Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Аналіз динамічних графів знань для «розумних» міських застосунків

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Мицишин П.М.
(підпис) (прізвище та ініціали)

Керівник _____ Дуда О.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Марценко С.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Гащин Н.Б.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_26_» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Мицишину Павлу Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз динамічних графів знань для «розумних» міських застосунків

Керівник роботи Дуда Олексій Михайлович, канд. техн.наук, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 20*24* року № *4/7-470*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Наукові публікації щодо застосунків «розумних міст» та динамічних графів знань

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Галузь динамічних графів знань для цифрового подання даних «розумних» застосунків. 1.1 Стан досліджень в галузі динамічних графів знань для «розумних міст». 1.2 Походження даних для робочих процесів «розумного міста». 1.3 Цифрові двійники реального світу «розумних міст». 1.4 Онтологія деривації даних «розумних» міських застосунків. 2. Аналіз динамічних графів знань для «розумних» міських застосунків. 2.1 Елемент онтології «OntoDerivation ABox». 2.2 Підключення «розумних» міських застосунків до «OntoAgent». 2.3. Агент деривації «розумних» міських застосунків. 2.4. Синхронний режим зв'язку агента «розумних» міських застосунків. 2.5. Асинхронний режим зв'язку агента «розумних» міських застосунків. 3. Робота з графами знань застосунків «розумних міст». 3.1. Оновлення змішаного типу клієнтів «розумних» міських застосунків. 3.2. Автоматизоване заповнення та оновлення підграфа виведення «розумних» міських застосунків. 3.3. Оцінювання онтології «розумного міста». 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд. 2. Тема та мета роботи. 3. Завдання роботи. 4. Актуальність роботи. 5. Практичне значення одержаних результатів. 6. Розумне місто. 7. Динамічні графи знань. 8. Схема графа знань реального світу – «розумного міста». 9. Деривація. 10. Концепції та зв'язки онтології «розумного міста». 11. Структури похідних підграфів як спрямовані ациклічні графи. 12. Діаграма активності UML шаблону агента деривації «розумних» міських застосунків. 13. Висновки. 14. Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці		12.06.2024	15.06.2024

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	
2.	Підбір джерел про застосунки «розумних міст» та	31.01.2024-03.02.2024	
	динамічні графи знань		
3.	Опрацювання джерел про застосунки «розумних	04.02.2024-06.02.2024	
	міст» та динамічні графи знань		
4.	Виконання дослідження щодо аналізу динамічних	07.02.2024-11.02.2024	
	графів знань для «розумних» міських застосунків		
5.	Оформлення розділу «Галузь динамічних графів	03.06.2024-05.06.2024	
	знань для цифрового подання даних «розумних»		
	застосунків»		
6.	Оформлення розділу «Аналіз динамічних графів	06.06.2024-08.06.2024	
	знань для «розумних» міських застосунків»		
7.	Оформлення розділу «Робота з графами знань	09.06.2024-11.06.2024	
	застосунків «розумних міст»»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	12.06.2024-13.06.2024	
9.	Виконання завдання до підрозділу «Основи охорони	14.06.2024-15.06.2024	
	праці»		
10.	Оформлення кваліфікаційної роботи	16.06.2024-17.06.2024	
11.	Нормоконтроль	18.06.2024-19.06.2024	
12.	Перевірка на плагіат	20.06.2024	
13.	Попередній захист кваліфікаційної роботи	21.06.2024	
14.	Захист кваліфікаційної роботи	27.06.2024	

Студент _____ Мицишин П.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дуда О.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Аналіз динамічних графів знань для «розумних» міських застосунків // Кваліфікаційна робота освітнього рівня «Бакалавр» // Мицишин Павло Михайлович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2024 // С. 64, рис. – 15, табл. – 0, кресл. – 14, додат. – 0, бібліогр. – 47.

Ключові слова: динамічний граф знань, похідна інформація, походження даних, спрямований ациклічний граф, розумні міста.

Кваліфікаційна робота присвячена аналізу динамічних графів знань для «розумних» міських застосунків. В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» розглянуто стан досліджень в галузі динамічних графів знань для «розумних міст». Описано походження даних для робочих процесів «розумного міста». Подано опис цифрових двійників реального світу. Розглянуто цифрові двійники реального світу «розумних міст». Висвітлено онтологія деривації даних «розумних» міських застосунків. Описано елемент онтології «OntoDerivation TBox».

В другому розділі кваліфікаційної роботи описано елемент онтології «OntoDerivation ABox». Висвітлено підключення «розумних» міських застосунків до «OntoAgent». Розглянуто агент деривації «розумних» міських застосунків. Проаналізовано синхронний режим зв'язку агента «розумних» міських застосунків. Висвітлено асинхронний режим зв'язку агента «розумних» міських застосунків. Описано паралельність і багатопотоковість агента «розумних» міських застосунків. Розглянуто клієнт деривації «розумних» міських застосунків. В третьому розділі кваліфікаційної роботи подано оновлення змішаного типу клієнтів «розумних» міських застосунків. Проаналізовано автоматизоване заповнення та оновлення підграфа виведення «розумних» міських застосунків. Описано процес оцінювання онтології «розумного міста».

ANNOTATION

Analysis of Dynamic Knowledge Graphs for "Smart" Urban Applications // Qualification work of the educational level "Bachelor" // Mytsyshyn Pavlo Mykhailovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-42 // Ternopil, 2024 // P. 64, fig. - 15, tabl. - 0, chair. - 14, annexes. – 0, references - 47.

Keywords: dynamic knowledge graph, derived information, data origin, directed acyclic graph, smart cities.

The qualification work is devoted to the analysis of dynamic knowledge graphs for "smart" urban applications. In the first section of the qualifying work of the "Bachelor" educational level, the state of research in the field of dynamic knowledge graphs for "smart cities" is considered. Data origins for smart city workflows are described. A description of digital doubles of the real world is provided. The digital counterparts of the real world of "smart cities" are considered. The ontology of data derivation of "smart" urban applications is highlighted. The ontology element "OntoDerivation TBox" is described. In the second section of the qualification work, the ontology element "OntoDerivation ABox" is described. The connection of "smart" city applications to "OntoAgent" is highlighted. The agent of derivation of "smart" urban applications is considered. The synchronous communication mode of the agent of "smart" urban applications is analyzed. The asynchronous communication mode of the agent of "smart" urban applications is highlighted. Parallelism and multithreading of the agent of "smart" urban applications are described. The client of the derivation of "smart" urban applications is considered. In the third section of the qualification work, an update of the mixed type of clients of "smart" urban applications is presented. The automated filling and updating of the output subgraph of "smart" urban applications was analyzed.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

BIBO (англ. Bibliographic Ontology) – бібліографічна онтологія.

DAG (англ. Directed Acyclic Graphs) – спрямовані ациклічні графи.

DC Terms (англ. Dublin Core Terms) – умови дублінського ядра.

DARP (англ. Distributed Application Runtime) – розподілена програма виконання.

HPC (англ. High-Performance Computing) – високопродуктивні обчислення.

IRI (англ. internationalized resource identifiers) – інтернаціоналізовані ідентифікатори ресурсів.

OPMO (англ. Open Provenance Model Ontology) – онтологія відкритої моделі походження.

PROVO (англ. W3C Provenance Ontology) – онтологія походження W3C.

SOA (англ. Service-Oriented Architecture) – сервіс-орієнтована архітектура.

WMS (англ. Work Process Management Systems) – системи керування робочим процесом.

ЗМІСТ

ВСТУП

Актуальність теми. Концепція «розумних» міст здобуває все більшу популярність. Сучасні міста активно впроваджують концепції «розумних» міст, що включають використання новітніх технологій для підвищення ефективності управління міською інфраструктурою, покращення якості життя мешканців та зменшення впливу на довкілля. «Розумні міста» використовують інформаційно-комунікаційні технології (ІКТ) для збору та аналізу даних про різні аспекти міського життя, включаючи транспорт, енергетику, охорону здоров'я та безпеку. ані про «розумні» міста є складними та динамічними, що утруднює їх аналіз традиційними методами. З розвитком інтернету речей (IoT) та інших технологій, які генерують великі обсяги даних у режимі реального часу, виникає необхідність у нових підходах до їх обробки та аналізу. Графи знань – це форма представлення знань, яка добре підходить для моделювання складних та динамічних систем. Динамічні графи знань надають можливість інтегрувати та аналізувати ці дані, що сприяє точнішому прийняттю рішень у міському управлінні. Використання динамічних графів знань може покращити ефективність аналізу даних про «розумні» міста та сприяти розробці нових та інноваційних «розумних» міських застосунків. Тому аналіз динамічних графів знань для «розумних» міських застосунків є актуальним напрямком досліджень, науково-новим та практично значущим та має потенціал зробити значний внесок у розвиток галузі «розумних» міст.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробка та дослідження методів і моделей використання динамічних графів знань для підвищення ефективності та якості управління міськими процесами, а також для оптимізації функціонування інфраструктури «розумних» міст. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Проаналізувати методи та алгоритми аналізу та візуалізації графів знань.

– Розробити методи та алгоритми для представлення даних «розумних» міст у вигляді динамічних графів знань.

– Розробити прототипи «розумних» міських застосунків, які використовують динамічні графи знань.

Практичне значення одержаних результатів. У кваліфікаційній роботі проаналізовано результуючу інформаційну структуру, щоб семантично анотувати, як частина інформації «розумного міста» може бути отримана від інших даних у динамічному графі знань. Відбувається кодування з використанням поняття «виведення» та фіксації метаданих за допомогою онтології. Розглянуто реалізацію синхронний та асинхронних режимів зв’язку для агентів, які взаємодіють із графом знань. При поєднанні можуть виникати орієнтовані ациклічні графи похідних, зі змінними даними, що поширюються через граф знань за допомогою програмно-алгоритмічних агентів. Хоча сама структура не залежить від окремого домену, її застосовано в контексті «розумних міст». При цьому продемонстровано, що вона здатна обробляти послідовні події в різних часових масштабах.

РОЗДІЛ 1. ГАЛУЗЬ ДИНАМІЧНИХ ГРАФІВ ЗНАНЬ ДЛЯ ЦИФРОВОГО ПОДАННЯ ДАНИХ «РОЗУМНИХ» ЗАСТОСУНКІВ

1.1 Стан досліджень в галузі динамічних графів знань для «розумних міст»

Натхненні технологією семантичного вебу, графи знань набувають популярності як у корпоративних застосунках [1], так і в наукових колах [2]. Вони розглядаються як інноваційний підхід до інтеграції різноманітних джерел інформації та сприяння спільному розумінню серед експертів у галузі «розумних міст» [3]. Відомими прикладами статичних графів знань є DBpedia [4] та Wikidata [5].

Динамічний аспект графів знань вивчався в [6], де вони використовуються як концентратори для інтеграції складних систем програмних агентів, з'єднуючи різні домени в конкретних випадках використання. Це дає змогу аналізувати сценарії «що-якщо» та автоматизовано приймати рішення, які імітують людську поведінку. Це важливий крок до оригінального бачення семантичних мереж, які є повністю анотованою мережею машиночитаних даних, що можуть автономно оброблятися програмними агентами [7]. Однак це також підкреслює потребу в надійних методах управління змінами та взаємозалежностями в складній інформаційній мережі, особливо в багатому на дані середовищі «розумних міст», яке може містити дезінформуючі джерела даних.

Ключовим сприятливим фактором, визначеним науковою спільнотою, є походження даних [8], тобто те, звідки походить частина інформації та як вона виникла. Походження даних охоплює обширний спектр аспектів, зокрема:

– Опубліковані інформаційні джерела.
– Інтернет.

– дані, безпосередньо отримані за допомогою вимірювального IoT-пристрою.

У [9] розроблено низку онтологій походження даних, наприклад:

– PAV.

– W3C Provenance Ontology (PROVO)

– Dublin Core Terms (DC Terms)

– Бібліографічна онтологія (BIBO).

– Онтологія відкритої моделі походження (OPMO).

Більш повний огляд розробок у галузі моделей даних та їх походження подано в [10].

У динамічних графах знань можна розглянути конкретну підпроблему походження, а саме коли деякі фрагменти інформації безпосередньо обчислюються або виводяться з інших фрагментів інформації програмними агентами, дані з яких зберігаються в графі знань.

Коли декілька фрагментів інформації залежать один від одного певним чином, отриманий каскад інформації просувається в часі через серію скоординованих зв’язків агентів, де обчислені значення одного процесу є вхідними даними для інших наступних процесів.

Щоб зробити граф знань справді динамічним, система також повинна забезпечувати автоматичне розповсюдження збурень у вихідних вхідних даних. Це вводить іншу точку зору – кешування. Окрім надання функцій агентів для обчислення результату, цей результат зберігається, тобто кешується, у графі знань «розумного міста». Обширний спектр технічних задач схожі, наприклад, можливість визначити, коли кеш застарів. При цьому забезпечуються функції оновлення або перерахунку кешу.

Будь-який підхід до вирішення цієї задачі може скористатись результатами, отриманими в інших сферах. У наукових обчисленнях така композиція обчислювальних завдань називається робочим потоком або конвеєром [11].

Робочі процеси зазвичай виражаються як спрямовані ациклічні графи (DAG), де вузли представляють завдання, а ребра представляють потоки даних [12]. Кожне завдання можна розпочати, лише якщо виконано всі його попередні завдання. Існує обширна множина різних систем керування робочим процесом (WMS), які можна використовувати для оркестрування цих завдань, починаючи від традиційних парадигм до сучасних підходів:

– Pegasus [13].
– Kepler.
– Apache Airflow.
– Nextflow.
– Lightning AI [14].

Деякі дослідження зосереджені на адаптивних робочих процесах, де зміни у вхідних даних можуть бути включені в обчислення на льоту, щоб забезпечити відповіді в режимі реального часу на динамічні події. Однак ці системи часто покладаються на неоднорідні та неструктуровані моделі даних без семантичних анотацій [15], що може ускладнити досягнення сумісності між різними системами, перешкоджаючи об’єднанню спільноти робочих процесів.

Ще одна сфера, з якої ми можемо навчитися, це мікросервісна архітектура [16]. Це сервіс-орієнтована архітектура (SOA), яка наголошує на слабкому зв’язку та високому рівні відношень. Це передбачає незалежну розробку, розгортання та підтримку кожної служби в екосистемі «розумного міста» невеликою спеціальною командою [17]. Коли відбувається подія, яка є подібною концепції до екземпляра виконання одного попередньо визначеного робочого процесу, мікросервіси складаються та координуються за допомогою передачі повідомлень. Ця архітектура не накладає обмежень на розробників щодо технологій, які використовуються для реалізації кожного мікросервісу «розумного міста», якщо узгоджено уніфікований інтерфейс. Тим не менш, він висуває значні вимоги до мережі для забезпечення успішного зв’язку. У цій парадигмі «Distributed Application Runtime» (DARP) [18] використовує контексні відношення для спрощення зв’язку за допомогою прямого обміну

повідомленнями про публікацію, підписку або на основі подій, об’єднуючи обидва режими зв’язку на одній платформі «розумного міста.

Роблячи нотатки з цих подій, ми можемо підсумувати та запропонувати рішення для графа знань. Згідно з дизайном, дані в графі знань «розумного міста» можна унікально ідентифікувати за допомогою інтернаціоналізованих ідентифікаторів ресурсів (IRI). Після надання привілеїв доступу всі активні агенти мають однакове бачення світу «розумного міста». Подібно до шини повідомлень в архітектурі мікросервісу, зв’язок між агентами, що працюють на графі знань, можна делегувати через обслуговування коректних IRI. Це усуває необхідність у великій одноранговій передачі даних. Його можна додатково поєднати з ідеєю, отриманою в науковому робочому процесі, щоб моделювати залежності обчислень як DAG у графі знань «розумного міста». Кодуючи повідомлення агентів як записи про походження, ми усуваємо потребу в прямому спілкуванні між агентами та даємо можливість інформації проходити через граф знань. Це роз’єднує систему та дає змогу створювати розподілену екосистему агентів без необхідності знати про існування один одного.

Як «цифровий двійник», що реалізований як динамічний граф знань [19], цифровий двійник вважається перспективним кандидатом для оцінки реалізації цієї технології. У поточному стані існує переплетена мережа концепцій, що охоплюють часові та просторові масштаби, від молекули та хімічного механізму до «розумної» лабораторії чи «розумного міста» [20]. Цифровий двійник постійно розвивається, оскільки підтримується мережею активних агентів, які регулярно вводять нові дані та оновлюють існуючі. Для ефективного керування діями цих агентів і забезпечення точного відстеження їхньої діяльності потрібна всеосяжна архітектура «розумного міста». Враховуючи широкий спектр даних і випадків використання, охоплених цифровим двійником «розумного міста», загальна і надійна реалізація такої архітектури є обов’язковою.

