

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

КАСЬЯНЧУК ІГОР ВЯЧЕСЛАВОВИЧ

УДК 004.046; 004.896; 004.021

ДИСЕРТАЦІЯ

**СЕМАНТИЧНА СТРУКТУРА ДЛЯ ОРКЕСТРОВКИ ТА ХОРЕОГРАФІЇ
КОГНІТИВНИХ ВЕБ-СЕРВІСІВ**

122 – Комп'ютерні науки

12 – Інформаційні технології

Подається на здобуття наукового ступеня

Дисертація містить результати власних досліджень. Використання ідей,
результатів і текстів інших авторів мають посилання на відповідне джерело
 І.В. Касьянчук

Науковий керівник:
ПЕТРЕНКО Анатолій Іванович
Доктор технічних наук, професор

КИЇВ – 2025

АНОТАЦІЯ

Касьянчук І.В. Семантична структура для оркестровки та хореографії когнітивних веб-сервісів. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії з галузі знань 122 Комп'ютерні науки за спеціальністю 122 Комп'ютерні науки. – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2025.

Дисертаційна робота присвячена розробці моделі координації когнітивних веб-сервісів, яка реалізується через семантичну структуру для їх вибору, інтеграції та автоматизації композиції. Дослідження зосереджене на вдосконаленні методів семантичного узгодження сервісів шляхом використання онтологічного підходу, адаптивного планування на основі контексту виконання запитів та інтеграції параметрів QoS для оптимізації вибору сервісів.

Основна увага приділяється розширенню семантичного опису сервісів шляхом використання онтологічного підходу та доповнення логічними правилами SWRL для формалізації зв'язків між сервісами. Проведено порівняльний аналіз із альтернативними методами. Запропонований підхід демонструє покращену продуктивність, точність і масштабованість семантичної обробки у динамічних середовищах у порівнянні з традиційними методами оркестрації (наприклад, BPEL) та хореографії (WS-CDL).

Дисертація відповідає науково-технічним пріоритетам Національного технічного університету України «КПІ імені Ігоря Сікорського» та спрямована на розвиток семантичних веб-технологій, когнітивних обчислень та інтелектуальної обробки запитів. Дослідження спрямоване на розробку методів і алгоритмів для підвищення продуктивності, адаптивності веб-сервісів та відповідності вимогам якості обслуговування (QoS).

Результати роботи узгоджуються з напрямками розвитку сучасних інформаційних технологій, зокрема, в аспектах автоматизованого семантичного аналізу, інтеграції онтологій та розумного вибору сервісів для оптимізації процесів обробки запитів.

Запропоновані рішення забезпечують ефективну інтеграцію та автоматизацію веб-сервісів через семантичну структуру, яка реалізує модель координації та композиції. Результати дослідження мають практичну цінність у контексті реалізації наукових програм із впровадження інноваційних цифрових рішень у різних галузях, оскільки проведене дослідження відповідає пріоритетам розвитку цифрової економіки, які закладені в стратегічних програмах, спрямованих на впровадження інтелектуальних систем, таких як «Інтернет речей» (IoT) та когнітивні обчислення.

Метою дисертації є розробка моделі координації когнітивних веб-сервісів, яка реалізується через семантичну структуру, забезпечуючи ефективне оркестрування і хореографію сервісів у розподілених середовищах.

Об'єкт дослідження – процеси опису, пошуку та інтеграції когнітивних веб-сервісів у розподілених системах.

Предмет дослідження – семантичні методи опису, пошуку та інтеграції когнітивних веб-сервісів, які забезпечують автоматизацію їх використання та взаємодії.

Наукова новизна отриманих результатів полягає у тому, що в дисертаційній роботі:

1. Вперше запропоновано модель координації когнітивних веб-сервісів, яка забезпечує автоматизований вибір, оркестрацію та хореографію на основі семантичного опису функціональності, параметрів QoS та контексту виконання. Відмінність від існуючих підходів полягає у формалізованому описі когнітивних можливостей сервісів (наприклад, здатність до навчання, обробки природної мови, аналізу зображень), їх функціональності, вхідних/вихідних даних та обмежень, що забезпечує автоматизовану інтеграцію.

2. Модифіковано метод пошуку та вибору веб-сервісів, який базується на семантичному описі та забезпечує динамічне ранжування сервісів за їх відповідністю до контексту виконання (наприклад, час відповіді, специфіка даних, ресурси).

3. Вперше запропоновано інтеграцію когнітивних веб-сервісів на основі семантичного підходу, що дозволяє автоматизувати процес вибору, оркестрації та хореографії сервісів у динамічних середовищах, враховуючи змінність умов виконання та недоступність окремих компонентів. Отримані результати розширюють можливості когнітивних веб-сервісів та інтеграції штучного інтелекту в складні системи, підвищуючи ефективність прийняття рішень у стратегічно важливих сферах.

Практичне значення отриманих результатів полягає у наступному:

1. Розроблена модель семантичної структури може бути використана для створення адаптивних і масштабованих систем у різних сферах, таких як управління ресурсами, автоматизація бізнес-процесів, інтеграція сервісів в Інтернеті речей (IoT) та смарт-середовищах.

2. Розроблений алгоритм інтелектуального пошуку та вибору когнітивних веб-сервісів може бути інтегрований у програмні платформи, що використовуються для побудови систем автоматизації, аналітики великих даних, розумних рекомендаційних систем або інтеграції сервісів у хмарних середовищах.

3. Запропоновані підходи є основою для розробки програмних продуктів, що забезпечують динамічний пошук, інтеграцію та управління когнітивними веб-сервісами в системах із високими вимогами до адаптивності та швидкості реагування, таких як медицина, фінанси чи логістика.

Результати проведеного дослідження були представлені та обговорені у провідних наукових виданнях, що підтверджує їхню актуальність і наукову новизну: стаття «Finding a conceptual approach to developing an architecture of general-purpose services for economic researches» опублікована в журналі Technology Audit and Production Reserves (Vol. 3 № 4(71), 2023). Наукова стаття «Модель розумної оркестрації веб-сервісів на прикладі статистичних досліджень» опублікована у Таврійському науковому віснику. Серія: Технічні науки (№ 4, 2023). Наукова стаття «Огляд хореографії веб-сервісів WSMO для виконання синхронних та асинхронних запитів» опублікована у Таврійському науковому віснику. Серія: Технічні науки (№ 2, 2024). Наукове дослідження на тему

«Orchestration of service-oriented applications with reactive programming techniques» опубліковане в журналі Technology Audit and Production Reserves (Vol. 4 № 2(78), 2024). Наукова стаття «Розробка семантичної структури для композиції когнітивних веб-сервісів», опублікована у Technology Audit and Production Reserves (2025), представляє основні результати дослідження щодо автоматизації оркестрації та хореографії сервісів.

Ключові слова: семантичний веб, онтології, когнітивні сервіси, обробка запитів, QoS, SWRL, семантична сумісність, інтелектуальне планування, оптимізація веб-сервісів, динамічне композиціювання, машинне навчання.

ANOTATION

Kasianchuk I.V. Semantic structure for orchestration and choreography of cognitive web-services – A Research Paper (Manuscript). Dissertation for the degree of Doctor of Philosophy in the field of knowledge 122 Computer Science, specialty 122 Computer Science. – National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute,” Kyiv, 2025.

The dissertation is devoted to the development of a coordination model for cognitive web services, implemented through a semantic structure for their selection, integration, and automation of composition. The research focuses on improving methods of semantic alignment, adaptive service planning, and the integration of Quality of Service (QoS) parameters in the query processing workflow.

The main attention is paid to expanding the semantic description of services by using an ontological approach and supplementing it with SWRL logical rules to formalize the relationships between services. A comparative analysis with alternative methods has been conducted. The proposed approach demonstrates higher productivity, accuracy, and scalability of semantic processing in dynamic environments.

The dissertation corresponds to the scientific and technical priorities of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" and is aimed at the development of semantic web technologies, cognitive computing, and intelligent query processing. The research is focused on the development of methods and algorithms to improve the productivity, adaptability of web services, and their compliance with Quality of Service (QoS) requirements.

The results of the work align with the directions of modern information technology development, particularly in aspects of automated semantic analysis, ontology integration, and intelligent service selection to optimize query processing workflows.

The proposed solutions ensure effective integration and automation of web services through a semantic structure that implements the coordination model. The research results have practical value in the context of implementing scientific programs for introducing innovative digital solutions in various fields, as the conducted research corresponds to the priorities of digital economy development, established in strategic programs aimed at the

implementation of intelligent systems such as the Internet of Things (IoT) and cognitive computing.

The aim of the dissertation is to develop a coordination model for cognitive web services, implemented through a semantic structure, ensuring effective orchestration and choreography of services in distributed environments.

The object of research is the processes of describing, searching, and integrating cognitive web services in distributed systems.

The subject of research is semantic methods of describing, searching, and integrating cognitive web services, which ensure the automation of their use and interaction.

The scientific novelty of the obtained results lies in the fact that in the dissertation:

1. A coordination model for cognitive web services is proposed for the first time, ensuring automated selection, orchestration, and choreography based on the semantic description of functionality, QoS parameters, and execution context. The difference from existing approaches lies in the formalized description of cognitive capabilities of services (such as the ability to learn, natural language processing, image analysis), their functionality, input/output data, and constraints, which ensures automated integration.

2. The method of searching and selecting web services has been modified, which is based on semantic description and provides dynamic ranking of services according to their relevance to the execution context (e.g., response time, data specificity, resources).

3. The integration of cognitive web services based on a semantic approach has been proposed for the first time, allowing the automation of service selection, orchestration, and choreography in dynamic environments, taking into account changing execution conditions and the unavailability of individual components. The obtained results expand the capabilities of cognitive web services and the integration of artificial intelligence into complex systems, increasing decision-making efficiency in strategically important areas.

The practical significance of the obtained results is as follows:

1. The developed semantic structure model can be used to create adaptive and scalable systems in various fields such as resource management, business process automation, service integration in the Internet of Things (IoT), and smart environments.

2. The developed algorithm for intelligent search and selection of cognitive web services can be integrated into software platforms used for building automation systems, big data analytics, intelligent recommendation systems, or service integration in cloud environments.

3. The proposed approaches form the basis for the development of software products that provide dynamic search, integration, and management of cognitive web services in systems with high requirements for adaptability and response speed, such as healthcare, finance, or logistics.

The results of the conducted research have been presented and discussed in leading scientific journals, confirming their relevance and scientific novelty: the article "Finding a conceptual approach to developing an architecture of general-purpose services for economic researches" was published in the journal *Technology Audit and Production Reserves* (Vol. 3 No. 4(71), 2023). The scientific article "A model of intelligent orchestration of web services using statistical research as an example" was published in the *Tavriya Scientific Bulletin. Series: Technical Sciences* (No. 4, 2023). The scientific article "A review of WSMO web service choreography for synchronous and asynchronous request execution" was published in the *Tavriya Scientific Bulletin. Series: Technical Sciences* (No. 2, 2024). The scientific research "Orchestration of service-oriented applications with reactive programming techniques" was published in the journal *Technology Audit and Production Reserves* (Vol. 4 No. 2(78), 2024). The scientific article "Development of a semantic structure for cognitive web service composition," published in *Technology Audit and Production Reserves* (2025), presents the main research results on the automation of service orchestration and choreography.

Keywords: semantic web, ontologies, cognitive services, query processing, QoS, SWRL, semantic interoperability, intelligent planning, web service optimization, dynamic composition, machine learning.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

1. Касьянчук І. В. Модель розумної оркестрації веб сервісів на прикладі статистичних досліджень. *Таврійський науковий вісник. Серія: Технічні науки*. 2023. № (4). С. 48-53.
2. Касьянчук І. В. Огляд хореографії веб-сервісів WSMO для виконання синхронних та асинхронних запитів. *Таврійський науковий вісник. Серія: Технічні науки*. 2024. № (2). С. 46-52.
3. Kasianchuk I. Orchestration of service-oriented applications with reactive programming techniques. *Technology Audit and Production Reserves*. 2-24. №4(2(78)). P. 24–29.
4. Lumpova T., Kasianchuk I. Finding a conceptual approach to developing an architecture of general-purpose services for economic researches. *Technology Audit and Production Reserves*. 2023. №3(4(71)). P. 25–31.
5. Kasianchuk, I., & Petrenko, A. (2025). Development of a semantic structure for the composition of cognitive web services. *Technology Audit and Production Reserves*, 1(81). <https://doi.org/10.15587/2706-5448.2025.322370>

ЗМІСТ

ВСТУП.....	12
РОЗДІЛ 1. ОГЛЯД НАУКОВИХ ДЖЕРЕЛ ТА АНАЛІЗ НАПРАЦЮВАНЬ ІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	18
1.1. Дослідження підходів до оркестрування та хореографії веб-сервісів.....	18
1.2. Класифікація семантичних структур та моделей.....	29
1.3. Аналіз когнітивних веб-сервісів.....	32
1.3.1. Ключові характеристики когнітивних веб-сервісів.....	34
1.3.2. Особливості когнітивних веб-сервісів та їх відмінності від традиційних.....	38
Висновки до розділу 1.....	46
РОЗДІЛ 2. ТЕОРЕТИЧНІ ПІДХОДИ ДО МОДЕЛЮВАННЯ СЕМАНТИЧНОЇ СТРУКТУРИ.....	48
2.1. Основи семантичного вебу: застосування технологій і стандартів.....	48
2.1.1. Принципи побудови RDF моделей.....	49
2.1.2. Можливості OWL для моделювання когнітивних сервісів.....	51
2.1.3. Рушії логічного виведення OWL. Принципи та застосування.....	57
2.1.4. Використання SPARQL і SQWRL для семантичних запитів.....	65
2.2. Методи оркестрації: принципи, переваги та недоліки.....	71
2.3. Методи хореографії: підходи та практичне застосування.....	77
2.4. Семантика в когнітивних веб-сервісах.....	83
2.5. Аналіз існуючих методів оркестровки та хореографії на основі семантики....	87
2.6. Моделі оркестрування та хореографії веб-сервісів у когнітивних системах...	93
2.6.1. Використання фреймворків для композиції когнітивних сервісів.....	93
Висновки до розділу 2.....	102
РОЗДІЛ 3. РОЗРОБКА СЕМАНТИЧНОЇ СТРУКТУРИ.....	105
3.1. Вимоги до семантичної структури когнітивних веб-сервісів.....	105
3.1.1. Функціональні вимоги.....	105
3.1.2. Нефункціональні вимоги.....	107
3.2. Формулювання задачі розробленої платформи.....	109

3.3. Опис запропонованої моделі оркестрування та хореографії.....	112
3.4. Зміст та структура онтології для опису когнітивних веб-сервісів.....	115
3.5. Реалізація додатку для обробки запитів на основі семантичної структури....	118
3.5.1. Концептуальна архітектура додатку.....	119
3.5.2. Алгоритмічна модель обробки запитів.....	123
Висновки до розділу 3.....	126
РОЗДІЛ 4. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОЇ СЕМАНТИЧНОЇ СТРУКТУРИ.....	129
4.1. Методологія оцінки ефективності семантичної структури.....	129
4.2. Оцінка функціональних можливостей.....	130
4.3. Експериментальне дослідження ефективності пошуку та композиції сервісів.....	132
4.4. Порівняльний аналіз із альтернативними підходами.....	135
Висновки до розділу 4.....	141
ВИСНОВКИ.....	142
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	148
ДОДАТОКИ.....	159

ВСТУП

Актуальність теми.

Сучасні веб-сервіси стають дедалі більш спеціалізованими та складними, що ускладнює їхню взаємодію. Для ефективної координації таких сервісів необхідна семантична модель, яка формалізує їхню функціональність, забезпечує автоматизовану інтеграцію та підтримує адаптивне управління взаємодією.

Когнітивні веб-сервіси, які мають здатність до самонавчання, аналізу неструктурованих даних та адаптації до змін середовища, потребують більш гнучкої та інтелектуальної моделі взаємодії, ніж традиційні веб-сервіси. Використання семантичної структури дозволяє таким сервісам "розуміти" функціональність один одного, узгоджувати спільні дії та автоматично адаптуватися до змінних умов виконання завдань.

Семантичний підхід забезпечує інтелектуальний вибір сервісів, їх динамічне компонування та ефективне управління потоками даних, що значно спрощує виконання складних бізнес-процесів, зменшуючи витрати часу та ресурсів на їх реалізацію.

Зі зростанням кількості IoT-пристроїв виникає необхідність у їх семантичній інтеграції для автоматизованого обміну даними та координації виконання завдань. Використання онтологій дозволяє формалізувати зв'язки між IoT-пристроями та когнітивними веб-сервісами, що дає змогу підвищити сумісність, адаптивність та автоматизацію їхньої взаємодії в розподілених середовищах.

Крім того, семантична інформація про користувачів та їхні потреби може використовуватися для персоналізації веб-сервісів, підвищуючи якість їхньої роботи та відповідність результатів очікуванням користувачів.

Зважаючи на викладене, тема дослідження «Семантична структура для оркестрування і хореографії когнітивних веб-сервісів» є актуальною, оскільки вона спрямована на вирішення ключових проблем автоматизації інтеграції когнітивних сервісів у складних динамічних середовищах.

Зв'язок роботи з науковими програмами, планами, темами.

Тема роботи «Семантична структура для оркестрування і хореографії когнітивних веб-сервісів» безпосередньо пов'язана із сучасними напрямками досліджень у галузі інформаційних технологій, зокрема Semantic Web, когнітивних веб-сервісів, а також методів оркестрування та хореографії. Результати роботи узгоджуються з цілями та завданнями наукових програм і дослідницьких проєктів у сфері цифрової трансформації, розвитку семантичних технологій, автоматизації управління сервісами та впровадження когнітивних обчислень.

Дане дослідження спрямоване на вирішення ключових проблем інтеграції та автоматизації веб-сервісів через використання семантичного підходу. Результати мають практичне значення у контексті реалізації наукових програм із впровадження інноваційних цифрових рішень у різних галузях. Зокрема, робота відповідає пріоритетам розвитку цифрової економіки та стратегічним ініціативам, орієнтованим на створення інтелектуальних систем, Інтернету речей (IoT) та когнітивних обчислень.

Мета і завдання дослідження.

Мета дисертаційної роботи – розробка семантичної структури, що забезпечує ефективне оркестрування і хореографію когнітивних веб-сервісів для підвищення ефективності їх взаємодії та автоматизації різноманітних завдань. Задля досягнення поставленої мети було сформульовано ряд завдань дослідження:

1. Дослідити існуючі стандарти, мови та платформи для оркестрування веб-сервісів, визначити їхні переваги та обмеження в контексті когнітивних сервісів.
2. Запропонувати модель семантичної структури для когнітивних веб-сервісів, що формалізує їхню функціональність, когнітивні можливості, вхідні та вихідні дані, обмеження, а також підтримує інтеграцію в розподілених системах.
3. Модифікувати метод пошуку та вибору когнітивних веб-сервісів на основі семантичного опису, забезпечуючи врахування контексту виконання та специфічних потреб користувача.
4. Розробити прототип програмного інструментарію, що базується на запропонованих підходах до семантичного опису та інтелектуального пошуку когнітивних веб-сервісів, та порівняти його з існуючими рішеннями на практиці.

5. Провести аналіз отриманих результатів, оцінити ефективність запропонованого підходу та визначити перспективи подальших досліджень.

Об'єктом дослідження є процеси опису, пошуку та інтеграції когнітивних веб-сервісів у розподілених системах.

Предметом дослідження виступають семантичні методи опису, пошуку та інтеграції когнітивних веб-сервісів, які забезпечують автоматизацію їх використання та взаємодії.

Методи дослідження: аналіз літературних джерел, спостереження, порівняння, аналіз, формалізація, онтологічне моделювання, моделювання процесів, експеримент.

Наукова новизна отриманих результатів полягає у тому, що в дисертаційній роботі:

1. Вперше запропоновано та розроблено модель координації когнітивних веб-сервісів, яка забезпечує автоматизований вибір, оркестрацію та хореографію на основі семантичного опису функціональності, параметрів QoS та контексту виконання. Відмінність від існуючих підходів полягає у формалізованому описі когнітивних можливостей сервісів (наприклад, здатність до навчання, обробки природної мови, аналізу зображень), їх функціональності, вхідних/вихідних даних та обмежень, що забезпечує автоматизовану інтеграцію.

2. Модифіковано метод пошуку та вибору веб-сервісів, який базується на семантичному описі та забезпечує динамічне ранжування сервісів за їх відповідністю до контексту виконання (наприклад, час відповіді, специфіка даних, ресурси).

3. Вперше запропоновано інтеграцію когнітивних веб-сервісів на основі семантичного підходу, що дозволяє автоматизувати процес вибору, оркестрації та хореографії сервісів у динамічних середовищах, враховуючи змінність умов виконання та недоступність окремих компонентів. Отримані результати розширюють можливості когнітивних веб-сервісів та інтеграції штучного інтелекту в складні системи, підвищуючи ефективність прийняття рішень у стратегічно важливих сферах.

Практичне значення отриманих результатів полягає у наступному:

1. Розроблена модель семантичної структури може бути використана для створення адаптивних і масштабованих систем у різних сферах, таких як управління ресурсами, автоматизація бізнес-процесів, інтеграція сервісів в Інтернеті речей (IoT) та смарт-середовищах.

2. Розроблений алгоритм інтелектуального пошуку та вибору когнітивних веб-сервісів може бути інтегрований у програмні платформи, що використовуються для побудови систем автоматизації, аналітики великих даних, розумних рекомендаційних систем або інтеграції сервісів у хмарних середовищах.

3. Запропоновані підходи є основою для розробки програмних продуктів, що забезпечують динамічний пошук, інтеграцію та управління когнітивними веб-сервісами в системах із високими вимогами до адаптивності та швидкості реагування, таких як медицина, фінанси чи логістика.

