles of actors in the ecosystem are indicated, connections between actors are described. The types, rules, and attributes of actors, relationships, and actions can be refined for specific software artifacts ecosystem models. The same applies to analyzing ecosystem properties.

Based on the generic model of the software artifacts ecosystem, a declarative model of the programming style ecosystem has been developed. The programming style

is an artifact that plays an important role in the development and maintenance of software. The description of the processes of creation and the use of the programming style is made by using the ontology.

2. Related works

***Software artifacts*.** In the software life cycle to support the processes of creating and maintaining a software product, many different artifacts are created and used. Wide ranges of components are considered as artifacts, from documentation, work products and their parts, to auxiliary tools. Interacting, artifacts ensure the efficient execution of software life cycle processes.

In [1], artifacts are analyzed in the context of reuse as equipment in the sense of work [2]. At the same time, three goals (writing, processing and transferring artifacts) and three aspects of equipment (the in-order-to of equipment, readiness-to-hand, presence-and-hand) are considered. In addition, since artifacts are analyzed as reusable components that are embedded in the created software product, their characteristics are taken into account: holism, commonality, reusability and maturity. Considering an artifact as hardware – a thing built into the context of a software product, the interaction of the specified characteristics of software artifacts is investigated. In [3], artifacts are considered in the context of a software product line and are divided into three types – architecture, shared components, and components made from shared ones. For each type of artifacts, three levels of maturity are identified, depending on the degree of integration of the artifact of the corresponding type into the software product line. In [4], artifacts are considered as information parts that are created, modified and used in the RUP processes. Artifacts can be of different types and take different forms, from UML models to executable code, and can be used in the creation and maintenance of a software product. Artifacts are the input and output of actions in RUP processes. In [5], software documentation as an artifact is considered. Artifact as a means of representing information about software is defined. A maintenance model of documentation as a software artifact is introduced.

Artifact modeling. In the following works, attempts are made to build a model of the artifact.

The paper [6] presents a metamodel for software artifacts aiming at providing a new and structured way to represent artifact content, other than current sections hierarchy. This work defines an extension to UML/MOF and SPEM meta-models by means of layers. The paper [7] discusses the theoretical foundations for the representation and interpretation of software artifacts. Based on different levels of perception of artifacts by a person – the user of artifacts introduces three levels of representation of artifacts – physical (physical representation), structural (syntactic structure) and semantic (semantic content). In addition, two steps for processing artifacts - parsing the physical representation, and analyzing the syntactic structure – the result of the first step (interpretation) are introduced. A meta model of artifacts is built on the basis of presentation levels and processing steps. The work [8] considers the architecture of tools that provide the creation and maintenance of metadata about software artifacts, which form an environment consisting of resources – development artifacts. Tools to manage the artifact environment are used.

***Artifacts in software development*.** The experience of using artifacts in life cycle processes in several works is considered.

In [9], artifact-oriented development of embedded systems is considered. A conceptual model of artifact-oriented development is proposed, examples of its use are given. In [10], artifact-oriented model-driven development is considered. Details a better understanding on how explicating artifacts and their relations facilitates traceability of artifacts, change impact analysis, and interoperability of software tools are considered. The paper [11] concentrates on the paradigm artefact-orientation in requirements engineering and presents a meta model. This meta model is inferred from two concrete domain specific requirements engineering models: one for the application domain of embedded systems and one for the application domain of business information systems. In [12] shown, that ccollaborative development of software products across organizational boundaries in software

ecosystems adds new challenges to existing software engineering processes. A new approach offered for handling the diverse software artefacts in ecosystems by adapting features from social network sites. In paper [13], an industrial survey to create an Activity-Based Artifact Quality Model to define what this means from a stakeholder's viewpoint is proposed. Specifically was conducted. Quality factors of test artifacts that have a positive or negative impact on the activities of Agile testers are explored. Quality model contains 16 quality factors for six test artifacts that are reportedly relevant to at least five stakeholders in the process. In paper [14] reference model and a metamodel for traceability are proposed. The reference model, defined by the conceptual basis, may be used in the creation of traceability approaches. The reference model was used to develop a metamodel. In paper [15], a generic artifact model based on an empirical investigation is proposed. The results of a mapping study in combination with a systematic literature review to analyze the usage of artifacts in agile methods are presented.

Towards a software artifacts ecosystem. We are not aware of any work directly devoted to the consideration of problems associated with the study of software artifacts ecosystems. However, there are works, the results of which can be used to solve these problems. In [16], attention is rightly drawn to the fact that in software ecosystems, attention is now paid to the participants only at the top level - these are organizations and teams that create, implement and maintain software products. However, there is a lower level – artifacts, the role of which in the life cycle processes can hardly be overestimated. In [17], there are requirements for describing and analyzing software ecosystems, which in our paper to model software artifacts ecosystems are used.

3. The generic model of software artefacts ecosystem

This section discusses a generic model of the software artifact ecosystem. Several methods are now used to model software ecosystems [18]. The application of a particular method depends on the type of ecosystem and the goals of the modeling. To represent the software artifacts ecosystem, this work uses the i* modeling approach [19]. In contrast to the most commonly used SSN method, which focuses on describing the software ecosystem at the top level (product, developer, vendor, user), the i* approach provides a description of the ecosystem of a more detailed software presentation layer that corresponds to the level software artifacts. Fig. 1 presents generic model of the software artifacts ecosystem. When designing an ecosystem, two groups of requirements are used [17]: descriptive and analytical.

The first group includes the requirements for the definition of actors, connections between them and their actions. In addition, the requirements for determining the types, rules and attributes of actors, connections and actions are formulated, as well as the requirements for determining the specific characteristics of both the ecosystem as a whole and its elements, for example, productivity, efficiency, security.

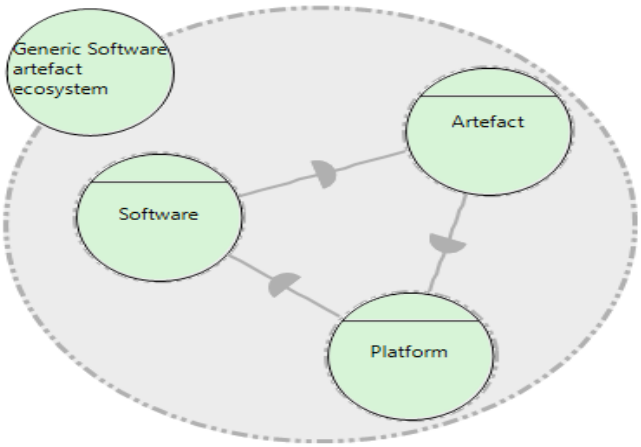

Fig. 1. The generic model of the software artifacts ecosystem

The second group includes requirements for defining characteristics that provide analysis of the ecosystem from incentives and motivation to sustainability and productivity.

Table the actors and roles in the software artifacts ecosystem are given (Tabl 1). The software artifacts ecosystem belongs to the Cornerstone type, since the basis of the ecosystem is a technological platform for the development and maintenance of software, the functionality of which is extended by using artifacts [20]. Thus, the actors of the ecosystem are a platform with a management role, software with a software product role, an artifact with a support service provider role. Common connections between actors can be indicated (Fig. 2). The platform, in the context of which such components as the life cycle model, organizational and technical support for development and maintenance are considered, defines and uses the artifact as an auxiliary means of implementing processes and filling the structure of a software product. The software depends on the platform, which is the main mean for the implementation of development and maintenance processes. The platform uses the artifact directly as a component in the software structure or indirectly as a means of improving the efficiency of the platform's processes.

The types, rules, and attributes of actors, relationships, and actions can be refined for specific ecosystem models of a software artifact. The same applies to meeting the requirements for the analysis of ecosystem properties [17].

Table 1. Actors and roles in the software artifacts ecosystem

Ecosystem type	**Actors**	**The role of the actor in the ecosystem**
Cornstoun ecosystem	Platform	Orchestration
	Software	Product
	Artefact	Support service provider

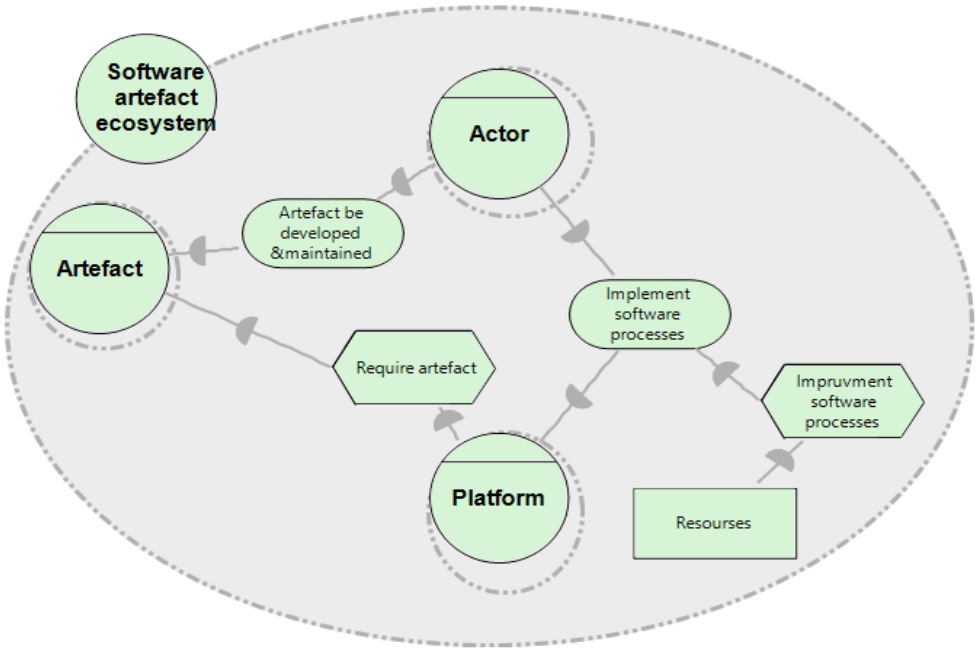

Fig. 2. The SD model of the software artefact ecosystem

4. Case study. The programming style ecosystem

Today, methods and tools that are based on reuse have become widespread for the development and maintenance of software products. The application of these methods and tools requires the developer to read, analyze and understand a significant number of representations of work products from different phases of the software life cycle. Reuse is now widespread from requirements specifications to source code and documentation. Therefore, one of the main requirements for software is understandability. The developer's activities will be more efficient, the software is understandable, and the development and maintenance is cheaper when the styles (standards) are use. They will ensure that the work products of different phases of the life cycle are understandable [21].