Метою кваліфікаційної роботи освітнього рівня «бакалавр» є підтвердження концепції технологічно-агностичної реалізації похідної

інформаційної основи «розумного міста» для динамічних графів знань. Водночас потрібно розглянути виведення для реалізації архітектури графа знань «розумного міста». При цьому доцільно використати спрощену онтологію для позначення походження, шаблон агента для стандартизації операцій агента та автоматизовану структуру для поширення змін інформації в динамічному графі знань «розумного міста». Розробка спрямована на зниження бар'єру входу для дослідників, щоб моделювати будь-які каскадні події реального світу, зокрема «розумного міста», з мінімальними зусиллями, надаючи зручний шаблон. Важливість цієї інфраструктури полягає в її подвійній здатності не лише відстежувати та документувати процес обчислення, але й автоматично повторно виконувати обчислення при доступі до історичної інформації «розумного міста». В епоху, яка характеризується як складністю, так і великою кількістю даних, структура дає змогу автоматизовано інтегрувати швидкий приплив нової інформації «розумного міста», забезпечуючи постійний доступ до актуальної інформації про предмет інтересу.

Зокрема, автори [21] демонструють це за допомогою оцінки впливу в рамках проекту цифрових двійників. Вибір домену «розумних міст» мотивований його необхідністю обробки як швидкої, так і повільної міської динаміки процесів [22], що робить його актуальним контекстом для демонстрації здатності структури враховувати події реального світу з різною частотою. Процеси в середовищі «розумного міста», вимагають оперативного прийняття рішень і цілісної перспективи реагування. Тому цінним атрибутом є можливість завжди мати актуальну інформацію. Однак важливо підкреслити, що універсальність інфраструктури виходить за межі сфери «розумних міст».

1.2 Походження даних для робочих процесів «розумного міста»

Проведемо короткий огляд відповідних онтологій щодо походження робочих процесів «розумного міста». Адже досвід у розробці цифрових двійників «розумних міст» сприяє обґрунтуванню структури виведення [23].

Де-факто стандарт PROV-O в галузі цифрових двійників [24] приймає три базові класи для опису походження:

– «prov:Entity» для речей, що підлягають опису;

– «prov:Activity» для подій, що відбуваються протягом певного періоду, що призводить до трансформації сутностей;

– «prov:Agent» – відповідальний за здійснення діяльності.

PROV-O також надає кваліфіковані терміни як детальну інформацію про бінарні зв’язки між цими базовими класами. Наприклад, «prov:Association» кваліфікує зв’язок «prov:wasAssociatedWith» між «prov:Activity» і «prov:Agent», вказуючи на екземпляр «prov:Plan» за допомогою «prov:hadPlan». Ця кваліфікація вказує на дії, які здійснює агент для досягнення своїх цілей. Однак «prov:Plan» надається як досить ізольована концепція, без додаткових специфікацій щодо того, як її можна поєднати з іншими концепціями, які мають відношення до виконання. Таким чином, PROV-O більше підходить для ретроспективних записів подій, а не для активного виклику агентів та виконання завдань.

Онтологія P-Plan [25] частково заповнює цю прогалину, розширюючи PROV-O термінами «p-plan:Step» і «p-plan:Variable» в наукових процесах. Це розширення допомагає публікувати методи та процеси наукових робочих процесів як зв’язані даних. Пізніша наукова робота [26] поєднується з відкритою моделлю походження (OPM), що призводить до OPMW, яка підтримує зв’язок між шаблоном робочого процесу та його конкретним застосуванням.

Примітно, що OPM була застарілою моделлю даних, розробленою з серії «Provenance Challenge» [27] і фактично була еталонною для створення PROV-O. Автори OPMW визнали, що одним з аспектів, який ще не підтримується OPMW, є автоматичне повторне виконання опублікованих робочих процесів у гетерогенних обчислювальних середовищах [26]. У цьому плані сучасні контейнерні рішення та хмарні сервіси можуть бути потенційним рішенням [28].

1.3 Цифрові двійники реального світу «розумних міст»

У цифрових двійниках фізичного світу, відбувається сканування реальності в якій ми живемо. Вона представляється як базовий світ. Гіпотетичні версії світу, в яких певні змінні чи припущення відрізняються, представлені як паралельні світи. Цими альтернативними цифровими всесвітами керують програмні агенти, які можуть виконувати різноманітні завдання. Важливо, що цифровий двійник розглядає цих агентів як частину графа знань (див. рисунок 1.1) [21].

Рисунок 1.1 – Схема графа знань цифрових двійників реального світу [21]

Зверніть увагу, що рисунок є лише ілюстративним і не відображає фактичних даних.

Використовуючи цей стек технологій, цифровий двійник реального світу є універсальним у трьох аспектах:

1) відповідає на міждоменні запитання про базовий світ [29];

2) керування об’єктами реального світу;

3) підтримка аналізу сценарію «що-якщо» з паралельними світами [30].

Оскільки проект цифровий двійник реального світу продовжує розвиватися, він прагне точніше представити складні явища «розумного міста» в просторово-часових масштабах. Для цього потрібен інструмент ефективної координації дій агентів, які оновлюють і реструктуризують граф знань «розумного міста».

В даний час програмні агенти представлені подібно до семантичних веб-служб [31], які викликаються через HTTP-запити. Для обчислень, що потребують багато часу, доступний асинхронний спостерігач за завданнями, який делегує завдання високопродуктивним обчисленням (HPC). Це було застосовано для оцінки впливу квантових розрахунків на дисперсію забруднення повітря та для автоматизації калібрування механізмів горіння [32]. Ці підходи в основному дотримуються парадигми статичного віддаленого виклику процедури. Щоб повністю розкрити потенціал моделі динамічного світу, краще використовувати більш гнучку та адаптивну архітектуру «розумного міста», яка може сприяти автономній взаємодії між агентами та графом знань.

1.4 Онтологія деривації даних «розумних» міських застосунків

Починемо з представлення онтології, створеної для анотування розмітки походження даних «розумних» міських застосунків.

«Деривація» – це запис про одиничний випадок того факту, що деякі частини інформації були отримані або обчислені з деяких інших частин інформації «розумних» міських застосунків. Термін «підграф деривації» використовується для опису підграфа всієї розмітки «розумного міста», пов’язаної з деривацією, коли представлено набір взаємозалежних процесів. Оскільки інформація в графі знань «розумного міста» фіксується у форматі висловлювань трійки «Суб’єкт–Предикат–Об’єкт», зберігання розмітки для

фіксації цього факту можна розглядати як спосіб приєднання довільних метаданих до трійок [33]. Для цього були розроблені або зараз знаходяться в розробці загальні рішення, такі як «reification» і «W3C RDFstar» [34]. Хоча деякі реалізації існують, наприклад Blazegraph's Reification Done Right (RDR) або GraphDB [35]. Наразі вони не мають достатньо широкої підтримки для реалізації фреймворку деривації технологічно агностичним способом, без прив’язки до конкретного «розумного» міського застосунку. Тому доцільно вказати необхідні метадані явно як трійки та представити OntoDerivation як полегшену онтологію «розумного міста».

Спочатку пояснимо термінологічний компонент (TBox), а потім наведемо приклад екземпляра ассертивного компонента (ABox). Зв’язок між OntoDerivation і OntoAgent [31], онтологією, що використовується для визначення можливостей агентів, додатково демонструється тим, як їх можна використовувати разом для керування діями агентів.

Концепція з PROV-O безпосередньо не буде повторно використовуватись щоб полегшити реалізацію конкретного аспекту походження даних «розумного міста» та їх оновлень – як частина інформації обчислюється з інших джерел і коли це відбувається . Зокрема, з таких причин:

– Необхідність врахування різноманітних часових масштабів у часі відповіді щодо діяльності, яку виконують агенти. Ця диференціація є важливою для полегшення автоматизованих оновлень отриманої інформації через програмні агенти «розумних» міських застосунків, зокрема щодо синхронних і асинхронних процесів, оскільки останні потребують запису статусу завдання. Це відноситься до рівня TBox.

– Вимога щодо уніфікованого представлення позначок часу для позначення своєчасності як чистих вхідних даних «розумних» міських застосунків, так і екземплярів похідних даних, що спростить реалізацію алгоритму оновлення для визначення того, чи є похідне застарілим. PROVO коментує ці позначки часу, використовуючи різні властивості даних із діапазоном «xsd:dateTime», що призводить до додаткового перетворення для

обробки похідних екземплярів із різних часових поясів. Це стосується створення екземплярів записів про походження даних «розумних» міських застосунків.

– Створення екземпляра кожної частини інформації «prov:Entity» унеможливлює розрізнення концепцій у сигнатурах введення та виведення агентів, не кажучи вже про диференціацію можливостей окремих агентів у складному підграфі похідних структур даних «розумного міста». Одним із можливих обхідних шляхів є оголошення всіх концепцій у вводі-виводі як підкласів «prov:Entity», однак це вводить додатковий рівень абстракції даних «розумних» міських застосунків практично для всіх концепцій в онтологіях домену.

1.5 Елемент онтології «OntoDerivation TBox»

На рисунку 1.2 зображено два типи похідних даних, тобто синхронну «Derivation» і асинхронну «DerivationAsyn» для адаптації до ситуацій, які відповідають у різних часових масштабах «розумного міста», коли надходить запит [21]. Усі класи та властивості належать до простору імен «OntoDerivation» «розумного міста», якщо не вказано інше.

У синхронному режимі зв’язок здійснюється через кінцеву точку «розумного» міського застосунку, яку відкриває програмний агент. Таким чином, він швидший і, отже, призначений для програмно алгоритмічних засобів «розумного міста», які вимагають відповідей у реальному часі. Асинхронний режим взаємодіє виключно через граф знань . Перевагою є запис кожного етапу операції в знаннях графу, але він повільніший і, отже, краще підходить для відносно дорогих робіт. Інформаційні залежності деривації послідовно позначаються незалежно від протоколу зв’язку «розумних» міських застосунків, причому отримана інформація належить до деривації, яка одержується з вихідної інформації – вхідних даних і одержується за допомогою агента, визначеного в «OntoAgent» [31].

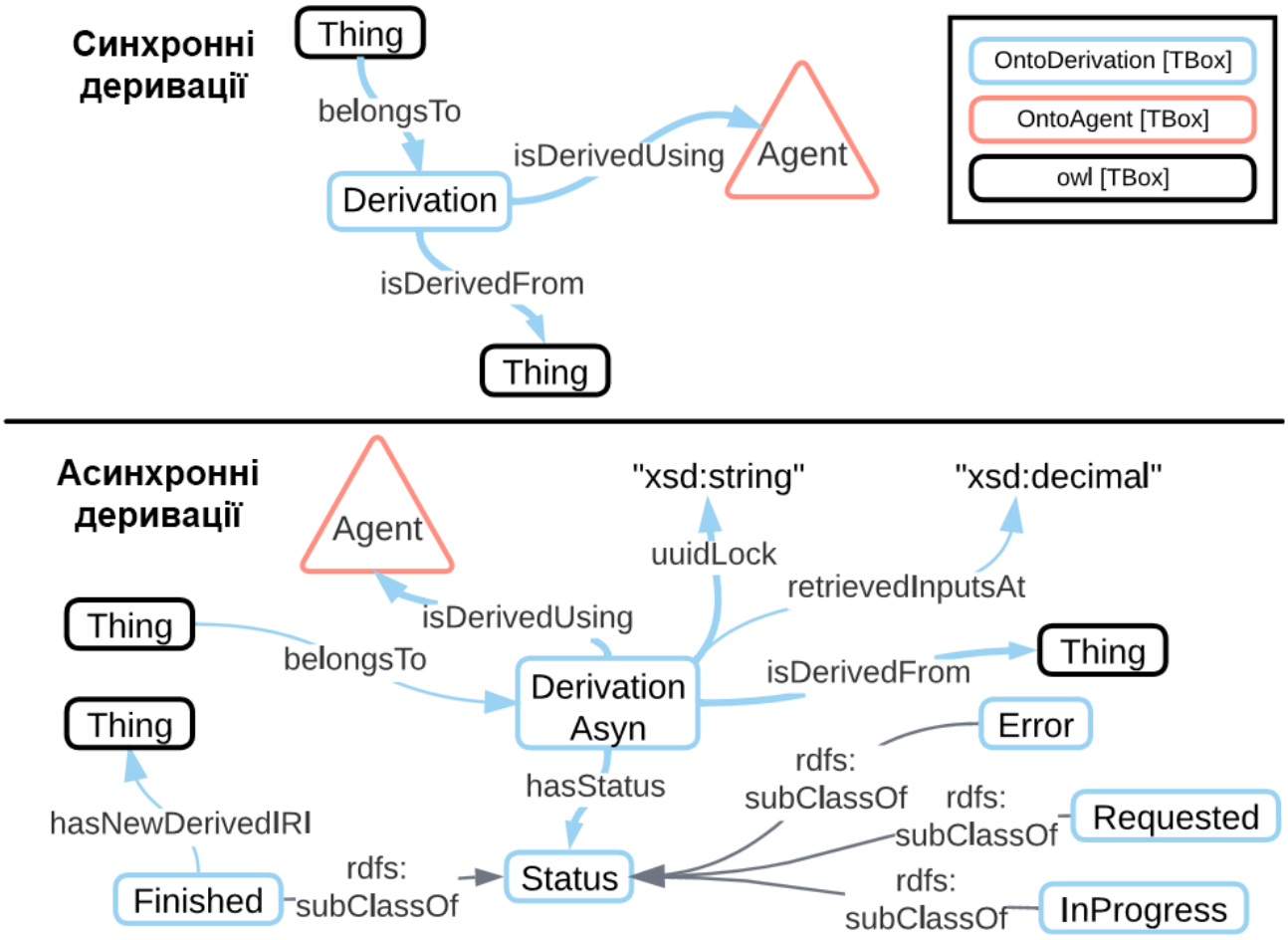

Рисунок 1.2 – Концепції та зв’язки онтології «розумного міста» «OntoDerivation» [21]

Доцільно зазначити, що немає обмежень на кількість входів або виходів для деривації, але одна вихідна сутність «розумних» міських застосунків не може належати більш ніж одному екземпляру деривації. Як вихідна, так і похідна інформація абстрагуються за допомогою «owl:Thing», тому також можливо, що вхідні дані похідної є частиною іншого екземпляра похідної, тобто вхідні дані є частиною похідної інформації та належать іншій похідній.

Для підтримки асинхронної роботи «розумних» міських застосунків введено концепцію «Статус», яка позначає стан асинхронної деривації з доступними параметрами «Запитано», «Виконується», «Завершено» та «Помилка». Властивість даних «retrievedInputsAt» записує мітку часу, коли вхідні дані для виведення були прочитані, щоб почати обчислення, яке пізніше буде використано для оновлення мітки часу для примірника виведення

«розумних» міських застосунків. Властивість даних «uuidLock» унікально ідентифікує потік агента, який обробляє похідне, і запобігає будь-яким змінам з боку інших потоків «розумних» міських застосунків, які не містять правильний ключ. Це забезпечує потокобезпечні операції, коли використовується декілька потоків для підвищення пропускної здатності обробки похідних даних. Спеціальна властивість об’єкта «hasNewDerivedIRI» використовується у статусі «Готово» для тимчасового зв’язування будь-якої щойно отриманої інформації «розумних» міських застосунків. Ці вихідні об’єкти зрештою будуть повторно підключені до екземпляра деривації після того, як агенти очистять статус. Як і кінцеві результати, немає обмежень на кількість нових сутностей «розумних» міських застосунків, які можна підключити через «hasNewDerivedIRI» для кожного екземпляра деривації [36].

1.6 Висновок до першого розділу

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» розглянуто стан досліджень в галузі динамічних графів знань для «розумних міст». Описано походження даних для робочих процесів «розумного міста». Подано опис цифрових двійників реального світу. Розглянуто цифрові двійники реального світу «розумних міст». Висвітлено онтологія деривації даних «розумних» міських застосунків. Описано елемент онтології «OntoDerivation TBox».

РОЗДІЛ 2. АНАЛІЗ ДИНАМІЧНИХ ГРАФІВ ЗНАНЬ ДЛЯ «РОЗУМНИХ» МІСЬКИХ ЗАСТОСУНКІВ

2.1 Елемент онтології «OntoDerivation ABox»

На рисунку 2.1 показаний екземпляр виведення «розумних» міських застосунків та його спрощене представлення. Після ініціалізації кожна похідна інформаційна сутність анотується міткою часу відповідно до стандарту W3C [37]. Мітка часу Unix вибирається замість «xsd:dateTime», щоб увімкнути пряме числове порівняння. Прості цілі числа використовуватимуться замість фактичної Unix-позначки часу для зручності читання. Ця позначка часу використовується протягом життєвого циклу деривації «розумних» міських застосунків, щоб визначити, чи вона застаріла.