Особистий внесок здобувача

Касьянчук І.В. здійснив внесок у розвиток сучасних підходів до оркестрації та хореографії веб-сервісів, що відображено у його наукових публікаціях. У своїй науковій праці «Модель розумної оркестрації веб-сервісів на прикладі статистичних досліджень» він розробив модель, яка дозволяє ефективно інтегрувати веб-сервіси для обробки статистичних даних. Завдяки аналізу існуючих підходів та формалізації процесу інтеграції, було створено алгоритм, здатний оптимізувати обробку запитів у цій сфері.

У публікації «Огляд хореографії веб-сервісів WSMO для виконання синхронних та асинхронних запитів» І. Касьянчук систематизував знання про хореографію веб-сервісів у контексті WSMO (Web Service Modeling Ontology). Його внесок включає огляд існуючих моделей, адаптацію методології WSMO для роботи з різними типами запитів, а також детальний аналіз їхніх переваг і недоліків у конкретних застосуваннях.

В іншій науковій праці «Orchestration of service-oriented applications with reactive programming techniques», присвяченій реактивному програмуванню, студент дослідив, як ці техніки можуть використовуватися для оркестрації сервіс-

орієнтованих додатків. Ним була розроблена концепція, що враховує сучасні вимоги до інтеграції сервісів, та продемонстрована її ефективність на практичних прикладах.

Крім того, спільно з Т. Лумповою, І. Касьянчук взяв участь у проєкті з розробки архітектури універсальних сервісів для економічних досліджень. Його внесок полягав у створенні концептуального підходу, що забезпечує гнучкість і масштабованість сервісів, та аналізі ключових компонентів для їхнього успішного впровадження.

Наукові праці автора дисертаційного дослідження охоплюють як теоретичні, так і практичні аспекти оркестрації та хореографії веб-сервісів, сприяючи впровадженню сучасних технологій для розв'язання актуальних задач у різних сферах.

Апробація результатів дисертації.

Результати проведеного дослідження були представлені та обговорені у провідних наукових виданнях, що підтверджує їхню актуальність і наукову новизну: стаття «Finding a conceptual approach to developing an architecture of general-purpose services for economic researches» опублікована в журналі Technology Audit and Production Reserves (Vol. 3 № 4(71), 2023). Наукова стаття «Модель розумної оркестрації веб-сервісів на прикладі статистичних досліджень» опублікована у Таврійському науковому віснику. Серія: Технічні науки (№ 4, 2023). Наукова стаття «Огляд хореографії веб-сервісів WSMO для виконання синхронних та асинхронних запитів» опублікована у Таврійському науковому віснику. Серія: Технічні науки (№ 2, 2024). Наукове дослідження на тему «Orchestration of service-oriented applications with reactive programming techniques» опубліковане в журналі Technology Audit and Production Reserves (Vol. 4 № 2(78), 2024). Наукова стаття «Розробка семантичної структури для композиції когнітивних веб-сервісів», опублікована у Technology Audit and Production Reserves (2025), представляє основні результати дослідження щодо автоматизації оркестрації та хореографії сервісів.

Публікації.

Касьянчук І. В. Модель розумної оркестрації веб сервісів на прикладі статистичних досліджень. *Таврійський науковий вісник. Серія: Технічні науки*. 2023. № (4). С. 48-53.

Касьянчук І. В. Огляд хореографії веб-сервісів WSMO для виконання синхронних та асинхронних запитів. *Таврійський науковий вісник. Серія: Технічні науки*. 2024. № (2). С. 46-52.

Kasianchuk I. Orchestration of service-oriented applications with reactive programming techniques. *Technology Audit and Production Reserves*. 2-24. №4(2(78)). P. 24–29.

Lumpova T., Kasianchuk I. Finding a conceptual approach to developing an architecture of general-purpose services for economic researches. *Technology Audit and Production Reserves*. 2023. №3(4(71)). P. 25–31.

Kasianchuk, I., & Petrenko, A. (2025). Development of a semantic structure for the composition of cognitive web services. *Technology Audit and Production Reserves*, 1(81). <https://doi.org/10.15587/2706-5448.2025.322370>

Структура дисертації та її обсяг.

Дисертація складається із вступу, чотирьох розділів, шістнадцяти підрозділів, двох пунктів до розділів, загальних висновків, списку використаних джерел та додатків. Загальний обсяг роботи складає 164 сторінки, із яких 136 сторінок основного тексту. У роботі міститься 25 рисунків, 15 таблиць. Список використаних джерел налічує 91 найменування, із яких 30 іноземною мовою. У роботі міститься 2 додатки, опублікованих на 4 сторінках.

РОЗДІЛ 1. ОГЛЯД НАУКОВИХ ДЖЕРЕЛ ТА АНАЛІЗ НАПРАЦЮВАНЬ ІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Дослідження підходів до оркестрування та хореографії веб-сервісів

Сьогодні активно формується єдиний глобальний економічний, правовий та інформаційний простір для забезпечення вільної та ефективної комерційної діяльності всіх суб'єктів господарювання в мережі Інтернет. У процесі трансформації до інформаційного суспільства з домінуванням сфери сервісів збереглися як промисловість, так і сільське господарство. Проте саме роль сервісів зростає настільки, що вони беруть на себе головну роль у виробництві ВВП. Адже інформаційні сервіси задовольняють потреби споживачів за допомогою ІКТ, а не просто надають певну інформацію.

Веб-сервіси – це набір функціональних можливостей, опублікованих у мережі, які спочатку (наприкінці 1990-х років) були започатковані IBM, Microsoft та W3C, а сьогодні стали визнаною технологією серед усіх учасників ІТ-індустрії [63].

Іншими словами з точки зору підтримки ІКТ, веб-сервіси є стандартизованим способом інтеграції програм на основі стандартів XML, SOAP, WSDL і UDDI. Веб-сервіси використовуються для обміну даними різнорідних програм на основі уніфікованих стандартів і протоколів. За допомогою веб-сервісів функціональність будь-якої програми може бути реалізована через Інтернет, тобто можна створювати розподілені програми, компоненти яких можуть вільно взаємодіяти один з одним. Це дає можливість надавати підприємствам спільне обслуговування без зміни економічної та технічної основи підприємства [37, с. 137].

Іноді для опису Web-сервісів використовують терміни «архітектура, орієнтована на сервіси» (Service Oriented Architecture – SOA) або «архітектура Web-сервісів» (Web Services Architecture – WSA). Застосування сервіс-орієнтованих архітектур (SOA) і розподілених реєстрів сервісів дозволяє значно знизити собівартість та підвищити якість прикладного програмного забезпечення, що

створюється, а також забезпечує шляхи інтегрування додатків в межах сукупності великої кількості інформаційних систем підприємства чи кількох підприємств в межах країни або континенту [41].

Для реалізації Web-сервісів потрібно забезпечити:

- 1) інтероперабельність інформаційних систем, що надають та отримують Web-сервіси;
- 2) підтримку протоколів і технологій мережі Internet;
- 3) стандартизацію інтерфейсів;
- 4) підтримку різних мов програмування;
- 5) підтримку розподіленого середовища [45, с. 636].

Концепція Web-сервісів означає, що вони мають певну обмежену функціональність. Для вирішення складних завдань потрібно використовувати функціональність кількох сервісів. Тому в процесі розвитку архітектури Web-сервісів виникло поняття компонування Web-сервісів і потік Web-сервісів, або ще використовують термін оркестровка (Web Service Choreography) і хореографія (Web Service Choreography) Web-сервісів. Ці поняття відображають взаємодію сервісів і послідовність їх виконання. Застосунки, побудовані з використанням Web-сервісів, базуються на потоках робіт (Workflow-based applications) [1, с. 164].

Зважаємо на те, що саме оркестрування та хореографія веб-сервісів є ключовими концепціями для інтеграції та управління сервіс-орієнтованими системами. Ці підходи дозволяють забезпечувати взаємодію між різними сервісами, синхронізувати їхню роботу та досягати поставлених цілей у межах складних бізнес-процесів чи технічних систем. Адже оркестрування веб-сервісів передбачає централізоване управління процесами, у якому є єдиний контролюючий компонент (оркестратор), який визначає порядок виконання операцій веб-сервісами. У цьому контексті оркестрація подібна до диригента в оркестрі, який керує діями всіх учасників.

Спираючись на наукові погляди W. Fdhila відзначаємо, що в даний час існують різні мови для оркестрування або хореографії, найважливішими з яких є

BPEL4WS, WSCDL, BPML та WSCI. На рисунку 1.1 зображаємо хронологію появи різних мов для опису оркестрування та хореографії веб-сервісів [71].

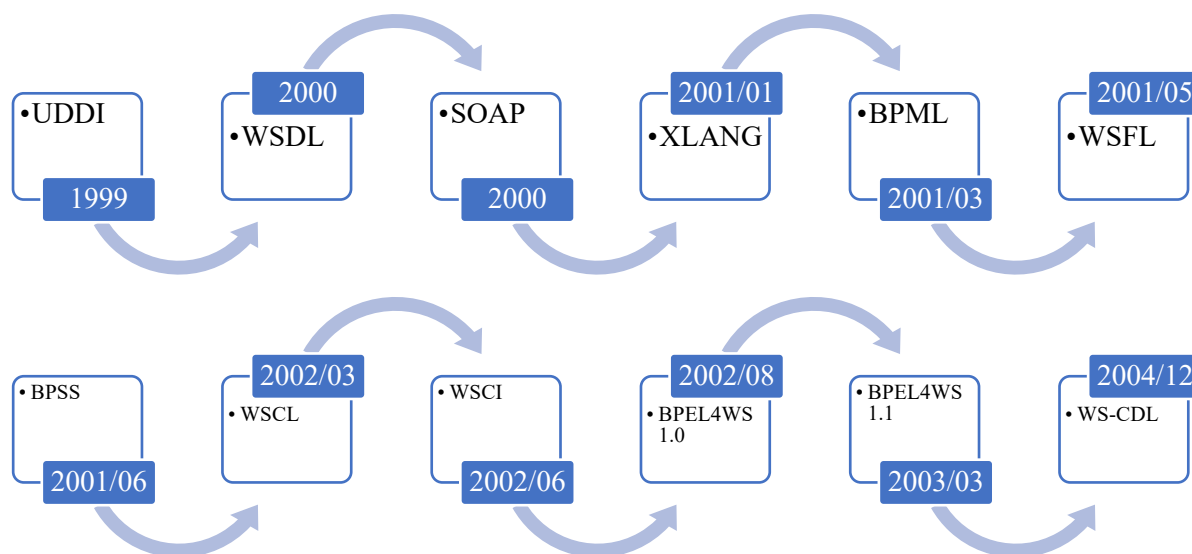


Рис. 1.1. Еволюція специфікацій опису веб-сервісів [71, с. 35]

Деякі з цих мов можуть бути використані для створення композицій орієнтованих оркестровки, а інші – хореографії.

UDDI є стандартом для реєстрації та пошуку веб-сервісів. UDDI – це своєрідний каталог, де організації можуть публікувати інформацію про свої веб-сервіси, щоб інші могли їх знаходити та використовувати. Він підтримує централізовану модель реєстру, що забезпечує інтеграцію та взаємодію бізнес-додатків через Інтернет [42].

WSDL – це XML-формат, який використовується для опису функціональності веб-сервісів. Він надає інформацію про те, як отримати доступ до веб-сервісу, які методи він підтримує, а також структуру вхідних і вихідних даних, що дозволяє клієнтам автоматично генерувати код для взаємодії з веб-сервісом [50].

SOAP – це протокол обміну повідомленнями, який використовується для передачі структурованих даних між додатками через мережу. Він базується на XML і забезпечує платформи-незалежний спосіб передачі даних, що робить його популярним для інтеграції систем у різних середовищах. SOAP забезпечує додаткові гарантії конфіденційності та цілісності даних, а також пропонує

вбудовану повторювану логіку для компенсації невдалих спроб обміну даними [91, с. 51].

XLANG є мовою, що використовується для опису логіки обміну повідомленнями між веб-сервісами. Вона дозволяє визначати послідовність дій та умови для взаємодії, а також забезпечує опис довготривалих транзакцій між різними системами [37, с. 135-142].

BPML – це стандарт моделювання бізнес-процесів, який використовує XML для опису логіки процесів, їхньої структури та інтеграції з різними системами. Він підтримує складні сценарії, включаючи обробку помилок і контроль за потоками даних.

WSFL є мовою для опису потоків процесів, які базуються на веб-сервісах. Вона дозволяє моделювати як оркестрацію (централізоване управління процесами), так і хореографію (розподілену взаємодію). WSFL використовується для інтеграції складних бізнес-процесів із залученням кількох веб-сервісів.

BPSS – це схема для опису бізнес-процесів у стандартизованій формі, яка використовується для забезпечення узгодженості між партнерами. BPSS є частиною ініціативи ebXML і зосереджується на забезпеченні сумісності бізнес-процесів для автоматизованої взаємодії між організаціями.

WSCL – це мова для визначення послідовності обміну повідомленнями між веб-сервісами. Вона дозволяє описувати, які повідомлення передаються, у якому порядку та за яких умов. WSCL спрощує інтеграцію веб-сервісів, особливо у сценаріях із багатоступеневими діалогами.

WSCI є стандартом для опису хореографії веб-сервісів, тобто взаємодії між ними з розподіленої перспективи. Мова допомагає визначити поведінку учасників у процесі, зокрема порядок дій, умови виконання та обробку помилок. WSCI підтримує координацію процесів між різними організаціями [2].

BPEL4WS 1.0 – це перша версія мови для опису бізнес-процесів, які виконуються за допомогою вебсервісів. Вона дозволяє створювати складні процеси шляхом інтеграції кількох вебсервісів у єдину транзакцію. BPEL4WS поєднує оркестрацію дій і управління виключеннями. Business Process Execution Language

for Web Services (BPEL4WS, 2002) був вперше запропонований компаніями BEA, IBM та Microsoft. Він складається з набору XML-тегів, які визначають нотацію для специфікації поведінки процесів на основі веб-сервісів. BPEL4WS надає мову на основі XML для формальної специфікації бізнес-процесів та протоколів взаємодії в бізнесі. У цьому розділі всі елементи BPEL4WS класифіковані на 5 категорій, як показано в таблиці 1.1, для кращого розуміння та аналізу.

Таблиця 1.1

Категорії елементів у BPEL4WS

Категорія	Елементи	Категорія	Елементи
Structural Control Elements	<process>	Data Structure Elements	<property>
	<scope>		<propertyAlias>
	<flow>		<correlationSets>
	<sequence>		<correlations>
	<receive>		<containers>
	<pick>	Service Related Elements	<serviceLinkType>
	<reply>		<serviceReference>
	<switch>		<partners>
	<while>	Exception Handling Elements	<faultHandlers>
	<links>		<compensateHandler>
Functional Elements	<invoke>		<compensate>
	<assign>		<throw>
	<wait>		<catch>
	<empty>		<catchAll>
	<terminate>		

Джерело: [89, с. 113-127]

Як бачимо із табл. 1.1, структурні елементи управління включають елементи, які визначають структуру та логіку виконання бізнес-процесів. Наприклад, елемент <process> задає загальний контекст для всього процесу, тоді як <scope> окреслює

область видимості, в межах якої елементи можуть взаємодіяти. Елементи, такі як `<flow>`, `<sequence>`, і `<while>`, дозволяють налаштовувати порядок виконання дій, включаючи паралельні та повторювані процеси, що, як наслідок, забезпечує гнучкість в організації бізнес-логіки, що дозволяє оптимізувати виконання завдань.

У категорії «елементи структури даних» зосереджено ті елементи, які відповідають за визначення та організацію даних, які використовуються в бізнес-процесах. Наприклад, `<property>` і `<propertyAlias>` дозволяють описувати властивості даних та створювати їхні псевдоніми для зручності. Елементи `<correlationSets>` і `<correlations>` використовуються для групування та встановлення зв'язків між повідомленнями, що забезпечує ефективне управління даними в процесі.

«Елементи, пов'язані зі сервісами» – категорія, яка містить елементи, що визначають взаємодію з зовнішніми сервісами. `<serviceLinkType>` та `<serviceReference>` забезпечують зв'язок між процесами та зовнішніми системами, що дозволяє бізнесу інтегрувати різні сервіси для виконання складних завдань. Елемент `<partners>` визначає партнерів, з якими процес може взаємодіяти, що сприяє створенню більш інтерактивних і гнучких бізнес-моделей.

Елементи обробки винятків: включають ті елементи, які спеціалізуються на обробці помилок і винятків під час виконання процесу. `<faultHandlers>` та `<compensateHandler>` дозволяють визначати дії у разі виникнення помилок, а також механізми компенсації для відновлення роботи процесу. Інші елементи, такі як `<throw>`, `<catch>`, і `<catchAll>`, забезпечують гнучку обробку винятків, що допомагає підтримувати стабільність і надійність бізнес-процесів.

Функціональні елементи – це категорія, яка містить елементи, що реалізують основні функції в бізнес-процесах. Наприклад, `<invoke>` дозволяє викликати зовнішні сервіси, а `<assign>` відповідає за призначення значень змінним. Інші елементи, такі як `<wait>` і `<terminate>`, управляють часовими аспектами виконання процесу та забезпечують його завершення в разі необхідності.

Версія 1.1 BPEL4WS удосконалила попередню версію, додавши більше можливостей для обробки складних сценаріїв, таких як управління помилками,

паралельне виконання процесів і підтримка транзакцій. Ця версія стала ще більш придатною для широкого впровадження в бізнес-середовищах.

WS-CDL – це мова для опису хореографії вебсервісів, яка дозволяє моделям поведінки охоплювати множинних учасників. Вона орієнтована на координацію взаємодій між сторонами, зберігаючи незалежність кожного учасника. WS-CDL використовується для забезпечення узгодженості у багатосторонніх бізнес-процесах [11].

Найчастіше моделі оркестрації будуються на основі таких стандартів, як BPEL (Business Process Execution Language) та BPMN (Business Process Model and Notation) [70]. Вони дозволяють описувати процеси, визначати послідовності дій та інтегрувати сервіси з урахуванням їхніх функцій. Переваги використання означених моделей полягають в тому, що вони дозволяють забезпечити чіткий контроль за виконанням процесів, можливість відстеження й оптимізації, простота впровадження в централізованих системах. Проте ці моделі характеризуються і окремими недоліками, які полягають у обмеженнях щодо масштабованості та складності інтеграції в розподілені системи.

Поряд із оркестрацією хореографія передбачає децентралізовану взаємодію між сервісами, де кожен компонент виконує свої дії на основі визначених правил, не маючи єдиного центру управління. Вона подібна до танцю, у якому всі учасники знають свої ролі та взаємодіють синхронно.

Одним із популярних підходів до хореографії веб-сервісів є WSMO (Web Service Modeling Ontology), який визначає правила взаємодії між сервісами через онтології та семантичну інформацію. Ця модель відрізняється гнучкістю, масштабованістю, можливістю адаптації до розподілених систем. Проте разом із тим для неї характерна складність координації, необхідність високого рівня семантичної сумісності між сервісами. WSMO є одним з найбільш значущих фреймворків семантичних веб-сервісів, запропонованих до цього часу (рис. 1.2). Він доповнює існуючі синтаксичні стандарти веб-сервісів, надаючи концептуальну модель і мову для семантичного маркування всіх відповідних аспектів загальних веб-сервісів. Характеристики сервісу визначаються через опис, статичної

семантики та динамічної семантики у кількох документах з урахуванням метамоделі WSMO [25, с. 47].

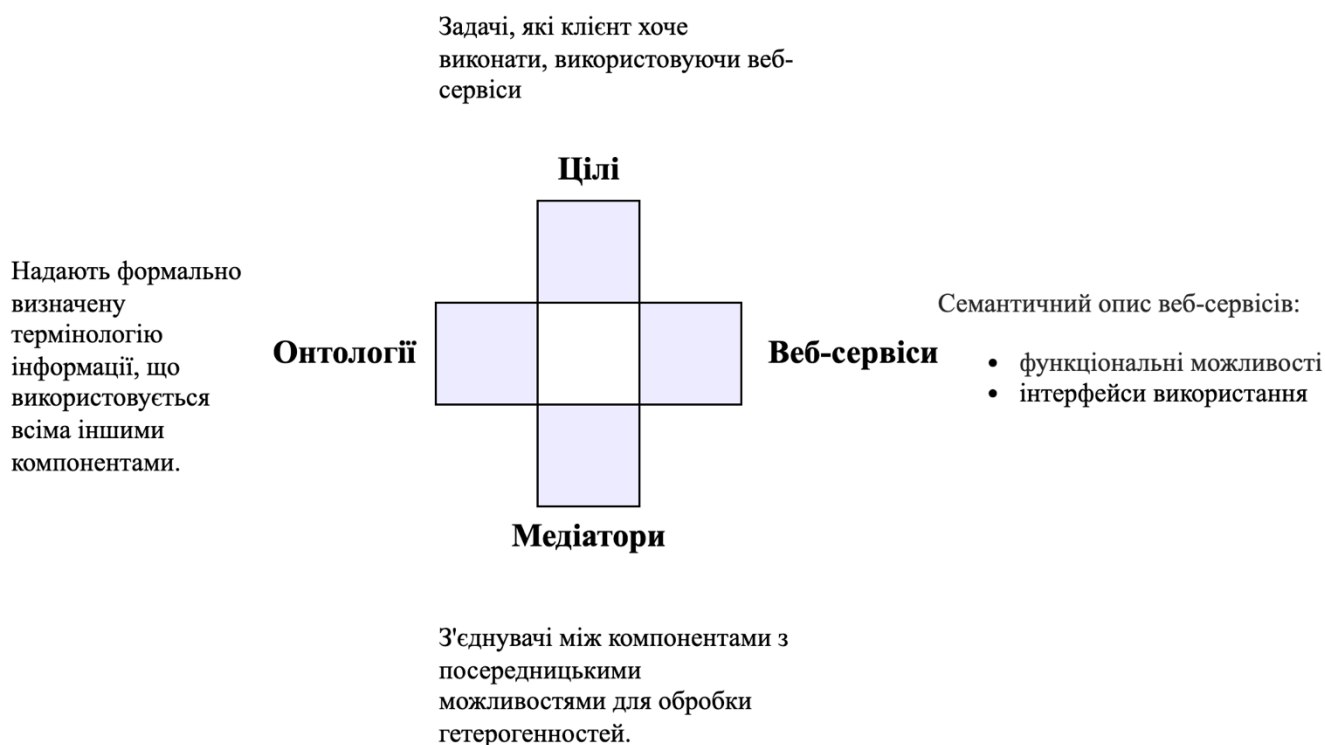


Рис. 1.2. Структура WSMO [25]

Онтології описуються у WSMO на мета-рівні. Мета-онтологія підтримує опис всіх аспектів онтологій, які надають термінологію для інших елементів WSMO. Типова онтологія складається з нефункціональних властивостей (з існуючого набору попередньо визначених основних властивостей), імпортованих онтологій та визначення понять, відносин, аксіом, функцій та екземплярів онтології. Крім того, вона вказує oo-посередники (ooMediators), які використовуються для імпорту онтологій, які вирішують проблеми підлаштування, злиття або перетворення [25].