Fig. 3 shows the model of a programming style ecosystem, which is built based on a generic model of a software artifact ecosystem (Fig. 1).

The artifact in this model is the programming style, and the actor, the software, is represented by that part of it – the source code for which the programming style is applied. Artifact – the programming style is platform-specific, as the style rules depend on a number of platform conditions, such as the programming language, management goals, schedule, risks, and project budget. The programming style is used in the source code construction (the phase of software live cycle) and affects the efficiency of the construction and maintenance processes.

Based on the artefact model from work [7], described the programming style artefact by the three levels of perception and the two processing steps (Fig. 4).

Level 1 – Semantic content. The content represents the meaning of an artefact. The content is interpreted in the context of the individual knowledge of the stakeholder (programmer) or the interpreter of the machine (in this case – Protégé The content is based on the rules of the programming style, which are described by the ontology.

Level 2 – Syntactic Structure. The structure of an artifact represents the syntactic expression of its content. The structure of the artifact is described in Web Ontology Language (OWL).

Level 3 – Physical Representation. The artefact is represented in the file of OWL text format.

There are two the processing steps.

Processing Step 1 – Parsing. The outcome of the parsing process is the syntax structure of the artefact. This process is performed by the OWL parser implemented using the OWL API [22].

Processing Step 2 – Interpretation. Interpretation is the process of extracting the content (i.e. the meaning) from the structure. This process is performed by Protégé system.

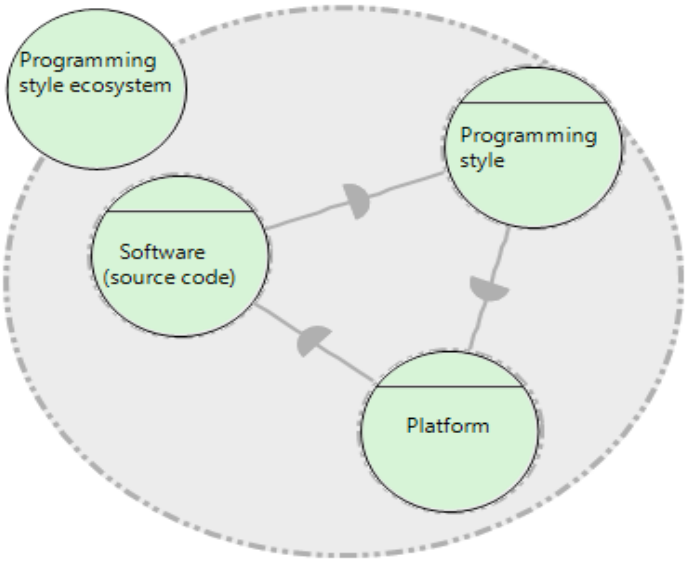

Fig. 3. The SD model of programming style ecosystem

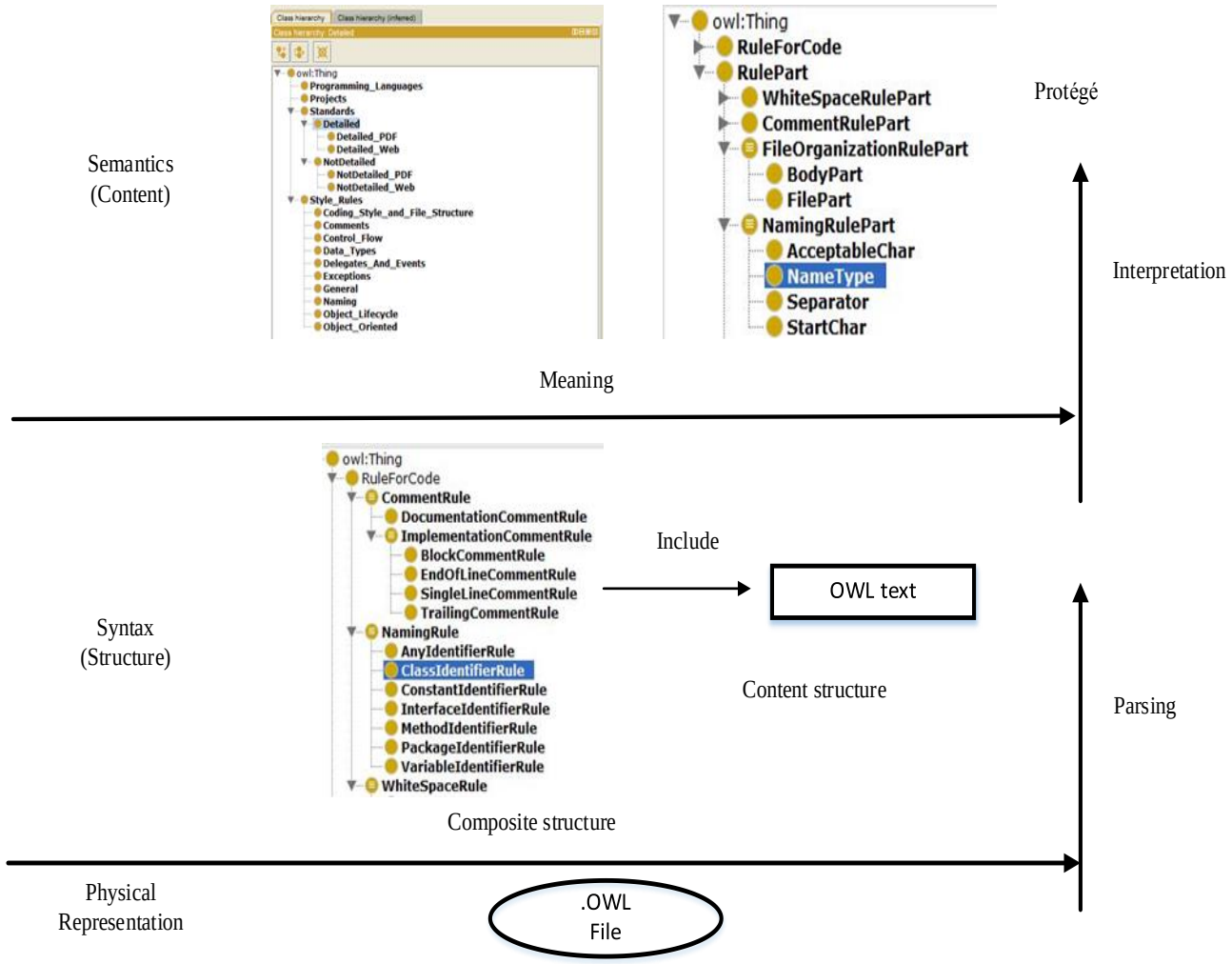

Fig. 4. The levels of perception of programming style artefact

The characteristics of the domain in which the style is applied are given in tab. 2. [21]. The activities of a programmer in a domain are shown in Fig. 5. The use of the programming style as an artifact involves the implementation of three processes [23]: the creation of an artifact, as a result of which the programming language style is built, the use of the style when programs are writing and the process changed of the artifact.