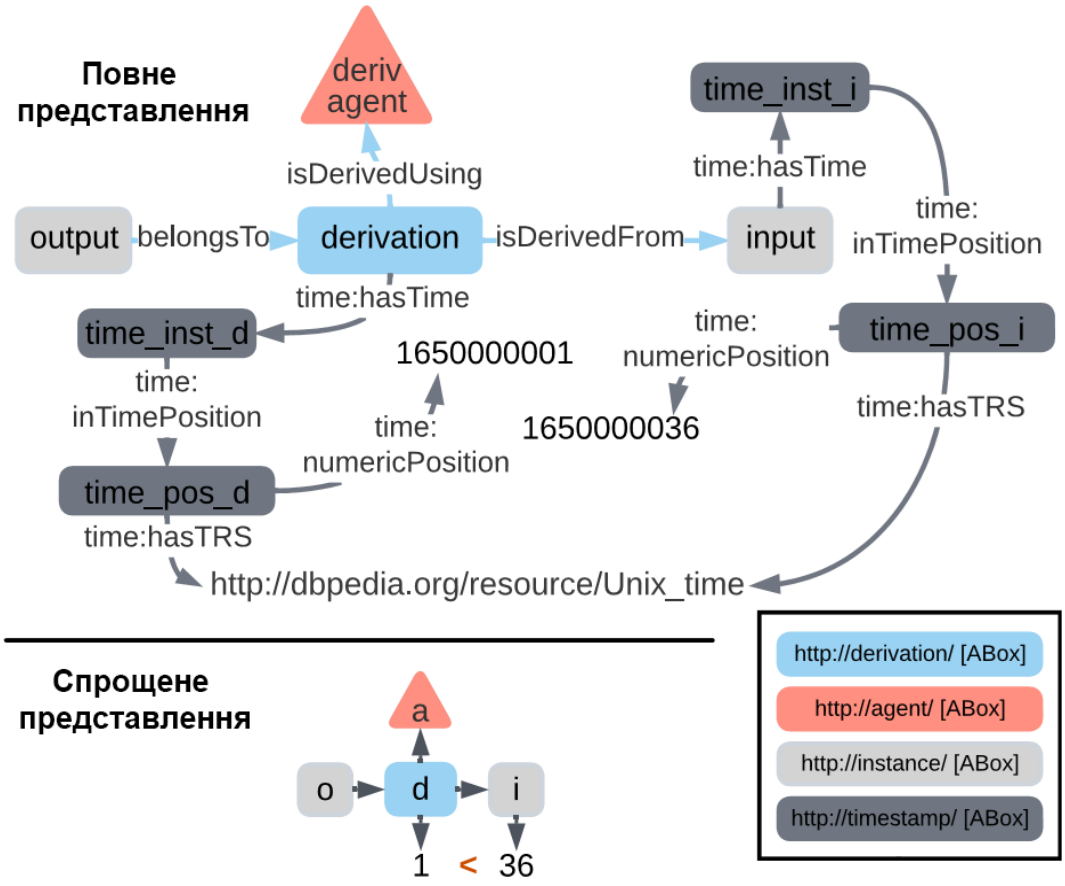

Рисунок 2.1 – Приклад похідного екземпляра, повністю анотованого метаданими [21]

У цьому прикладі, оскільки позначка часу виведення менша, ніж позначка часу його вихідної інформації, тобто «1 < 36», можна зробити висновок, що ці похідні дані застаріли і, отже, потрібне оновлення похідної інформації «розумних» міських застосунків. Однак слід зазначити, що хоча результати похідних даних «розумних» міських застосунків можуть бути введені для інших, вони не пов’язані безпосередньо з міткою часу, але є непрямими через похідні, до яких вони належать.

Наприклад, вихідний екземпляр на рис. 2.1 не пов’язаний безпосередньо з міткою часу, як і всі інші екземпляри, які належать до похідного.

Стрілки між екземплярами вказують на розмітку для залежностей даних. «Інформація» «розумних» міських застосунків тече у зворотному напрямку, тобто справа наліво. Такий вибір зроблено для зменшення надлишкової інформації в графі знань «розумного міста», оскільки своєчасність похідної інформації вже відображається міткою часу екземпляра похідного, якому вона належить. У результаті, коли підграф похідних форм «розумних» міських застосунків складається з декількох пов’язаних похідних, пряме порівняння часових позначок стає неможливим для тих похідних інформаційних сутностей, вхідні дані яких є виходами іншого екземпляра похідних «розумних» міських застосунків. Тому необхідно використовувати додаткові критерії для визначення того, чи є деривація застарілою.

На рисунку 2.2 показано три рівні загальності в похідному підграфі «розумних» міських застосунків, від основного лінійного ланцюга до нелінійного полідерева та загального спрямованого ациклічного графа.

Зв’язок між екземплярами деривації та екземплярами агента використовується для регулювання операцій агента. Усі властивості об’єктів належать до простору імен «OntoDerivation» «розумного міста», якщо не вказано інше.

На відміну від наукових робочих процесів, де стрілки часто вказують у тому ж напрямку, що й потік даних, розмітка в графі знань «розумних» міських застосунків позначає залежності даних і вказує на джерело інформації.

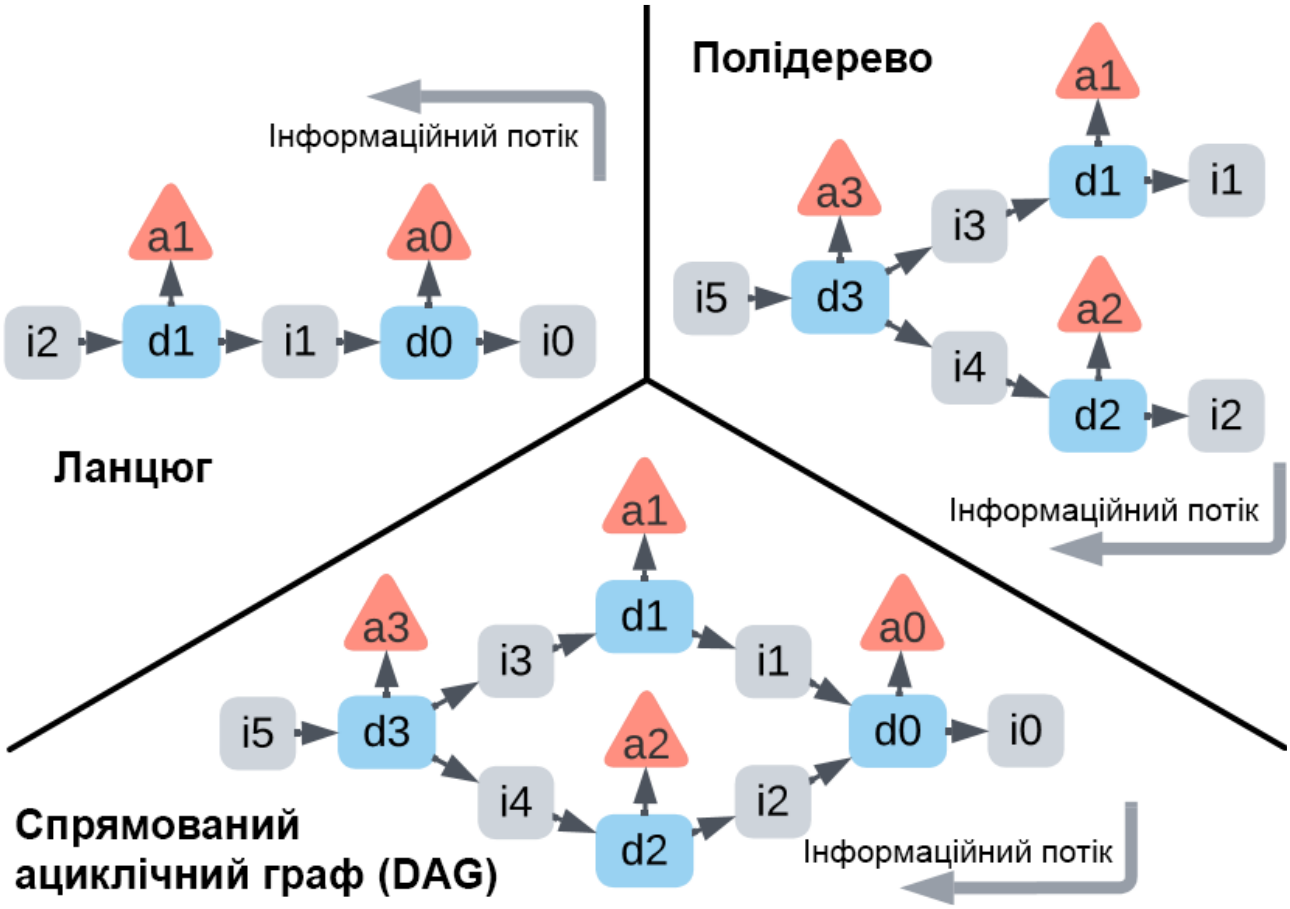

Рисунок 2.2 – Структури похідних підграфів як спрямовані ациклічні графи DAG різної спільності «розумних» міських застосунків [21]

Зміни у вихідній інформації рухаються в протилежному напрямку в межах графа знань «розумних» міських застосунків, як показано напрямком «потоку інформації». У цьому контексті визначаємо «вгору» та «вниз» для посилання на відносне розташування екземпляра деривації «розумних» міських застосунків в цьому потоці. Ключова особливість дизайну полягає в тому, що під час доступу до нього оновлюється лише релевантна інформація «розумних» міських застосунків.

Доцільно підкреслити, що основним призначенням структури похідних інформаційних сутностей «розумних» міських застосунків є представлення логічних залежностей як таких, як історичних даних інформаційних записів, а не розглядати їх як «кроки» в робочому процесі або алгоритмі. Це означає, що, на відміну від робочих процесів [38], циклічні залежності не допускаються в системі похідних, оскільки вони призведуть до логічних протиріч. Незважаючи

на це, інформаційні сутності можна використовувати для запису залежностей фрагментів інформації, які були отримані з алгоритмів «розумних» міських застосунків, що містять цикли та інші циклічні конструкції.

Онтологія «OntoDerivation» розроблена таким чином, що її легко використовувати та легко запитувати для потреб «розумних» міських застосунків. Завдяки цьому надаються інструменти для автоматичного створення похідної розмітки для всіх трьох типів інформаційних сутностей і перевірки згенерованого похідного підграфа «розумного міста». Можна виконувати SPARQL-запити для отримання всіх похідних даних у заданому графі знань «розумного міста».

2.2 Підключення «розумних» міських застосунків до «OntoAgent»

Цифровий двійник реального світу «OntoAgent» [31] використовується «розумними» міськими застосунками для розмітки підпису вводу-виводу агентів та полегшення виявлення агентів. Ця розмітка вказує на концепції в онтологіях домену «розумного міста» та вказує на можливості агента шляхом ідентифікації концепцій, необхідних чи вироблених агентами.

На рисунку 2.3 подано приклад створення екземпляра з використанням «OntoDerivation» і «OntoAgent» [21], де вхідні та похідні виходи екземпляра деривації створюються з підпису вводу-виводу, визначеного в екземплярі «OntoAgent».

Повністю анотованого метаданими похідний екземпляр [21], та його спрощене представлення, яке буде використано нижче. Усі властивості належать простору імен «розумного міста» «OntoDerivation», якщо не вказано інше. Навпаки, «OntoDerivation» зосереджується на рівні екземплярів даних «розумних» міських застосунків, тобто фактичних даних, які обробляються та створюються агентами, що відповідають кожному етапу обчислень, відкриваючи можливість використання обох онтологій для регулювання операцій агента.

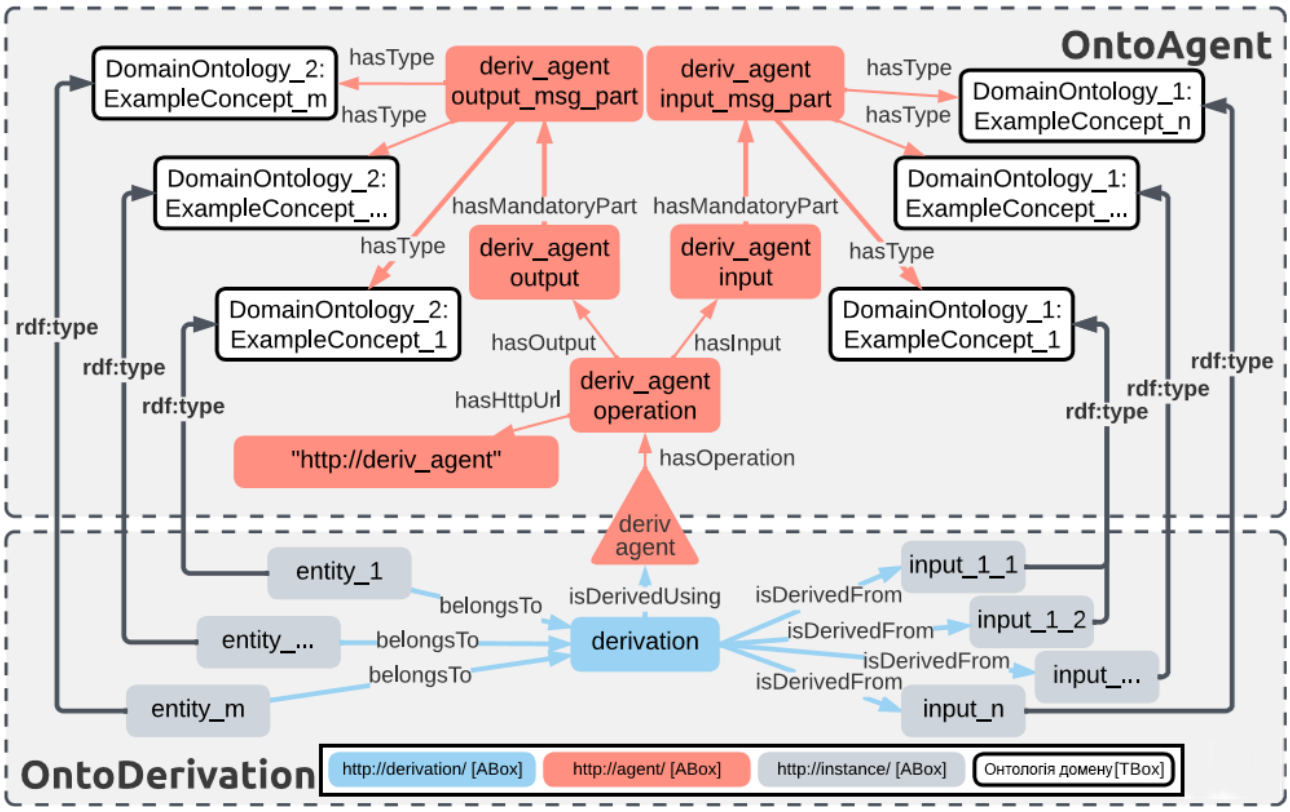

Рисунок 2.3 – Екземпляр, який з’єднує «OntoDerivation» і «OntoAgent» [21]

Агент деривації «розумних» міських застосунків підтримує як синхронні, так і асинхронні деривації. Розробникам потрібно лише надати вузол активності «ProcessRequestParameters» для забезпечення можливостей агента. Дії, позначені жовтим і пурпуровим кольором, представляють операції пошуку даних графа знань «розумного міста» і заповнення відповідно.

Зокрема, зв’язок «OntoAgent:hasType» може вказувати на те, що повідомлення агента охоплює різні концепції з онтологій домену «розумного міста». Для даного екземпляра похідної інформації вхідні дані можна класифікувати за парами ключ–значення, де IRI кожної концепції є ключем, а список екземплярів – значенням. Наприклад, значенням пари з ключем «DomainOntology_1:ExampleConcept_1» буде «input_1_1» і «input_1_2». Цей дизайн пов’язує концептуальні можливості агентів із конкретними завданнями, які їм доручено виконувати. Дотримуючись цієї практики, розробку агентів можна зосередити на рівні концепції, спрощуючи реалізацію.

2.3 Агент деривації «розумних» міських застосунків

Діаграма активності шаблону агента деривації «розумних» міських застосунків в форматі уніфікованої мови моделювання (UML) на рисунку 2.4.