У WSMO визначаються цілі, які можуть бути у клієнта при взаємодії з веб-сервісом. Вони складаються з нефункціональних властивостей, імпортованих онтологій, використаних посередників, після умов та ефектів. Після умови та ефекти описують стан інформаційного середовища, що потребується замовником. Онтології можуть бути безпосередньо імпортовані як термінологія для визначення

цілі, коли не потрібно вирішувати конфлікти (наприклад, неспівпадіння схеми відповіді тощо). Однак, якщо необхідне вирівнювання, злиття або вирішення конфліктів, вони імпортуються через посередників типу `ooMediators`.

Описи веб-сервісів у WSMO надають семантичний опис самостійних веб-сервісів з анотаціями, включаючи їх функціональні та нефункціональні властивості, а також інші аспекти, необхідні для взаємодії з ними. Необхідна термінологія, як і для цілей, може бути імпортована безпосередньо або через посередників типу `ooMediators`, коли потрібно вирішити конфлікти. Крім того, описується можливості (*capabilities*) та інтерфейси (*interfaces*) сервісу: можливість визначає функціональні аспекти запропонованого сервісу, моделювані в термінах передумов (*preconditions*), припущень (*assumptions*), після умов (*postconditions*) та ефектів (*effects*); тоді як інтерфейси представляють інтерфейси, орієнтовані на дані, сервісу (наприклад, ті, що визначені за допомогою WSDL), та надають деталі щодо його функціонування з точки зору його хореографії та оркестрації [25].

Поняття посередників було введено WSMO з метою вирішення проблем гетерогенності, і вони є спеціальними елементами, що використовуються для зв'язку гетерогенних компонентів, що беруть участь у моделюванні веб-сервісу. Вони визначають необхідні відображення, трансформації або зведення між зв'язаними елементами. Чотири різновиди посередників визначені як `ggMediators`, `ooMediators`, `wgMediators` і `wwMediators`.

WSMO надає онтологічні специфікації для ключових елементів семантичних веб-сервісів. Семантичні веб-сервіси спрямовані на створення інтегрованої технології для наступного покоління вебу, поєднуючи технології семантичного вебу та веб-сервісів. Це перетворює Інтернет з інформаційного сховища для людського використання на глобальну систему розподілених веб-обчислень. Таким чином, відповідні фреймворки для семантичних веб-сервісів мають інтегрувати основні принципи дизайну вебу, визначені для семантичного вебу, а також принципи дизайну для розподілених сервісно-орієнтованих обчислень у вебі [83].

Хореографія визначає послідовність та умови, за яких кілька співпрацюючих, незалежних агентів обмінюються повідомленнями для виконання завдання з

досягненням цільового стану. Мови, такі як WS-Choreography (яка базується на Pi-Calculus), описують взаємодії зі сповіщаючою стороною (якою можуть бути інші веб-сервіси, додатки або люди) та можуть складатися з кількох окремих взаємодій, чиє поєднання становить повну транзакцію. Це поєднання, разом з її протоколами повідомлень, інтерфейсами, послідовностями та пов'язаною логікою, розглядається як хореографія (рис. 1.3).

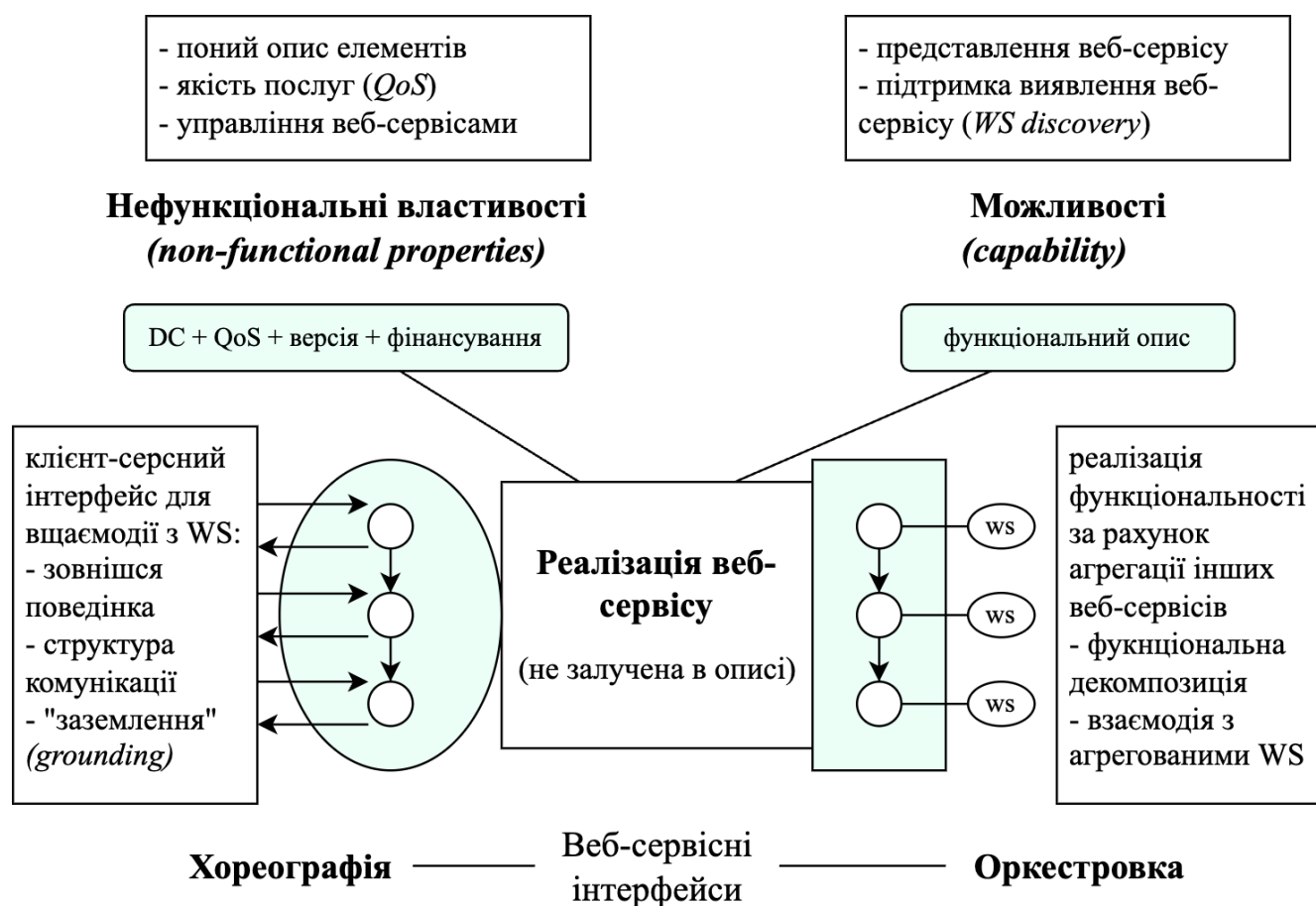


Рис. 1.3. Структура опису веб-сервісу через WSMO

Таким чином, для узагальнення, хореографія описує зразок комунікації, який дозволяє користуватися функціональністю веб-сервісу іншими клієнтами, у той час як оркестровка описує, як різні веб-сервіси повинні взаємодіяти для досягнення загальної функціональності веб-сервісу. Отже, інтерфейс хореографії в цьому прикладі має три правила переходу з урахуванням визначень термінології [25].

Наступний приклад описує хореографію системи мовою WSMO для визначення окупованої площі території Донецької області (фрагмент коду наводимо у вигляді рисунку 1.4.

```
<wsmo:Choreography>
  <wsmo:Role name="DataCollector">
    <wsmo:Interaction>
      <wsmo:Input name="GeospatialDataRequest"/>
      <wsmo:Output name="GeospatialData"/>
    </wsmo:Interaction>
  </wsmo:Role>
  <wsmo:Role name="DataProcessor">
    <wsmo:Interaction>
      <wsmo:Input name="GeospatialData"/>
      <wsmo:Output name="OccupiedArea"/>
    </wsmo:Interaction>
  </wsmo:Role>
  <wsmo:Role name="Visualizer">
    <wsmo:Interaction>
      <wsmo:Input name="OccupiedArea"/>
      <wsmo:Output name="Visualization"/>
    </wsmo:Interaction>
  </wsmo:Role>
</wsmo:Choreography>
```

Рис. 1.4. Фрагмент коду, який описує хореографію системи мовою WSMO

З метою спрощення, ми припускаємо наступну поведінку: першим кроком є отримання актуальних карт та даних про межі Донецької області та тимчасово окупованих районів, які можуть бути отримані з різних джерел, таких як державні кадастри, супутникові знімки або спеціалізовані геоінформаційні системи. Отримані дані обробляються за допомогою спеціалізованого програмного забезпечення, яке дозволяє аналізувати геопросторову інформацію. Цей етап включає визначення точних меж окупованих територій та обчислення їх площі. Після обчислення площі результати представляються у вигляді карт або звітів, що дозволяє краще зрозуміти масштаби тимчасово окупованих територій. Цей процес вимагає використання спеціалізованих інструментів та програмного забезпечення, таких як геоінформаційні системи (ГІС), для точного визначення та аналізу територій.

Як видно з опису концепцій взаємодії, сервіси повинні мають бути централізовано зареєстровані, тобто мати певну базу знань, яка містить описи всіх сервісів: онтологію запитів, перед- та післяумов, ефектів тощо. Хореографія вимагає чітко визначеного запиту від клієнта для подальшого виконання бізнес-логіки. Сучасні додатки, які оброблюють потоки даних великого обсягу працюють за асинхронною моделлю, наприклад використовуючи чергу повідомлень (рис. 1.5).

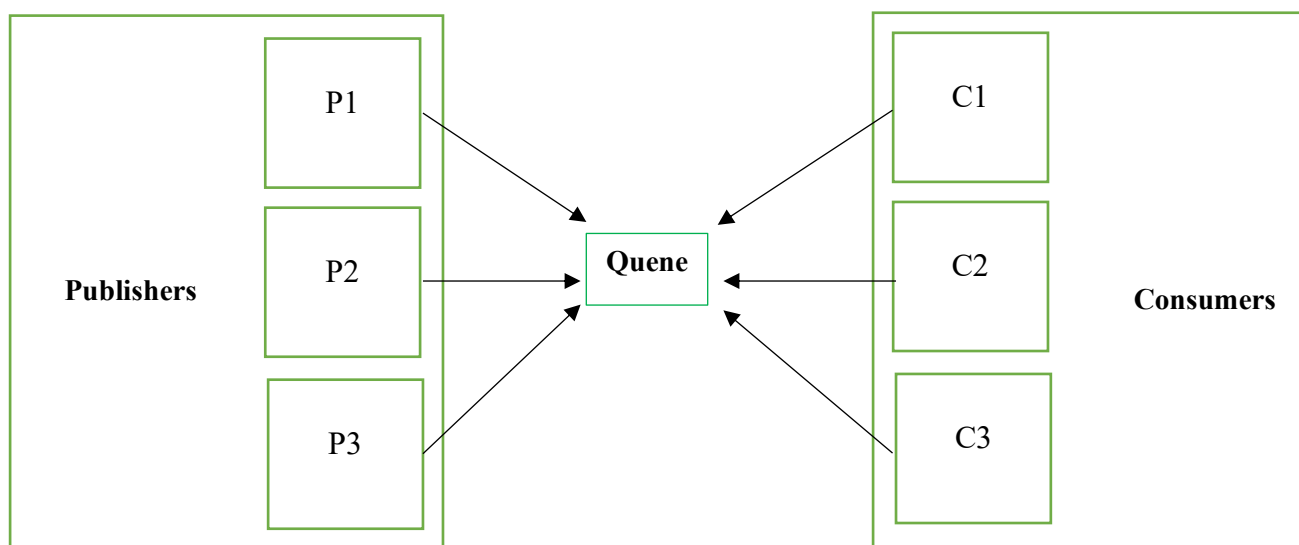


Рис. 1.5. Хореографія веб-сервісів за допомогою черги [25]

Сервіси, залучені для хореографії виступають споживачами (subscribers) даних, які очікують на нове повідомлення з черги від постачальника (publishers). Подальша оркестровка дозволяє обробити ці дані, та помістити відповідь до іншої відповідної черги, зберегти її до своєї бази даних тощо.

Отже, оскільки WSMO є одним з повноцінних доробок у сфері семантичних описів веб-сервісів, можна дійти висновку що інші фреймворки так само не мають здатності адаптації свого опису до реактивних систем або потоків даних.

1.2. Класифікація семантичних структур та моделей

Семантичні структури охоплюють широкий спектр теорій і підходів, кожна з яких пропонує унікальне розуміння того, як створюється і розуміється значення.

Когнітивна семантика, наприклад, заглиблюється в концептуальні структури, які лежать в основі мови, підкреслюючи, як ці структури формують лінгвістичну інтерпретацію. Важливим аспектом когнітивної семантики є дослідження втілення, яке розглядає, як фізичний досвід впливає на значення виразів. Цей підхід доповнюється вивченням категоризації, когнітивної евристики, яка допомагає в управлінні інформацією шляхом систематичного групування різних об'єктів, що сприяє більш ефективній обробці та розумінню. Крім того, когнітивна семантика досліджує універсальну або відносну до мови природу цих концептуальних структур, надаючи ширший погляд на взаємозв'язок мови та мислення в різних мовних контекстах. Ці аспекти семантичних структур розкривають складну взаємодію між когнітивними процесами, лінгвістичними типологіями та фізичним втіленням користувачів мови, підкреслюючи необхідність всебічного дослідження, яке з'єднує ці сфери, щоб поглибити наше розуміння семантики та її застосування.

Семантичні моделі класифікуються на кілька структур, кожна з яких має свої особливості та цілі. Однією з відомих категорій є використання онтологій, які є основоположними у визначенні комплексного набору понять і категорій у певній області. Онтології відіграють вирішальну роль не лише у визначенні цих концепцій, але й у з'ясуванні зв'язків між ними, забезпечуючи тим самим структуровану структуру, яка підтримує розширену сумісність даних та інтеграцію. На додаток до онтологій, таксономії використовуються, щоб запропонувати ієрархічну класифікацію даних, що дозволяє користувачам більш ефективно орієнтуватися та розуміти складні набори даних. Ця ієрархічна структура полегшує організацію інформації таким чином, що відображає внутрішні зв'язки між різними елементами даних, сприяючи більш інтуїтивно зрозумілим процесам пошуку даних. Разом ці класифікації семантичних моделей покращують здатність керувати великими обсягами даних і використовувати їх, підкреслюючи важливість прийняття цих структур у сферах, які потребують точної обробки даних і механізмів пошуку. Щоб оптимізувати ефективність семантичних моделей, існує потреба в постійному вдосконаленні та узгодженні цих структур із ландшафтом

даних, що розвивається, забезпечуючи їхню актуальність і надійність у вирішенні нових викликів [71].

У сфері класифікації семантичних структур і моделей були розроблені різні методології для підвищення точності та ефективності семантичної інтерпретації. Відомий підхід передбачає інтеграцію синтаксичних і семантичних методологій для формування лексично розподіленої семантичної граматичної системи. Ця інтеграція забезпечує більш детальне відображення між текстовими рядками та предикатно-структурованими системами знань, полегшуючи більш точну семантичну класифікацію. У цій структурі лексичні шаблони розглядаються як структурні відображення, які поєднують текстовий вміст із його семантичним позначенням, ефективно поєднуючи синтаксичний аналіз із семантичною інтерпретацією. Крім того, фреймворк підтримує поєднання шаблонів, пов'язаних з окремими словами в контексті, що забезпечує послідовну та узгоджену інтерпретацію більших текстових конструкцій. Ці комбіновані методології не тільки покращують ефективність класифікації, але й дозволяють детально витягувати семантичну інформацію зі складних текстових структур, що вимагає інноваційних втручань для подальшого вдосконалення цих процесів на практиці [88].

Таблиця 1.2

Класифікація семантичних структур та моделей

Критерій класифікації	Категорії	Приклади
За типом структурованості	-Онтології -Тезисаури -Реляційні моделі - Мережеві моделі	OWL, WordNet, RDF, Neo4j
За рівнем семантики	-Синтаксична семантика -Формальна семантика - Прикладна семантика	XML, Description Logic, Knowledge Graphs
За сферою застосування	-Інтеграція даних -Інтелектуальний аналіз - Автоматизація процесів	SPARQL, Knowledge Graphs, BPMN
За рівнем формалізації	-Формальні моделі -Напівформальні моделі - Неформальні моделі	OWL-DL, UML
За підходом до побудови	-Дедуктивні моделі -Індуктивні моделі - Гібридні моделі	Логічні правила, ML-алгоритми

За технологічними платформами	<ul style="list-style-type: none"> - RDF/OWL онтології - Graph-based моделі - NLP семантики 	RDF, Graph DB, онтології NLP
--------------------------------------	--	------------------------------

Джерело: [73]

Класифікація семантичних структур та моделей допомагає систематизувати різні підходи до роботи з інформацією в інтелектуальних системах. Перший критерій, тип структурованості, відображає спосіб організації даних: від високоформалізованих онтологій до гнучких мережових моделей, які дозволяють працювати із складними зв'язками між об'єктами.

За рівнем семантики виділяють три основних рівні: синтаксична семантика зосереджується на структурі даних, формальна – на їхньому точному значенні, а прикладна – на використанні цих даних у конкретних задачах. Наприклад, Knowledge Graphs є важливим інструментом прикладної семантики.

Сфери застосування охоплюють інтеграцію даних з різних джерел, інтелектуальний аналіз інформації, як у знаннях про системи, та автоматизацію бізнес-процесів із використанням семантичних технологій.

Рівень формалізації структур варіюється від повністю формальних моделей, таких як OWL-DL, до напівформальних (UML) і неформальних, що дозволяють більш інтуїтивний підхід.

Підходи до побудови поділяються на дедуктивні, індуктивні та гібридні. Наприклад, дедуктивні підходи використовують правила логіки, а індуктивні базуються на навчанні з даних.

Останній критерій – за технологічними платформами – визначає конкретні інструменти та методи реалізації. RDF/OWL онтології застосовуються для семантичного пошуку, графові моделі для роботи зі складними зв'язками, а NLP семантики інтегрують обробку природної мови з онтологіями.

Така класифікація допомагає зрозуміти, які інструменти обрати для конкретного завдання, забезпечуючи гнучкість і ефективність роботи з даними.

1.3. Аналіз когнітивних веб-сервісів

Веб-сервіси та когнітивні веб-сервіси – це два типи сервісів, що надаються через Інтернет, але вони відрізняються за своїми можливостями та функціями.

Веб-сервіси – це, по суті, програмні компоненти, доступні через мережу, що дозволяють різним програмам взаємодіяти між собою. Вони зазвичай виконують певні задачі, такі як обробка даних, доступ до баз даних, або виконання бізнес-логіки. Веб-сервіси, як правило, працюють з структурованими даними і чітко визначеними правилами.

Когнітивні веб-сервіси – це новий клас веб-сервісів, що використовують технології штучного інтелекту (ШІ), такі як машинне навчання, обробка природної мови та комп'ютерний зір, для надання більш інтелектуальних та адаптивних функцій. Вони можуть аналізувати неструктуровані дані, такі як текст, зображення та мова, розуміти контекст та надавати більш релевантні та персоналізовані результати. Іншими словами когнітивні сервіси – це набір алгоритмів машинного навчання, які розроблюються для вирішення проблем у сфері штучного інтелекту (AI). Штучний інтелект впливає на SEO, допомагаючи автоматизувати процес управління та оптимізації діяльності в галузі цифрового маркетингу [29].

Когнітивні веб-сервіси є новим досягненням в області інформаційних технологій, що призводить до штучного інтелекту та веб-сервісу для створення більш інтерактивних, адаптивних і розумних систем. Ці сервіси працюють за допомогою обробки природної мови, машинного навчання та інших передових технологій, які не можуть системам аналізувати дані, навчатися від користувачів і покращувати свої функції з часом. Когнітивні веб-сервіси мають на меті зменшити різницю між людськими очікуваннями та можливостями машин, що створюють сервіси, які можуть краще розуміти та прогнозувати потреби користувачів [7].

Штучний інтелект забезпечує систему здатності до автономного прийняття рішень, у той час як обробка природної мови дозволяє розуміти і відповідати на людські запити природним чином. Машинне навчання дає системам можливість покращувати свої функції за допомогою аналізу попередніх даних і результатів, а

аналітика великих даних у виявлених тенденціях і шаблонах інформації, яка постійно змінюється [40]. Ці технології разом створюють потужні інструменти, здатні трансформувати спосіб, яким користувачі взаємодіють з веб-сервісами, надаючи більш персоналізований та ефективний досвід.

Ключові характеристики, які використовують когнітивні веб-сервіси, включають здатність до самонавчання, адаптивність до змін умов та інтелектуальну взаємодію з користувачами. Ці сервіси здатні обробляти великі обсяги даних і аналізувати їх у реальному часі, що дозволяє їм швидко реагувати на зміни в середовищі або в поведінці користувачів. Вони також інтегрують здатність до природної обробки мови, що дозволяє користувачам взаємодіяти із системами через голосові команди або текстові запити. Крім того, когнітивні веб-сервіси можуть адаптуватися до нових вимог і вводити нові функції без необхідності втручання з боку користувача або розробника [21].