Table 2. Characteristics of the style of the program domain

Epoch	**Characteristic**						
	Property			**Factors**			**Means**
	Period	**Idea**	**Principle of importance**	**Historical**	**Social**	**Style**	**Elements**
Application of standards in coding	2000	Readability, quality, safety	Productivity, maintenance	Standardization	Programming team, requirements	Modular, mega programming	Composition, classification in programming languages

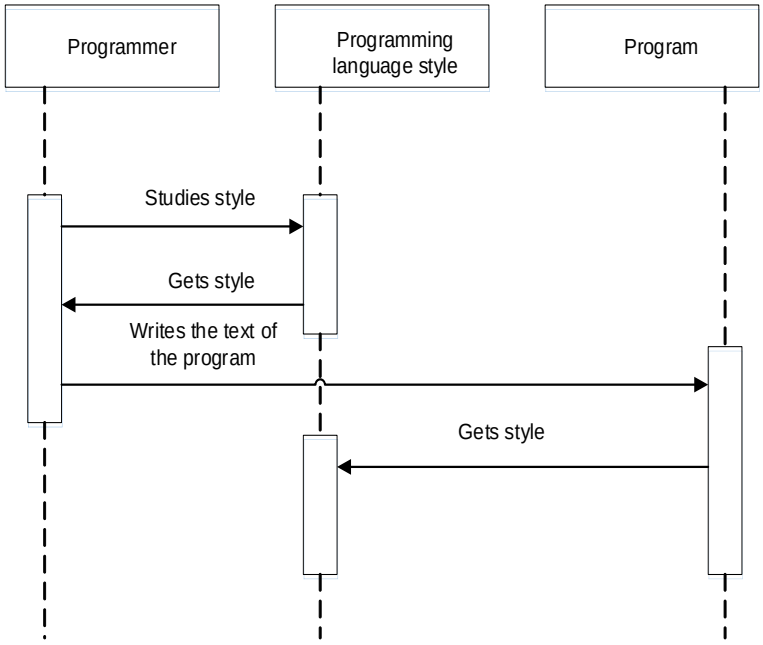

Fig. 5. Domain sequence diagram

In fig. 6. The ontology of creating a programming style is presented. The ontology describes in detail the participants and actions taking place in this regard in the programming style ecosystem. All ontology concepts are categorized as resources in i* terminology, with the exception of the <<event>> concept, which represents a goal. At the same time, the concepts Coding phase, Party, Programming language refer to the Platform actor, and the concepts Creating work product style, Style party create guide, Style and Programming language style to the Programming style actor.

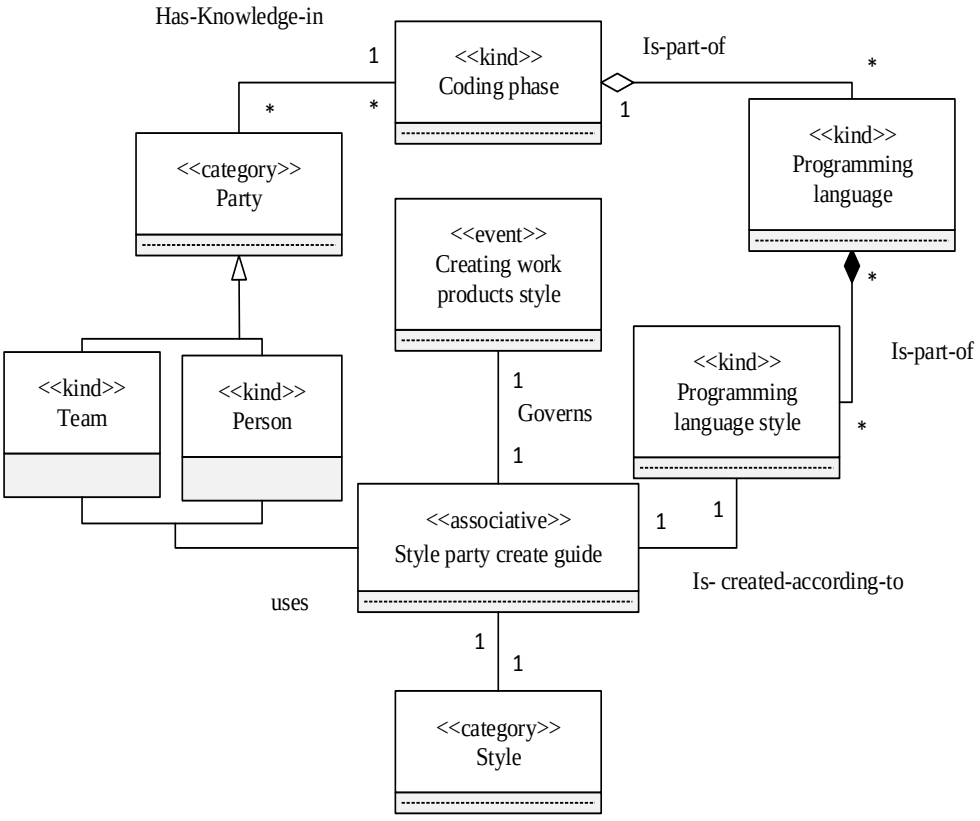

Fig. 6. Ontology of programming style creation

Fig. 7. The ontology of using the programming style is presented. An ontology describes the relevant actors and activities in a programming style ecosystem. The Party, Coding phase concepts belong to the Platform actor, the Program, Program style concepts to the Software actor, and the Using work product style, Style party using guide, Program language style concepts to the Programming style actor.

To implement the processes of creating and using a programming style, tools are created that can be considered, on the one hand, as resources of the Programming style artifact, and on the other hand, as artifacts as part of the Platform artifact. These include the programming style knowledge base and the Reasoner. Thus, the programmer, while coding the program, applies the ontology of the programming style, both for learning the style and for checking the observance of the style in the program. Therefore, two tools are needed - one to create an ontology and support the programmer in the coding process, and the second, to control the application of the programming style in the source code of the program (Fig. 8) [23].

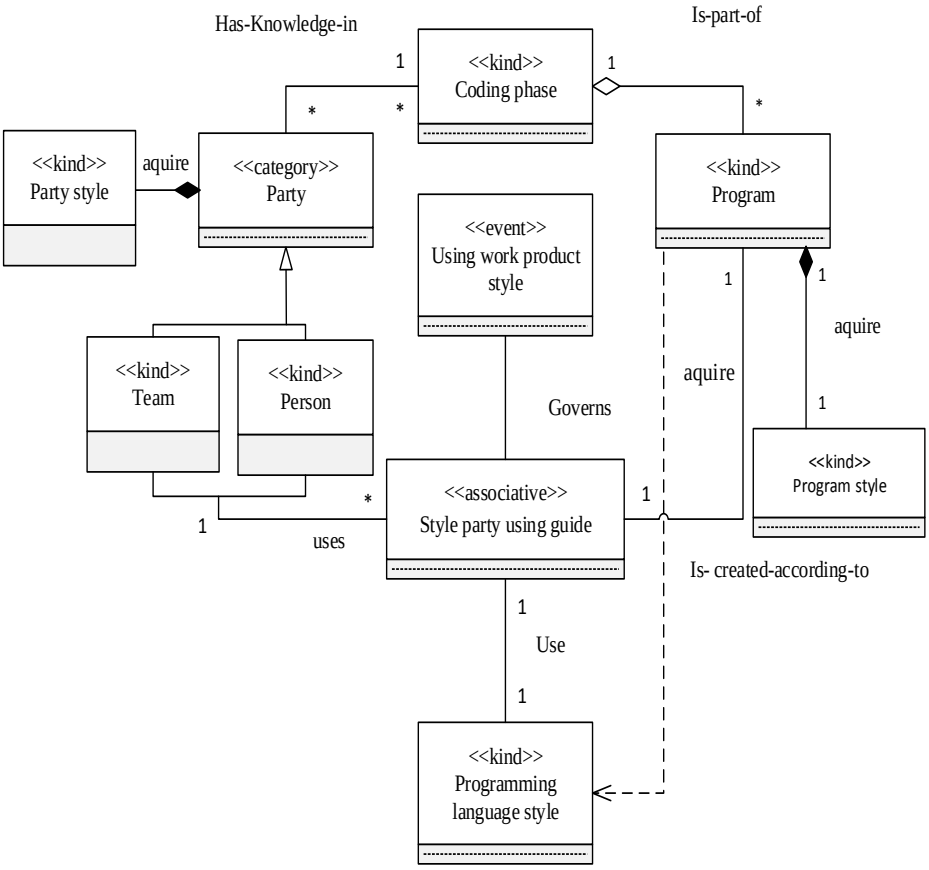

Fig. 7. Ontology of using programming style

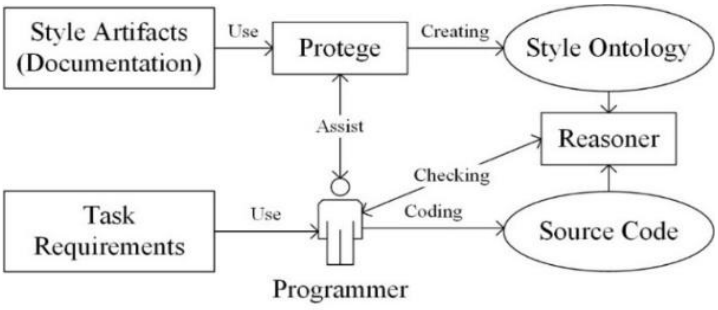

Fig. 8. Tools usage diagram

The style analyst, using the first tool - Protégé, setting up the ontology to the appropriate programming style, creating a TBox (Fig. 6). After setting up, the programmer is introduced to the programming style with the help of Protégé. The second tool is functionally similar to the reasoner, but adds a function for identifying style errors. In terms of descriptive logic, the reasoner verifies the consistency of the ontology (Fig. 9).