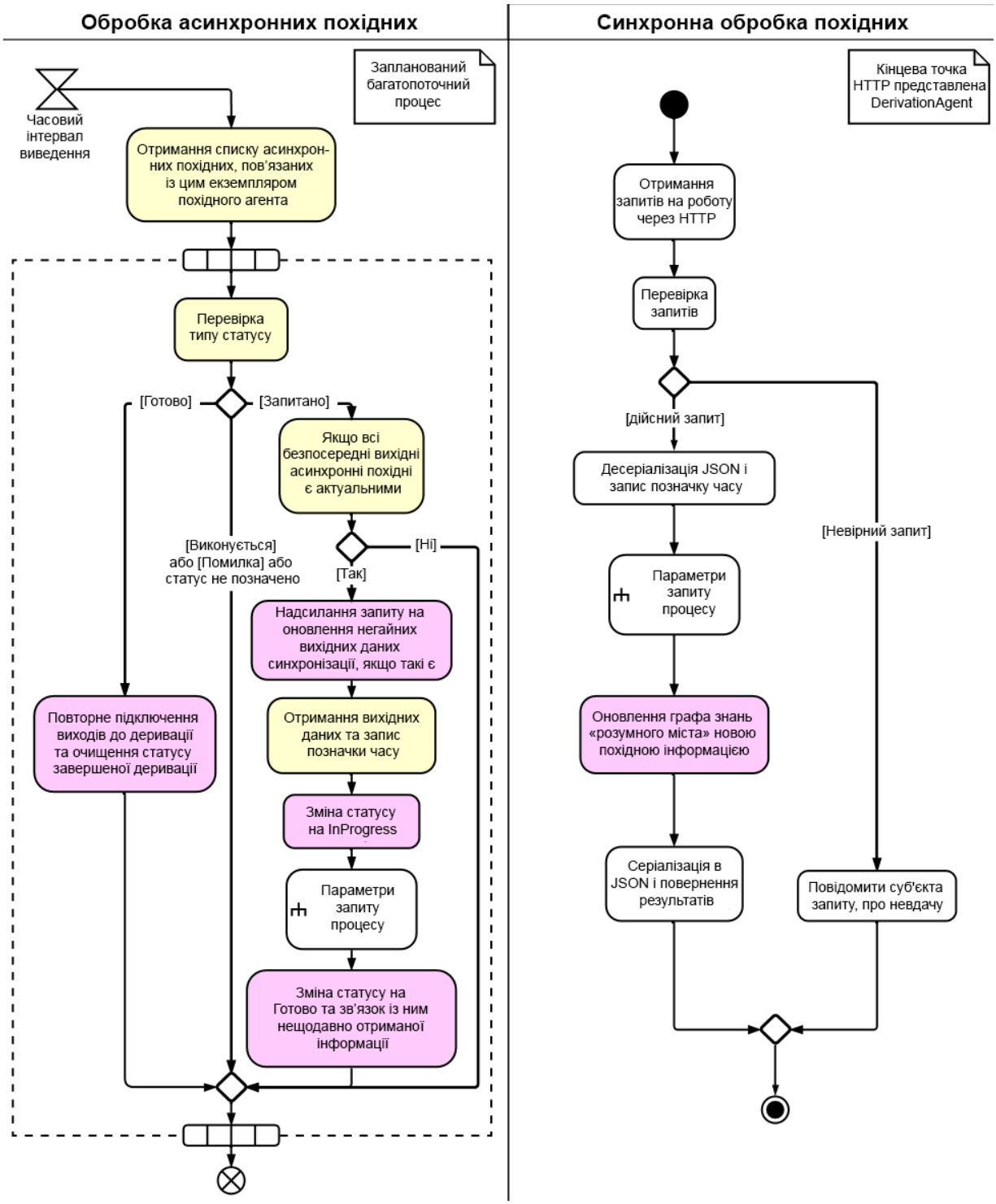

Рисунок 2.4 – Діаграма активності UML шаблону агента деривації «розумних» міських застосунків [21]

Розробники «розумних» міських застосунків можуть використовувати шаблон агента деривації, як відправну точку під час розробки нових програмних агентів.

Подамо опис клієнтської бібліотеки, яку можна використовувати для керування екземплярами деривації даних «розумних» міських застосунків.

Використання онтологічної розмітки для запису кожного кроку в процесі оновлення похідної інформації «розумного міста» значною мірою обмежує зв’язок, який агент повинен здійснювати безпосередній доступ до графу знань, а не до інших агентів. Шаблон агента потрібен для підтримки цього як у синхронному, так і в асинхронному режимах. Шаблон використовує класи контейнерів даних для розміщення вхідних і вихідних даних агентів «розумних» міських застосунків. Ці класи є парами ключ-значення, які можна відобразити за допомогою запитів. Розробникам надаються службові функції для доступу та перевірки відображених вхідних даних і створення виходів для «розумних» міських застосунків. Логіка агента, яка обчислює виходи з вхідних даних, є єдиним кодом, який вимагається від розробника «розумних» міських застосунків. При цьому доцільно зробити шаблон доступним як на Java, так і на Python, щоб підвищити доступність. Основні елементи дизайну детально описано нижче.

2.4 Синхронний режим зв'язку агента «розумних» міських застосунків

Синхронний режим зв'язку агента «розумних» міських застосунків реалізується через прямі запити та відповіді. Отримавши запит на звичайну деривацію, агент «розумних» міських застосунків серіалізує вміст запиту до екземпляра класу контейнера.

Момент часу записується безпосередньо перед передачею вхідних даних для обробки кодом розробників. Цей момент вважається міткою часу, коли вхідні дані «розумних» міських застосунків були прочитані. Після створення

виходів операція оновлення буде сформульована та виконана агентом «розумних» міських застосунків для оновлення графа знань. Якщо помилок немає, оновлення похідних даних «розумних» міських застосунків вважається успішним, і буде повернено відповідь, яка містить отриману похідну інформацію та записану позначку часу.

DAG використовуються в пам’яті для підтримки зворотного відстеження алгоритму DFS під час оновлення підграфа виведення «розумних» міських застосунків. Похідні екземпляри розглядаються як вершини, а їхні з’єднання як ребра. Залежно від вибраного кореневого походження можна створити різні DAG для потреб «розумних» міських застосунків.

2.5 Асинхронний режим зв'язку агента «розумних» міських застосунків

В асинхронному режимі зв’язку агенти «розумних» міських застосунків відстежують стан похідних даних у графі знань і виконують будь-які запитані завдання. Коли агент «розумних» міських застосунків виявляє похідні дані зі статусом «Запитано», він спочатку перевіряє, чи всі залежності даних для цих похідних задоволені, використовуючи запит. Якщо вимоги задовольняються, агент «розумних» міських застосунків отримує вхідні дані з графу знань, записує поточну позначку часу та змінює статус похідного походження на «InProgress». Потім вхідні дані «розумних» міських застосунків передаються в метод, наданий розробником, і перетворюються на вихідні дані. На цьому етапі статус деривації буде змінено на «Завершено», а щойно отримані виходи тимчасово підключено до нього. Цей статус означає різницю між завданням «розумних» міських застосунків, яке виконано, але все ще потребує постобробки чи очищення, і завданням, яке завершено в тому сенсі, що воно не потребує подальших дій. Він використовується, щоб запобігти одночасному виконанню одних і тих самих завдань «розумних» міських застосунків з очищення декількома агентами, і видаляється, коли підграф деривації

прибирається протягом наступного запланованого періоду моніторингу. Процес очищення включає видалення старих екземплярів, з’єднання нових екземплярів з оригінальною деривацією та будь-якими наявними подальшими деривaціями, видалення всієї інформації про стан «розумних» міських застосунків і, нарешті, оновлення мітки часу деривації, щоб підграф деривації залишався актуальним. Моніторинг виконується через запланований проміжок часу, і його частоту може визначати користувач, що водночас є розробником «розумних» міських застосунків. Якщо під час будь-якої операції виникає помилка, агент змінює статус похідного походження на «Помилка» та записує трасування винятку.

2.6 Паралельність і багатопотоковість агента «розумних» міських застосунків

Будучи децентралізованою системою за дизайном, цифровий двійник реального світу «розумних міст» містить обширний перелік агентів, які одночасно працюють над графом знань. У ситуаціях, коли декілька агентів «розумних» міських застосунків запитують оновлення однієї дочірньої версії, відповідний агент повинен правильно й ефективно обробляти одночасні запити.

Наприклад, така ситуація виникає, коли доступ до примірників «i1» та «i2», зображених у загальній формі DAG на рисунку 2.2, здійснюється одночасно. У режимі синхронного зв’язку поточна реалізація ґарантує, що дубльована інформація не додається до графу знань завдяки використанню оновлення SPARQL, описаного як запит.

В асинхронному режимі зв’язку агент використовує властивість даних «uuidLock», щоб ідентифікувати та заблокувати потік, який зараз обробляє похідне, уникаючи повторного виконання. Ці заходи гарантують правильну обробку паралелізму без шкоди для високої пропускної здатності багатопоточності, прокладаючи шлях для розгортання розподіленого агента «розумних» міських застосунків.

2.7 Клієнт деривації «розумних» міських застосунків

Створивши онтологію для захоплення процесу деривації та шаблон агента «розумних» міських застосунків для виконання оновлення деривації, представимо клієнт деривації, здатний керувати підграфами деривації. Це передбачає визначення, чи похідні дані застаріл. Якщо так, то буде виконано запит на оновлення даних «розумних» міських застосунків.

Тому доцільно розглянути випадки, коли оновлення підграфа похідних даних «розумних» міських застосунків обробляються за допомогою різних режимів зв’язку, а саме:

– пріоритетного синхронного;
– пріоритетного асинхронного;
– змішаного типу.

Для кожного випадку доцільно спочатку розглянути загальний алгоритм, а потім навести приклади для опису очікуваного результату.

2.8 Режим пріоритетного синхронного оновлення клієнта «розумних» міських застосунків

Визначення своєчасності кожної деривації «розумних» міських застосунків та виконання необхідних оновлень у деривaційному підграфі є рекурсивним процесом.

З огляду на будь-який екземпляр похідної, з якого походить інформація, до якої здійснюється доступ, структура розглядає її як кореневу, проходить вгору аж до похідної інформації, яка походить від вихідної інформації «розумних» міських застосунків, тобто всі вхідні дані якої не є похідними ні з чого, і, нарешті, оновлює похідні в зворотному порядку. Це можна описати як алгоритм пошуку в глибину (DFS), поданий лістингу 2.1 [21].

Лістинг 2.1 – Алгоритм синхронного оновлення похідних інформаційних сутностей [21]

```
Input: IRI of the root derivation instance
Result: The root derivation and all its upstream derivations are
updated if deemed outdated
Create an empty directed acyclic graph G;
Cache root derivation d and all its upstream derivations
recursively in G;
updateSyncDerivation(d, G);
Function updateSyncDerivation(d, G):
   U ← d.upstreams() ; /* get immediate upstream derivations */
   if vertex d ∉ G.vertices then
      Add d as vertex in G;
   end
   for u ∈ U do
      if vertex u ∉ G.vertices then
         visited_u ← false;
         Add u as vertex in G;
      else
         visited_u ← true;
      end
      if edge(d, u) ∉ G.edges then
         traversed_d,u ← false;
         Add (d, u) as edge in G ;
         /*will throw error if circular dependency detected */
      Else
         traversed_d,u ← true;
      end
      if visited_u == false and traversed_d,u == false then
         updateSyncDerivation(u, G);
      end
   end
   Fire request to update d if it is deemed outdated;
   Update outputs of d in cache;
```

Рисунок 2.5 ілюструє помітну деталь алгоритму з лістингу 2.1, який використовує DAG G для відстеження обходу графа [21]. Для цього алгоритм додає екземпляри похідних як вершини та зв’язки між ними як спрямовані ребра до G. Залежно від похідного кореня, алгоритм DFS може призвести до двох версій G, пропускаючи паралельні гілки, які не потребують оновлення.

Оновлення виконується лише для похідних у результуючому графі під час зворотного відстеження алгоритму DFS. Слід також зазначити, що обчислення триває лише тоді, коли ні вузол, ні ребро раніше не видно.

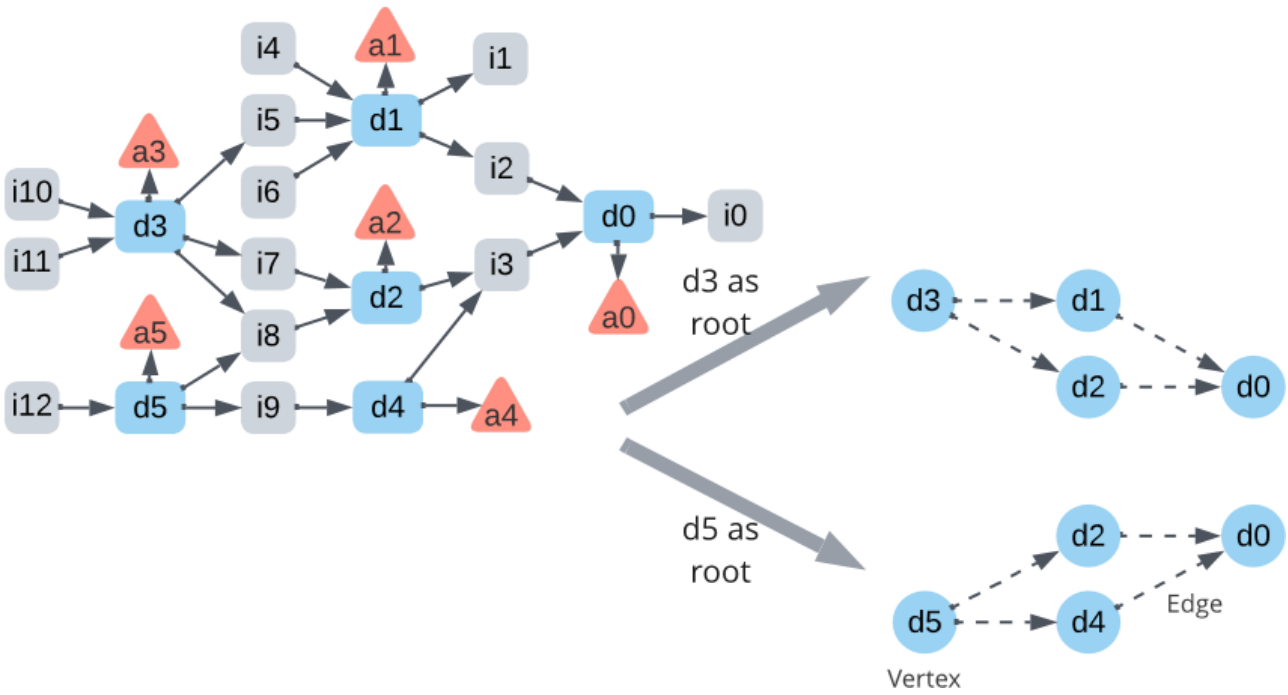

Рисунок 2.5 – DAG використовуються в пам’яті для підтримки зворотного відстеження алгоритму DFS під час оновлення підграфа виведення [21]

Похідні екземпляри розглядаються як вершини, а їхні з’єднання як ребра. Залежно від вибраного кореневого походження можна створити різні DAG. Наприклад, незалежно від того, яка гілка пройдена першою, деривація d1 або d2 у верхньому результуючому графі, деривація d0 відвідується лише вперше під час розгалуження [21]. Ця конструкція ґарантує, що релевантна інформація вгорі відвідується лише один раз, щоб уникнути дублювання роботи.

Рисунок 2.6 [21] ілюструє найпростішу форму оновлення деривації даних «розумних» міських застосунків, тобто оновлення однієї синхронної деривації.

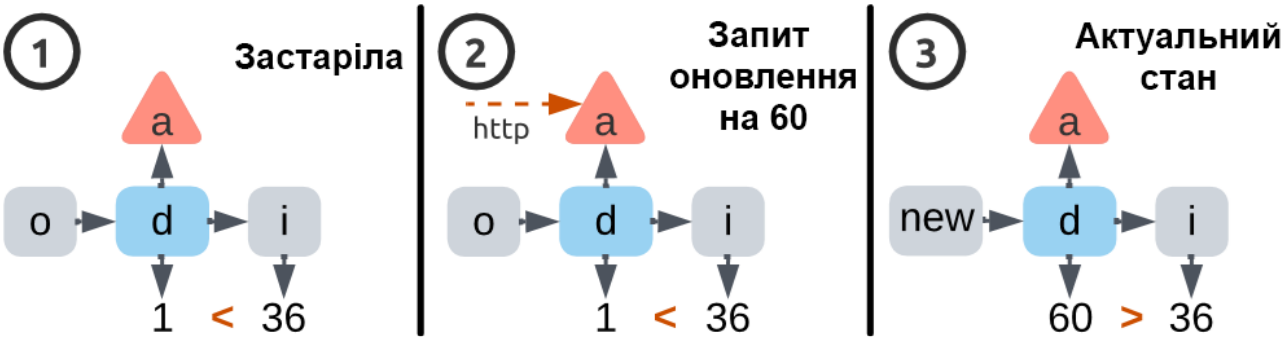

Рисунок 2.6 – Процес оновлення однієї синхронної деривації [21]

Для демонстрації розглянемо спрощене представлення виведення, виражене на рис. 2.1 у мітці часу «36», як початкову точку. Як зазначалося вище, ці похідні дані вважається застарілими, якщо порівняти їх позначку часу з міткою часу вхідних даних. Припускаючи, що вихідна інформація доступна на «60», структура запускає запит на оновлення до агента, пов'язаного з екземпляром похідного походження. Отримавши запит на оновлення, агент негайно починає обчислення та оновлює вихідну сутність у графі знань «розумного міста» новою отриманою інформацією. Таким чином, екземпляр деривації «розумного» міського застосунку є актуальним, а результати повертаються.

Щоб розширити цей приклад до трохи складнішої ситуації, розглянемо доступ до інформації «I2» у лінійному ланцюжку двох синхронних похідних інформації, як показано на рисунку 2.7.