Наскрізна архітектурна схема когнітивних сервісів для великих даних є досить складною. Використовуючи архітектуру, розробники додатків можуть збагатити свої бази даних як контейнерними, так і веб-орієнтованими інтелектуальними сервісами [72].

1.3.1. Зміст когнітивних веб-сервісів

Когнітивні веб-сервіси – це передовий клас сервісів, які використовують можливості когнітивних обчислень для моделювання процесів мислення людини в комп'ютеризованій моделі. Ці сервіси визначаються їхньою здатністю розуміти, вивчати та передбачати потреби користувачів, забезпечуючи таким чином персоналізований і залежно від контексту досвід. Ключові характеристики когнітивних веб-сервісів включають адаптивність, взаємодію в реальному часі та здатність до самонавчання. Вони використовують когнітивне моделювання для аналізу та прогнозування впливу різних факторів на процеси прийняття рішень [33]. Використовуючи когнітивні карти, ці сервіси можуть створювати схематичні

зображення сприйняття окремої проблеми даної проблеми, сприяючи більш інтуїтивній взаємодії. Інтеграція цих характеристик гарантує, що когнітивні веб-сервіси не тільки швидко реагують, але й розвиваються з часом для підвищення їх точності та ефективності.

Штучний інтелект (ШІ) відіграє вирішальну роль у розробці та функціонуванні когнітивних веб-сервісів. Технології штучного інтелекту дають змогу цим сервісам приймати рішення, розпізнавати мовлення та обробляти природну мову, імітуючи когнітивні функції людини. Використовуючи штучний інтелект, когнітивні веб-сервіси можуть вирішувати складні завдання, такі як аналіз даних, розпізнавання образів і прогнозна аналітика, що є важливим для надання індивідуальної взаємодії користувачам. Застосування ШІ в цих сервісах дозволяє безперервно навчатися та адаптуватися, що досягається за рахунок використання когнітивних нейронних мереж [49, с. 149-156]. Ці мережі можуть розширити можливості веб-сервісів щодо прийняття рішень, роблячи їх ефективнішими та ефективнішими для задоволення потреб користувачів. Платформи, керовані ШІ, такі як віртуальні помічники та чат-боти, є прикладом того, як когнітивні веб-сервіси можуть забезпечувати цілодобову підтримку та взаємодію [36, с. 31-44].

Когнітивні веб-сервіси суттєво відрізняються від традиційних веб-сервісів у різних аспектах. Традиційні веб-сервіси зазвичай дотримуються попередньо визначеного набору правил і відповідають на запити користувачів на основі статичної логіки. Когнітивні веб-сервіси, навпаки ж, використовують штучний інтелект для інтерпретації та розуміння намірів користувачів, що дозволяє їм надавати більш динамічні та відповідні відповіді. На відміну від традиційних сервісів, які можуть вимагати ручного оновлення та налаштування, когнітивні веб-сервіси можуть автономно навчатися та адаптуватися до змін у поведінці та уподобаннях користувачів. Ця здатність до самонавчання дозволяє їм постійно вдосконалювати надання сервісів, підвищуючи задоволеність користувачів з часом. Крім того, когнітивні веб-сервіси включають передові методи обробки даних, які дозволяють їм обробляти великі обсяги даних і отримувати значущу інформацію,

пропонуючи тим самим вищий рівень персоналізації та взаємодії у більш стислі терміни [60].

Обробка природної мови (NLP) є наріжним каменем когнітивних веб-сервісів, відіграючи ключову роль у тому, як ці сервіси взаємодіють з користувачами. NLP – це міждисциплінарна галузь, яка поєднує в собі елементи інформатики, штучного інтелекту та математичної лінгвістики, щоб дозволити комп'ютерам розуміти людську мову та реагувати на неї осмислено. Його значення є очевидним у повсякденних програмах (до прикладу, віртуальні помічники, такі як Siri та Alexa покладаються на NLP для обробки та відповіді на голосові команди). У своїй основі NLP використовує різні алгоритми для перетворення неструктурованих даних у структуровану інформацію, що дозволяє машинам ефективно інтерпретувати та генерувати людську мову. Ця трансформація полегшує ряд додатків, від простого аналізу тексту до розуміння складної природної мови, що має вирішальне значення для розробки інтелектуальних чат-ботів і покращення взаємодії з користувачем [20; 38].

Алгоритми машинного навчання утворюють основу когнітивних веб-сервісів, дозволяючи системам навчатися на основі даних і вдосконалюватися з часом. Як підмножина штучного інтелекту, машинне навчання фокусується на розробці та застосуванні статистичних алгоритмів, які дозволяють системам ідентифікувати закономірності та робити прогнози на основі вхідних даних. Ці алгоритми є важливими для таких завдань, як навчання моделі, оцінка якості та моделювання рекомендацій, усі з яких сприяють введенню в дію та створенню інтелектуальних веб-сервісів. Використовуючи машинне навчання, когнітивні веб-сервіси можуть автоматизувати складні процеси та адаптуватися до нових даних, підвищуючи свою здатність надавати персоналізовані та ефективні рішення. Інтеграція машинного навчання не тільки підтримує розробку надійних когнітивних веб-додатків, але й прокладає шлях для прогресу в дослідженнях штучного інтелекту.

Інтеграція аналітики даних і когнітивних обчислень є трансформаційним аспектом когнітивних веб-сервісів, що забезпечує організації інструментами,

необхідними для обробки великих обсягів даних і отримання корисної інформації. Когнітивні обчислення застосовують машинне навчання та інші методи штучного інтелекту, щоб імітувати процеси людського мислення, дозволяючи системам аналізувати величезні набори даних і визначати значущі шаблони. Ця можливість має вирішальне значення для компаній, які прагнуть підвищити ефективність прийняття рішень і операцій за допомогою аналізу даних. Впроваджуючи когнітивні обчислення, організації можуть оптимізувати аналіз даних, підвищити точність прогнозів і зробити обґрунтований стратегічний вибір [44, с. 202-210]. Синергія між аналітикою даних і когнітивними обчисленнями у веб-сервісах не тільки покращує аналітичні можливості компаній, але й стимулює інновації в різних галузях.

Когнітивні веб-сервіси значно покращують роботу користувачів, надаючи персоналізовані сервіси, адаптовані до індивідуальних уподобань. Одним із яскравих прикладів є використання чат-ботів, які використовують обробку природної мови (NLP) для безперебійної взаємодії з користувачами, збираючи стандартні персоналізовані дані, щоб відповідно адаптувати відповіді та рекомендації [34]. Ця персоналізована взаємодія не тільки дає користувачам відчуття, що їх цінують, але й підвищує їхню задоволеність сервісом. Оскільки чат-боти збирають дані та вивчають попередні взаємодії, вони можуть точніше передбачати потреби користувачів, забезпечуючи динамічний та індивідуальний досвід, який розвивається з часом. Цей рівень персоналізації має вирішальне значення в сучасному цифровому середовищі, де користувачі очікують, що сервіси будуть інтуїтивно зрозумілими та відповідатимуть їхнім унікальним вимогам.

Включення когнітивних веб-сервісів у бізнес-процеси призводить до значного підвищення автоматизації та ефективності. Адже штучний інтелект (ШІ), до прикладу, відіграє ключову роль в автоматизації рутинних завдань, зменшенні ручної роботи та підвищенні продуктивності в різних областях. Звертаємо вагу й на те, що автоматизація на основі штучного інтелекту може оптимізувати такі процеси, як обслуговування клієнтів, введення даних і управління запасами, дозволяючи співробітникам зосередитися на більш стратегічних завданнях, які

вимагають креативності та критичного мислення. Крім того, інтеграція цифрових технологій оптимізує бізнес-процеси шляхом організації та автоматизації робочих процесів, що призводить до підвищення гнучкості та ефективності. Це не тільки зменшує операційні витрати, але й прискорює прийняття рішень, дозволяючи підприємствам швидко реагувати на зміни ринку та вимоги клієнтів.

Когнітивні веб-сервіси були успішно реалізовані в численних тематичних дослідженнях, демонструючи їх трансформаційний вплив на бізнес. Наприклад, посилилася інтеграція цифрових програмних рішень, що призвело до помітних покращень у функціонуванні підприємства в сучасних умовах [12, с. 3-8]. Компанії, які запровадили когнітивні веб-сервіси, спостерігають підвищення ефективності завдяки автоматизованим системам, які беруть на себе виконання повторюваних завдань і звільняють людські ресурси для вирішення складніших проблем. Крім того, використання штучного інтелекту в управлінні взаємовідносинами з клієнтами (CRM) було офіційно визнано.

Отже, когнітивні веб-сервіси використовують штучний інтелект для моделювання процесів людського мислення, дозволяючи їм розуміти, вивчати та передбачати потреби користувачів для персоналізованого досвіду. На відміну від традиційних веб-сервісів, які дотримуються статичних правил, когнітивні веб-сервіси адаптуються та розвиваються з часом, використовуючи машинне навчання та обробку природної мови для покращення взаємодії та задоволення користувачів. Ці сервіси значно підвищують ефективність роботи, автоматизуючи рутинні завдання та надаючи динамічні, залежні від контексту відповіді, остаточно змінюючи бізнес-практику та стимулюючи інновації в різних галузях.

1.3.2. Особливості та відмінності когнітивних веб-сервісів

Когнітивні веб-сервіси являють собою розширену ітерацію веб-сервісів, призначених для імітації когнітивних процесів, подібних до людських, сприяючи більш розумній та адаптивній взаємодії між комп'ютерами та користувачами. На відміну від традиційних веб-сервісів, які в основному зосереджені на обміні даними

та сумісності, когнітивні веб-сервіси включають елементи штучного інтелекту (ШІ) для аналізу, прогнозування та реагування на потреби користувачів у режимі реального часу. Ця еволюція включає в себе когнітивне моделювання, яке дозволяє системам визначати вплив різних факторів на процеси прийняття рішень [33]. Використовуючи такі технології штучного інтелекту, як обробка природної мови (NLP), машинне навчання та розпізнавання образів, когнітивні веб-сервіси спрямовані на покращення взаємодії з користувачами, надаючи більш персоналізовані та контекстно-орієнтовані рішення.

Ключові характеристики, які визначають когнітивні веб-сервіси, включають адаптивність, масштабованість і усвідомлення контексту, які відрізняють їх від звичайних веб-сервісів. Адаптивність означає здатність системи навчатися на основі взаємодії та вдосконалюватися з часом, що має вирішальне значення для задоволення різноманітних вимог користувачів. Масштабованість гарантує, що ці сервіси можуть ефективно обробляти зростаючі обсяги даних і запити користувачів без шкоди для продуктивності. Усвідомлення контексту є ще однією важливою функцією, яка дозволяє когнітивним веб-сервісам розуміти та обробляти інформацію на основі середовища та намірів користувача. Ці характеристики спільно сприяють створенню більш інтуїтивно зрозумілого та чуйного досвіду користувача, що забезпечує бездоганну інтеграцію в різні програми та домени.

Кілька технологій лежать в основі функціональності когнітивних веб-сервісів, кожна з яких сприяє їх здатності обробляти та інтерпретувати складні набори даних. Яскраві приклади включають обробку природної мови (NLP), яка дозволяє системам розуміти та створювати людську мову, роблячи взаємодію більш інтуїтивно зрозумілою. Алгоритми машинного навчання дозволяють когнітивним веб-сервісам ідентифікувати закономірності та робити прогнози на основі історичних даних, тим самим покращуючи можливості прийняття рішень. Крім того, такі технології, як комп'ютерне бачення та розпізнавання мови, розширюють сферу застосування цих сервісів, спрощуючи розширену обробку зображень і аудіоданих. Інтеграція цих технологій гарантує, що когнітивні веб-

сервіси залишаються в авангарді надання інтелектуальних та адаптивних рішень у різних секторах.

Когнітивні веб-сервіси та традиційні веб-сервіси принципово відрізняються своїми основними можливостями та функціями. У той час як традиційні веб-сервіси в першу чергу зосереджені на обміні та обробці даних на основі попередньо визначених правил, когнітивні веб-сервіси створені для імітації людського розуміння та процесів прийняття рішень. Це досягається за рахунок інтеграції штучного інтелекту, машинного навчання та технологій обробки природної мови, які дозволяють цим сервісам навчатися на основі даних, адаптуватися до уподобань користувачів і надавати контекстуалізовані відповіді [39]. Ці розширені функції дозволяють когнітивним веб-сервісам пропонувати більш персоналізований і проникливий досвід користувача, відрізняючи їх від традиційних аналогів.

Однією з найважливіших переваг когнітивних веб-сервісів перед традиційними є їх здатність забезпечувати персоналізовану та адаптивну взаємодію. На відміну від традиційних веб-сервісів, які працюють на основі статичних алгоритмів, когнітивні веб-сервіси використовують штучний інтелект і машинне навчання, щоб постійно навчатися на основі взаємодії користувачів і з часом покращувати їхні відповіді. Ця адаптивність сприяє підвищенню задоволеності та залученості користувачів, надаючи більш відповідний вміст і рекомендації. Наприклад, когнітивний веб-сервіс може аналізувати поведінку та вподобання користувачів, щоб адаптувати доставку контенту, підвищуючи загальну ефективність і результативність сервісу [22]. Ця динамічна можливість не тільки покращує взаємодію з користувачами, але й надає підприємствам цінну інформацію про поведінку споживачів.

Незважаючи на численні переваги, когнітивні веб-сервіси також стикаються з потенційними обмеженнями та проблемами, які можуть вплинути на їх широке впровадження. Однією з основних проблем є значні обчислювальні ресурси, необхідні для обробки та аналізу великих обсягів даних у режимі реального часу, що може призвести до збільшення операційних витрат. Крім того, розробка та підтримка когнітивних веб-сервісів вимагають спеціалізованих знань у галузі

штучного інтелекту та машинного навчання, які можуть бути доступні не всім організаціям. Інше занепокоєння викликають етичні наслідки використання технологій штучного інтелекту, такі як конфіденційність даних і ризики безпеки, якими необхідно ретельно керувати, щоб захистити інформацію користувачів. Ці виклики підкреслюють необхідність постійних досліджень і розробок для усунення обмежень і забезпечення сталого зростання когнітивних веб-сервісів.

Когнітивні веб-сервіси стали дуже впливовими в різних галузях завдяки використанню передових технологій, таких як штучний інтелект і машинне навчання. Одним із найбільш помітних секторів, які отримують вигоду від цих сервісів, є охорона здоров'я, де когнітивні системи підвищують точність діагностики за допомогою аналізу даних і розпізнавання образів. Роздрібна торгівля – ще одна галузь, яка значно виграє, використовуючи когнітивні сервіси для персоналізації досвіду клієнтів і оптимізації управління ланцюгом поставок. Крім того, фінансовий сектор використовує ці сервіси для виявлення шахрайства та управління ризиками, використовуючи прогнозу аналітику для передбачення ринкових тенденцій. Освіта також бачить багатообіцяючі застосування, оскільки когнітивні веб-сервіси полегшують персоналізований досвід навчання та покращують залучення студентів [54]. Ці галузі є прикладом того, як когнітивні веб-сервіси можуть трансформувати традиційні процеси, роблячи їх більш ефективними та адаптивними.

Конкретні застосування когнітивних веб-сервісів виходять далеко за рамки основних галузевих функцій, пропонуючи інноваційні рішення, адаптовані до різноманітних потреб. Наприклад, у сфері обслуговування клієнтів чат-боти на базі когнітивних технологій надають допомогу в режимі реального часу та підвищують задоволеність користувачів, навчаючись у взаємодії, щоб пропонувати дедалі релевантніші відповіді. У юридичній сфері когнітивні системи аналізують величезну кількість юридичних документів, щоб отримати відповідну інформацію, допомагаючи в підготовці справи та прийнятті рішень. Крім того, у маркетингу когнітивна аналітика розшифровує моделі поведінки споживачів, дозволяючи компаніям створювати високоцільові кампанії. Такі програми демонструють

універсальність і потенціал когнітивних веб-сервісів для революції в галузевих завданнях, забезпечуючи значну цінність завдяки розширеним можливостям.

Потенціал когнітивних веб-сервісів у майбутньому величезний завдяки новим тенденціям і постійному технологічному прогресу. Однією з ключових тенденцій є інтеграція когнітивних сервісів з Інтернетом речей (IoT), що забезпечує розумніші середовища та чутливіші системи. Ще одна багатообіцяюча розробка – розширення когнітивних можливостей в автономних системах, таких як безпілотні автомобілі та дрони, де прийняття рішень і адаптивність є вирішальними. Більше того, розвиток периферійних обчислень посилить ефективність когнітивних веб-сервісів шляхом обробки даних ближче до джерела, зменшення затримки та покращення прийняття рішень у реальному часі [35]. Ці тенденції не тільки підкреслюють зростаючий обсяг когнітивних веб-сервісів, але й підкреслюють трансформаційний вплив, який вони готові мати в багатьох областях.

Ключові відмінності між веб-сервісами та когнітивними веб-сервісами відображено у табл. 1.3.

Таблиця 1.3

Ключові відмінності між веб-сервісами та когнітивними веб-сервісами

	Веб-сервіси	Когнітивні веб-сервіси
Тип даних	Структуровані дані	Структуровані та неструктуровані дані
Обробка даних	За чітко визначеними правилами	За допомогою ШІ та машинного навчання
Функції	Виконання певних задач	Розуміння, навчання, адаптація
Приклади	API для доступу до бази даних, сервіси оплати	Розпізнавання мови, аналіз зображень, чат-боти

Аналізуючи наведену у табл. 1.2 інформацію бачимо, що традиційні веб-сервіси зазвичай працюють зі структурованими даними, такими як бази даних, тоді як когнітивні веб-сервіси можуть обробляти як структуровані, так і неструктуровані дані (наприклад, текст, зображення, аудіо). Веб-сервіси виконують завдання за чітко визначеними алгоритмами та правилами. В той же час когнітивні веб-сервіси, натомість, використовують алгоритми штучного інтелекту

та машинного навчання для навчання на основі даних та адаптації до нових умов. Веб-сервіси виконують конкретні завдання, тоді як когнітивні веб-сервіси мають здатність розуміти контекст, навчатися та адаптуватися до потреб користувачів. Веб-сервіси включають прості API та платежі, тоді як когнітивні веб-сервіси надають більш складні функції, такі як розпізнавання мови, аналіз зображень і взаємодія через чат-боти.

Когнітивні веб-сервіси та традиційні веб-сервіси принципово відрізняються своїм дизайном і функціональністю. Традиційні веб-сервіси в основному дотримуються попередньо визначеного набору операцій, зосереджуючись на сумісності та стандартизації для полегшення зв'язку між різними системами. Ці сервіси, як правило, засновані на правилах і не мають можливості динамічно пристосовуватися до мінливих умов або вводу користувача. Когнітивні веб-сервіси створені для імітації когнітивних здібностей людини, таких як навчання та міркування, що дозволяє їм розвиватися на основі нових даних і взаємодії. Ця адаптивність дозволяє когнітивним сервісам розуміти контекст, розпізнавати шаблони та самостійно приймати обґрунтовані рішення. У той час як традиційні сервіси працюють у межах суворих параметрів, когнітивні сервіси використовують передові технології, такі як машинне навчання та обробка природної мови, щоб підвищити свій інтелект і швидкість реакції [25, с. 46-52].

Таким чином, із аналізу наведених відмінностей між веб-сервісами та когнітивними веб-сервісами бачимо, що когнітивні веб-сервіси надають ряд переваг порівняно зі звичайними веб-сервісами (рис. 1.6).

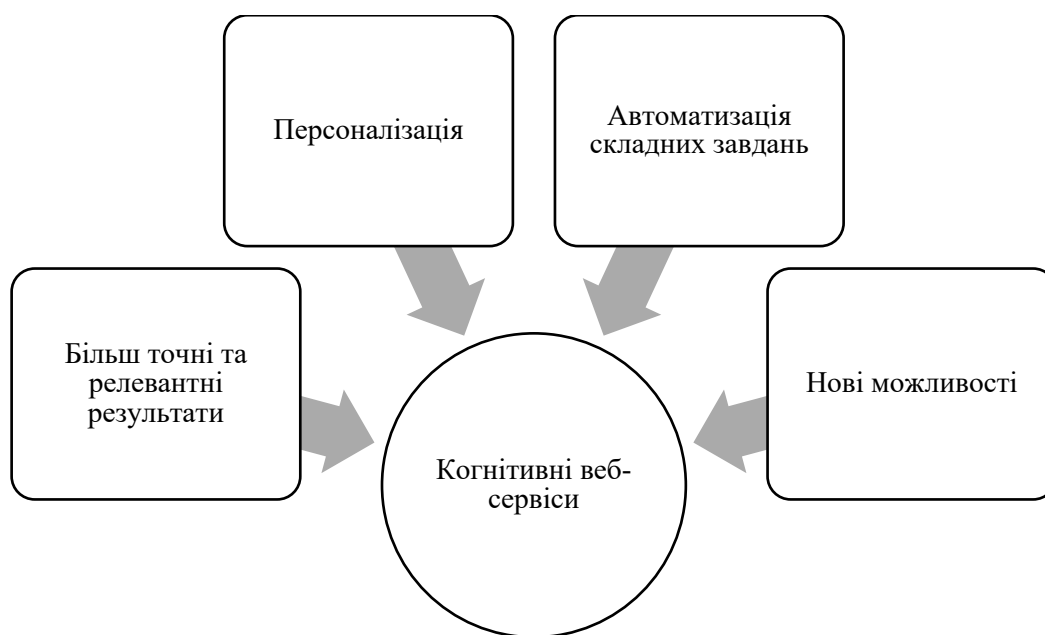


Рис. 1.6. Переваги когнітивних веб-сервісів у порівнянні із звичайними веб-сервісами

Оскільки когнітивні веб-сервіси використовують алгоритми штучного інтелекту та машинного навчання для аналізу великої кількості даних, коли відбувається пошук інформації в системі, когнітивний веб-сервіс може враховувати не лише ключові слова, але й контекст запиту, історію пошуку користувача, а також історію пошукових запитів і інших користувачів, що, як наслідок, забезпечує релевантні результати, що значно підвищує ефективність пошуку інформації.