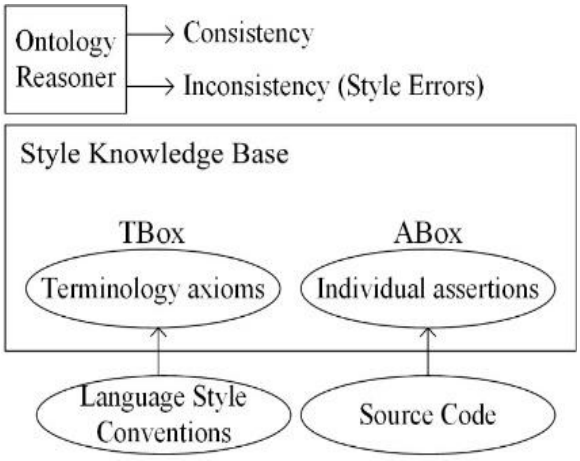

Fig. 9. Knowledge base of programming style

Protégé is used to create TBox. It is part of an ontology with terms describing a programming style. Assertions about the source code (ABox) written by the programmer are created by the corresponding part of the reasoner. It provides the appropriate service using the knowledge base (TBox and ABox). The service includes, firstly, the verification of the consistency of the ontology (a direct function of the reasoner), and secondly, the search for stylistic errors in the source code of the program.

5. Results Analysis and Discussion

The results are a development of the solutions obtained in the works of the author [21–24]. For the first time, the concept of the software artifact ecosystem is proposed. Within the framework of the concept, the generic model of the software artifact ecosystem is described. Model belongs to the Cornstoun ecosystem type and consists of three actors – platform, software, and artifact. The roles of actors in the ecosystem are indicated, connections between actors are described. Based on the generic model of the software artifact ecosystem, a declarative model of the programming style ecosystem has been developed. The programming style is an artifact that plays an important role in the development and maintenance of software. Using [7], a three-level artifact model of - programming style is proposed. The description of the processes of creating and using a programming style is made by applying the ontology. In continuation of research of the software artifact ecosystem, the description of the actors of the ecosystem will be expanded and the types, rules and attributes of actors, links and actions will be developed. In addition, the metric provision of the ecosystem in relation to determining the effectiveness, sustainability and reliability of the ecosystem of software artifacts will consider.

The work done in research “Research on software artifacts ecosystems”, № 0120U104329.

References

1. Nuwangi S.M., Darshana S. Software artefacts as equipment: a new conception to software development using reusable software artefacts. Thirty-Sixth International Conference on Information Systems. 2015. Texas, USA.
2. Heidegger M. (1927/1962) Being and Time, Translated by John Macquarrie & Edward Robinson. USA: Harper & Row.
3. Bosch J. Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization, Software Product Lines, Second International Conference, SPLC 2, San Diego, CA, USA, August 19–22, 2002,
4. Rational Unified Process: Best Practices for Software development Teams, Rational Software White Paper TP026B, Rev. 11/01. 1998. 18 p.
5. Glass R. Software maintenance documentation, SIGDOC '89, Pittsburg, Pennsylvania, USA, ACM Press. 1989. P. 18 – 23.
6. Silva M., Oliveira T., Bastos R., Software Artifact Metamodel, XXIII Brazilian Symposium on Software Engineering, 2009. P. 176 – 186.
7. Fernandez D M., Bohm W., Broy M. Artefacts in Software Engineering: A Fundamental Positioning, International *Journal on Software and Systems Modeling*. 2018. 26. 9 p.

8. Dewar R.G. Managing Software Engineering Artefact Metadata, Department of Computer Science, Heriot-Watt University, Edinburgh, UK. (2005)
9. Bohm W., Vogelsang A. An Artifact-oriented Framework for the Seamless Development of Embedded Systems, *Model-Based Engineering of Embedded Systems*. Springer Berlin Heidelberg. 2012. P. 225–234.
10. Butting, A., Greifenberg T, Rumpe B. Wortmann: A. On the Need for Artifact Models in Model-Driven Systems Engineering Projects. In: *Software Technologies: Applications and Foundations*, LNCS 10748. Springer. 2018. P. 146–153.
11. Fernández D.M., Penzenstadler B., Kuhrmann M., Broy M., A Meta Model for Artefact-Orientation:Fundamentals and Lessons Learned in Requirements Engineering, Lecture Notes in Computer Science. October 2010.
12. Seichter D., Dhungana D., Pleuss A., Hauptmann B. Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens. ECSA 2010. August 23–26, 2010. Copenhagen. Denmark. P. 119–126.
13. Fischbach J., Mendez D. What Makes Agile Test Artifacts Useful? An Activity-Based Quality Model from a Practitioners' Perspective, ESEM '20, October 8–9, 2020, Bari, Italy.
14. Azevedo B., Jino M., Modeling Traceability in Software Development: A Metamodel and a Reference Model for Traceability, ENASE, School of Electrical and Computer Engineering. University of Campinas, Brazil, 8 p.
15. Kuhrmann M., Fernández D., Towards Artifact Models as Process Interfaces in Distributed Software Projects, IEEE workshop proceedings, 10 p.
16. Seichter D., Dhungana D., Pleuss A., Hauptmann B. Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens, ECSA 2010 August 23–26, 2010. Copenhagen. Denmark. P. 119–126.
17. Sadi M., Yu E. Designing Software Ecosystems: How Can Modeling Techniques Help? Springer-Verlag, Berlin Heidelberg. 2015. 15 p.
18. Sydorov N. Software Ecology. *Software Engineering*. 2010. P. 53–61.
19. Yu E. Modelling Strategic Relationships for Business Process Reengineering. Ph.D., thesis. Dept. of Computer Science, University of Toronto. 1995.
20. Knodel J., Manikas K. Towards a typification of software ecosystems. In Fernandes et al. Software Business – 6th International Conference. ICSOB 2015. Braga, Portugal. June 10–12, 2015. Proceedings 2015. vol. 210 of Lecture Notes in Business Information Processing. Springer. P. 60–65.
21. Sydorov N.A. Software Stylistics. *Problems of Programming*. 2005. 2,3. P. 245–254.
22. Sidorov N., Sidorova N., Pirog A. Ontology-driven tool for utilizing programming styles. *Вісник НАУ*. 2017. Том 71. № 2. С. 84–93.
23. Sydorov N., Sydorova N., Sydorov E., Cholyshkina O., Batsurovska I. Development of the approach to using a style in software engineering. *Eastern-European Journal of Enterprise Technologies*. 2019. 4/2 (100). P. 41–51.
24. Sydorov N.A., Sydorova N.N., Sydorov E.N. Description model of programming style ecosystem. Problems in programming, special issue. Proceeding of the UkrProg'2020. N 2–3. P. 74–81.

Література

1. Nuwangi S.M., Darshana S. Software artefacts as equipment: a new conception to software development using reusable software artefacts. Thirty-Sixth International Conference on Information Systems. 2015. Texas, USA.
2. Heidegger M. (1927/1962) Being and Time, Translated by John Macquarrie & Edward Robinson. USA: Harper & Row.
3. Bosch J. Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization, Software Product Lines, Second International Conference, SPLC 2, San Diego, CA, USA, August 19–22, 2002,
4. Rational Unified Process: Best Practices for Software development Teams, Rational Software White Paper TP026B, Rev. 11/01. 1998. 18 p.
5. Glass R. Software maintenance documentation, SIGDOC '89, Pittsburg, Pennsylvania, USA, ACM Press. 1989. P. 18 – 23.
6. Silva M., Oliveira T., Bastos R., Software Artifact Metamodel, XXIII Brazilian Symposium on Software Engineering, 2009. P.176–186
7. Fernandez D M., Bohm W., Broy M. Artefacts in Software Engineering: A Fundamental Positioning, International *Journal on Software and Systems Modeling*. 2018. 26. 9 p.
8. Dewar R.G. Managing Software Engineering Artefact Metadata, Department of Computer