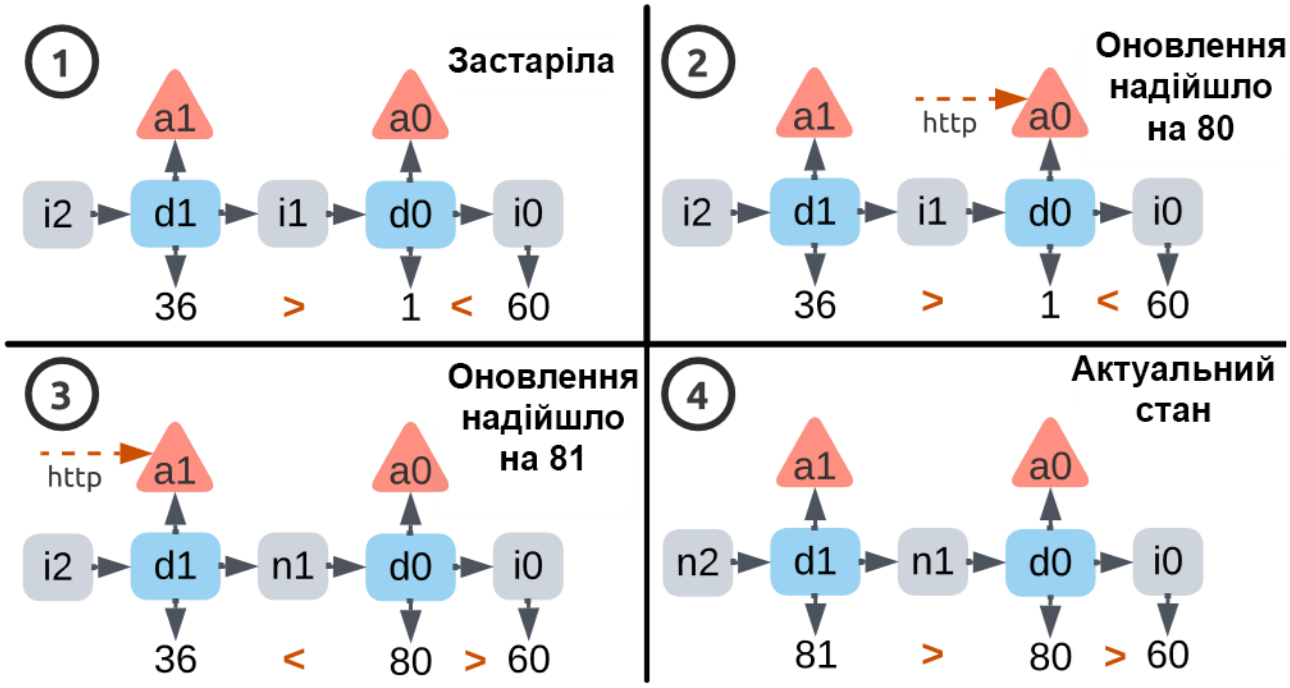

Рисунок 2.7 – Процес оновлення похідної DAG, що складається лише з синхронних похідних структур даних [21]

У цьому прикладі вхідна інформація похідної «d1» належить похідній «d0», отже, своєчасність «d1» визначається шляхом порівняння його з

деривацією «d0». Однак просто порівняння часових позначок цих двох інформаційних елементів призведе до неправильного висновку, що «d1» є актуальним. Навпаки, «d1» слід вважати застарілим, оскільки він залежить від застарілої деривації.

Таким чином, у ситуації оцінки своєчасності похідних даних, які залежать від отриманої інформації «розумних» міських застосунків, структура спочатку визначає своєчасність вихідних даних «d0» перед виконанням будь-яких дій щодо вихідних даних «d1». Для представленого прикладу, припускаючи, що доступ до інформації «I2» здійснюється в момент часу «80», структура спочатку оновлює «d0», а потім відразу після цього «d1». З новими виходами, пов’язаними в графі знань «розумного міста», обидва похідні інформаційні набори будуть розглядатися як актуальні.

2.9 Режим пріоритетного асинхронного оновлення клієнта «розумних» міських застосунків

Наприклад, у сфері наукових обчислень для потреб «розумних міст» часто виникають ситуації, коли потрібні масштабні обчислення на основі вихідних та вхідних даних вимагають тривалого періоду часу, перш ніж результати стануть доступними. Асинхронне оновлення інформаційних сутностей, придатне для таких ситуацій. Алгоритм асинхронного оновлення (див. лістинг 2.2) подібний до попереднього алгоритму, за винятком того, що алгоритм для асинхронних похідних не кешує підграф похідних даних «розумних» міських застосунків.

Лістинг 2.2 – Алгоритм асинхронного оновлення похідних інформаційних сутностей [21]

```
Input: IRI of the root derivation instance
Result: The root derivation and all its upstream derivations are
requested for update if deemed outdated
Create an empty directed acyclic graph G;
```

```
Query derivation instance d from the KG using the given
rootDerivationIRI;
updateAsyncDerivation(d, G);
Function updateAsyncDerivation(d, G):
   Query the list U of all immediate upstream derivations of d;
   if vertex d ∉ G.vertices then
      Add d as vertex in G;
   end
   for u ∈ U do
      if vertex u ∉ G.vertices then
         visited_u ← false;
         Add u as vertex in G;
      else
         visited_u ← true;
      end
      if edge (d, u) ∉ G.edges then
         traversed_{d,u} ← false;
         Add (d, u) as edge in G ;
         /* will throw error if circular dependency detected */
      else
         traversedd,u ← true;
      end
      if visited_u == false and traversedd,u == false then
         updateAsyncDerivation(u, G);
      end
   end
   Mark d as Requested if it is deemed outdated;
```

Швидше, безпосередні висхідні похідні дані запитуються на льоту, і, отже, їх своєчасність визначається виключно на основі запитів «розумних» міських застосунків до графа знань у режимі реального часу. Метою цього дизайну є врахування того факту, що через відносно довгі часові масштаби будь-яка інформація в підграфі асинхронних похідних «розумних» міських застосунків може змінюватися під час виконання процесу оновлення.

Як показано на рисунку 2.8, починаємо з моменту часу, коли похідні дані «розумних» міських застосунків щойно створено [21], тобто асинхронні похідні ініціалізуються зі статусом «Запитано» та міткою часу 0, без обчислення результату.

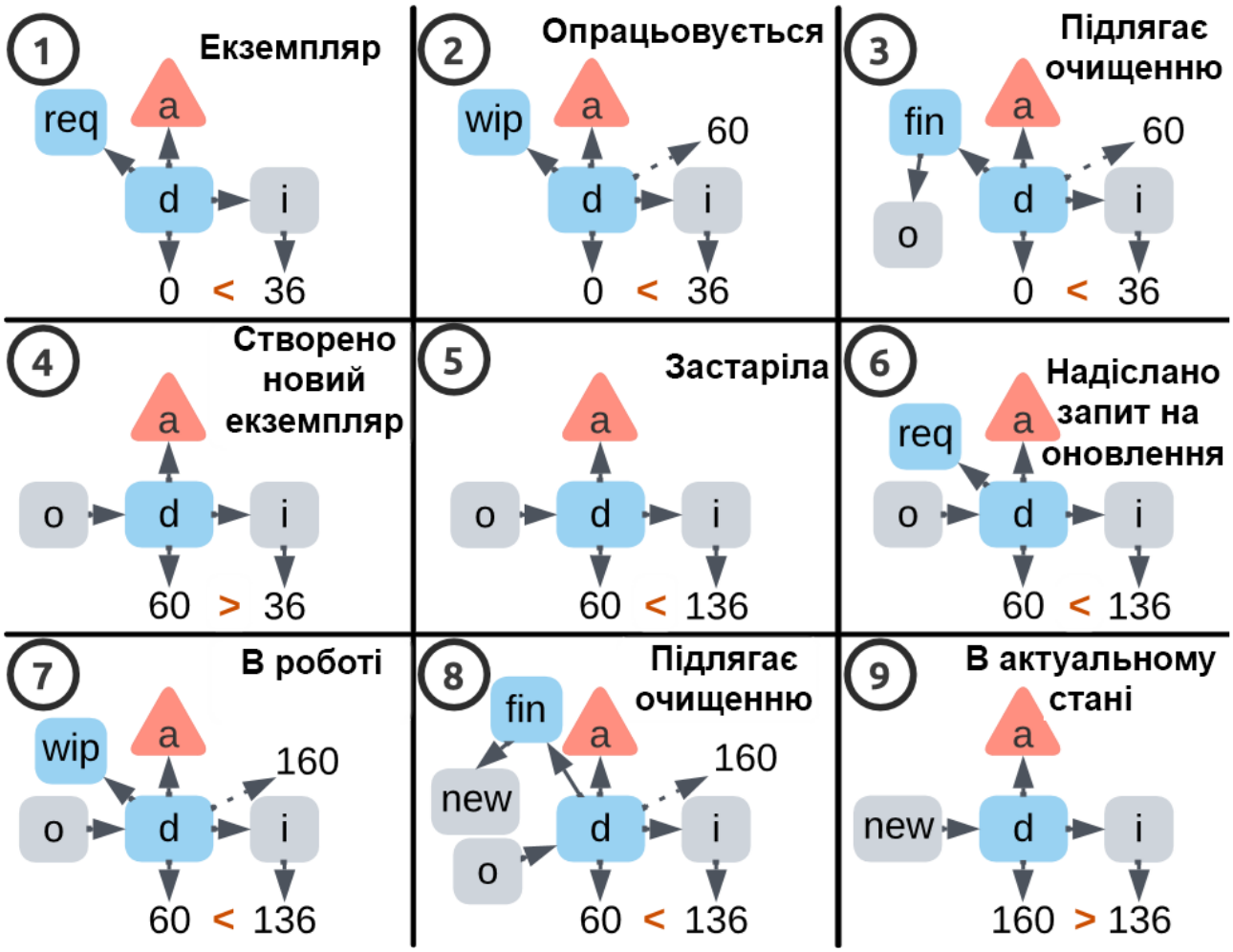

Рисунок 2.8 – Процес оновлення однієї асинхронної деривації [21]

Цілі числа, додані до екземплярів похідних за допомогою пунктирних стрілок, позначають мітки часу, записані властивістю даних retrievedInputsAt.

Фактичне оновлення асинхронної деривації «розумних» міських застосунків виконуватиметься агентом деривації, і воно не відбувається одночасно з запит на це оновлення. Оскільки агент «розумних» міських застосунків періодично перевіряє статус похідних даних, отриманих за допомогою самого себе, запитана похідна інформаційна сутність буде перетворено на «InProgress» під час наступного запуску, а позначка часу, коли вхідні дані були зчитані, буде записана. Успішне завершення завдання буде відображено в його статусі «Завершено». Потім агент «розумних» міських застосунків з’єднає згенерований вихід із екземпляром деривації, оновить мітку часу й, нарешті, повністю видалить статус. Процес оновлення подібний до

початкового обчислення. Єдина відмінність полягає в тому, що агент виконуватиме зіставлення екземплярів під час повторного підключення нової отриманої інформації до існуючих низхідних похідних у підграфі похідних даних «розумних» міських застосунків.

Цілі числа, додані до екземплярів похідних даних «розумних» міських застосунків за допомогою пунктирних стрілок, позначають мітки часу, записані властивістю даних «retrievedInputsAt».

Той самий приклад можна розширити до лінійного ланцюга з двох асинхронних похідних наборів даних, створених лише з одним фрагментом вхідних даних, як показано на рисунку 2.9.

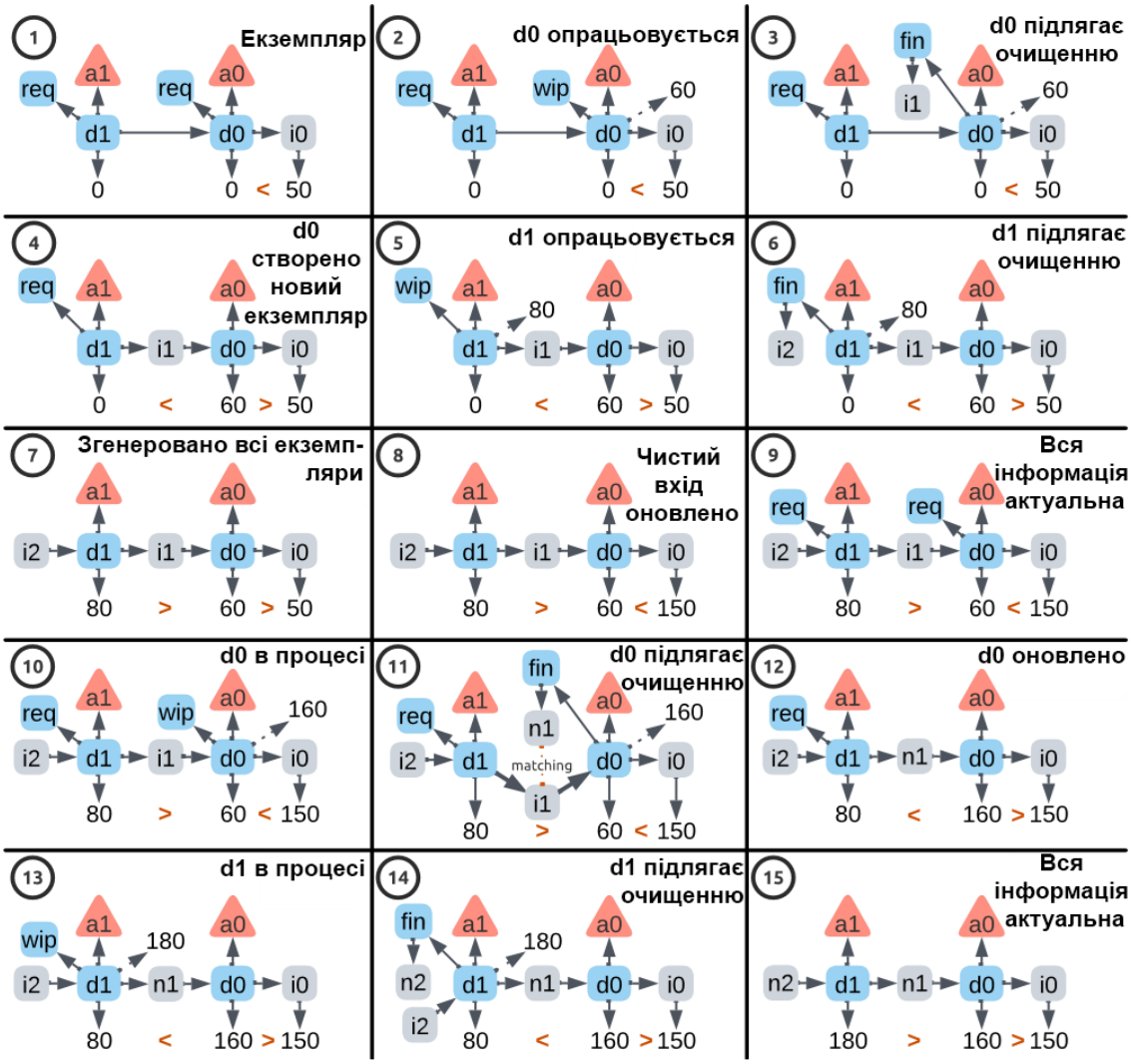

Рисунок 2.9 – Процес оновлення похідної DAG, що складається лише з асинхронних похідних [21]

У цьому випадку, оскільки жоден із виходів «розумних» міських застосунків не обчислюється, вихідний вихід «d1» безпосередньо позначається як «isDerivedFrom» його висхідного похідного «d0». Агент «розумних» міських застосунків, відповідальний за «d0», працюватиме так само, як згадано вище, з тією лише різницею, що агент повторно з’єднає новий похідний екземпляр «i1» як вхідні дані для похідного «d1» та усуне прямий зв’язок між двома похідними «розумних» міських застосунків. Подібно до наукового робочого процесу, агент, який керує вихідним процесом «d1», може розпочати його виконання лише після успішного завершення його попередників.

Таким чином, блоки з першого по сьомий на рисунку 2.9 демонструють одне бажане використання структури деривації для автоматичного завершення попередньо визначеного робочого процесу на основі вхідних даних «розумних» міських застосунків. Після того, як усі похідні екземпляри обчислено, проілюструємо кроки для оновлення похідних у блоці з восьмого по п’ятнадцятий, де оновлюються вихідні вхідні дані «розумних» міських застосунків. За запитом алгоритм проходить ланцюжок похідних даних і визначає своєчасність похідних даних «розумних» міських застосунків. На відміну від синхронного оновлення, алгоритм лише позначає похідні як запитувані, залишаючи фактичне оновлення окремим агентам «розумних» міських застосунків таким самим чином, як зазначено вище.