Звертаємо увагу й на те, що когнітивні веб-сервіси здатні адаптуватися до індивідуальних потреб і вподобань користувачів. Вони аналізують поведінку користувача, його інтереси, звички та уподобання, щоб надавати персоналізований контент та рекомендації. Наприклад, в онлайн-магазині такий сервіс може пропонувати товари, які відповідають вашим попереднім покупкам або переглядам, створюючи таким чином унікальний досвід для кожного користувача.

Інша відмінність та перевага полягає в тому, що когнітивні веб-сервіси можуть автоматизувати виконання складних завдань, що зазвичай вимагають значних витрат часу та ресурсів. Наприклад, у бізнесі вони можуть обробляти великі обсяги даних, аналізувати їх та надавати рекомендації щодо стратегій прийняття рішень. Це дозволяє компаніям зосередитися на більш важливих

аспектах діяльності, знижуючи витрати та підвищуючи ефективність роботи. Крім того, когнітивні веб-сервіси можуть покращити взаємодію з користувачами, забезпечуючи персоналізовану взаємодію, оскільки вони здатні розуміти вподобання користувача та реагувати відповідно них. Цій персоналізації сприяють такі технології, як системи рекомендацій, керовані штучним інтелектом, і аналіз настроїв, які дозволяють когнітивним сервісам пристосовувати свої результати до індивідуальних потреб, тим самим підвищуючи задоволеність і залученість користувачів [28].

Завдяки використанню когнітивних веб-сервісів організації отримують доступ до нових можливостей, які раніше були недоступні, до прикладу: інноваційних рішень для аналізу даних, вдосконалених методів взаємодії з клієнтами, або нових способів оптимізації бізнес-процесів. Когнітивні сервіси також сприяють розвитку нових продуктів і сервісів, адаптуючи їх до змінюваних умов ринку і потреб споживачів, що дає змогу компаніям залишатися конкурентоспроможними.

Незважаючи на численні переваги, когнітивні веб-сервіси стикаються з потенційними обмеженнями та проблемами, які необхідно вирішити (рис. 1.7).

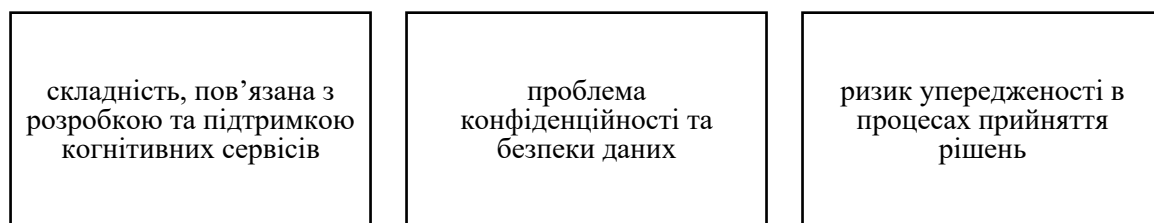


Рис. 1.7. Проблеми функціонування когнітивних веб-сервісів

Однією з першочергових проблем є складність, пов'язана з розробкою та підтримкою цих сервісів, які часто потребують складних алгоритмів і передових обчислювальних ресурсів. Крім того, існує проблема конфіденційності та безпеки даних, оскільки для ефективного функціонування когнітивні сервіси зазвичай покладаються на великий збір даних. Забезпечення захисту конфіденційної інформації при підтримці продуктивності сервісу є критичним завданням. Крім того, існує ризик упередженості в процесах прийняття рішень, якщо базові набори

даних, які використовуються для навчання когнітивних моделей, не є репрезентативними або мають недоліки. Вирішення цих проблем передбачає не тільки технічні рішення, але й етичні міркування, щоб забезпечити ефективність і відповідальність когнітивних веб-сервісів.

Отже, когнітивні веб-сервіси представляють передову еволюцію традиційних веб-сервісів, які включають штучний інтелект для забезпечення більш розумної, адаптивної та персоналізованої взаємодії з користувачем. На відміну від звичайних веб-сервісів, які зосереджуються на обміні даними на основі попередньо визначених правил, когнітивні веб-сервіси використовують такі технології, як обробка природної мови та машинне навчання, щоб аналізувати неструктуровані дані, розуміти контекст і автоматизувати складні завдання. Когнітивні веб-сервіси пропонують значні переваги, включаючи підвищену точність, персоналізацію та здатність обробляти великі обсяги даних, а також стикаються з проблемами, такими як високі обчислювальні вимоги та етичні проблеми щодо конфіденційності даних. Проте навіть попри це мають значні перспективи у майбутньому.

Висновки до розділу 1

WSMO є одним з повноцінних доробок у сфері семантичних описів веб-сервісів, можна дійти висновку що інші фреймворки так само не мають здатності адаптації свого опису до реактивних систем або потоків даних.

Класифікація семантичних структур та моделей допомагає зрозуміти, які інструменти обрати для конкретного завдання, забезпечуючи гнучкість і ефективність роботи з даними.

Веб-сервіси та когнітивні веб-сервіси – це два типи сервісів, що надаються через Інтернет, але вони відрізняються за своїми можливостями та функціями. Ключові характеристики, які використовують когнітивні веб-сервіси, включають

здатність до самонавчання, адаптивність до змін умов та інтелектуальну взаємодію з користувачами.

Когнітивні веб-сервіси використовують штучний інтелект для моделювання процесів людського мислення, дозволяючи їм розуміти, вивчати та передбачати потреби користувачів для персоналізованого досвіду. На відміну від традиційних веб-сервісів, які дотримуються статичних правил, когнітивні веб-сервіси адаптуються та розвиваються з часом, використовуючи машинне навчання та обробку природної мови для покращення взаємодії та задоволення користувачів. Ці сервіси значно підвищують ефективність роботи, автоматизуючи рутинні завдання та надаючи динамічні, залежні від контексту відповіді, остаточно змінюючи бізнес-практику та стимулюючи інновації в різних галузях.

Когнітивні веб-сервіси представляють передову еволюцію традиційних веб-сервісів, які включають штучний інтелект для забезпечення більш розумної, адаптивної та персоналізованої взаємодії з користувачем. На відміну від звичайних веб-сервісів, які зосереджуються на обміні даними на основі попередньо визначених правил, когнітивні веб-сервіси використовують такі технології, як обробка природної мови та машинне навчання, щоб аналізувати неструктуровані дані, розуміти контекст і автоматизувати складні завдання. Когнітивні веб-сервіси пропонують значні переваги, включаючи підвищену точність, персоналізацію та здатність обробляти великі обсяги даних, а також стикаються з проблемами, такими як складність, пов'язана з розробкою та підтримкою когнітивних сервісів, проблема конфіденційності та безпеки даних, ризик упередженості в процесах прийняття рішень. Проте навіть попри це мають значні перспективи у майбутньому.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ПІДХОДИ ДО МОДЕЛЮВАННЯ СЕМАНТИЧНОЇ СТРУКТУРИ

2.1. Основи семантичного вебу: застосування технологій і стандартів

Сьогодні семантичні технології стають все більш популярними та затребуваними. Насамперед це пов'язано зі зростанням кількості та складності інформації. Загальновідомо, що семантичні мережі є технологічним втіленням семантичних технологій.

Сьогодні існує багато визначень семантичної мережі. Скористаємось визначенням, представленим у науковій публікації Ю. Барташевської: «Семантична мережа – це модель предметної області, представлена у вигляді графа, де у вершинах стоять поняття, а ребра – відносини між ними» [3]. Тобто семантична мережа відображає семантику предметної області у вигляді понять та відносин, кількість типів яких визначається її автором.

Семантична павутина – це мережа знань, що складається з пов'язаних даних та інтелектуального вмісту, що дозволяє машинам інтерпретувати та обробляти вміст, метадані та інші інформаційні елементи в масштабі. Це метод, за допомогою якого комп'ютери швидко розуміють і реагують на складні людські запити на основі їх змісту. Цей рівень розуміння вимагає семантичного структурування відповідних джерел знань, що є складним процесом [4].

Основна ідея Semantic Web полягає в тому, щоб зробити інформацію, передану в Web, більше формалізованою і зручною для машинного сприйняття, зокрема, для того щоб її можна було ідентифікувати й класифікувати. На думку авторів технології Semantic Web, це може досягатися за допомогою введення метаданих, які повинні супроводжувати будь-яку інформацію й розповідати про її походження, формат, що повинно радикальним способом полегшити пошук інформації в Web і її обробку. Ґрунтуючись на відкритих стандартах, технології Semantic Web дозволяють описувати й виділяти значущу інформацію (семантику) з довільних даних, зокрема змісту документів або коду додатків. Говорячи, що

машина розуміє семантику документа, мається на увазі не тільки інтерпретація набору символів, що втримуються в документі, але й те, що машина розуміє зміст документа, тобто значення документа в цілому [8].

Технології семантичного Веб включають:

- XML (eXtensible Markup Language) – розширювана мова розмітки;
- RDF (Resource Description Framework) – мова моделювання даних для семантичної мережі. Вся інформація семантичної павутини зберігається і представляється в RDF;
- SPARQL (протокол SPARQL і мова запитів RDF) – мова запитів семантичної мережі. Він спеціально призначений для запиту даних в різних системах;
- OWL (мова веб-онтологій) – мова схеми або мову подання знань семантичної мережі. OWL дозволяє визначати поняття спільно, щоб ці поняття можна було використовувати як можна частіше [3].

2.1.1. Принципи побудови RDF моделей

Структура опису ресурсів (RDF) слугує основоположним будівельним блоком семантичної мережі, подібно до того, як HTML був для Інтернету [3].

RDF – це стандартна модель для обміну даними в Інтернеті, призначена для полегшення машинного розуміння інформаційних ресурсів та, яка дозволяє об'єднувати дані, навіть якщо базові схеми відрізняються, таким чином забезпечуючи взаємодію між різними системами.

RDF досягає цього, представляючи інформацію за допомогою трійок, які складаються з суб'єкта, предиката та об'єкта. Ця структура схожа на прості речення людською мовою, що робить її потужним інструментом для опису зв'язків між об'єктами даних (рис. 2.1).

Як відзначає у науковій публікації І. Кравчук, стандарт W3C, RDF відіграв важливу роль у розробці систем електронного навчання та інших веб-додатків,

який допомагає в точному описі та інтеграції інформації [31]. Зважаємо на те, що прийняття RDF як стандарту підкреслює його значення в поточній еволюції семантичної мережі, де він продовжує підтримувати зусилля зі створення більш взаємопов'язаної та інтелектуальної мережі даних.

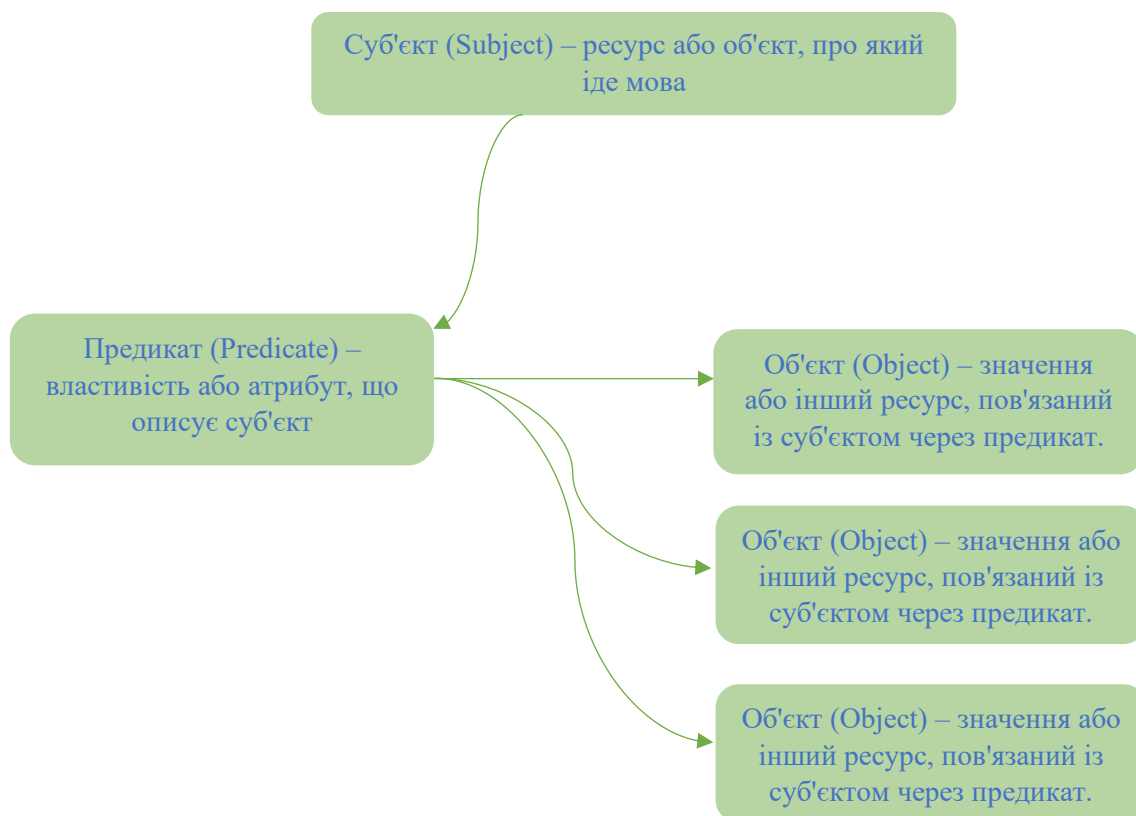


Рис. 2.1. Зразок шаблону Rdf Data Model (Resource Description Framework)

Схема RDF (RDF(S)) базується на принципах RDF, щоб забезпечити структуру для опису зв'язків між ресурсами та їхніми властивостями. Як розширення RDF, RDF(S) вводить поняття «Ресурс», «Клас» і «Властивість», які є вирішальними для визначення семантики або значення даних у семантичній мережі. Пропонуючи словник для опису груп пов'язаних ресурсів і зв'язків між ними, RDF(S) дозволяє створювати більш складні та значущі моделі даних. Ця можливість має важливе значення для розробки онтологій, які визначають предметно-специфічні знання, необхідні для просунутих семантичних веб-додатків. Також варто відзначити, що RDF(S) полегшує формулювання кінцевого результату, дозволяючи системам виводити нову інформацію на основі наявних

даних і зв'язків та, таким чином, підвищуючи здатність машин обробляти та розуміти веб-дані.

Інша специфіка моделі даних RDF полягає в тому, що ресурси й властивості ідентифікуються за допомогою глобальних ідентифікаторів (URI). RDF описує предметну область у термінах ресурсів, властивостей ресурсів і значень властивостей. Наступний рівень у піраміді технологій Semantic Web займає RDF Schema – набір класів із певними властивостями, що використовують розширювану модель даних представлення знань RDF, що містить основні елементи для опису онтологій. RDFS слугує підґрунтям опису онтологій предметної області, які дозволяють адаптувати до Web системи логіки й забезпечити семантичну обробку даних. Схема RDF являє собою систему типів для Semantic Web і дозволяє визначити класи ресурсів і властивості як елементи словника, зокрема задати, які властивості з якими класами можуть бути використані [8].

Отже, формат RDF найбільш корисний у забезпеченні спільного використання інформації, зміст якої може однаково інтерпретуватися різними програмними агентами.

2.1.2 Можливості OWL для моделювання когнітивних сервісів

Мова веб-онтології (OWL) є ще одним компонентом семантичної мережі, що забезпечує мову схем або мову представлення знань, яка покращує семантичне багатство веб-даних [3]. OWL дозволяє створювати онтології, які є формальним представленням набору понять у межах домену та зв'язків між цими поняттями. Також вона дозволяє детально описувати класи, властивості та зв'язки між сутностями, що є важливим для побудови складних моделей даних. Ця мова підтримує більшу виразність у визначенні та спільному використанні складних інформаційних структур, забезпечуючи більш складну взаємодію з даними та можливості аргументації. Дозволяючи вказувати ідеї модульним способом, OWL полегшує інтеграцію та повторне використання онтологічних компонентів у різних програмах і доменах. Ця модульність має вирішальне значення для розробки веб-

сервісів і програм, які вимагають високого ступеня гнучкості та адаптивності. Використання OWL у семантичному вебi підкреслює його роль у просуванні бачення мережі, де дані можуть безперешкодно обмінюватися та розумітися на різних платформах і контекстах.

OWL надає структуру, яка підтримує більш багату інтеграцію та взаємодію веб-вмісту, ніж це можливо з XML, RDF та RDFS [13]. Забезпечуючи точне представлення знань, OWL відіграє вирішальну роль у створенні структурованої інформації, яка може оброблятися машинами, сприяючи розробці складних когнітивних сервісів.

Неможливо переоцінити важливість OWL у покращенні моделей когнітивних сервісів. Когнітивні сервіси, які передбачають моделювання процесів людського мислення в комп'ютеризованій моделі, потребують надійних структур для представлення знань і міркувань. Як відзначають у колективній монографії В.Плескач, Ю.Рогущина, OWL робить значний внесок, забезпечуючи стандартизований підхід до моделювання даних, який підтримує машинне розуміння та міркування [45, с. 228]. Його здатність представляти складні зв'язки та ієрархії між об'єктами даних дозволяє когнітивним сервісам виконувати такі завдання, як обробка природної мови, виявлення знань і підтримка прийняття рішень з більшою точністю та ефективністю. Таким чином, використання OWL забезпечує основу для більш інтелектуальних і контекстно-залежних програм.

Застосування OWL у когнітивних обчислювальних завданнях різноманітне та ефективне. З точки зору колективу авторів (Ю.Рогущина, А.Гладун, В.Осадчий, С.Прийма) незалежно від того, чи використовується модель OWL для розширення можливостей пошукових систем чи для покращення функціональності веб-сервісів і програмних агентів, її роль є незамінною [52, с. 70-74]. З свого боку підкреслюємо, що використовуючи онтології, OWL полегшує організацію та пошук інформації, полегшуючи системам розуміння запитів користувачів і реагування на них. Також зважаємо й на те, що у контексті штучного інтелекту OWL підтримує розробку систем, які можуть міркувати про сутності та їхні взаємозв'язки, що веде до прогресу в таких сферах, як машинне навчання та інтелектуальний аналіз даних.

Таким чином, універсальність OWL гарантує, що він залишається ключовим компонентом у розвитку когнітивних обчислень.

OWL має три діалекти (у порядку зростання виразності): Lite, DL та Full. Порівняльну характеристику діалектів OWL наводимо у табл. 2.1.

Таблиця 2.1

Порівняльна характеристика діалектів OWL

Назва діалекту	Зміст діалекту	Особливості
OWL Lite	Базовий рівень виразності. Призначений для простих онтологій із чітко визначеними обмеженнями та класифікаціями.	Мінімальна складність, низька вимогливість до ресурсів, простота впровадження.
OWL DL	Підтримує логічну завершеність та можливість виведення. Орієнтований на користувачів, які потребують збалансованості між складністю онтологій та продуктивністю.	Забезпечує повну логічну строгість, оптимізований для виконання обчислень, гарантує вирішувальність задачі.
OWL Full	Найвища виразність серед діалектів. Дозволяє найбільш гнучке створення онтологій без обмежень на синтаксис.	Максимальна гнучкість, відсутність обмежень на використання синтаксису RDF, інтеграція з іншими RDF-документами.

Джерело: складено автором на основі джерела [32, с. 274; 19; 9, с. 23-43]

OWL Lite призначений для користувачів, яким потрібна спрощена версія мови веб-онтології для потреб моделювання когнітивних сервісів. Цей варіант діалекту особливо ефективний для проєктів, де простота використання та швидка реалізація мають пріоритет над розширеною виразністю та складністю. OWL Lite надає базовий набір функціональних можливостей, необхідних для побудови простих онтологій і передачі існуючих таксономій і тезаурусів [32, с. 274]. Його нижча формальна складність робить його ідеальним для користувачів, які є новачками в моделюванні онтології, і для програм. Крім того, OWL Lite часто обирають через його здатність оптимізувати процес представлення знань у менш вимогливих сценаріях, гарантуючи, що користувачі зможуть досягти своїх цілей без непотрібних ускладнень.

OWL DL є розширенням OWL Lite і пропонує збалансований підхід між виразністю та повнотою обчислень. Цей діалект спеціально розроблений для

користувачів, які потребують більш складних онтологічних структур, але все ще повинні підтримувати логічну послідовність і гарантувати, що всі висновки є обчислювально можливими [53, с. 23-33]. OWL DL підтримує ширший діапазон конструкцій, ніж OWL Lite, дозволяючи моделювати складні зв'язки та залежності в когнітивних сервісах. Його дизайн гарантує, що розробники можуть створювати надійні та деталізовані онтології, уникаючи пасток нерозбірливості. OWL DL є особливо вигідним у середовищах, де важливі точні міркування та комплексна інтеграція даних.

Для користувачів, яким потрібна максимальна гнучкість і виразність у моделюванні онтології, OWL Full представляє собою вершину діалектів OWL. Він пропонує найповніший набір функцій, уможливорюючи представлення дуже складних і нюансованих онтологій без обмежень обчислювальної повноти, які є в OWL DL [32, с. 275]. OWL Full особливо підходить для завдань, які вимагають широких можливостей представлення знань і де прийнятний компроміс між виразністю та розв'язуваністю. Цей діалект дозволяє бездоганно інтегрувати схему RDF і словники OWL, що робить його чудовим вибором для проектів, які потребують багатого та різноманітного середовища моделювання. Хоча він надає неперевершені можливості моделювання, користувачі повинні пам'ятати про його потенційні обчислювальні проблеми, особливо у широкомасштабних моделях когнітивних сервісів.