Science, Heriot-Watt University, Edinburgh, UK. (2005)
9. Bohm W., Vogelsang A. An Artifact-oriented Framework for the Seamless Development of Embedded Systems, *Model-Based Engineering of Embedded Systems*. Springer Berlin Heidelberg. 2012. P. 225–234.
10. Butting, A., Greifenberg T, Rumpe B. Wortmann: A. On the Need for Artifact Models in Model-Driven Systems Engineering Projects. In: *Software Technologies: Applications and Foundations*, LNCS 10748. Springer. 2018. P.146–153.
11. Fernández D.M., Penzenstadler B., Kuhrmann M., Broy M., A Meta Model for Artefact-Orientation:Fundamentals and Lessons Learned in Requirements Engineering, Lecture Notes in Computer Science. October 2010.
12. Seichter D., Dhungana D., Pleuss A., Hauptmann B. Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens. ECSA 2010. August 23–26, 2010. Copenhagen. Denmark. P.119–126.
13. Fischbach J., Mendez D. What Makes Agile Test Artifacts Useful? An Activity-Based Quality Model from a Practitioners' Perspective, ESEM '20, October 8–9, 2020, Bari, Italy.
14. Azevedo B., Jino M., Modeling Traceability in Software Development: A Metamodel and a Reference Model for Traceability, ENASE, School of Electrical and Computer Engineering. University of Campinas, Brazil, 8 p.
15. Kuhrmann M., Fernández D., Towards Artifact Models as Process Interfaces in Distributed Software Projects, IEEE workshop proceedings, 10 p.
16. Seichter D., Dhungana D., Pleuss A., Hauptmann B. Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens, ECSA 2010 August 23–26, 2010. Copenhagen. Denmark. P. 119–126.
17. Sadi M., Yu E. Designing Software Ecosystems: How Can Modeling Techniques Help? Springer-Verlag, Berlin Heidelberg. 2015. 15 p.
18. Сидоров Н.А. Экология программного обеспечения. *Інженерія програмного забезпечення*. 2010. № 1. С. 53–61.
19. .Yu E. Modelling Strategic Relationships for Business Process Reengineering. Ph.D., thesis. Dept. of Computer Science, University of Toronto. 1995.
20. Knodel J., Manikas K. Towards a typification of software ecosystems. In Fernandes et al. Software Business – 6th International Conference. ICSOB 2015. Braga, Portugal. June 10–12, 2015. Proceedings 2015. vol. 210 of Lecture Notes in Business Information Processing. Springer. P. 60–65.
21. Сидоров Н.А. Стилистика программного обеспечения. *Проблеми програмування*. 2018. 2, 3. С. 245–254.
22. Sidorov N., Sidorova N., Pirog A. Ontology-driven tool for utilizing programming styles. *Вісник НАУ*. 2017. Том 71. № 2. С. 84–93.
23. Sydorov N., Sydorova N., Sydorov E., Cholyshkina O., Batsurovska I. Development of the approach to using a style in software engineering. *Eastern-European Journal of Enterprise Technologies*. 2019. 4/2 (100). P. 41–51.
24. Сидоров Н.А., Сидорова Н.Н., Сидоров Е.Н. (2020) Дескриптивная модель экосистемы стиля программирования, Проблеми програмування, Спеціальний випуск, Матеріали конференції, УкрПрог. 2020. № 2-3. С. 74–81.

Received 04.11.2020

About author:

Sydorov Nikolay,
Doctor of Technical Sciences, Professor,
Number of scientific publications in Ukrainian publishing houses – 105.
Number of scientific publications in foreign publishing houses – 35.
https://orcid.org/0000-0002-3794-780X.

Affiliation:

National Technical University of Ukraine Igor Sikorsky Kyiv Polytechnic Institute,
Department of Automated Information Processing and Control Systems,
03056, Kiev - 56, Prosp. Peremohy, 37.

E-mail: nyksydorov@gmail.com

УДК 612.51.001.57+519.6

Програмна технологія для підтримки процедур налаштування кількісних моделей гемодинаміки людини / Р.Д. Григорян, О.І. Юрчак, А.Г. Дегода, Т.В. Людовик. – С. 3 – 13.

Математичне моделювання та спеціалізовані програмні симулятори (СПС), які засновані на кількісних моделях (КМ), є сучасними інструментами дослідження, що розширюють можливості вивчення фізіології людини. Зокрема, багато разів автори пропонували моделі з зосередженими параметрами (МЗП) серцево-судинної системи (ССС). Однак більшість моделей не включають складних механізмів, що забезпечують загальний контроль кровообігу людини. Щоб заповнити цю прогалину, ми запропонували три спеціальні моделі та моделювальну концепцію для їх функціональної інтеграції. Проблема полягає у тому, що інтегральна модель занадто складна, щоб її можна було налаштувати вручну. Для забезпечення ефективних процедур налаштування пропонується спеціальне програмне забезпечення, що містить автономні модулі для розв'язування рівнянь кожної моделі у відомих умовах вхідних навантажень. Загалом, побудована та запрограмована складна математична модель (СММ), що включає опис як фізіології, так і вхідних динамічних впливів. Програмне забезпечення, враховуючи три основні блоки моделей, забезпечує їх кількісні процедури налаштування. Перший блок описує саморегуляцію гемодинаміки людини в 23-компартментній МЗП ССС. Другий блок описує вісім фізіологічних механізмів, які незалежно забезпечують гострий і довгостроковий контроль ССС у горизонтальному, сидячому та вертикальному положенні тіла. Третій блок описує зовнішній та внутрішній динамічний вплив на ССС. Завдяки

UDC 612.51.001.57+519.6

A software technology providing tuning procedures of a quantitative model of human hemodynamics / R.D. Grygoryan, O.I. Yurchak, A.G. Degoda, T.V. Lyudovyk. – P. 3 – 13.

Mathematical modeling and specialized software simulators (SSS) based on quantitative models (QM) are modern research tools expanding research opportunities in human physiology. In particular, lumped-parametric models (LPM) of the cardiovascular system (CVS) have been proposed by many authors. However, most models do not include complex mechanisms providing overall control of human circulation. To fill this gap, we had proposed three special models and a concept of their functional integration. The problem is that the integral model is too complex to be manually tuned. To provide effective tuning procedures, special software containing autonomic modules for the solving equations of each model is proposed. In general, the complex mathematical model (CMM), including both the physiology and the external (inpdynamic ut) influences, is constructed and programmed. The software, taking into account three main blocks of models, provides their quantitative tuning procedures. The first block describes the self-regulation of human hemodynamics in a 23-compartmental lumped-parametric model (LPM) of the cardiovascular system (CVS). The second block describes eight physiological mechanisms independently providing CVS's acute and long-term control in body horizontal, sitting, and vertical positions. The third block describes external/internal dynamic influences on CVS. The model creator, due to SSS, can manually set values of both compartments' 92 parameters and sensitivity constants of every physiological mechanism. Special tuning tools allow the modeler to imitate a certain number of tests and to

СПС розробник моделі може вручну встановлювати значення 92 параметрів 23 відсіків ССС та константи чутливості кожного фізіологічного механізму. Спеціальні інструменти налаштування дозволяють розробнику моделі імітувати певну кількість тестів і будувати графіки гемодинамічних реакцій на обраний тест.

Ключові слова: метамова, список, множина, предикат, рекурсія, визначення.

УДК 004.4'24

До питання оптимізації хмарних обчислень з урахуванням їх вартості / А.Ю. Дорошенко, О.С. Новак. – С. 14 – 21.

В роботі запропоновано підхід до архітектурних налаштувань паралельних обчислень на хмарній платформі, що дозволяє в напівавтоматичному режимі здійснити оптимізацію паралельної програми з цільовою функцією мінімуму вартості обчислень. Для розв'язання оптимізаційної задачі використано лінійне програмування і доступний програмний солвер, який за допомогою методу гілок і границь в напівавтоматичному режимі підбирає значення архітектурних параметрів конфігурації програми, що істотно впливають на вартість обчислень. Таким чином узагальнено попередній метод автотюнінгу, що розроблявся авторами раніше, та поширено його на випадок комплексу сервісів, що виконуються на хмарній платформі. Проведено аналітичне випробування розробленої системи на моделі хмарного мультипроцесорного кластеру, що показує можливість суттєвого скорочення вартості хмарних обчислень внаслідок проведених оптимізацій.

Ключові слова: хмарні платформи, паралельні обчислення, методи оптимізації, лінійне програмування, вартість обчислень.

build graphs of hemodynamic responses to the chosen test.

Key words: cardiovascular system, modeling, software technology, simulation.

UDC 004.4'24

To the issue of optimizing cloud computing based on their cost / A. Doroshenko, O. Novak. – P. 14 – 21.

The paper offers an approach to the architectural settings of parallel computing on the cloud platform, which allows in semi-automatic mode to perform optimization of a parallel program with the goal function of minimum cost of computations. To solve the optimization problem, it is proposed to use linear programming and an available software solver, which with the help of the method of branches and boundaries in semi-automatic mode selects the value of the architecture parameters of the program configuration which significantly affect the cost of calculations. Therefore, the method of autotuning developed by the authors earlier is generalized and spread to the complex of services performed on the cloud platform. An analytical test was conducted on the model of cloud multiprocessor cluster, which presents the possibility of significantly reducing the cost of cloud computing due to the optimizations carried out.

Key words: cloud platforms, parallel calculations, optimization methods, linear programming, cost of computations.

УДК 004.94

Основні аспекти семантичного анотування великих даних / О.В. Захарова. – С. 22 – 33.

Семантичні анотації, в силу своєї структурованості, є невід'ємною складовою ефективного вирішення задач великих даних. Але, сама проблема визначення семантичних анотацій є досить не тривіальною. Ручне анотування є не прийнятним для великих даних з огляду на їх розмір та різнорідність, а також трудомісткість та вартосність самого процесу, задача повністю автоматичного анотування для великих даних поки що не має вирішення. Тобто вирішення задачі семантичного анотування вимагає сучасних змішаних підходів, які б на основі та із застосуванням існуючого теоретичного апарату, а саме методів та моделей машинного навчання, статистичного навчання, роботи з контентами різних форматів представлення, обробки текстів природньою мовою, тощо, забезпечували вирішення основних задач анотування: виявлення та витягнення сутностей та відношень з контенту будь-якого типу та визначення семантичних анотацій за основі існуючих джерел знань (словників, онтологій, тощо). Отримані анотації повинні бути точними та забезпечувати подальшу можливість вирішення прикладних задач з анотованими даними. Слід зазначити, що контенти великих даних є дуже різноманітними, як наслідок, дуже різняться їх властивості, що підлягають анотуванню. Це вимагає різних метаданих для опису даних та обумовлює наявність великої кількості різних стандартів метаданих для даних різних типів чи форматів представлення. Але, для ефективного вирішення задачі анотування треба мати узагальнену характеристику типів метаданих, в межах якої розглядати їх специфіку. Визначення загальної класифікації метаданих та спільних аспектів та підходів до семантичного анотування контенту великих даних за їх допомогою і є метою даної роботи.