2.10 Висновок до другого розділу

В другому розділі кваліфікаційної роботи описано елемент онтології «OntoDerivation ABox». Висвітлено підключення «розумних» міських застосунків до «OntoAgent». Розглянуто агент деривації «розумних» міських застосунків. Проаналізовано синхронний режим зв'язку агента «розумних» міських застосунків. Висвітлено асинхронний режим зв'язку агента «розумних» міських застосунків. Описано паралельність і багатопотоковість агента «розумних» міських застосунків. Розглянуто клієнт деривації «розумних»

міських застосунків. Проаналізовано режим пріоритетного синхронного оновлення клієнта «розумних» міських застосунків. Подано аналіз режиму пріоритетного асинхронного оновлення клієнта «розумних» міських застосунків.

РОЗДІЛ 3. РОБОТА З ГРАФАМИ ЗНАНЬ ЗАСТОСУНКІВ «РОЗУМНИХ МІСТ»

3.1 Оновлення змішаного типу клієнтів «розумних» міських застосунків

Розглянемо приклад похідного підграфа, що складається з похідних змішаного типу «розумних» міських застосунків. Зокрема, асинхронні похідні даних «розумних» міських застосунків залежать від синхронних похідних, тобто тривале обчислення покладається на результати швидких обчислень.

Варто зазначити, що інший шлях призведе до суто асинхронного сценарію. Якщо синхронна деривація залежить від асинхронної, то синхронна деривація фактично стає асинхронною через необхідність довго чекати відповіді «розумних» міських застосунків. З цієї причини випадок асинхронних похідних залежно від синхронних є єдиним змішаним випадком, який необхідно розглянути без втрати загальності «розумних» міських застосунків.

Як було показано раніше, потрібно визначити, чи похідні дані застаріли, і виконати запит оновлення. Ці дві частини поєднуються та виконуються в рекурсивному алгоритмі для підграфів похідних, які складаються лише з синхронних похідних даних «розумних» міських застосунків [21]. Однак, щоб заощадити обчислювальні ресурси для оновлення підграфів похідних змішаного типу, доцільно спроектувати структуру, яка б працювала таким чином, щоб оновлення синхронних похідних вихідних даних «розумних» міських застосунків обчислювалося лише тоді, коли агент оновлює асинхронні похідні. Таким чином, коли алгоритм рекурсивно визначає своєчасність асинхронних похідних даних «розумних» міських застосунків, розмітка необхідна в графі знань, щоб вказати, що нижча асинхронна похідна фактично застаріла і, отже, має бути позначена як «Запитана».

Уніфікований метод надається як функція-обгортка двох раніше представлених алгоритмів. Залежно від інстанційованого типу виведення

кореня функція вибирає інший алгоритм входу [21]. Рисунок 3.1 демонструє життєвий цикл такого прикладу.

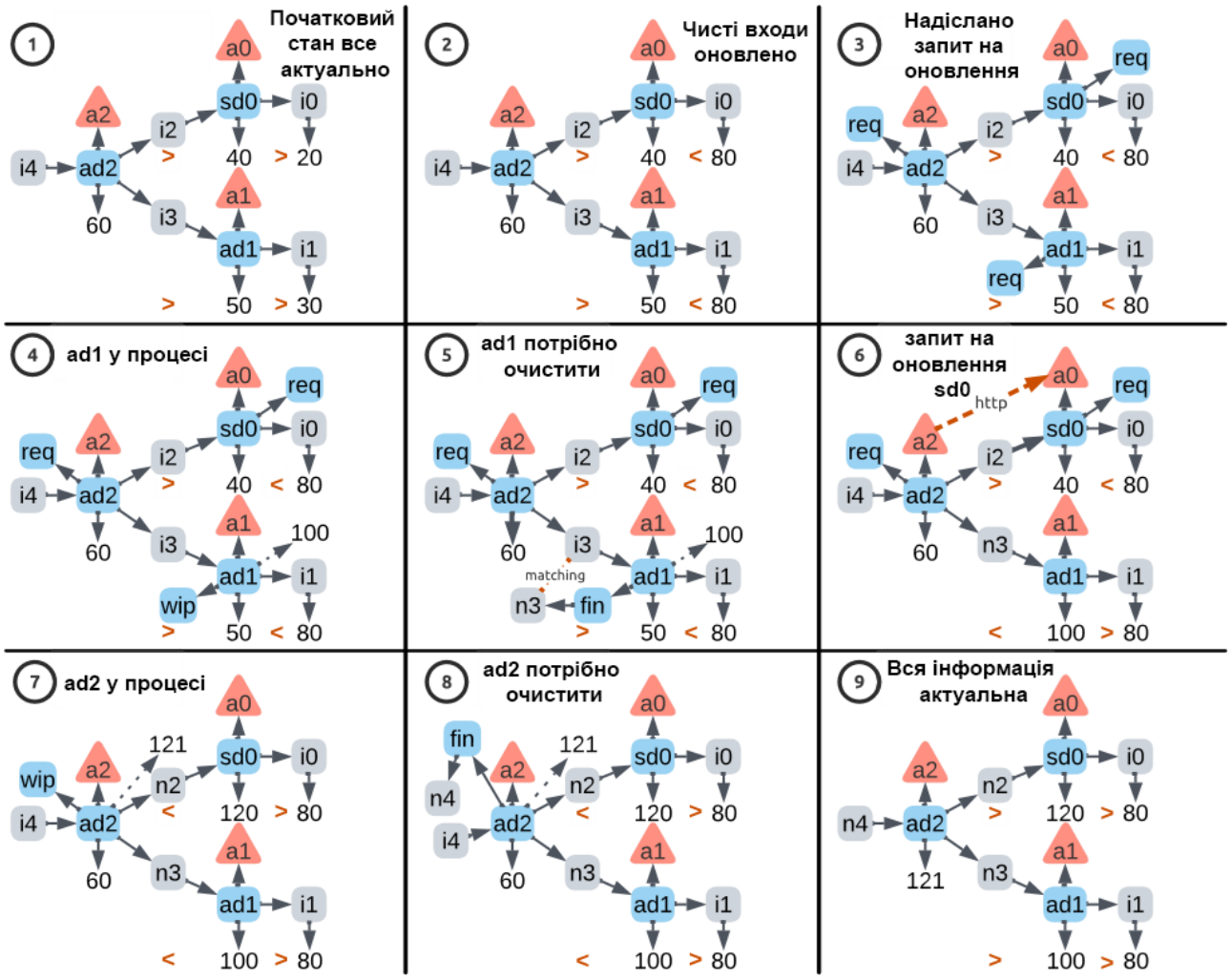

Рисунок 3.1 – Процес оновлення похідної DAG, що складається з асинхронних похідних, які залежать від синхронних похідних [21]

Синхронне виведення «sd0» буде позначено як «Запитане» під час проходження алгоритмом, що служить сигналом, коли алгоритм визначає своєчасність асинхронного виведення «ad2» за потоком «розумних» міських застосунків. Агент моніторингу «ad2» запитає оновлення для «sd0» і розпочне його виконання після того, як він підтвердить, що всі його безпосередні вихідні дані «розумних» міських застосунків асинхронні похідні є актуальними.

Три частини каркаса похідної інформації «розумних» міських застосунків, які працюють разом у згуртованому вигляді:

– онтологія похідного;

– агент похідного;

– клієнт похідного.

Основна мета проведеного в кваліфікаційній роботі полягає в тому, щоб уся інформація про походження була чітко задокументована в графі знань «розумного міста», фіксуючи:

– деталі виконаних обчислень;

– їх походження;

– використовувані функції-агенти.

Це забезпечує часову узгодженість, тобто впорядкування часових позначок подій «розумних» міських застосунків. Таким чином, його правильність можна перевірити за допомогою наданої функції «validateDerivations», яка ґарантує відсутність циклічних залежностей і кожен екземпляр чисті вхідні та похідні дані «розумних» міських застосунків має дійсну позначку часу.

Крім того слід включити тестові випадки, щоб охопити ключові функції «розумних» міських застосунків. Існують також мінімальні робочі приклади як на Java, так і на Python як навчальні посібники для програмістів новачків. Щоб отримати ці ресурси, слід звернутися до відкритого репозиторію «TheWorldAvatar» на «GitHub» і «PyPI» [21]. Така прозорість полегшує перевірку точності розробленої основи.

3.2 Автоматизоване заповнення та оновлення підграфа виведення «розумних» міських застосунків

Для оцінювання результатів аналізу використаємо дані з різних джерел, зокрема:

– прикладні програмні інтерфейси (API) моніторингу повеней у режимі реального часу Агентства охорони навколишнього середовища [39];

– сертифікати енергоефективності (EPC);

– відкриті дані земельного кадастру [40].

Програмно-алгоритмічні агенти введення регулярно створюють і оновлюють ці дані в графі знань цифрового двійника реального світу «розумних міст». Використовуючи вихідну інформацію, оцінка впливу включає різні агенти похідних, які працюють разом, щоб заповнити підграф похідних даних «розумних» міських застосунків. Цей процес охоплює два типи дій на похідному підграфі:

– створення нової отриманої інформації, такої як вплив нещодавно виданого попередження про повінь;

– оновлення існуючої інформації «розумних» міських застосунків, такої як оновлений вплив існуючого попередження про повінь, коли частина вихідної інформації є оновлено.

Один із цих програмно-алгоритмічних агентів, «Flood Assessment Agent», розраховує вплив повені, ідентифікуючи «розумні» будівлі, розташовані в зоні впливу, і визначаючи загальну ринкову вартість майна «розумного міста», що знаходиться під загрозою. Вартість нерухомості кожної «розумної» будівлі оцінюється шляхом масштабування останніх записів про транзакції на основі місцевого індексу цін на нерухомість або шляхом множення її площі на середню ціну квадратного метра для її поштового індексу [21]. Ця репрезентативна середня ціна може бути обчислена «розумним» агентом цін за середній квадратний метр з урахуванням усіх останніх записів транзакцій у «розумному» регіоні.

Рисунок 3.2 ілюструє прогрес у підграфі виведення при оцінці потенційного впливу виданого попередження про повінь в «розумному місті».

Сутності з низькою непрозорістю представляють існуючі сутності в графі знань «розумного міста», тобто додані на попередніх етапах роботи агентів до того, як агент додасть щойно отриману інформацію «розумних» міських застосунків. Цілі числа, додані до сутностей, позначають типові часові мітки, коли ця інформація була додана до графу знань «розумного міста» [21].

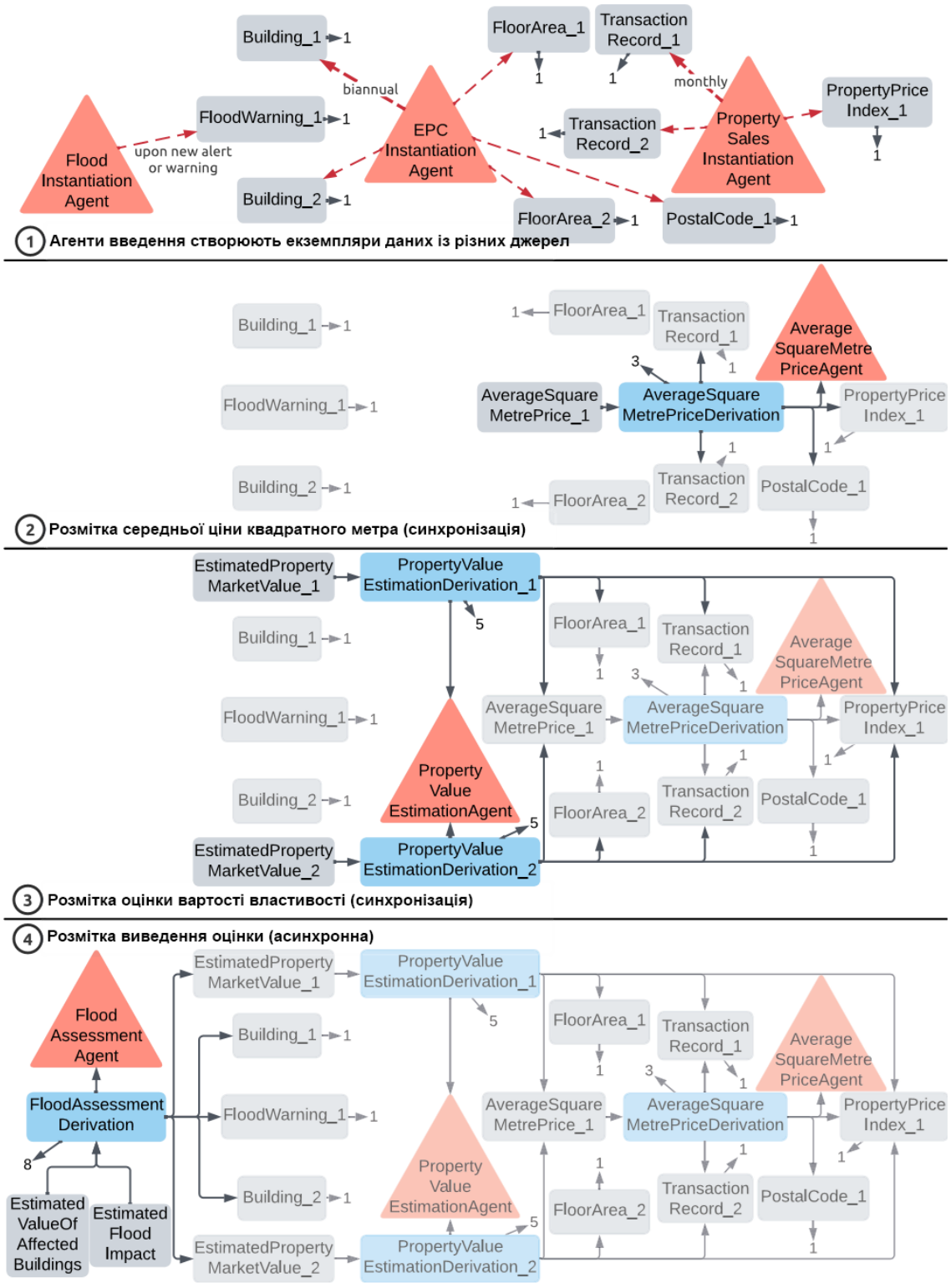

Рисунок 3.2 – Процес заповнення деривaційного підграфа для оцінки впливу повені [21]

Як позначено червоною пунктирною стрілкою, екземпляр попередження про створюється агентом створення «розумного» міського застосунку. Він визначає інформацію про очікувану серйозність та конкретний геопросторовий

масштаб «розумного міста», який знаходиться під загрозою. У цьому спрощеному прикладі передбачається, що геопросторовий багатокутник охоплює поштовий індекс, який включає дві «розумні» будівлі, кожна з яких містить дані про площу приміщення та його останній запис транзакції. Крім того, індекс цін на нерухомість у «розумній локації» [21], який фіксує зміни у вартості житлової нерухомості, створюється як часовий ряд на графіку знань і оновлюється щомісяця.

Після розгортання агентів «розумних» міських застосунків виведення похідної інформації починається з розмітки виведення середньої ціни квадратного метра для всіх поштових індексів у «розумному регіоні». Далі для всіх «розумних» будівель робиться розмітка виведення оцінки вартості майна. Оскільки ці обчислення є відносно швидкими, вони позначаються як синхронні похідні для отримання миттєвих відповідей «розумних» міських застосунків. Виведення оцінки «розумних» міських застосунків розмічується після створення екземпляра попередження.

На практиці попередження «розумних» міських застосунків може охоплювати більше десятка поштових індексів із сотнями будівель. Його обчислення може зайняти деякий час, тому воно позначається як асинхронне виведення. Отже, відсутність статусу, пов’язаного з визначенням оцінки впливу «розумних» міських застосунків, вказує на те, що отримана інформація є актуальною, що означає завершення процесу оцінки. Усі похідні розмітки створюються програмним шляхом за допомогою ретельно перевірених SPARQL і геопросторових запитів, щоб забезпечити точність захоплених залежностей «розумного міста» (див. рисунок 3.3) [21].

Оскільки кожен програмно-алгоритмічний агент введення «розумних» міських застосунків працює на різних частотах, вплив може змінитися, коли вихідна інформація оновлюється, поки попередження все ще активне. Ці розмітки динамічно оновлюються агентами «розумних» міських застосунків для інтеграції нової доступної інформації.