Специфічні переваги кожного діалекту OWL підкреслюють їхні сильні та слабкі сторони. Спрощений характер OWL Lite є особливо вигідним для додатків, які включають прості завдання класифікації або де швидка розробка онтології має першочергове значення. Він добре слугує у контекстах, де онтологія відносно проста і не вимагає повної виразності OWL DL або OWL Full. Навпаки, чітко визначена структура OWL DL робить його кращим вибором для складних завдань, що включають детальні онтологічні ієрархії та зв'язки, такі як ті, які можна знайти в біоінформатиці або семантичних веб-сервісах. Його здатність забезпечувати обчислювальну повноту та розв'язність має вирішальне значення для цих програм, дозволяючи складні міркування, зберігаючи передбачуваність. OWL Full

призначається для користувачів, які хочуть мати максимальну виразність і синтаксичну свободу RDF без обчислювальних гарантій. OWL Full уможливорює такі онтології, які розширюють склад зумовленого (RDF або OWL) словника [19, с. 33]. Його здатність представляти складні зв'язки та обмеження пропонує значні переваги в цих контекстах, незважаючи на підвищену обчислювальну складність.

У розробці інтелектуальних систем аналізу даних, наприклад, OWL Lite часто вибирають через його можливості швидкого розгортання та меншу складність, забезпечуючи базову онтологічну структуру без накладних витрат на складніші діалекти [32]. Комплексні можливості моделювання OWL Full використовуються у великомасштабних системах штучного інтелекту, де потреба в інтеграції різноманітних джерел даних і представленні складних шаблонів є першочерговою [19]. Ці тематичні дослідження підкреслюють важливість розуміння сильних сторін кожного діалекту OWL і вибору правильного на основі конкретних вимог і цілей поставленого завдання.

Можливості OWL для моделювання когнітивних сервісів полягають в тому, що Web Ontology Language (OWL) є потужним інструментом для створення онтологій, які допомагають організовувати, описувати і представляти знання в структурованому вигляді. Однією з ключових можливостей OWL є здатність описувати семантичні відносини між концептами, що у результаті дозволяє системам краще розуміти контекст даних і взаємозв'язки між різними об'єктами та є особливо важливим для когнітивних сервісів, які покладаються на глибоке розуміння інформації. Крім того, OWL надає можливість створення ієрархічних структур класів і підкласів. Адже можна організовувати знання в зрозумілі та логічні групи, що полегшує управління інформацією. Відзначимо, що ієрархії є важливими для когнітивних сервісів, оскільки вони дозволяють моделювати складні концепти, зберігаючи при цьому їхні властивості і зв'язки. Наприклад, у системі, що обробляє інформацію з карт, можна створити клас «Об'єкти місцевості» з підкласами, такими як «Річки», «Гори», «Міста» тощо. Такий підхід дозволить групувати географічні об'єкти за категоріями. Для запиту на обробку інформації, наприклад: *«Знайти всі міста поблизу річки Дністер»*, система може

використовувати ієрархічну структуру для швидкого визначення відповідних зв'язків між класом «Міста» і класом «Річки». Або ж, як приклад, у військовій задачі пошуку стратегічних точок можна створити клас «Стратегічні об'єкти» з підкласами «Мости», «Аеропорти», «Тунелі». Створення такої структури групування даних дозволить швидко обробляти запити, наприклад: *«Які мости розташовані в зоні 10 км від міста Львів?»*. Завдяки OWL система не лише знайде потрібні об'єкти, але й врахує їхні зв'язки з іншими елементами карти, такими як дороги тощо. Відтак, вважаємо, що цей підхід дозволяє ефективно моделювати складні інформаційні системи, забезпечуючи їхню точність і швидкість.

Однією з найпотужніших можливостей OWL є підтримка логічних запитів і інференції. У контексті аналізу даних у військовій сфері це означає, що на основі наявної інформації система може автоматично виводити нові факти, які раніше не були явно зазначені. Така особливість дозволяє створювати когнітивні сервіси, здатні адаптуватися до змін у бойових умовах і виявляти приховані залежності в даних. Наприклад, якщо система обробляє карти і володіє інформацією стосовно того, що «всі мости, розташовані на стратегічних річках, є ключовими точками», вона може автоматично визначити, що міст через річку Дністер, який використовується для логістики, також є стратегічно важливим об'єктом. На підставі здійсненого аналізу вважаємо, що завдяки таким інструментам OWL дозволяє значно підвищити ефективність планування та ухвалення рішень у відповідній сфері, забезпечуючи швидкий і точний аналіз великих обсягів інформації.

Отже, вибір між діалектами OWL залежить від специфіки завдання, вимог до логіки та обсягу даних, які потрібно моделювати. Якщо задача є простою і не вимагає складного логічного висновку, доцільно обрати OWL Lite. У випадках, коли потрібна точність і автоматизовані висновки, варто звернутися до OWL DL. Для експериментальних проектів або досліджень, де необхідна максимальна гнучкість, можна обирати OWL Full, проте потрібно бути готовими до потенційних труднощів. Таким чином, розуміння можливостей OWL та характеристик його

діалектів дозволяє ефективно вибирати відповідний підхід для моделювання когнітивних сервісів, адаптуючи його під конкретні вимоги і завдання.

2.1.3 Рушії логічного виведення OWL. Принципи та застосування

Розуміння фреймворків OWL (мова веб-онтології) має вирішальне значення для використання логічних міркувань у семантичній мережі. OWL слугує надійною мовою, яка дозволяє описувати онтології, які включають визначення класів і зв'язків між ними [51, с. 37-40]. Забезпечуючи формалізовану структуру, OWL полегшує представлення складних даних у форматі, який зчитується машинними методами. Ця можливість має важливе значення для програм, які вимагають складної обробки даних і можливостей міркування. Конструкція мови забезпечує безперебійну інтеграцію з різними інструментами міркування, дозволяючи вдосконалити процеси інтерпретації даних і дедукції. Крім того, сумісність OWL із семантичними веб-технологіями гарантує, що він відіграє ключову роль у покращенні сумісності даних на різноманітних платформах.

Дослідження семантичних веб-технологій виявило значні досягнення у взаємодії даних, які є ключовими для еволюції цифрового середовища. Технології, такі як Resource Description Framework (RDF) і OWL, дозволяють комп'ютерам інтерпретувати та обробляти дані способом, подібним до людського розуміння. Встановлюючи загальну структуру для представлення даних, семантичні веб-технології забезпечують ефективний зв'язок між різними системами, долаючи таким чином традиційні роз'єднані дані. Зважаємо на те, що у військовій сфері ці технології мають критично важливе значення. Наприклад, у системах управління бойовими діями RDF може використовуватися для інтеграції даних з різнорідних джерел, таких як карти, розвідувальні звіти та дані з дронів. Завдяки цьому командування отримує єдину, узгоджену картину ситуації на полі бою.

Використання семантичних технологій для обробки геопросторових даних може полягати у інтеграції даних про місцезнаходження військових баз, стратегічних об'єктів та зон із підвищеним ризиком може бути реалізована за

допомогою RDF і OWL. Таким чином бачимо, що семантичні веб-технології змінюють підхід до керування інформацією, дозволяючи ефективніше досягати поставлених цілей.

Методи логічного виведення, особливо в контексті OWL, відіграють життєво важливу роль у застосуванні складних систем даних. Ці методи дозволяють витягувати неявні знання з явно викладеної інформації, тим самим підвищуючи глибину та широту аналізу даних []. Такі інструменти, як Pellet, HermiT і Fact++, служать аргументами, які полегшують цей процес, виконуючи автоматизовані завдання міркування в онтологіях. Кожен інструмент має унікальні функції та сильні сторони, що дозволяє користувачам вибрати найбільш підходящий варіант на основі їхніх конкретних потреб []. У військовій сфері логічне виведення знаходить застосування у багатьох аспектах. Наприклад, такі інструменти, як Pellet, можуть використовуватися для аналізу логістичних даних з метою оптимізації маршруту постачання ресурсів.

```
PREFIX ex: <http://example.com/ontology#>
```

```
ASK {
  ex:BridgeA ex:status ex:Unusable .
}
```

```
{
  "boolean": true
}
```

Якщо відомо, що «дорога через міст А непридатна для використання через руйнування», система автоматично виведе, що «маршрути постачання, які включають цей міст, потребують перегляду».

```
SELECT ?route
WHERE {
  ?route ex:includes ex:BridgeA .
}
```

```
{
  "results": {
    "bindings": [
      {
        "route": { "value": "http://example.com/routes/Route1" }
      },
      {
        "route": { "value": "http://example.com/routes/Route2" }
      }
    ]
  }
}
```

HermiT, відомий своїми можливостями роботи з дуже складними онтологіями, може застосовуватися для аналізу розвідувальних даних. Наприклад, якщо розвідка надає інформацію, що «зона X контролюється ворожими силами», а зона Y знаходиться у 10 км від зони X, система може зробити висновок, що зона Y знаходиться під потенційною загрозою.

Нижче наводимо приклад опису онтології OWL.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:ex="http://example.com/ontology#">
  <owl:Ontology rdf:about="http://example.com/ontology"/>

  <!-- Класи -->
  <owl:Class rdf:about="ex:Zone"/>
  <owl:Class rdf:about="ex:ThreatenedZone"/>

  <!-- Властивості -->
  <owl:ObjectProperty rdf:about="ex:isControlledBy"/>
  <owl:ObjectProperty rdf:about="ex:isWithinDistanceOf"/>

  <!-- Особливості -->
  <rdf:Description rdf:about="ex:ZoneX">
    <rdf:type rdf:resource="ex:Zone"/>
    <ex:isControlledBy rdf:resource="ex:EnemyForces"/>
  </rdf:Description>

  <rdf:Description rdf:about="ex:ZoneY">
    <rdf:type rdf:resource="ex:Zone"/>
    <ex:isWithinDistanceOf rdf:resource="ex:ZoneX"/>
  </rdf:Description>
</rdf:RDF>
```

Формулюємо запит у форматі SPARQL.

```
PREFIX ex: <http://example.com/ontology#>

ASK {
  ?zone rdf:type ex:ThreatenedZone .
}
```

Таким чином, HermiT, працюючи з правилами та аксіомами, виведе «Зона Y належить до класу ThreatenedZone, оскільки вона знаходиться близько до Зони X, контрольованої ворогом».

```
{
  "boolean": true
}
```

У такому випадку система може рекомендувати зміну плану розміщення або підвищення безпеки у зоні Y.

Fast++ можна використовувати для обробки геопросторових даних. Якщо є знання стосовно того, що «стратегічні об'єкти повинні бути розташовані на відстані понад 20 км від лінії фронту», то система автоматично визначить, які об'єкти знаходяться ближче до лінії фронту, і позначить їх як критичні для переміщення або захисту.

Нижче наводимо приклад опису онтології OWL.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:ex="http://example.com/ontology#">
  <owl:Ontology rdf:about="http://example.com/ontology"/>

  <!-- Класи -->
  <owl:Class rdf:about="ex:StrategicObject"/>
  <owl:Class rdf:about="ex:CriticalObject"/>

  <!-- Властивості -->
  <owl:ObjectProperty rdf:about="ex:isLocatedAt"/>
  <owl:DatatypeProperty rdf:about="ex:distanceFromFrontLine">
    <rdfs:domain rdf:resource="ex:StrategicObject"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
  </owl:DatatypeProperty>

  <!-- Аксиома -->
  <rdf:Description rdf:about="ex:CriticalObject">
    <rdfs:subClassOf rdf:resource="ex:StrategicObject"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="ex:distanceFromFrontLine"/>
        <owl:someValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
        <owl:withRestrictions>
          <rdf:List>
            <rdf:first>
              <rdf:Description>
                <xsd:maxExclusive rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">20</xsd:maxExclusive>
              </rdf:Description>
            </rdf:first>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
          </rdf:List>
        </owl:withRestrictions>
      </owl:Restriction>
    </rdfs:subClassOf>
  </rdf:Description>

  <!-- Особливості -->
  <rdf:Description rdf:about="ex:StrategicObject1">
    <rdf:type rdf:resource="ex:StrategicObject"/>
    <ex:distanceFromFrontLine
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">15</ex:distanceFromFrontLine>
  </rdf:Description>

  <rdf:Description rdf:about="ex:StrategicObject2">
    <rdf:type rdf:resource="ex:StrategicObject"/>
    <ex:distanceFromFrontLine
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">25</ex:distanceFromFrontLine>
  </rdf:Description>
</rdf:RDF>
```

Після аналізу онтології Fact++ встановить:

- StrategicObject1 (на відстані 15 км) належить до класу CriticalObject.
- StrategicObject2 (на відстані 25 км) не належить до класу CriticalObject.

SPARQL-запит у даному випадку матиме такий вигляд:

```
PREFIX ex: <http://example.com/ontology#>

SELECT ?object
WHERE {
  ?object rdf:type ex:CriticalObject .
}
```

Тоді як результат запити буде представлено у такому форматі:

```
{
  "results": {
    "bindings": [
      {
        "object": {
          "type": "uri",
          "value": "http://example.com/ontology#StrategicObject1"
        }
      }
    ]
  }
}
```

У підсумку може бути встановлено, що StrategicObject1 позначається як критичний для переміщення або захисту. StrategicObject2 знаходиться поза критичною зоною. Як наслідок, виходячи із згенерованої інформації, система може рекомендувати: перемістити StrategicObject1 у безпечну зону або ж посилити захист об'єкта, якщо переміщення неможливе.

Як бачимо, ці інструменти виконують автоматизовані завдання міркування в онтологіях, що є надзвичайно корисним для планування, прогнозування можливих сценаріїв та аналізу ризиків. Кожен інструмент має унікальні функції, які дозволяють обирати найкращий варіант для специфічних завдань [13].

HermiT – це reasoner (обмірковувач) для OWL 2, написаний мовою Java. Він використовує обчислювальний метод гіпертаблиці (hypertableau calculus) і

підтримує перевірку наслідків (entailment checking), а також надає інші сервіси логічного виведення, такі як класифікація класів і властивостей або відповіді на запити SPARQL [13, с. 9].

Проаналізувавши наукові публікації іноземних видань, науковаців, відзначимо, що метод гіпертаблиці (hypertableau calculus) – це обчислювальний підхід, який розширює класичний метод таблиць для логічного виведення. Ключова особливість hypertableau calculus полягає в тому, що цей метод уникає генерації зайвих проміжних даних під час обчислень, що значно підвищує ефективність роботи із складними логічними моделями. У гіпертабличному обчисленні обробляються множини, а не окремі літерали, що дозволяє уникнути зайвих розгалужень у процесі доказів. Такий підхід, відповідно, робить метод особливо корисним для роботи з великими та складними онтологіями, наприклад, у рамках OWL 2 [69; 84; 82; 64].

Він спеціально розроблений для підтримки OWL 2 DL, підмови OWL, розробленої для додатків, які вимагають високої виразності та повноти обчислень. Однією з ключових особливостей Hermit є його алгоритм аргументації на основі таблиць, який дозволяє виконувати перевірку узгодженості, підтвердження класу та відповіді на запити з високою точністю та швидкістю [77]. Ця особливість робить Hermit ідеальним вибором для додатків на основі онтології в таких сферах, як біоінформатика та семантичні веб-сервіси, де точне логічне міркування є критичним. Крім того, інтеграція Hermit із інструментами редагування онтології, такими як Protégé – потужний інструмент для створення, редагування та управління онтологіями, який підтримує різні формати представлення знань, такі як OWL (Web Ontology Language) та RDF (Resource Description Framework) [54], покращує зручність використання, забезпечуючи безперебійний доступ до розробки онтології та можливостей міркувань на одній платформі.

Pellet виділяється як резонер OWL 2, оптимізований для роботи з великомасштабними онтологіями, що робить його улюбленим вибором серед розробників, які працюють із великими наборами даних. Його ефективність впливає з його здатності реалізовувати поступове міркування, яке мінімізує

обчислювальні витрати, зосереджуючись лише на змінах в онтології, а не на повторній оцінці всього набору даних [13, с. 24]. Підтримка Pellet для різних профілів OWL, включаючи OWL DL і OWL Lite, гарантує, що він може задовольняти різні вимоги міркування, зберігаючи високу продуктивність. Крім того, здатність Pellet обробляти складні запити та його сумісність із SPARQL, мовою запитів для RDF, роблять його універсальним інструментом для семантичних веб-додатків. Ці особливості в поєднанні з відкритим вихідним кодом сприяють широкому застосуванню Pellet в академічному та промисловому контекстах.

Порівнюючи Fact++ і RacerPro, кожен резонер демонструє унікальні особливості, які відповідають конкретним аспектам онтологічного міркування. Fact++ відомий своїми ефективними процесами міркування, які часто використовуються в сценаріях, де дуже важливий швидкий час відповіді. Його конструкція зосереджена на сприянні ієрархії ролей і складних виразах класів, які є ключовими в детальних структурах онтології. Fact++ – це аргументація для онтологій OWL, яка зосереджена на високопродуктивному міркуванні. Fact++ підтримує OWL 2 і забезпечує ефективні алгоритми для класифікації та перевірки узгодженості. Забезпечує високу ефективність аргументації, сумісність з різними форматами онтологій та інтеграція з різними редакторами онтологій. Fact++ може не підтримувати деякі розширені завдання міркування, які виконують інші міркувачі [10, с. 44]. З іншого боку, RacerPro – DL-резонер [10, с. 211] відомий своєю надійною підтримкою міркувань як OWL, так і RDF, пропонуючи розширені функції, які допомагають у розумінні та налагодженні онтологічних моделей. Показники продуктивності вказують на те, що хоча Fact++ вирізняється швидкістю, RacerPro надає комплексні можливості міркування, що робить його придатним для додатків, які вимагають детальних механізмів висновку та розширеного керування онтологіями. Цей порівняльний аналіз підкреслює важливість вибору резонера на основі конкретних потреб проекту, враховуючи такі фактори, як розмір онтології, складність і бажані функції міркування.

Таким чином, вибираючи інструмент міркування, важливо враховувати такі фактори, як складність вашої онтології, завдання міркування, які вам потрібно виконати, легкість інтеграції та підтримка спільноти.

Отже, в різних галузях логічні міркування (Hermit, Pellet, Fact++, RacerPro), продемонстрували значний вплив на практичних прикладах. Наприклад, у військовій сфері логічні міркування в OWL-інструментах, таких як HermiT, Pellet, Fact++ та RacerPro, суттєво підвищують ефективність аналізу даних та управління інформацією в військовій сфері, що, у свою чергу, може позитивно вплинути на результати військових операцій (табл. 2.2).

Таблиця 2.2

Застосування логічних міркувань у військовій сфері

Застосування	Інструмент	Опис
Аналіз бойової ситуації	HermiT	Інтеграція та аналіз різноманітних даних з розвідувальних систем (супутникові зображення, дані з дронів, інформація про війська противника); автоматичне виявлення загроз.
Оптимізація логістики	Pellet	Оптимізація логістичних маршрутів; швидке переосмислення маршрутів постачання у разі непригодності доріг, забезпечення своєчасного доставлення ресурсів підрозділам.
Управління ресурсами	Fact++	Класифікація та управління військовими активами; автоматичне класифікування військової техніки за типом, станом та доступністю для оптимального використання ресурсів.
Оцінка ризиків	RacerPro	Проведення аналізу ризиків у військових операціях; автоматична оцінка ризиків, пов'язаних із використанням певних типів озброєння, на основі історичних даних.
Планування операцій	HermiT, Pellet	Створення детальних планів операцій з урахуванням метеорологічних умов, розташування ворога та доступності ресурсів; ухвалення обґрунтованих рішень та адаптація стратегій.

На підставі здійсненого аналізу відзначаємо, що оцінка переваг і обмежень аргументів OWL у практичних застосуваннях розкриває складну картину. Однією з головних переваг є їх здатність міркувати над складними онтологіями з високою точністю, що важливо для забезпечення точності та узгодженості даних у різних областях. Ця можливість особливо корисна в середовищах, де цілісність даних є критичною, наприклад у сфері охорони здоров'я та фінансів. Крім того, аргументи OWL полегшують автоматизацію вилучення та представлення знань, значно

скорочуючи час і зусилля, необхідні для ручної обробки даних. Однак існують обмеження, такі як накладні витрати на обчислення, пов'язані з великомасштабними онтологіями, які можуть перешкоджати продуктивності та масштабованості. Крім того, складність впровадження та підтримки цих систем може створити проблеми, особливо для організацій з обмеженим технічним досвідом. Незважаючи на ці проблеми, переваги резонерів OWL часто переважають недоліки, що робить їх цінним надбанням у багатьох програмах.

Отже, здійснивши опрацювання наукових джерел та узагальнивши його результати, вважаємо, що майбутні перспективи та нові тенденції в розвитку інструментів міркування на основі OWL є багатообіцяючими та відображають зростаючий попит на більш складні та ефективні можливості міркування. На нашу думку, досягнення в галузі штучного інтелекту та машинного навчання зіграють значну роль у підвищенні продуктивності логічних засобів OWL, дозволяючи їм обробляти ще більші набори даних із більшою швидкістю та точністю. Крім того, прогнозуємо, що інтеграція технологій хмарних обчислень запропонує масштабовані рішення, які зможуть задовольнити потреби різноманітних галузей без шкоди для продуктивності. Таким чином формулюємо висновок стосовно того, що перелічені тенденції вказують на майбутнє, де інструменти міркування на основі OWL будуть не тільки потужнішими, але й ширше застосовуються в різних секторах, сприяючи інноваціям і ефективності в додатках, заснованих на знаннях.