Ключові слова: великі дані, анотування великих даних, класифікація метаданих, семантичні анотації, процес

UDC 004.94

Main Aspects of Big Data Semantic Annotation / O. Zakharova. – P. 22 – 33.

Semantic annotations, due to their structure, are an integral part of the effective solution of big data problems. However, the problem of defining semantic annotations is not trivial. Manual annotation is not acceptable for big data due to their size and heterogeneity, as well as the complexity and cost of the annotation process, the automatic annotation task for big data has not yet decision. So, resolving the problem of semantic annotation requires modern mixed approaches, which would be based on and using the existing theoretical apparatus, namely methods and models of machine learning, statistical learning, working with content of different types and formats, natural language processing, etc. It also should provide solutions for main annotation tasks: discovering and extracting entities and relationships from content of any type and defining semantic annotations based on existing sources of knowledge (dictionaries, ontologies, etc.). The obtained annotations must be accurate and provide a further opportunity to solve application problems with the annotated data. Note that the big data contents are very different, as a result, their properties that should be annotated are very different too. This requires different metadata to describe the data. It leads to large number of different metadata standards for data of different types or formats appears. However, to effectively solve the annotation problem, it is necessary to have a generalized description of the metadata types, and we have to consider metadata specificity within this description. The purpose of this work is to define the general classification of metadata and determinate common aspects and approaches to big data semantic annotation.

Key words: big data, big data annotation, metadata classification, annotation process, semantic metadata, manual annotation, automatic annotation, semi-automatic semantic annotation, machine learning, semantic annotation aspects, annotator, annotation models, ontology-based annotation tools, ontology-based

анотування, кероване машинне навчання, некероване машинне навчання, витягнення сутностей, витягнення відношень, домени анотування, онтологічна модель анотування, засоби онтологічного анотування, ручне анотування, автоматичне анотування, напівавтоматичне семантичне анотування, анотатор, аспекти семантичного анотування.

УДК 004.89

Класифікаційна система з підбору персоналу, базована на аналізаторі української мови / О.П. Жежерун, М.С. Рєпкін. – С. 34 – 40.

У статті розглядається класифікаційна системах, яка базується на аналізі природньої мови. В багатьох таких системах використовуються нейронні мережі, проте вони потребують даних для навчання, які не завжди наявні. Автори пропонують використання онтологій в подібних системах аналізу природньої мови. В якості прикладу представлено класифікаційну систему, яка допомагає сформувати список найкращих кандидатів під час підбору персоналу. Представлено огляд методів побудови онтологій та мовних аналізаторів, доречних для класифікаційних систем. Побудовано систему у вигляді бази знань. Здійснена підтримка української та англійської мов у класифікаційній системі. Описані можливості розширення системи.

Ключові слова: класифікаційна система, база знань, онтологія, Protégé, аналіз природньої мови, аналіз української мови.

УДК 004.62

Реалізація відображень між дескриптивною логікою та бінарною реляційною моделлю даних на рівні RDF / І.С. Чистякова. – С. 41 – 54.

Ця публікація присвячена проблемі інтеграції даних, а саме – створенню

annotation model, entity extraction, relation extraction, supervised machine learning, unsupervised machine learning, annotation domains.

UDC 004.89

System of classification for personnel selection based upon Ukrainian language analyzer / O.P. Zhezherun, M.S. Ryepkin. – P. 34 – 40.

The article describes a classification system with natural language processing. Many systems use neural networks, but it needs massive amounts of data for training, which is not always available. Authors propose to use ontologies in such systems. As example of such approach it is shown the classification system, which helps to form a list of the best candidates during the recruitment process. An overview of the methods for ontologies constructing and language analyzers appropriate for classification systems are presented. The system in the form of a knowledge base is constracted. Described system supports Ukrainian and English languages. The possible ways of system expansion is regarded.

Key words: classification system, knowledge base, ontology, Protégé, natural language processing, Ukrainian language processing.

UDC 004.62

Implementation of mappings between the description logic and the binary relational data model on the RDF level / I.S. Chystiakova. – P. 41 – 54.

This paper is dedicated to the data integration problem. In article the task of

механізму відображень між дескриптивною логікою та бінарною реляційною моделлю даних. Метод створення відображень був запропонований та описаний на теоретичному рівні. З метою його практичної перевірки, у поточній публікації пропонується метод реалізації відображень між дескриптивною логікою та бінарною реляційною моделлю даних за допомогою RDF. У роботі наведено огляд існуючих методів та засобів перетворення реляційних баз даних в онтологію та навпаки за допомогою RDF. Було сформульовано постановку задачі реалізації відображень та розглянуто відображення дескриптивної логіки ALC, її основних розширень, а також аксіоматики у RDF-трійки. У статті відзначено ряд складнощів при відображенні дескриптивної логіки на рівень RDF.

Ключові слова: бінарна реляційна модель даних, дескриптивна логіка, відображення, RDF, DL, RM^2, ALC, OWL.

practical implementation of mappings between description logic and a binary relational data model is discussed. This method was formulated earlier at a theoretical level. A practical technique to test mapping engines using RDF is provided in the current paper. To transform the constructs of the description logic ALC and its main extensions into RDF triplets the OWL 2-to-RDF mappings are used. To convert RDB to RDF graph, the R2R Mapping Language (R2R ML) was chosen. The mappings DL ALC and its main extensions to the RDF triplets are described in the publication. The mapping of the DL axioms into an RDF triplet also is considered in the publication. The main difficulties in describing DL-to-RDF transformations are given in the corresponding section. For each constructor of concepts and roles a corresponding expression in OWL 2 and its mapping into the RDF triplet. A schematic representation of the resulting RDF graph for each mapping is created. The paper also provides an overview of existing methods that relate to the use of RDF when mapping RDB to ontology and vice versa.

Key words: binary relational data model, description logic, mapping, RDF, DL, RM^2, ALC, OWL.

УДК 004.853, 004.55

UDC 004.853, 004.55

Застосування онтологічного аналізу для обробки метаданих при інтерпретації Big Data на семантичному рівні / Ю.В. Рогушина, А.Я. Гладун. – С. 55 – 70.

Application of ontological analysis for metadata processing in the interpretation of Big Data at the semantic level / J.V. Rogushina, A.Ya. Gladun. – P. 55 – 70.

В роботі розглядаються основні аспекти застосування сучасних технологій менеджменту знань для здобуття інформації з Big Data. Як показує аналіз сучасного стану досліджень у цій сфері, для того, щоб ефективно визначати, яку саме інформацію можна отримати з певних наборів Big Data, так і зробити це здобуття більш корисним (наприклад, недоцільно здобувати вже відомі або наочні правила), потрібно застосовувати фонові знання, які містяться в онтологіях предметних областей, що цікавлять користувачів. За

The paper considers the main aspects of modern technologies applied for knowledge analysis to obtain information from Big Data. The analysis of the current state of research in this area shows that background knowledge subject areas of user interest represented by domain ontologies can be used both in order to effectively analysis of information acquried from certain sets of Big Data, and to make this acquisition more useful. With the help of such ontologies, users can formally describe the scope of their information needs,

допомогою таких онтологій користувачі можуть формально описувати сферу своїх інформаційних потреб, задавати структуру потрібних інформаційних об'єктів та явно виділяти ті аспекти предметної області, які є важливими для поточної задачі. Це викликає необхідність у засобах пошуку або створення онтологій, які відповідають задачі користувача. Предметом обробки в процесі аналізу семантики Big Data є їх метадані, в яких відомості про зміст Big Data, як правило, представлені неструктурованим природномовним описом. Тому виникає потреба у стандартизації подання метаописів з використанням відповідних онтологій, які визначають структуру та семантику окремих елементів метаданих. Застосування методів Data Mining дозволяє здобувати необхідні знання з неструктурованих елементів таких метаданих. Новизна досліджень, які запропоновані у цій роботі, полягає у тому, що фонові знання, які використовуються для аналізу Big Data та їх метаописів, генеруються автоматизовано відповідно до поточної задачі користувача (на основі семантично розмічених Wiki-ресурсів та пов'язаних з ними онтологій), що забезпечує більш пертинентний підбір наборів Big Data, з яких здобуваються потрібні користувачеві знання. Такий підхід дозволяє зменшити обсяг вибірки, що обробляється, та зменшити час та складність її аналізу.

Ключові слова: Big Data, онтологія, метадані, семантична розмітка.

УДК 004.04

Автоматизація розв'язування задач з планіметрії, записаних природною українською мовою / О.П. Жежерун, О.Р. Смиш. – С. 71 – 80.

У роботі досліджено й описано створення системи для розв'язування задач з планіметрії за допомогою сучасних можливостей обробки природної української мови та розробленої сукупності алгоритмів опрацювання тексту задачі. Розробка базується на аналі-

define the structure of the required information objects and explicitly highlight critical for current task domain aspects. Subject of rocessing in the semantics analysis of Big Data is their metadata usually represented by unstructured natural language text. We need to standardize the representation of meta-descriptions wit use of appropriate ontologies that determine the structure and content of individual elements of metadata.