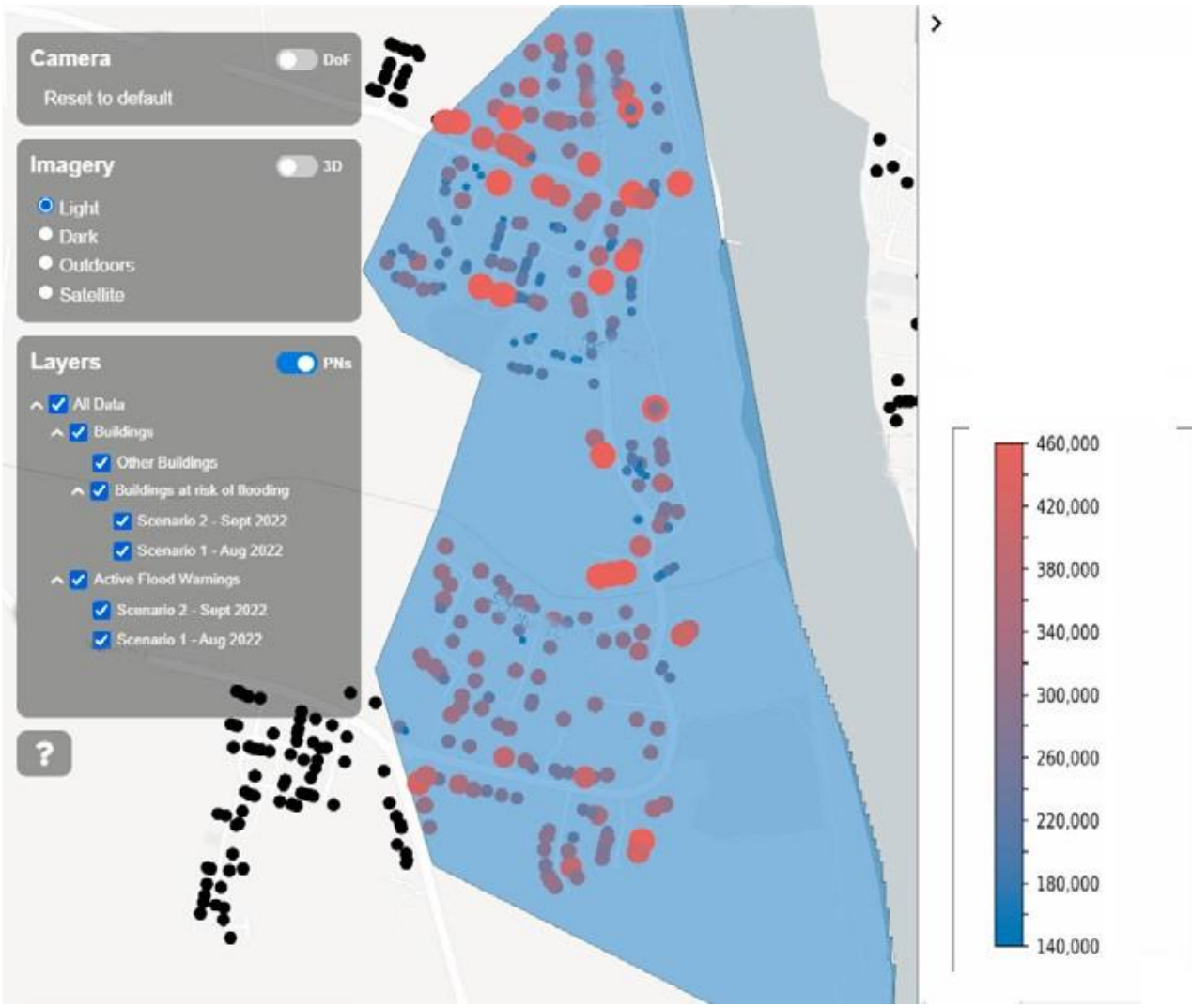

Рисунок 3.3 – Веб-інтерфейс, що візуалізує оцінку «розумних» міських застосунків з шарами для «розумних» будівель і попереджень [21]

Наприклад, індекси цін на нерухомість для всіх адміністративних округів «розумного міста» оновлюється щомісяця, і геопросторовий масштаб активного попередження про «розумних» міських застосунків також може змінюватися, що може призвести до застарілого впливу. Синя область позначає потенційну зону загрози «розумного міста». Колір і розмір точок у цій області відображають оціночну вартість власності. «Розумні» будівлі за межами прогнозного небезпечного регіону вважаються незачепленими загрозою, тому позначаються чорними крапками.

Як і оновлення будь-якої іншої похідної інформації, оцінка впливу розроблена таким чином, щоб здійснюватися за запитом «розумних» міських

застосунків, щоб заощадити обчислювальну потужність. Після запиту на оновлення структура деривації проходить деривaційний підграф «розумного міста», щоб перевірити, чи середня оцінка «розумних» міських застосунків актуальна, і якщо ні, вона повертається назад, щоб позначити деривації як застарілі, що запускає оцінку як «запитану». Згодом агент «розумних» міських застосунків ініціює оновлення шляхом прямого виклику програмно-алгоритмічного агента, тобто синхронно. Дані оновлюються в графі знань «розумного міста», який надалі використовується програмно-алгоритмічним агентом для оновлення оцінки. Лише коли вся вхідна інформація оновлена, агент оцінки повеней починає оцінку впливу. Це гарантує, що інформація, використана в оцінці, точно відображає реальний світ «розумного міста». Варіант використання демонструє обидва режими зв’язку похідної інформаційної структури «розумних» міських застосунків шляхом використання як синхронних, так і асинхронних похідних інформаційних структур.

3.3 Оцінювання онтології «розумного міста»

Процес оцінювання спродукованої в [21] онтології проводиться в три етапи . По-перше, відбувалось оцінювання «OntoDerivation TBox». Згодом були отримані в результаті тестів на масштабованість. Нарешті, проаналізована сумісність між «OntoDerivation» і PROV-O.

Для цієї оцінки використовувалися резонер «HermiT» [41] і плагін «OntoDebug» для редактора онтології «Protégé» [42]. «OntoDerivation» реалізує для «розумного міста» вісім класів, п’ять властивостей об’єктів і дві властивості даних. Резонер «HermiT» може класифікувати онтологію «розумного міста» «OntoDerivation». У режимі налагодження «OntoDebug» визначає онтологію як узгоджену та послідовну. «Protégé» не показує жодних помилок, пов’язаних із незаконним оголошенням сутностей «розумного міста»

або їх повторним використанням. Усі зміни в «OntoDerivation TBox» контролюються версіями в «GitHub» [21].

Рисунок 3.4 ілюструє лінійну масштабованість структури виведення для різної кількості перевірених «розумних» будівель.

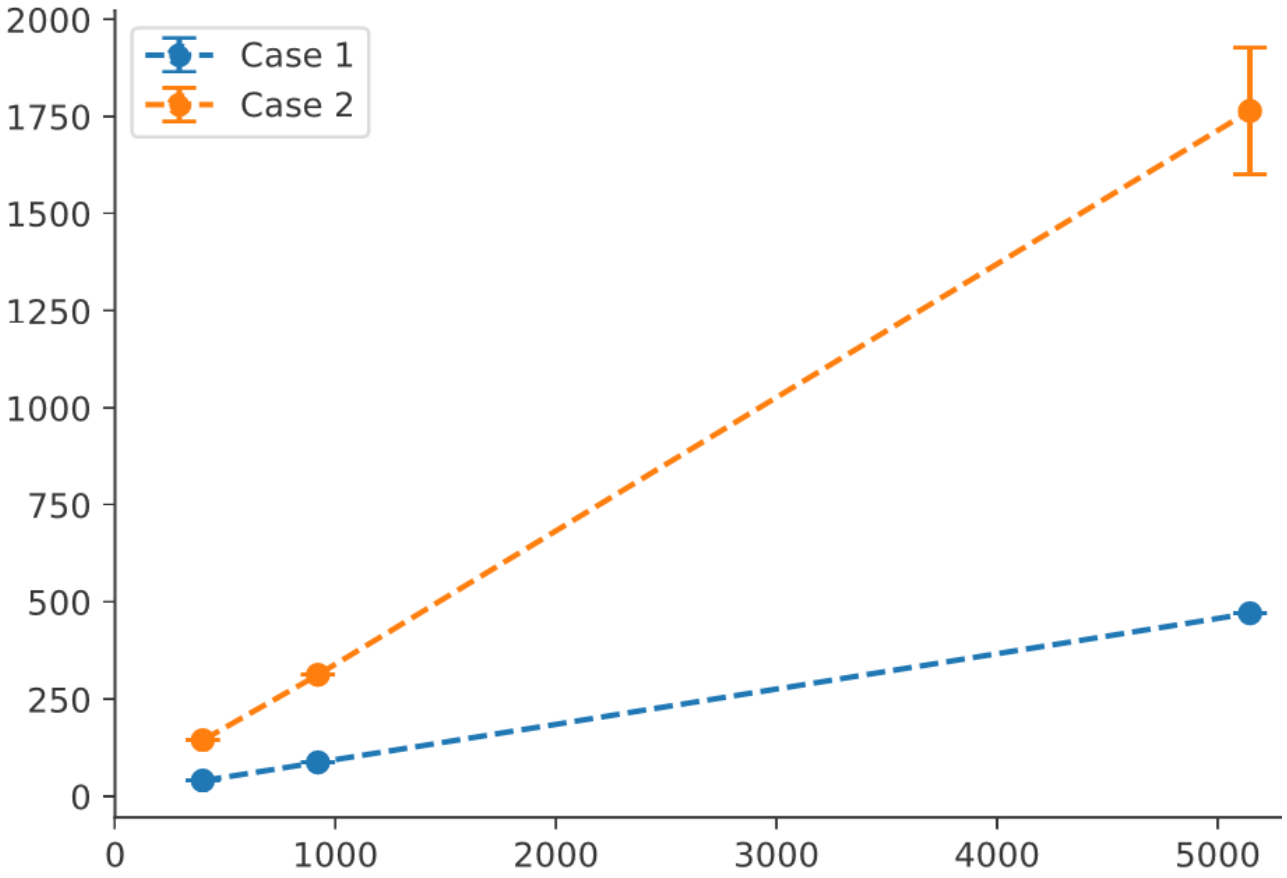

Рисунок 3.4 – Час обробки, коли структура виведення оновлює оцінку впливу для потенційних подій «розумного міста» з різною кількістю «розумних» будівель [21]

«Case 1» – створення лише нових попереджень щодо «розумних» будівель. «Case 2» – оновлення індексу «розумних» будівель з подальшим створенням нових попереджень «розумного міста». Однак слід зазначити, що фактична кількість часу обробки залежить від конкретного випадку використання «розумних» міських застосунків. Вираховуючи час обробки «Case 1» із «Case 2», можна помітити, що агенти обробляють приблизно чотири похідні запити «розумних» міських застосунків в секунду. Щоб оцінити приріст

продуктивності, досягнутий за допомогою розгортання кількох екземплярів певного програмно-алгоритмічного агента «розумних» міських застосунків, стає необхідним впровадження балансувальника навантаження, що, однак, виходить за рамки цієї роботи.

3.4 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи подано оновлення змішаного типу клієнтів «розумних» міських застосунків. Проаналізовано автоматизоване заповнення та оновлення підграфа виведення «розумних» міських застосунків. Описано процес оцінювання онтології «розумного міста».

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Ризик як кількісна оцінка небезпек

Динамічні графи знань можуть допомогти у виявленні та управлінні ризиками в міському середовищі. Наприклад, аналіз даних про дорожньо-транспортні пригоди, пожежі, природні катастрофи та інші небезпечні ситуації може бути використаний для розробки стратегій зниження ризиків. Розгляд ризику як кількісної оцінки небезпек надає можливість проводити більш детальний та об'єктивний аналіз потенційних загроз.

Нещасні випадки, аварії та катастрофи, що призводять до смертельних випадків, травм, скорочення тривалості життя, шкоди здоров’ю та довкіллю, є результатом прояву небезпек. Виникає проблема оцінки наслідків таких подій. Кількісна оцінка збитків, завданих небезпекою, залежить від багатьох факторів, таких як кількість людей у небезпечній зоні, кількість та якість матеріальних цінностей, що знаходилися там, та природних ресурсів.

Кожен вид шкоди має своє кількісне вираження, наприклад, кількість загиблих, поранених чи хворих, площа зараженої території, площа вигорілого лісу або вартість зруйнованих споруд. Перший спосіб кількісного визначення шкоди – це її вартісна оцінка у грошовому еквіваленті. Інша, універсальна та найбільш поширена оцінка небезпечності – це ризик, який можна назвати фактором потенційної небезпеки.

У тлумачному словнику ризик визначається як "усвідомлена можливість небезпеки". Проте більш точним визначенням є "усвідомлена ймовірність небезпеки". Ризик – це кількісна оцінка небезпеки, яка є відношенням кількості небажаних реалізованих наслідків (n) до максимально можливої їх кількості (N) за конкретний період часу: $R = n/N$, тобто це частота реалізації небезпек. Ризик супроводжує будь-яку діяльність людини [43].

Подана формула дозволяє розрахувати розміри індивідуального, групового та загального ризику. При оцінці загального ризику величина N

означає максимальну кількість усіх подій. У разі оцінки групового ризику, N представляє максимальну кількість подій у певній соціальній групі, вибраній за конкретною ознакою з загальної кількості. Такою групою можуть бути люди однієї професії, віку, статі або ж один клас суб'єктів господарської діяльності.

Для визначення індивідуального ризику, наприклад, ризику людини потрапити в транспортну аварію, слід поділити кількість постраждалих (наприклад, 46 000 осіб) на загальну кількість людей, що можуть травмуватися за рік (46 мільйонів осіб). Таким чином, ризик (N - фактична частота небезпечності на транспорті) буде складати 0,001 (46 000 : 46 000 000).

В охороні праці для характеристики рівня травматизму використовують коефіцієнт частоти (Кч), який показує кількість травмованих або загиблих на 1 000 працівників.

Ризик для суспільства можна класифікувати як знехтуваний, прийнятний, гранично допустимий і надмірний. Знехтуваний ризик настільки малий, що перебуває в межах природних відхилень. Прийнятний ризик - це рівень, який суспільство готове прийняти, враховуючи технічні, економічні та соціальні можливості на певному етапі розвитку. Гранично допустимий ризик - це максимальний рівень, який не можна перевищувати, незалежно від очікуваних результатів. Надмірний ризик має настільки високий рівень, що майже завжди призводить до негативних наслідків [44].

Досягнення нульового рівня ризику, тобто абсолютної безпеки, наразі неможливе. Абсолютна безпека не може бути гарантована жодній людині, незалежно від її способу життя чи соціального статусу. Ми живемо завдяки щоденному уникненню небезпек. Вимога абсолютної безпеки, хоч і приваблива своєю гуманністю, може мати трагічні наслідки. На сучасному етапі розвитку суспільства знехтуваний ризик також є недосяжним через технічні та економічні обмеження. Тому сучасна концепція безпеки життєдіяльності базується на досягненні допустимого ризику. Ідея концепції "допустимого ризику" полягає у створенні такого низького рівня ризику, який суспільство може прийняти в даний час. Допустимий ризик враховує технічні, економічні,

соціальні та політичні аспекти, будучи компромісом між рівнем безпеки та можливостями її забезпечення.

Для оцінки прийнятного ризику використовують витратний механізм, який дозволяє розподіляти державні витрати на забезпечення заданого рівня безпеки між природною, техногенною та соціальною сферами. Важливо підтримувати баланс витрат у цих сферах, оскільки порушення цього балансу може призвести до різкого зростання ризику, що виходить за межі допустимих значень.

Збільшуючи витрати на технічні потреби, можна зменшити ризик аварій на обладнанні, але при цьому зростає соціально-економічний ризик. Підвищення витрат на безпеку технічних систем, в умовах обмеженого бюджету, може призвести до занедбання соціальних сфер, наприклад, аварійно-рятувальних служб і медичної допомоги [45].

Оптимальне співвідношення витрат між технічною та соціальною сферами дозволяє мінімізувати загальний ризик. Максимально прийнятним рівнем індивідуального ризику загибелі людини вважається ризик, що дорівнює 10^{-6} на рік. Ризик загибелі людей, що дорівнює 10^{-8} на рік, вважається малим.

Вивчення причин виникнення небезпек, їх характеристик і особливостей впливу допомагає розробляти ефективні заходи захисту, спрямовані на забезпечення нормальної життєдіяльності людини. Основною метою безпеки життєдіяльності є підвищення рівня безпеки в усіх видах людської діяльності. При створенні нових пристроїв, обладнання, технічних систем, необхідно включати в проекти елементи, що виключають небезпеку настільки, наскільки це можливо. Якщо небезпеку повністю усунути не вдається, необхідно знизити ймовірність ризику до мінімуму шляхом вибору відповідного рішення.

Збільшення витрат на технічні потреби зменшує ризик аварій на обладнанні, але водночас підвищує соціально-економічний ризик. Підвищуючи витрати на безпеку технічних систем за умов обмеженого бюджету, можна занедбати соціальні сфери, наприклад, аварійно-рятувальні служби та медичну допомогу.

4.2 Особливості заходів електробезпеки на підприємствах

Дана кваліфікаційна робота присв’ячена аналізу динамічних графів знань для «розумних» міських застосунків. Враховуючи активне використання електронних та електричних пристроїв та обладнання при впровадженні інновацій у галузі, необхідно приділити увагу специфіці заходів електробезпеки на підприємствах. Сучасне виробництво нерозривно пов'язане з використанням електроенергії, тому питання електробезпеки набуває особливого значення.