2.1.4. Використання SPARQL і SQWRL для семантичних запитів

SPARQL, що розшифровується як SPARQL Protocol and RDF Query Language, є спеціалізованою мовою запитів, розробленою для доступу до даних RDF та обробки даних [10, с. 54; 13, с. 114]. Будучи мовою запитів семантичної мережі, SPARQL дозволяє користувачам ефективно отримувати та запитувати дані, що зберігаються у форматі RDF. Він забезпечує стандартизований спосіб виконання складних запитів до наборів даних, дозволяючи отримувати певні шаблони даних і агрегувати інформацію з кількох джерел RDF. Ця можливість має важливе значення для програм, які вимагають динамічного пошуку даних і обробки

інформації в реальному часі, таких як семантичні пошукові системи та інтелектуальні інформаційні системи. Пропонуючи надійні функції запитів, SPARQL полегшує дослідження та аналіз великих семантичних наборів даних, таким чином підтримуючи більш обґрунтовані процеси прийняття рішень. Впровадження SPARQL є значним прогресом у семантичній мережі, надаючи інструменти, необхідні для використання повного потенціалу даних RDF та впровадження інновацій у різних сферах [13, с. 114].

Вважаємо, що ця потужна мова запитів є важливим компонентом семантичного вебу, представляючи рекомендації W3C, призначені для полегшення доступу до пов'язаних даних. Дозволяючи користувачам виконувати складні запити до наборів даних RDF, SPARQL дозволяє отримувати та маніпулювати структурованими даними в різних доменах. Як стандартизована мова запитів, SPARQL підтримує різноманітні операції, такі як фільтрація, агрегація та перетворення, що робить його незамінним для додатків, які потребують складного аналізу та інтеграції даних [6]. Ці можливості позиціонують SPARQL як наріжну технологію в сфері семантичної обробки даних.

Синтаксис і структура SPARQL є ключовими для його функціональності та ефективності запитів до даних RDF. Загальну схему запиту SPARQL представляємо на рис. 2.2.

```
PREFIX foo: <http://example.com/resources/>
# префіксні оголошення
FROM ...
# джерела запиту
SELECT ...
# пункт результату
WHERE {...}
# критерії запиту
ORDER BY ...
# модифікатори запиту
```

Рис. 2.2. Загальна схема запиту SPARQL

**де префіксні оголошення використовуються для спрощення універсальних ресурсних ідентифікаторів (URI); джерела запиту встановлюють, які RDF графи підлягають запиту; пункт результату надає набір даних (вибірку), які відповідають умовам запиту; критерії запиту визначають, які дані потрібно отримати з базового набору; модифікатори запиту обмежують, упорядковують та змінюють результати запиту різними способами.*

За своєю суттю запити SPARQL складаються з кількох основних компонентів: SELECT, WHERE та додаткових умов, таких як FILTER і ORDER BY. Означені компоненти дозволяють користувачам вказати точні дані, які вони бажають отримати, і умови, за яких ці дані повинні бути обрані. Речення SELECT визначає змінні, які потрібно повернути, тоді як речення WHERE визначає шаблон RDF-трийок, які мають бути зіставлені в наборі даних (рис. 2.3).

```
PREFIX ex: <http://example.org/>

SELECT ?author ?name
WHERE {
  ?author a ex:Author .
  ?author ex:name ?name .
}
```

Рис. 2.3. Зразки фрагменту запити мовою SPARQL із застосуванням компонентів SELECT, WHERE

* де PREFIX визначає префікс для скорочення URI, у даному випадку для <http://example.org/>; SELECT вказує, які змінні будуть повернуті (у цьому випадку ?author та ?name); WHERE використовується для опису шаблону, за яким виконуватиметься запит.

Додаткове положення FILTER можна використовувати для уточнення результатів шляхом накладення додаткових обмежень, тоді як ORDER BY дозволяє сортувати результати запити на основі заданих критеріїв (рис. 2.4) [85].

```
PREFIX ex: <http://example.org/>

SELECT ?author ?name ?age
WHERE {
  ?author a ex:Author .
  ?author ex:name ?name .
  ?author ex:age ?age .
  FILTER(?age > 30)
}
ORDER BY ?name
```

Рис. 2.4. Зразки фрагменту запити мовою SPARQL із застосуванням компонентів FILTER, ORDER BY

* де PREFIX визначає префікс для скорочення URI; SELECT вказує змінні, які будуть повернуті (у цьому випадку ?author, ?name, і ?age); WHERE описує шаблон запити, у якому визначено, що ?author є автором, і отримуються відповідні ім'я та вік; FILTER використовується для обмеження результатів запити, в даному випадку для вибору авторів, чий вік більше 30 років; ORDER BY сортує результати за іменем автора у алфавітному порядку.

Відображений структурований підхід до формулювання запитів гарантує, що користувачі можуть ефективно орієнтуватися в наборах даних RDF, витягуючи значущу інформацію та зв'язки зі складних структур даних.

Універсальність SPARQL очевидна в його широкому спектрі варіантів використання та застосувань у різних семантичних веб-технологіях. Одне з його основних застосувань – у сфері пов'язаних даних, де він полегшує інтеграцію та запит даних із багатьох джерел, забезпечуючи більш повну та зв'язану інформацію [6]. SPARQL також використовується в доступі до даних на основі онтології, де він забезпечує механізм для запитів до баз знань, які структуровані відповідно до конкретних онтологій [5, с. 131]. Крім того, SPARQL відіграє вирішальну роль у семантичному пошуку, покращуючи здатність пошукових систем розуміти та інтерпретувати значення термінів запиту в більш широкому контексті [86].

SQWRL, або мова веб-правил, розширена семантичними запитами, відіграє ключову роль у надсиланні запитів до онтологій у рамках семантичної мережі. Це розширення OWL (мова веб-онтології), яке полегшує складні можливості запитів, таким чином дозволяючи користувачам ефективно отримувати значущу інформацію з онтологічних даних. На відміну від традиційних мов запитів, SQWRL зосереджується на складних зв'язках та ієрархіях, визначених в онтологіях, уможливлуючи більш тонкий пошук даних. Ця здатність робити запити до онтологій робить SQWRL особливо корисним у додатках, де розуміння зв'язків між різними об'єктами даних має вирішальне значення, наприклад, у системах навчальних програм та їхніх зв'язків із типами даних для об'єктів онтології [87].

Однією з ключових особливостей, що відрізняє SQWRL від SPARQL, є його здатність обробляти логіку на основі правил у запитах. У той час як SPARQL в основному призначений для запиту даних RDF, SQWRL розширює ці можливості, інтегруючи логіку на основі правил, що дозволяє більш складну маніпуляцію даними та висновки. Ця інтеграція правил із можливостями запитів означає, що SQWRL може не лише отримувати дані, але й виконувати логічні операції з даними для отримання нових ідей. Крім того, SQWRL підтримує низку агрегатних функцій, що покращує його здатність виконувати комплексний аналіз даних. Цей рівень

складності особливо корисний у програмах, які вимагають глибокого розуміння зв'язків між даними та здатності виводити нові знання з існуючих структур даних.

Порівняльний аналіз SPARQL та SQWRL наводимо у табл. 2.3.

Таблиця 2.3

Порівняльний аналіз: SPARQL та SQWRL

Характеристика	SPARQL	SQWRL
Призначення	Запит і маніпуляція даними RDF у середовищах семантичного вебу.	Запит даних в онтологіях OWL з можливістю використання складних міркувань.
Синтаксис і використання	Базується на графах RDF, потребує знань про модель даних та складний синтаксис.	Інтуїтивно зрозумілий для користувачів, знайомих із конструкціями OWL.
Область застосування	Широкий спектр завдань: зв'язані середовища даних, академічні дослідження, комерційне використання.	Специфічні сценарії з використанням онтологій, де потрібна виразність OWL.
Обмеження	Схильний до проблем із декартовим добутком, що може знижувати ефективність.	Обмежена сфера застосування та менша підтримка спільноти порівняно зі SPARQL.
Підтримка складних міркувань	Відсутня.	Забезпечує складні логічні міркування завдяки експресивності OWL.
Ефективність	Надійний і оптимізований для роботи з RDF-даними.	Оптимізований для роботи з онтологіями та виявлення контекстно-залежних зв'язків.
Основна перевага	Виконання складних запитів до RDF-графів із високою ефективністю.	Потужний інструмент для розширеного аналізу онтологій і підтримки складних моделей.

Виходячи із даних наведених в табл. 2.3, відзначаємо, що PARQL, як рекомендація консорціуму W3C, виділяється як ключова технологія семантичного вебу. Його основна перевага полягає в його здатності виконувати складні запити до даних RDF, дозволяючи користувачам витягувати та ефективно маніпулювати пов'язаними даними. Однак одним із властивих обмежень SPARQL є його обробка декартового добутку, що може призвести до неефективності під час запиту непов'язаних частин даних [15]. Хоча це можна пом'якшити шляхом формулювання добре структурованих запитів, це вимагає глибокого розуміння як моделі даних, так і синтаксису запиту. Незважаючи на це, SPARQL залишається незамінним для завдань, які вимагають запитів до наборів даних RDF, наприклад,

у зв'язаних середовищах даних, де дані представлені у вигляді графіків. Його надійна структура дозволяє отримувати детальну інформацію, що робить його придатним для широкого спектру застосувань від академічних досліджень до комерційного використання.

SQWRL, з іншого боку, пропонує унікальні переваги, які відрізняють його від SPARQL, особливо в додатках, керованих онтологіями. На відміну від SPARQL, SQWRL розроблено для бездоганної роботи з онтологіями OWL, надаючи більш інтуїтивно зрозумілий механізм запитів для тих, хто знайомий з конструкціями OWL. Це робить його особливо корисним у сценаріях, де виразність OWL потрібна для моделювання складних зв'язків і обмежень. Однак одним із потенційних обмежень SQWRL є його обмежене застосування порівняно зі SPARQL, що може призвести до зменшення ресурсів і підтримки спільноти. Тим не менш, для користувачів, які глибоко заглиблені в розробку онтології та потребують складних можливостей міркування, SQWRL надає потужний набір інструментів, який доповнює експресивну силу OWL, дозволяючи створювати більш детальні та контекстно-залежні запити.

Отже, на підставі проведеного аналізу зважаємо на те, що вибір між SPARQL і SQWRL для семантичних веб-проектів часто залежить від конкретних вимог і існуючої інфраструктури проекту. SPARQL зазвичай віддають перевагу для проектів, що включають великі пов'язані дані в різноманітних наборах даних через його широке визнання та надійну продуктивність у обробці даних RDF. Він ідеально підходить для сценаріїв, де інтеграція даних із кількох джерел є пріоритетом, завдяки можливості обробляти та повертати складні зв'язки даних. Навпаки, SQWRL найкраще підходить для проектів, орієнтованих на онтологію, де міркування над складними ієрархіями класів і зв'язками є вирішальними. Наприклад, у семантичному веб-проекті, який значною мірою покладається на онтології OWL для визначення складних моделей домену, SQWRL пропонує необхідну виразність для виконання точних запитів. Таким чином, вибір між SPARQL і SQWRL повинен ґрунтуватися на характері даних, складності необхідних запитів і загальних цілях проекту.

2.2. Методи оркестрації: принципи, переваги та недоліки

Оркестровка веб-сервісів відноситься до систематичної організації та координації складних комп'ютерних систем, проміжного програмного забезпечення та сервісів. За своєю суттю оркестровка полягає в управлінні взаємодіями та залежностями між різними компонентами в системі для досягнення злагодженого робочого процесу. Такий підхід часто передбачає автоматизацію конфігурації, керування та координацію підсистем для забезпечення ефективної роботи. Принципи оркестровки ґрунтуються на сервісно-орієнтованій архітектурі (SOA), де окремі сервіси створюються для створення програми, що дозволяє створювати модульні та масштабовані рішення. Інтегруючи ці сервіси, оркестровка сприяє безперебійному спілкуванню та обміну даними між різними платформами та середовищами. Визначена інтеграція має ключове значення для створення гнучкої та оперативної ІТ-інфраструктури, яка може адаптуватися до мінливих потреб бізнесу.

У сучасній сфері розробки програмного забезпечення мікросервісна архітектура зарекомендувала себе як ефективний підхід для створення складних та масштабованих систем. Основна концепція мікросервісної архітектури полягає в розбитті програмного забезпечення на невеликі, незалежні один від одного сервіси, кожен з яких виконує окрему функцію. Такий підхід дозволяє покращити гнучкість та масштабованість системи, проте створює нові виклики, пов'язані з керуванням та координацією великої кількості сервісів. Саме тому, як вважають V.Diulher, A.Sorokin методи оркестрації мікросервісів є ключовими у забезпеченні стабільної та ефективної роботи таких систем [65, с. 58].

Ключовим компонентом оркестровки є ПЗ оркестровки або оркестратор (англ. Orchestrator). Її ключовою функцією є забезпечення робочих процесів для виконання автоматизованих завдань з метою досягнення бажаного результату. Робочий процес – це набір взаємопов'язаних завдань (англ. Tasks), які виконують бізнес-операцію. Програмне забезпечення оркестровки дозволяє автоматизувати організацію, координацію та керування завданнями. Це допомагає групувати та

визначати послідовність завдань із залежностями між ними в єдиний автоматизований робочий процес (workflow) [16, с. 43].

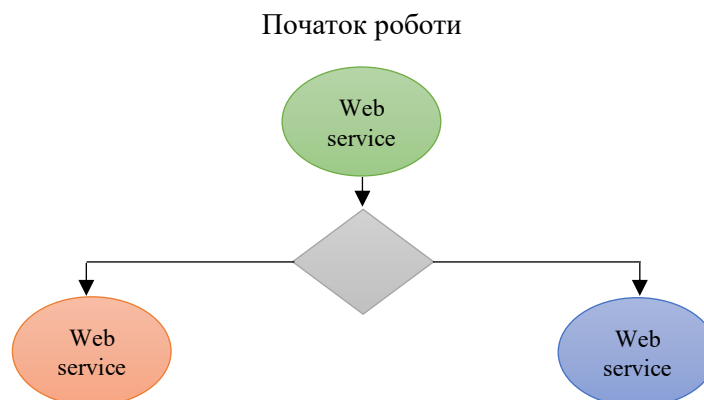


Рис. 2.5. Схема моделі оркестрації веб-сервісів

Одним з найпоширеніших інструментів для оркестрації мікросервісів є Kubernetes, розроблений Google. Kubernetes забезпечує автоматизоване розгортання, масштабування та управління контейнеризованими застосунками. Його основні функції включають управління контейнерами, автоматичне балансування навантаження, моніторинг стану сервісів та автоматичне відновлення після збоїв. Kubernetes використовує декларативний підхід до управління конфігураціями, що дозволяє спростувати процес керування складними системами (рис. 2.6) [61].

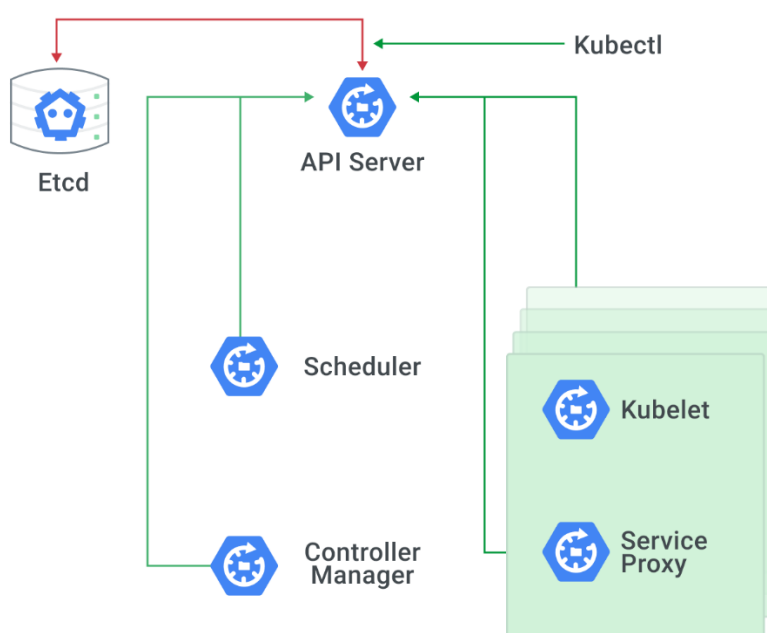


Рис. 2.6. Ключові елементи архітектури Kubernetes [90]

Іншим важливим інструментом оркестровки є Docker Swarm, який інтегрується з Docker та забезпечує оркестрацію контейнерів у кластері. Docker Swarm спрощує процес розгортання та управління контейнерами, надаючи можливість автоматичного масштабування та балансування навантаження. Хоча Docker Swarm поступається Kubernetes у функціональності, він є менш складним у налаштуванні та управлінні, що робить його привабливим для менших проєктів (рис. 2.7) [67].

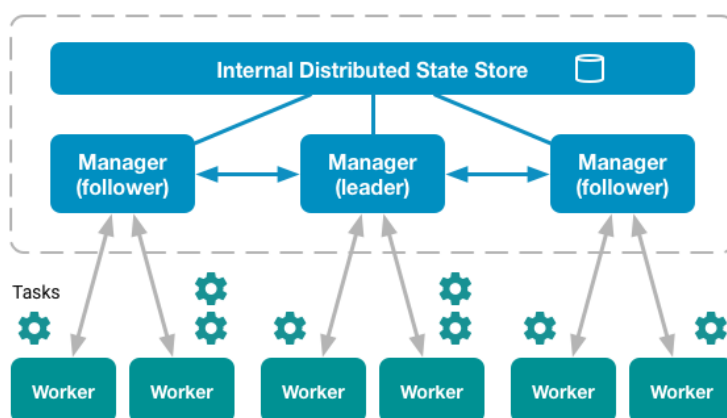


Рис. 2.7. Ключові елементи архітектури Docker Swarm [67]

Apache Mesos є ще одним інструментом для оркестрації, який забезпечує розподіл ресурсів та управління контейнерами. Mesos може працювати з різними типами контейнерів, такими як Docker та Mesosphere. Його особливість полягає в високій масштабованості та можливості інтеграції з іншими інструментами, такими як Marathon для оркестрації застосунків (рис. 2.8) [77, с. 88].

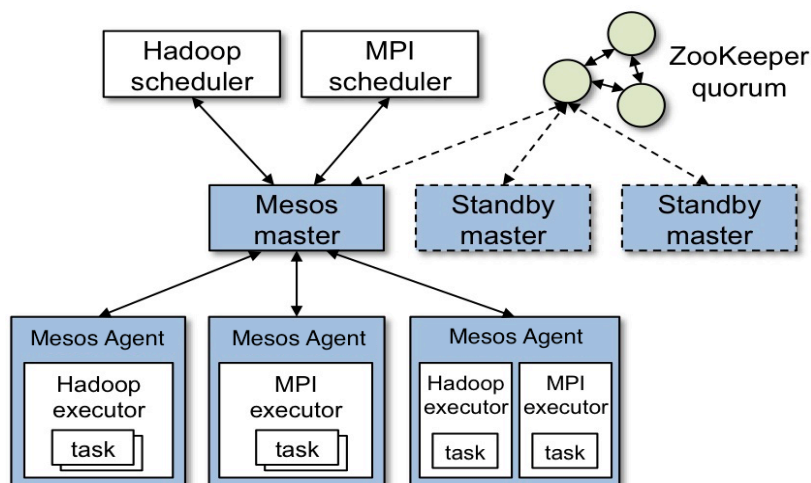


Рис. 2.8. Ключові елементи архітектури Apache Mesos [80]

Ключові інструменти розглянуті нами є важливими у її процесі, оскільки забезпечують узгоджену роботу різних підсистем для досягнення бажаних результатів. Вони ж зазвичай включають оркестратори, які слугують центральним рівнем керування, відповідальним за керування завданнями та робочими процесами в різних системах.

Важливим компонентом процесу оркестрації є реєстр сервісів, який підтримує базу даних доступних сервісів і їх розташування, що дозволяє оркестраторам динамічно розподіляти ресурси за потреби. Крім того, інструменти моніторингу відіграють життєво важливу роль у оркеструванні, надаючи інформацію про продуктивність і стан системи в режимі реального часу, дозволяючи проактивне керування та усунення несправностей. У своїй сукупності ці компоненти утворюють основу системи оркестровки, забезпечуючи плавну та ефективну роботу. У результаті організації можуть оптимізувати свої процеси та покращити надання сервісів.

Основною перевагою сучасних методів оркестрації є автоматизація процесів, що значно знижує навантаження на розробників та адміністративний персонал. Автоматичне розгортання та масштабування дозволяють швидко реагувати на змінні навантаження та забезпечувати стабільну роботу системи. Моніторинг та автоматичне відновлення після збоїв покращують надійність та доступність сервісів.

Автоматизація є фундаментальним аспектом оркестровки, оптимізації операцій і зменшення потреби в ручному втручанні. Завдяки автоматизації рутинних завдань оркестровка не тільки підвищує ефективність, але й мінімізує ризик людської помилки, що забезпечує більш надійну роботу системи. Інструменти автоматизації часто інтегруються в платформи оркестровки, щоб забезпечити автоматичне розгортання, масштабування та керування програмами в різних середовищах. Відзначимо, що можливість автоматизації оркестрації особливо важлива в динамічних хмарних середовищах, де робоче навантаження може швидко коливатися. Таким чином, автоматизація оркестровки слугує для узгодження ІТ-ресурсів з бізнес-цілями, гарантуючи, що системи одночасно

реагують і стійкі [59]. Отже, організації можуть швидко реагувати на зміни попиту, зберігаючи конкурентну перевагу на ринку.

Впровадження методів оркестровки значно підвищує ефективність і продуктивність робочих процесів за рахунок оптимізації і зменшення ручного втручання. Автоматизуючи рутинні завдання та координуючи складні операції, оркестровка дозволяє більш бездоганно інтегрувати різні компоненти в певний робочий процес. Також такий підхід мінімізує людські помилки та прискорює виконання завдань, забезпечуючи оптимальне використання ресурсів і виконання проектів за розкладом. Зокрема, підвищення продуктивності досягається за рахунок скорочення часу та зусиль, необхідних для виконання повторюваних завдань, що дозволяє командам зосередитися на стратегічних ініціативах та інноваціях. Результатом є більш гнучке та чутливе операційне середовище, де зміни можна впроваджувати швидко, не порушуючи поточну діяльність.