Key words: Big Data, ontology, metadata, semantic markup.

UDC 004.04

Automation of solving planimetry problems written in Ukrainian / O.P. Zhezherun, O.R. Smysh. – P. 71 – 80.

The article focuses on developing a software solution for solving planimetry problems that are written in Ukrainian. We discuss tendencies and available abilities in Ukrainian natural language processing. Presenting a comprehensive analysis of different types of describing a

зі текстів планіметричних задач та аналізі доступних засобів обробки живої української мови, що наразі наявні. Результатом роботи є кінцевий програмний продукт, написаний мовою Python, що дає змогу вирішувати прості завдання з планіметрії.

Ключові слова: планіметрія, обробка природньої мови, токенізація, лематизація, розмічування частин мови, сегментація тексту, видобування інформації, розмічений корпус.

problem, which shows regularities in the formulation and structure of the text representation of problems. Also, we demonstrate the similarities of writing a problem not only in Ukrainian but also in Belarusian, English, and Russian languages. The final result of the paper is a system that uses the morphosyntactic analyzer to process a problem's text and provide the answer to it. Ukrainian natural language processing is growing rapidly and showing impressive results. Huge possibilities appear as the Gold standard annotated corpus for Ukrainian language was recently developed. The created architecture is flexible, which indicates the possibility of adding both new geometry figures and their properties, as well as the additional logic to the program. The developed system with a little reformatting can be used with other natural languages, such as English, Belarusian or Russian, as the algorithm for text processing is universal due to the globally accepted representations for presenting such types of mathematical problems. Therefore, the further development of the system is possible.

Key words: planimetry, natural language processing, tokenization, lemmatization, Part-of-speech tagging, text segmentation, information extraction, annotated corpus.

УДК 004.85

UDC 004.85

Анализ методов машинного обучения в задачах классификации документов / А.П. Жиркова, О.П. Игнатенко. – С. 81 – 87.

Machine learning methods analysis in the document classification problem / A. Zhyrkova, O. Ignatenko. – P. 81 – 87.

Публикация рассматривает методы классификации документов по наличию в них печати. Для этого проанализировано уже существующие методы решения данной задачи, предложено модель сверточной нейронной сети для классификации документов, а также отображено зависимость корректности работы нейронной сети от количества входных данных, на которых обучается модель. В результате получено нейронную сеть, которая классифицирует документы по наличию печати с точностью немного больше 88 %.

Current situation with official documentary in the world, and especially in Ukraine, requires tools for electronical processing. One of the main tasks at this field is seal (or stamp) detection, which leads to documents classification based on mentioned criterion. Current article analyzes some of existed methods to resolve the problem, describes a new approach to classify documentary and reflects dependence of model accuracy to input data amount. As a result of this work is a convolutional neural network that classify 708 out of 804 images of official documents correctly. A corre-

Ключевые слова: машинное обучение, классификация, сверточные нейронные сети, последовательная модель, печать.

sponded percentage of model accuracy is 88.03, despite the fact of bias presence in input data.

Key words: machine learning, classification, convolutional neural networks, stamp, seal.

УДК 004.05

UDC 004.05

Алгебраїчне моделювання в системах міжнародної та місцевої обслуговуючої логістики / О. Летичевський, С. Горбатюк, В. Горбатюк. – С. 88 – 97.

Algebraic modeling in international and local service logistical systems / O. Letychevskyi, S. Horbatiuk, V. Horbatiuk. – P. 88 – 97.

Міжнародна та внутрішня обслуговуюча логістика розвиваються неймовірними темпами в сучасному житті, а прогнози розвитку для цієї галузі вкрай оптимістичні. З цих причин ми зіштовхнулись з задачею контролю, оптимізації та безпечної і надійної перевірки комплексних логістичних систем з великою кількістю внутрішніх агентів, які діють у мінливому та нестабільному середовищі. Ця робота має за мету показати як математичне моделювання, зокрема, алгебра поведінок, дає можливість передбачити поведінку та стабільність логістичних середовищ, перевірити їх властивості безпеки та надійність. Основними властивостями безпеки комплексних систем логістики різної складності та рівнів є стабільність функціонування, стійкість до зовнішніх загроз, виявлення та усунення вразливостей. При розробці програмних систем доцільно використовувати модельний підхід. Він передбачає створення моделей як інструментів на кожному етапі розробки програмного забезпечення для застосування методів верифікації, тестування та валідації. Алгебраїчні моделі логістичних систем можуть бути використані для аналізу поведінки всіх залучених агентів і доведення їх здатності виконувати свої цілі та здатності всієї системи постійно існувати та залишатися стабільною. Як приклади практичного використання розглянуто застосування на практиці алгебри поведінок на прикладі діючої за-

International and internal service logistics are advancing at a tremendous pace in our modern life, and future forecasts for this area are optimistic. For this reason, we face a task in the control, optimization and safety and security checking of complex logistical systems with many internal agents working in a changing environment. Our paper aims to show how mathematical modeling, especially behavior algebra, can provide an opportunity for predicting the behavior and stability of logistics environments and checking their safety and security properties. The main security properties of complex logistics systems of different capacities and levels are stability of operation, resistance to external threats and detection and elimination of vulnerabilities. During the development of software systems, it is advisable to use a model approach. It involves the creation of models as tools at every stage of software development for the application of verification, testing and validation methods. Algebraic models of logistics systems can be used to analyze the behavior of all agents involved and to prove their ability to fulfill their goals and the ability of the whole system to constantly exist and remain stable. As examples of practical use the application in practice of algebra of behaviors on an example of the operating closed logistic system of a farm is considered.

Key words: behavior algebra; insertion modeling; formal verification; supply chain; model-driven development; security properties..

критої логістичної системи фермерського господарства.

Ключові слова: алгебра поведінок, інсерційне моделювання, формальна верифікація, ланцюжок постачання, модельно-орієнтовані розробки, властивості безпеки.

УДК 004.5

Управління процесом координації у соціотехнічній системі / Л.А. Гірченко, А.Ю. Дорошенко, Ю.В. Зюбрицька. – С. 98 – 109.

У статті розглядаються теоретичні засади управління процесом координації у соціотехнічній системі; досліджено базу корпоративних знань ІТ-компанії; проаналізовано проблеми користування базою корпоративних знань компанії; накреслено напрями перетворення бази корпоративних знань; визначено типи просторів у базі корпоративних знань та методи керування ними. Розглянуто основні характеристики соціотехнічної системи, які забезпечують успіх у загостреній конкурентній боротьбі і одночасно характеризують рівень розвитку нового управлінського мислення. З проведеного дослідження стає зрозумілим наслідки впливу інформаційних технологій на діяльність людських організаціях та ринки, важливість координації у багатьох видах систем. У статті визначено основні компоненти координації, здійснено прогнози про можливі наслідки скорочення координаційних витрат. Значну увагу в статті приділяється особливостям побудови та налагодження процесів керування базами корпоративних знань соціотехнічних систем; методичним підходам і практиці забезпечення керування процесом координації у соціотехнічній системі. Прослідковується тенденція до прискорення появи різноманітних видів. Також визначено основні напрями перетворення бази корпоративних знань, окреслено перспективи подальшого розвитку керування процесом координації у соціотехнічній системі. В результаті визначено типи просторів у базі корпоративних знань

UDC 004.5

Management of the coordination process at the sociotechnic system / L.A. Girchenko, A.Yu. Doroshenko, Yu.V. Zyubrytska. – P. 98 – 109.

The article considers the theoretical principles of management of the coordination process in the socio-technical system; the base of corporate knowledge of the IT company is investigated; the problems of using the company's corporate knowledge base are analyzed; the directions of transformation of the corporate knowledge base are outlined; the types of spaces in the base of corporate knowledge and methods of their management are defined. The main characteristics of the socio-technical system are considered, which ensure success in the intensified competition and at the same time characterize the level of development of the new managerial thinking. The study clarifies the effects of information technology on the activities of human organizations and markets, the importance of coordination in many types of systems. The article identifies the main components of coordination, forecasts about the possible consequences of reducing coordination costs. Considerable attention in the article is paid to the peculiarities of construction and adjustment of management processes of corporate knowledge bases of socio-technical systems; methodological approaches and practice of ensuring the management of the coordination process in the socio-technical system. there is a tendency to accelerate the emergence of various species. The article identifies the main directions of transformation of the corporate knowledge base, outlines the prospects for further development of management of the coordination process in the socio-technical system. As a result,

та запропоновано рекомендації щодо методів керування ними.

Ключові слова: менеджмент знань, база корпоративних знань, поглиблене подання знань, інформаційний простір, IT-компанія, соціотехнічні системи.

the types of spaces in the corporate knowledge base are identified and recommendations for methods of managing them are offered.

Key words: knowledge management, corporate knowledge base, in-depth knowledge representation, information space, IT company, socio-technical systems.

УДК 004.415.2. (043.3)

UDC 004.415.2. (043.3)

Відносно екосистеми артефактів програмного забезпечення / М.О. Сидоров. – С. 110 – 120.

Toward softare artifacts ecosystem / N. Sydorov. – P. 110 – 120.