Електробезпека є комплексом організаційних і технічних заходів та засобів, які захищають людей від шкідливого і небезпечного впливу електричного струму, електричної дуги, електромагнітного поля та статичної електрики. На підприємствах електробезпека забезпечується шляхом дотримання вимог, викладених у законодавчих актах [46].

– Правила безпечної експлуатації електроустановок споживачів (далі – ПБЕЕС), затверджені наказом Держнаглядохоронпраці від 09.01.1998 № 4, вимоги яких поширюються на працівників, що обслуговують діючі електроустановки споживачів напругою до 220 кВ включно і є обов'язковими для всіх споживачів та виробників електроенергії, незалежно від їх відомчої належності і форм власності на засоби виробництва.

– Правила безпечної експлуатації електроустановок, дія яких поширюються на працівників, що виконують роботи в електроустановках Міністерства енергетики України (наказ Держнаглядохоронпраці України від 06.10.1997 № 257).

– Правила технічної експлуатації електроустановок споживачів (ПТЕЕС), затверджені наказом Мінпаливенерго України від 25.07.2006 № 258 (у редакції наказу від 13.02.2012 № 91), якими унормовано організаційні й технічні вимоги щодо експлуатації електроустановок споживачів.

– Правила експлуатації електрозахисних засобів, затверджені наказом Міністерства праці та соціальної політики України від 05.06.2001 № 253, в яких наведено перелік засобів захисту, вимоги до них, обсяги і норми випробувань,

порядок застосування, зберігання їх, а також норми комплектування засобами захисту електроустановок і виробничих бригад.

– Правила улаштування електроустановок (ПУЕ), які визначають будову, принципи улаштування, особливі вимоги до окремих систем, їх елементів, вузлів і комунікацій електроустановок. Наказ Мініністарства енергетики та вугільної промисловості України від 24.07.2017 № 476.

– ДСТУ 2843-94 «Електротехніка. Основні поняття. Терміни та визначення», який установлює терміни та визначення основних понять в галузі електротехніки.

– Правила пожежної безпеки в Україні, затверджені наказом МВС від 30.12.2014 № 1417.

Побутові електроприлади в умовах виробництва експлуатують відповідно до експлуатаційної документації підприємств-виробників і ПТЕЕС, п. 1.5 розд. I ПТЕЕС. Відповідальність за організацію безпечної експлуатації електроустановок ПБЕЕС покладають на роботодавця, який повинен [47]:

– призначити відповідального за справний стан і безпечну експлуатацію електроустановок;

– створити і укомплектувати електротехнічну службу з числа осіб, які досягли 18-річного віку, мають відповідну освіту та пройшли медичний огляд і не мають протипоказань;

– розробити і затвердити Положення про енергетичну службу підприємства, посадові інструкції працівників та інструкції з безпечного виконання робіт;

– забезпечити навчання і перевірку знань працівників, своєчасний огляд електроустановок, проведення профілактичних, протиаварійних та приймально-здавальних випробувань;

– встановити такий порядок, щоб працівники, на яких покладено обов’язки з обслуговування електроустановок, вели ретельні спостереження за дорученим їм обладнанням і мережами.

4.3 Висновок до четвертого розділу

В четвертому розділі кваліфікаційної роботи описано ризик як кількісну оцінка небезпек. Також розглянуто особливості заходів електробезпеки на підприємствах.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Розглянуто стан досліджень в галузі динамічних графів знань для «розумних міст».

– Описано походження даних для робочих процесів «розумного міста».

– Подано опис цифрових двійників реального світу.

– Розглянуто цифрові двійники реального світу «розумних міст».

– Висвітлено онтологія деривації даних «розумних» міських застосунків.

– Описано елемент онтології «OntoDerivation TBox».

В другому розділі кваліфікаційної роботи:

– Описано елемент онтології «OntoDerivation ABox».

– Висвітлено підключення «розумних» міських застосунків до «OntoAgent».

– Розглянуто агент деривації «розумних» міських застосунків.

– Проаналізовано синхронний режим зв'язку агента «розумних» міських застосунків.

– Висвітлено асинхронний режим зв'язку агента «розумних» міських застосунків.

– Описано паралельність і багатопотоковість агента «розумних» міських застосунків.

– Розглянуто клієнт деривації «розумних» міських застосунків.

– Проаналізовано режим пріоритетного синхронного оновлення клієнта «розумних» міських застосунків.

– Подано аналіз режиму пріоритетного асинхронного оновлення клієнта «розумних» міських застосунків.

В третьому розділі кваліфікаційної роботи:

– Подано оновлення змішаного типу клієнтів «розумних» міських застосунків.

– Проаналізовано автоматизоване заповнення та оновлення підграфа виведення «розумних» міських застосунків.

– Описано процес оцінювання онтології «розумного міста».

У розділі «Безпека життєдіяльності, основи охорони праці» описано ризик як кількісну оцінка небезпек. Також розглянуто особливості заходів електробезпеки на підприємствах.

ПЕРЕЛІК ДЖЕРЕЛ

1 N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, J. Taylor, Industry-scale knowledge graphs: Lessons and challenges, Commun. ACM 62 (8) (2019) 36–43, http://dx.doi.org/10.1145/3331166.

2 P. Hitzler, A review of the semantic web field, Commun. ACM 64 (2) (2021) 76–83, http://dx.doi.org/10.1145/3397512.

3 A. Hogan, E. Blomqvist, M. Cochez, C. D'Amato, G.D. Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier, A.-C.N. Ngomo, A. Polleres, S.M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, A. Zimmermann, Knowledge graphs, ACM Comput. Surv. 54 (4) (2022) 1–37, http://dx.doi.org/ 10.1145/3447772.

4 J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, C. Bizer, DBpedia – A large-scale, multilingual knowledge base extracted from wikipedia, Semant. Web 6 (2) (2015) 167–195, http://dx.doi.org/10.3233/SW-140134.

5 T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, L. Pintscher, From freebase to wikidata: The great migration, in: Proceedings of the 25th International Conference on World Wide Web, 2016, pp. 1419–1428, http://dx.doi.org/ 10.1145/2872427.2874809.

6 J. Bai, L. Cao, S. Mosbach, J. Akroyd, A.A. Lapkin, M. Kraft, From platform to knowledge graph: Evolution of laboratory automation, JACS Au 2 (2) (2022) 292–309, http://dx.doi.org/10.1021/jacsau.1c00438.

7 T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Sci. Am. 284 (5) (2001) 34–43, http://dx.doi.org/10.1038/scientificamerican0501-34, URL https: //www.jstor.org/stable/10.2307/26059207.

8 J. Zhao, C. Bizer, Y. Gil, P. Missier, S. Sahoo, Provenance requirements for the next version of RDF, 2010, W3C Workshop – RDF Next Steps, Stanford, CA, USA, June 26-27, URL https://www.w3.org/2009/12/rdf-ws/papers/ws08.

9 DCMI Usage Board, DCMI metadata terms, 2020, URL https://www.dublincore. org/specifications/dublin-core/dcmi-terms/.

10 L.F. Sikos, D. Philp, Provenance-aware knowledge representation: A survey of data models and contextualized knowledge graphs, Data Sci. Eng. 5 (3) (2020) 293–316, http://dx.doi.org/10.1007/s41019-020-00118-0.

11 E. Deelman, D. Gannon, M. Shields, I. Taylor, Workflows and e-science: An overview of workflow system features and capabilities, Future Gener. Comput. Syst. 25 (5) (2009) 528–540, http://dx.doi.org/10.1016/j.future.2008.06.012.

12 J. Liu, E. Pacitti, P. Valduriez, M. Mattoso, A survey of data-intensive scientific workflow management, J. Grid Comput. 13 (4) (2015) 457–493, http://dx.doi. org/10.1007/s10723-015-9329-8.

13 The Apache Software Foundation, Apache airflow, 2022, URL https://airflow. apache.org/.

14 W. Falcon, The PyTorch Lightning team, PyTorch lightning, 2019, http://dx.doi. org/10.5281/zenodo.3828935, URL https://github.com/Lightning-AI/lightning.

15 R.F. da Silva, H. Casanova, K. Chard, D. Laney, D. Ahn, S. Jha, C. Goble, L. Ramakrishnan, L. Peterson, B. Enders, D. Thain, I. Altintas, Y. Babuji, R.M. Badia, V. Bonazzi, T. Coleman, M. Crusoe, E. Deelman, F.D. Natale, P.D. Tommaso, T. Fahringer, R. Filgueira, G. Fursin, A. Ganose, B. Gruning, D.S. Katz, O. Kuchar, A. Kupresanin, B. Ludascher, K. Maheshwari, M. Mattoso, K. Mehta, T. Munson, J. Ozik, T. Peterka, L. Pottier, T. Randles, S. Soiland-Reyes, B. Tovar, M. Turilli, T. Uram, K. Vahi, M. Wilde, M. Wolf, J. Wozniak, Workflows community summit: Bringing the scientific workflows community together, 2021, URL https://arxiv.org/abs/2103.09181.

16 N. Dragoni, S. Giallorenzo, A.L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, L. Safina, Microservices: Yesterday, today, and tomorrow, in: Present and Ulterior Software Engineering, Springer, 2017, pp. 195–216, http://dx.doi.org/10.1007/ 978-3-319-67425-4_12.

17 Duda, O., et al, Selection of Effective Methods of Big Data Analytical Processing in Information Systems of Smart Cities. CEUR Workshop Proceedings 2631, pp. 68-78. 2020.

18 Dapr Authors, APIs for building portable and reliable microservices, 2022, URL https://dapr.io/.

19 J. Akroyd, S. Mosbach, A. Bhave, M. Kraft, Universal digital twin – a dynamic knowledge graph, Data-Centr. Eng. 2 (2021) e14, http://dx.doi.org/10.1017/dce. 2021.10.

20 Bodnarchuk I., Duda O., Kharchenko A., Kunanets N., Matsiuk O., Pasichnyk V. Choice method of analytical information-technology platform for projects associated to the smart city class. ICTERI 2020 ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer Proceedings of the 14th International Conference on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer. Volume I: Main Conference p.317-330.

21 Bai, Jiaru, et al. "A derived information framework for a dynamic knowledge graph and its application to smart cities." Future Generation Computer Systems 152 (2024): 112-126.

22 N. Mohammadi, J.E. Taylor, Thinking fast and slow in disaster decision-making with smart city digital twins, Nat. Comput. Sci. 1 (12) (2021) 771–773, http: //dx.doi.org/10.1038/s43588-021-00174-0.

23 Duda, O., Kunanets, N., Martsenko, S., Matsiuk, O., Pasichnyk, V., Building secure Urban information systems based on IoT technologies. CEUR Workshop Proceedings 2623, pp. 317-328. 2020.

24 T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, J. Zhao, PROV-O: The PROV ontology, 2013, W3C Recommendation, URL https://www.w3.org/TR/prov-o/.

25 D. Garijo, Y. Gil, Augmenting PROV with plans in P-PLAN: Scientific processes as linked data, in: Proceedings of the 2nd International Workshop on

Linked Science, Vol. 951, CEUR Workshop Proceedings, 2012, URL https://oa.upm.es/ 19478/.

26 D. Garijo, Y. Gil, O. Corcho, Abstract, link, publish, exploit: An end to end framework for workflow sharing, Future Gener. Comput. Syst. 75 (2017) 271–283, http://dx.doi.org/10.1016/j.future.2017.01.008.

27 L. Moreau, B. Ludäscher, I. Altintas, R.S. Barga, S. Bowers, S. Callahan, G. Chin, B. Clifford, S. Cohen, S. Cohen-Boulakia, S. Davidson, E. Deelman, L. Digiampietri, I. Foster, J. Freire, J. Frew, J. Futrelle, T. Gibson, Y. Gil, C. Goble, J. Golbeck, P. Groth, D.A. Holland, S. Jiang, J. Kim, D. Koop, A. Krenek, T. McPhillips, G. Mehta, S. Miles, D. Metzger, S. Munroe, J. Myers, B. Plale, N. Podhorszki, V. Ratnakar, E. Santos, C. Scheidegger, K. Schuchardt, M. Seltzer, Y.L. Simmhan, C. Silva, P. Slaughter, E. Stephan, R. Stevens, D. Turi, H. Vo, M. Wilde, J. Zhao, Y. Zhao, Special issue: The first provenance challenge, Concurr. Comput.-Pract. Exp. 20 (5) (2008) 409–418, http://dx.doi.org/10.1002/ cpe.1233.

28 R. Chard, J. Pruyne, K. McKee, J. Bryan, B. Raumann, R. Ananthakrishnan, K. Chard, I.T. Foster, Globus automation services: Research process automation across the space–time continuum, Future Gener. Comput. Syst. 142 (2023) 393–409, http://dx.doi.org/10.1016/j.future.2023.01.010.

29 J. Akroyd, Z. Harper, D. Soutar, F. Farazi, A. Bhave, S. Mosbach, M. Kraft, Universal digital twin: Land use, Data-Centr. Eng. 3 (2022) http://dx.doi.org/ 10.1017/dce.2021.21.

30 J. Akroyd, A. Bhave, G. Brownbridge, E. Christou, M. Hillman, M. Hofmeister, M. Kraft, J. Lai, K.F. Lee, S. Mosbach, D. Nurkowski, O. Parry, CReDo technical paper 1: Building a cross-sector digital twin, 2022, URL https://doi.org/10. 17863/CAM.81779.

31 X. Zhou, A. Eibeck, M.Q. Lim, N.B. Krdzavac, M. Kraft, An agent composition framework for the J-park simulator - a knowledge graph for the process industry, Comput. Chem. Eng. 130 (2019) 106577, http://dx.doi.org/10.1016/j. compchemeng.2019.106577.

32 J. Bai, R. Geeson, F. Farazi, S. Mosbach, J. Akroyd, E.J. Bringley, M. Kraft, Automated calibration of a poly(oxymethylene) dimethyl ether oxidation mechanism using the knowledge graph technology, J. Chem. Inf. Model. 61 (4) (2021) 1701–1717, http://dx.doi.org/10.1021/acs.jcim.0c01322.

33 O. Hartig, B. Thompson, Foundations of an alternative approach to reification in RDF, 2021, URL https://arxiv.org/abs/1406.3399v3.

34 World Wide Web Consortium (W3C), RDF-star, 2021, URL https://w3c.github. io/rdf-star/.

35 Ontotext, What is RDF-star?, 2022, URL https://www.ontotext.com/ knowledgehub/fundamentals/what-is-rdf-star/.

36 J. Bai, K.F. Lee, S. Mosbach, 2023, URL https://github.com/cambridge-cares/ TheWorldAvatar/tree/main/JPS_Ontology/ontology/ontoderivation.

37 S. Cox, C. Little, J.R. Hobbs, F. Pan, Time ontology in OWL, 2020, W3C Candidate Recommendation 26 March 2020, URL https://www.w3.org/TR/owltime/.

38 M. Krämer, H.M. Würz, C. Altenhofen, Executing cyclic scientific workflows in the cloud, J. Cloud Comput. 10 (1) (2021) 1–26, http://dx.doi.org/10.1186/ s13677-021-00229-7.

39 Department for Environment Food & Rural Affairs, Real Time floodmonitoring API, 2021, URL https://environment.data.gov.uk/flood-monitoring/ doc/reference.

40 HM Land Registry, HM land registry open data, 2022, URL https://landregistry. data.gov.uk/.

41 K. Schekotihin, P. Rodler, W. Schmid, OntoDebug: Interactive Ontology Debugging Plug-in for Protégé, in: Lecture Notes in Computer Science, Springer International Publishing, 2018, pp. 340–359, http://dx.doi.org/10.1007/978-3319-90050-6_19.

42 M.A. Musen, Protégé Team, The Protégé project: A look back and a look forward, AI Matters 1 (4) (2015) 4–12, http://dx.doi.org/10.1145/2757001.2757003.

43 Ризик як оцінка небезпеки. http://opcb.kpi.ua/wp-content/uploads/2014/09/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F

-%D0%A0%D0%B8%D0%B7%D0%B8%D0%BA-%D1%8F%D0%BA-%D0%BE%D1%86%D1%96%D0%BD%D0%BA%D0%B0-%D0%BD%D0%B5%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D0%.pdf.

44 Небезпека. Ризик – як оцінка небезпеки. http://samkult.com/wp-content/uploads/2020/04/%D0%9D%D0%B5%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D0%BA%D0%B0-%D0%A0%D0%B8%D0%B7%D0%B8%D0%BA-%D1%8F%D0%BA-%D0%BE%D1%86%D1%96%D0%BD%D0%BA%D0%B0-%D0%BD%D0%B5%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D0%BA%D0%B8.pdf.

45 Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання / укл.: Стручок В. С. Тернопіль: ФОП Паляниця В. А., 2022. 156 с.

46 Електробезпека: охорона праці. URL: https://www.sop.com.ua/article/745-elektrobezpeka.

47 Лекція 8. Заходи електробезпеки на підприємствах галузі. URL: http://opcb.kpi.ua/wp-content/uploads/2014/09/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F-8.pdf.