У процесі розробки та супроводження програмного продукту створюється і використовується багато речей, які називаються артефактами програмного забезпечення. Артефакти програмного забезпечення створюються, змінюються, повторно використовуються та змінюють взаємозв'язки у процесах розробки та супроводження програмного продукту. Складність та різноманітність взаємозв’язків артефактів програмного забезпечення вимагають адекватних засобів опису та керування. Вони можуть бути екосистемою артефактів програмного забезпечення. У статті вперше запропоновано концепцію екосистеми артефактів програмного забезпечення. Концепція описує загальну модель екосистеми артефактів програмного забезпечення, яка є типом екосистеми Cornerstone і складається з трьох суб'єктів – платформи, програмного забезпечення та артефакту. На основі загальної моделі описана SD модель екосистеми артефактів програмного забезпечення. Вказуються ролі суб'єктів у екосистемі, описуються взаємозв'язки між акторами. Діяльність розробника буде ефективнішою, програмне забезпечення зрозумілим, а розробка та обслуговування дешевше, коли використовуються стилі (стандарти). Як тематичне дослідження, на основі загальної моделі екосистеми артефактів програмного забезпечення була розроблена декларативна модель екосистеми стилю програмування. Запропоновано

In the process of developing and maintaining a software product, many things are created and used that are called software artefacts. Software artifacts are created, changed, reused, and change relationships in the development and maintenance processes of a software product. The complexity and variety of software artifact relationships require adequate means of description and management. They may be a software artifacts ecosystem. In the article, for the first time, a concept of a software artifact ecosystem is proposed. The concept describes a generic model of the software artifacts ecosystem, which is the Cornerstone ecosystem type and consists of three actors — the platform, the software, and the artifact. Based on the generic model, the SD model of the software artifacts ecosystem is described. The roles of actors in the ecosystem are indicated, the relationships between actors are described. The developer's activities will be more efficient, the software is understandable, and the development and maintenance is cheaper when the styles (standards) are used. As case study, based on the generic model of the software artifacts ecosystem, a declarative model of the programming style ecosystem has been developed. Three-level model of programming style artifact is proposed. The tools and processes for creating and using a programming style artifact are developed and described.

трирівнева модель артефакту стилю програмування. Розроблено та описано інструменти та процеси для створення та використання артефакту стилю програмування.

Ключові слова: інженерія програмного забезпечення, артефакт програмного забезпечення, екосистема програмного забезпечення, програмування, стиль програмування, онтологія.

Key words: software engineering, software artifact, software ecosystem, programming, programming style, ontology.

ДО УВАГИ АВТОРІВ!

У журналі "Проблеми програмування" публікуються наукові матеріали, які раніше не публікувалися в інших виданнях.

Мова статті: українська, англійська.* Обсяг статті - від 6 до16 сторінок формату А4.

Документ зберігається у форматі doc або docx. Ім'я подається транслітерацією, як прізвище автора (авторів), наприклад, “Petrenko.doc”.

Автори можуть користуватися електронною поштою і також телефаксом для ділової переписки та передачі до редакції тексту статті та правки при коректурі. E-mail редакції: alengoro@isofts.kiev.ua. FAX: +380 (44) 526 6263, Телефон: 526 5065.

1. Оформлення файлу з текстом статті.

При підготовці файлу використовуються: стиль нормальний (звичайний) або normal; шрифт Times New Roman, розмір шрифта 12 пт.; міжрядковий інтервал – 1,0; абзацний відступ -1,25 см; вирівнювання – по ширині. У тексті не допускається вирівнювання пропусками; розстановка переносів – автоматична. Формат паперу А4, розміри полів документа – 20 мм. Текст статті після анотації має бути оформлений **у 2 колонки,** ширина яких – 7,86 см, а пробіл між ними – 1,27 см.

2. Послідовність розміщення та оформлення матеріалу статті.

УДК: індекс за універсальною десятковою класифікацією.

Автори: ініціали та прізвища авторів, курсив (світлий).

Заголовок 1 **(назва статті):** не містить абревіатур та строго відповідає змісту статті. Шрифт 15 пт, напівжирний, регістр верхній.

Анотація **(мовою статті):** 50-100 слів, не містить абревіатур, зрозумілих із змісту статті. Шрифт 10 пт, звичайний.

Ключові слова **(мовою статті):** не більше10 слів, не містить абревіатур, зрозумілих із змісту статті, подаються в називному відмінку, розділені комами. Шрифт 10 пт, звичайний.

Заголовок 2 **(назва розділу):** шрифт 14 пт, напівжирний; абзац із центральним вирівнюванням, без переносів. Заголовки нижчого рівня (пункти і т.п.) у самостійний абзац не виділяються і проходять першим реченням текстового абзацу, шрифт 12 пт, напівжирний.

Основний текст статті має такі необхідні елементи:

постановка проблеми в загальному вигляді і її зв'язок з важливими науковими або практичними завданнями;

аналіз останніх досліджень і публікацій, у яких розпочато рішення даної проблеми і на які спирається автор, виділення невирішених раніше частин загальної проблеми, яким присвячується дана стаття;

формулювання цілей статті (постановка задачі);

виклад основного матеріалу дослідження з повним обґрунтуванням отриманих наукових результатів;

висновки з даного дослідження і перспективи подальших розробок у даному напрямку;

подяка (за наявності такої).

Формули створюються в редакторі Microsoft Equation 3.0 або MathType. Формули, на які є посилання в тексті, повинні мати наскрізну нумерацію. Номер формули друкується в круглих дужках біля краю правого поля. Розмір основного шрифту редактора формул – 12 пт. Розміри символів у формулах: звичайний – 12 пт, великий індекс – 9 пт, дрібний індекс – 7 пт, великий символ – 18 пт, дрібний символ – 11пт. Не допускається масштабування формульних об´єктів.

Рисунки мають бути створені вбудованим редактором Word Picture або експортовані з прикладних програм Windows у графічних форматах (bmp, pcx, gif, jpg або tif). Рисунки розташовуються по центру. Нумерація рисунків здійснюється відповідно до порядку

згадування у тексті. Нумеровані підписи розміщуються під рисунком з позначенням «Рис. », далі вказується номер рисунка і текст підпису.

Таблиці мають бути підготовлені стандартним вбудованим в Word інструментарієм "Таблиця". Таблиці нумеруються за порядком згадування. На номер таблиці повинно бути посилання в тексті. Номер таблиці вказується в окремому рядку з вирівнюванням по правій стороні (наприклад, «Таблиця 1»). Назви таблиць розміщуються над таблицею з вирівнюванням по центру. Мінімальний розмір шрифту в таблицях – 11 пт.

***Література*:** нумерований список джерел згідно ДСТУ 8302:2015 від 01.07.2016 р., шрифт 11 пт, відступ: спеціальний, навислий, 0,63 см.

Література англійською мовою (References): список використовуваних джерел згідно ***Harvard Style***. Джерела з заголовками на латиниці наводяться без перекладу. Для літератури джерел на мовах, що не використовують латинський алфавіт, необхідно забезпечити переведення назв джерел і вказати після них у дужках мову оригіналу. Прізвища та ініціали авторів слід транслітерувати за правилами, як для закордонного паспорта. Приклади оформлення бібліографічних посилань згідно з вимогами ***Harvard Style*** наведені в багатьох публікаціях, наприклад, за електронною адресою http://www.staffs.ac.uk/assets/harvard_referencing_examples_tcm44-39847.pdf

***Дані про авторів*:** мають починатися рядком "Про авторів:", напівжирний курсив. Далі вказуються для кожного з авторів ПІБ повністю, наукове звання, посада, адреса, кількість публікацій в українських виданнях (приблизна), кількість публікацій в зарубіжних індексованих виданнях (приблизна), індекс Хірша (за наявності), обов'язково номер ORCID (сайт ORCID http://orcid.org/).

***Дані про місце роботи авторів*:** починаються рядком "Місце роботи авторів:", напівжирний курсив. Далі вказуються місце роботи, адреса, телефон, факс, електронна пошта, контактний телефон.

3. Оформлення файлу з анотаціями.

Файл з анотаціями містить інформацію двома мовами (наприклад, якщо стаття написана українською мовою, то анотація та ключові слова – англійською і українською мовами) та має бути оформлений у дві колонки: УДК (шрифт – 8 пт); назва статті (шрифт – 12 пт, напівжирний); прізвища та ініціали авторів (шрифт – 12 пт); текст анотації, ключові слова (шрифт – 10 пт).

Вимоги до анотації англійською мовою: обсяг від 100 до 250 слів, інформативність, оригінальність (не є калькою української анотації), змістовність (відображає основний зміст статті і результати досліджень), структурованість (дотримується логіки опису результатів у статті).

Документ зберігається у форматі doc або docx. Ім´я подається транслітерацією, як прізвище автора (авторів), наприклад, "Petrenko_Annot.doc".

*16.07.2020 р. набули чинності положення Закону України «Про забезпечення функціонування української мови як державної». Відповідно до статті 22 «Державна мова у сфері науки» у наукових виданнях не повинно бути вміщено матеріалів іншими мовами, окрім державної, англійської та мов ЄС.

Примітка: Підписний індекс журналу "Проблеми програмування" – **90853.**