Методи оркестровки забезпечують покращену масштабованість і гнучкість операцій, дозволяючи організаціям адаптуватися до мінливих вимог і легко масштабувати свої процеси. Динамічний розподіл ресурсів, що забезпечується оркестровкою, гарантує, що системи можуть обробляти різні робочі навантаження без зниження продуктивності. Для підприємств, які відчувають зростання чи коливання попиту, така масштабованість є надзвичайно важливою, оскільки вона запобігає вузьким місцям і підтримує якість обслуговування. Крім того, гнучкість, яку пропонує оркестровка, означає, що системи можна легко переконфігурувати для адаптації до нових технологій або процесів, сприяючи інноваціям і конкурентній перевазі.

Ефективне управління ресурсами та економічна ефективність є ключовими перевагами використання методів оркестровки. Завдяки оптимізації розподілу та використання ресурсів оркестровка мінімізує відходи та максимізує віддачу від інвестицій за рахунок інтелектуального планування та розподілу ресурсів, які гарантують, що жоден компонент не буде використаний недостатньо або перевантажений. У результаті організації можуть досягти значної економії за рахунок зменшення непотрібних витрат і підвищення ефективності роботи. Крім

того, оркестровка забезпечує видимість використання ресурсів, уможливлюючи краще планування та прогнозування, що додатково сприяє економічно ефективному управлінню. Такий свого роду стратегічний підхід до управління ресурсами не тільки підтримує стає зростання, але й покращує загальний фінансовий стан організації.

Однак, існують і певні недоліки. Використання інструментів оркестрації вимагає додаткових ресурсів та навичок [48].

Складність у реалізації та підтримці методів оркестровки є одним із їхніх найбільш істотних недоліків. Ці системи часто вимагають глибокого розуміння базових технологій і складних конфігурацій для ефективного функціонування. Оскільки організації прагнуть автоматизувати та оптимізувати свої процеси, вони можуть зіткнутися з труднощами через багатогранність платформ оркестровки. Розробники та ІТ-спеціалісти повинні проходити складні процедури налаштування та поточні оновлення системи, що може призвести до збільшення часу та ресурсів, витрачених на навчання та усунення несправностей. Крім того, технічна складність може створити перешкоду для входу на ринок менших організацій, які можуть не мати необхідного досвіду чи бюджету для керування такими складними системами.

Методи оркестровки також можуть створювати потенційні вразливості безпеки та ризики, якщо ними не керувати належним чином [56]. Оскільки ці системи розроблені для автоматизації та інтеграції різних компонентів ІТ-інфраструктури організації, вони створюють численні точки взаємодії, які можна використати, якщо вони не захищені належним чином. Кожна точка входу створює можливість для несанкціонованого доступу, витоку даних або інших кіберзагроз. Попри це динамічний характер оркестровки означає, що заходи безпеки необхідно постійно оновлювати та контролювати, щоб усунути нові загрози. Якщо цього не зробити, це може призвести до серйозних порушень безпеки, загрози конфіденційних даних і критичних операцій. Тому організації обов'язково повинні інвестувати в надійні протоколи безпеки та регулярні аудити, щоб забезпечити цілісність і конфіденційність свого організованого середовища.

Інтеграція методів оркестровки з існуючими системами та технологіями створює ще одну помітну проблему [65, с. 58-60]. Організації часто покладаються на поєднання застарілих систем і сучасних технологій, що може створити проблеми сумісності під час впровадження рішень оркестровки. Потреба в бездоганній інтеграції може перешкоджати різним архітектурним дизайнам, форматам даних і протоколам зв'язку. Такі своєрідні перешкоди можуть призвести до збільшення складності в досягненні сумісності, вимагаючи додаткового часу та ресурсів для модифікації існуючих систем або розробки спеціальних інтерфейсів. Крім того, процес інтеграції може порушити поточні операції, що призведе до тимчасових простоїв або зниження ефективності. Окрім цього зважаємо на те, що складність налаштування та управління великими кластерами може призвести до помилок та збоїв у роботі системи. Незважаючи на це, переваги значно переважають недоліки, що робить оркестрацію невід'ємною частиною сучасних мікросервісних архітектур [65, с. 60].

Отже, аналіз сучасних методів оркестрації мікросервісів демонструє, що такі інструменти, як Kubernetes, Docker Swarm і Apache Mesos, значно підвищують гнучкість, масштабованість і надійність складних систем, автоматизуючи процеси розгортання, масштабування та управління контейнеризованими застосунками. Оркестрація є ключовим елементом сучасної IT-інфраструктури, забезпечуючи стабільну та ефективну роботу мікросервісних архітектур.

2.3. Методи хореографії: підходи та практичне застосування

Існуючі підходи до хореографії веб-сервісів є ефективним методом проектування та виконання складних взаємодій сервісів. У цих підходах використовуються формальні моделі, такі як UML (уніфікована мова моделювання) і BPMN (модель і нотація бізнес-процесу), щоб створити візуальне представлення хореографії. Побудовані на правилах підходи до хореографії веб-сервісів включають впровадження набору правил і логіки для керування взаємодією між різними сервісами.

UML (Unified Modeling Language) – уніфікована мова моделювання, що використовується розробниками програмного забезпечення для візуалізації процесів та роботи систем. Проте, вважаємо, що це скоріше не мова програмування, а набір правил та стандартів для створення діаграм. Вони дозволяють розробникам програмного забезпечення та інженерам «говорити однією мовою», не заглиблюючись у фактичний код свого продукту. Складання діаграм за допомогою UML – це чудовий спосіб допомогти іншим швидко зрозуміти складну ідею чи структуру. UML-діаграми поділяються на 14 типів (рис. 2.9).

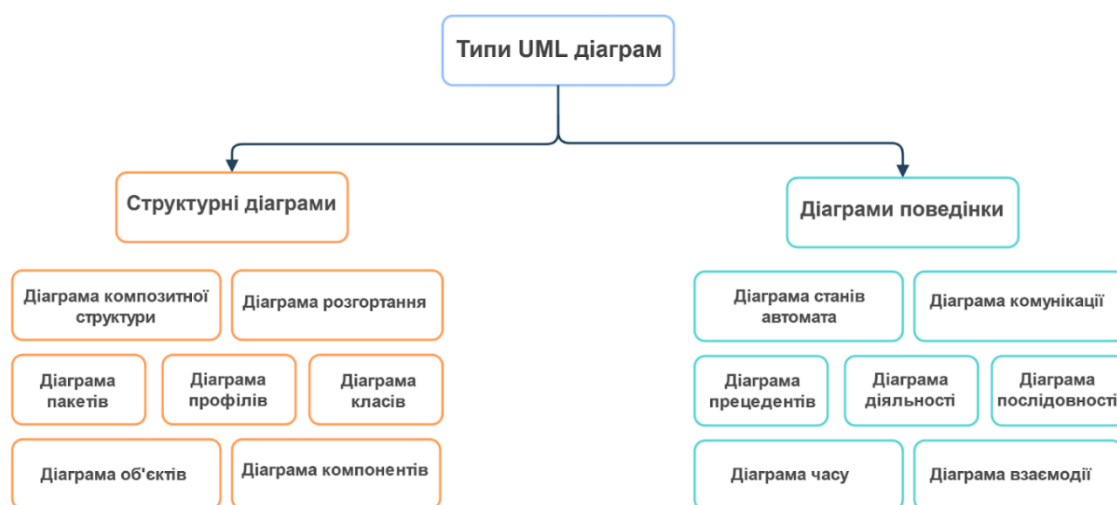


Рис. 2.9. Типи UML-діаграм [24]

Модель і нотація бізнес-процесів (BPMN) відіграє вирішальну роль у подоланні розриву між бізнес-підрозділами та ІТ-відділами під час впровадження хореографії у веб-сервіси. BPMN надає мову візуального моделювання, яка є інтуїтивно зрозумілою для бізнес-професіоналів, але достатньо детальною для впровадження ІТ-фахівцями. Пропонуючи стандартизований спосіб планування бізнес-процесів, BPMN гарантує, що всі зацікавлені сторони мають чітке розуміння того, як взаємодіють сервіси та перебіг процесів. Це загальне розуміння є життєво важливим для успішної хореографії, оскільки воно забезпечує безперебійне спілкування та співпрацю між різними командами всередині організації. Здатність BPMN перетворювати діаграми бізнес-процесів у виконуваний код робить його важливим інструментом для синхронізації виконання бізнес-процесу з бізнес-цілями. Використання BPMN в хореографії веб-сервісів не тільки підвищує чіткість

і ефективність, але також підтримує гнучкість, необхідну для адаптації до мінливого бізнес-середовища.

Мова опису хореографії веб-сервісів (WS-CDL) – це мова, призначена для опису спільної поведінки у середовищі веб-сервісів [74]. Вона є основним інструментом для визначення взаємодії та співпраці між веб-сервісами. WS-CDL полегшує проектування складних сервісних зв'язків, надаючи формальну структуру для опису послідовності та умов взаємодії [43, с. 36].

Основною перевагою WS-CDL є його здатність охоплювати хореографію з глобальної точки зору, пропонуючи комплексне уявлення про весь процес взаємодії. Вважаємо, що ця перспектива є важливою для забезпечення того, щоб усі сервіси-учасники дотримувалися узгоджених шаблонів взаємодії, тим самим зменшуючи ймовірність помилок і невідповідностей у взаємодії між сервісами.

Мова виконання бізнес-процесів (BPEL) в основному використовується для оркестровки веб-сервісів, але вона також пропонує розуміння відмінностей між оркестровкою та хореографією. На відміну від хореографії, яка зосереджується на спільному аспекті взаємодії сервісів, оркестровка більше стосується управління виконанням процесів з централізованої точки зору. BPEL дозволяє розробникам визначати складні бізнес-процеси, вказуючи послідовність і логіку взаємодії з єдиної точки управління.

Методи хореографії у веб-сервісах описують способи організації та координації взаємодій між різними сервісами, щоб досягти цілісної та ефективної роботи розподілених систем (табл. 2.4).

Таблиця 2.4

Характеристика методів хореографії у веб-сервісах

Метод хореографії	Опис	Галузь практичного застосування	Приклад практичного застосування
Централізована хореографія	Усі взаємодії між сервісами контролюються центральним модулем або координатором, який відповідає за синхронізацію їх дій.	Оркестрація API	управління процесом оформлення замовлення в e-commerce системах
		Моніторинг	контроль виконання транзакцій і виявлення помилок

Продовження табл. 2.4

Децентралізована хореографія	Сервіси взаємодіють безпосередньо між собою на основі заздалегідь визначених протоколів і контрактів.	Системи мікросервісів	взаємодія в потокових сервісах (наприклад, Netflix)
		IoT-платформи	автономний обмін даними між пристроями
Подійно-орієнтований підхід	Сервіси реагують на події, генеровані іншими сервісами, за допомогою механізмів публікації та підписки.	Обробка транзакцій	автоматичне повідомлення інвентаризації та доставки про платіж
		Масштабування	AWS Lambda для обробки подій у реальному часі
Контрактно-орієнтований підхід	Сервіси взаємодіють на основі формальних контрактів, що описують їхні обов'язки та правила обміну даними.	B2B-інтеграція	CRM і ERP-системи обмінюються даними через стандартизовані контракти
		Інтеграція SaaS-сервісів	Як приклад, Google Workspace
Орієнтація на доменний підхід	Зосереджує увагу на бізнес-доменах, де кожен домен відповідає за конкретну частину бізнес-логіки.	Фінансові платформи	поділ на домени "клієнти", "рахунки", "платежі"
			Розподілені облікові системи для масштабованості та автономності
Транзакційна хореографія	Організація довготривалих транзакцій між сервісами із забезпеченням узгодженості даних через патерни компенсації.	Бронювання квитків	відкат операцій у разі помилок
		Логістичні системи	узгодженість між постачальниками та транспортними компаніями
Автоматизація бізнес-процесів	Використання BPMN для створення сценаріїв хореографії, які включають ролі, завдання та сервіси.	Workflow-системи	автоматизація найму персоналу
		Інтеграція в DevOps	управління процесами CI/CD

Метод централізованої хореографії передбачає, що всі сервіси координуються центральним модулем, який визначає порядок і спосіб їхньої взаємодії. Такий підхід дозволяє легко контролювати процеси, оскільки всі дії записуються і моніторяться в одному місці. Наприклад, в інтернет-магазинах центральний координатор може обробляти замовлення, керувати оплатою, перевіряти наявність товару на складі та передавати дані для доставки.

При децентралізованій хореографії сервіси працюють незалежно і взаємодіють один з одним без посередника. Вони обмінюються інформацією за допомогою протоколів або контрактів, що забезпечує більшу гнучкість і

відмовостійкість. Наприклад, у потокових сервісах, як Netflix, модулі взаємодіють напрямую, обмінюючись запитами щодо рекомендацій або інформації про користувача.

Подійно-орієнтований підхід полягає в тому, що сервіси реагують на події, які генеруються іншими компонентами. Ця система часто використовує механізми публікації і підписки, що дозволяє швидко реагувати на зміни. До прикладу, якщо клієнт здійснює платіж, система доставки отримує сигнал і починає процес пакування та відправлення товару.

Контрактно-орієнтований підхід полягає в тому, що контракти визначають правила взаємодії між сервісами, що гарантує узгодженість обміну даними та, як наслідок, підвищує сумісність і запобігає помилкам. Наприклад, CRM-система компанії може без проблем взаємодіяти з ERP-системою, використовуючи стандартизовані формати для передачі даних, наприклад, інформації про клієнтів чи замовлення.

Орієнтація на доменний підхід (DDD) зосереджується на поділі системи на незалежні бізнес-домени, кожен з яких виконує конкретну роль та, у свою чергу, забезпечує зручність розробки та масштабування. Наприклад, фінансова платформа може поділятися на окремі модулі для роботи з клієнтами, обліковими записами та транзакціями, які взаємодіють через API.

Транзакційна хореографія – це метод, який передбачає управління тривалими транзакціями між сервісами, включаючи механізми відкату у разі помилки. До прикладу, під час бронювання квитків система забезпечує синхронізацію між модулями бронювання, оплати та підтвердження. Якщо один із них не виконав завдання, транзакція скасовується.

Автоматизація бізнес-процесів – підхід використовує інструменти моделювання процесів, такі як BPMN, для створення схем роботи сервісів, автоматизуючи складні процеси. Завдяки цьому можна забезпечити прозорість і ефективність виконання завдань.

Наводячи інші приклади застосування методів хореографії відзначаємо, що, до прикладу, у сфері електронної комерції практичне застосування хореографії веб-

сервісів значно покращило взаємодію з клієнтами, забезпечивши повну інтеграцію різноманітних сервісів. Можливість координації кількох сервісів, таких як обробка платежів, управління запасами та логістика доставки, забезпечує оптимізовану роботу користувача, яка є одночасно ефективною та зручною. Наприклад, коли клієнт робить покупку, хореографія веб-сервісів забезпечує автоматичну синхронізацію кожного компонента транзакції, мінімізуючи ймовірність помилок і затримок. Такий приклад інтеграції не тільки підвищує задоволеність клієнтів, але й підвищує ефективність роботи. Застосовуючи такі технології, як WS-CDL і BPMN, платформи електронної комерції можуть подолати розрив між складними бізнес-процесами та ІТ-рішеннями, що зрештою призведе до більш узгодженої та чутливої пропозиції сервісів.

Системи охорони здоров'я дедалі частіше застосовують хореографію веб-сервісів для координації управління даними пацієнтів між кількома сервісами, тим самим покращуючи догляд за пацієнтами та ефективність роботи. Завдяки інтеграції різноманітних медичних сервісів, таких як електронні записи про стан здоров'я, результати лабораторних досліджень і планування прийому, надавачі медичних сервісів можуть забезпечити точне та оперативне оновлення інформації про пацієнтів на всіх платформах. Цей цілісний підхід сприяє кращому спілкуванню між медичними працівниками та покращує загальну якість медичної допомоги.

У сфері управління ланцюгами поставок впровадження хореографії веб-сервісів зробило революцію в ефективності логістики, дозволивши взаємопов'язаним веб-сервісам функціонувати як єдине ціле. За допомогою хореографії таких сервісів, як обробка замовлень, відстеження запасів і планування доставки, менеджери ланцюга постачання можуть оптимізувати операції та скоротити час виконання. Цей взаємозв'язок дозволяє обмінюватися даними в режимі реального часу, гарантуючи, що кожен сегмент ланцюга постачання оновлюється найновішою інформацією, що є життєво важливим для прийняття своєчасних та ефективних рішень. Застосування хореографії в управлінні ланцюгом постачання не тільки підвищує операційну ефективність, але й підвищує

стійкість і адаптивність мереж постачання. У результаті підприємства можуть швидше реагувати на зміни ринку та вимоги клієнтів, зберігаючи конкурентну перевагу в галузі.

Отже, методи хореографії веб-сервісів відіграють ключову роль у забезпеченні ефективної та скоординованої роботи розподілених систем. Завдяки використанню формальних моделей, таких як UML, BPMN, WS-CDL і BPEL, а також застосуванню різних підходів – від централізованого й децентралізованого до контрактно-орієнтованого та подійного – компанії можуть оптимізувати свої бізнес-процеси, підвищити сумісність між сервісами та мінімізувати ймовірність помилок. Практичне впровадження хореографії дозволяє організаціям різних галузей, таких як електронна комерція, охорона здоров'я та логістика, досягати нових рівнів операційної ефективності та задоволеності клієнтів. Завдяки гнучкості, масштабованості та автоматизації бізнес-процесів, ці методи стають важливим інструментом у сучасному цифровому середовищі, забезпечуючи інтеграцію технологій із бізнес-цілями.

2.4. Семантика в когнітивних веб-сервісах

Семантика у веб-сервісах стосується осмисленої інтерпретації даних, що дозволяє машинам обробляти та розуміти інформацію у спосіб, подібний до людського пізнання. Такий підхід має вирішальне значення для створення цифрового контенту, доступного для всіх людей, незалежно від їхніх здібностей, віку чи доступності, оскільки це сприяє ефективній комунікації між різними системами [47]. Завдяки впровадженню семантичних технологій веб-сервіси можуть подолати розрив між людською мовою та машинним розумінням, забезпечуючи більш ефективний пошук даних і взаємодію, оскільки в сучасну цифрову епоху, коли обсяг даних постійно збільшується, а потреба в інтелектуальній обробці даних стає все більш виразною це є особливо важливим.

Онтології відіграють ключову роль у рамках семантичних технологій, слугуючи структурованими представленнями знань у певній області. Вони

дозволяють визначити поняття та зв'язки між ними, забезпечуючи таким чином чітку структуру для інтерпретації даних [17]. Онтології полегшують семантичну взаємодію між веб-сервісами, стандартизуючи значення термінів, що використовуються в різних системах. Відтак, стандартизація має важливе значення для встановлення значущих зв'язків між точками даних, зрештою підвищуючи точність і релевантність пошуку інформації. Наприклад, розробка онтологій, таких як PDO, є невід'ємною частиною концептуалізації та вдосконалення ієрархічних структур об'єктів, гарантуючи, що веб-сервіси можуть спілкуватися ефективно та узгоджено [30].

Порівнюючи семантичні технології з традиційними веб-сервісами, перші пропонують явну перевагу, забезпечуючи глибше розуміння даних завдяки розширеному контексту та значенню. Традиційні веб-сервіси часто покладаються на синтаксичну взаємодію, що може призвести до непорозумінь або неправильної інтерпретації даних через відсутність стандартизованого значення [81]. Навпаки, семантичні веб-сервіси використовують онтології та інші семантичні інструменти, щоб забезпечити більш багате та детальне розуміння даних. Така можливість забезпечує більш точний обмін даними, покращує автоматизацію процесів і здатність виконувати складні запити, з якими традиційним веб-сервісам важко впоратися. Оскільки цифровий ландшафт розвивається, впровадження семантичних технологій стає все більш важливим для організацій, які прагнуть залишатися конкурентоспроможними та інноваційними у своїх стратегіях управління даними.

Інтеграція семантики в когнітивні веб-сервіси значно підвищує ефективність пошуку та обробки даних шляхом автоматизації динамічного пошуку, композиції та виконання веб-сервісів. Семантика дозволяє системам розуміти й обробляти дані «за змістом», а не просто синтаксично, що призводить до більш точних і відповідних результатів. Цей підхід скорочує час і зусилля, які традиційно потрібні для пошуку та обробки наукових даних, пропонуючи спрощений спосіб доступу до інформації та маніпулювання нею. Використовуючи семантичні технології, когнітивні веб-сервіси можуть надавати користувачам інтуїтивно зрозумілі та

контекстно-залежні інтерфейси, тим самим покращуючи загальний досвід користувача. Здатність розпізнавати наміри та контекст користувача дає змогу цим сервісам ефективніше задовольняти конкретні потреби користувачів.

Впровадження методів машинного навчання для семантичного розуміння відіграє ключову роль у розвитку когнітивних веб-сервісів. Моделі машинного навчання можна навчити розуміти та обробляти семантичні дані, тим самим підвищуючи їх здатність інтерпретувати складні набори даних і надавати значущу інформацію [81]. Наприклад, евристичні моделі зосереджені на покращенні продуктивності агентів шляхом навчання шаблонів даних, що допомагає в ефективній обробці семантичної інформації [18]. Інтеграція машинного навчання з семантикою полегшує розробку інтелектуальних систем, які можуть розвиватися та адаптуватися до нової інформації, надаючи більш персоналізовані та точні відповіді на запити користувачів. Такі досягнення підкреслюють потенціал машинного навчання революціонізувати роботу когнітивних веб-сервісів шляхом впровадження глибшого рівня розуміння та аргументації.

Кілька когнітивних веб-сервісів уже використовують потужність семантичних технологій для надання інноваційних рішень. Наприклад, такі платформи, як EPrints і DSpace, використовують хмарні технології в поєднанні з семантичними рамками для оптимізації процесів зберігання та пошуку даних [57]. Ці сервіси призначені для керування величезною кількістю цифрових активів шляхом класифікації та представлення даних у спосіб, який відповідає потребам і вподобанням користувачів. Крім того, включення семантики в ці платформи дозволяє автоматизувати обробку даних, що значно скорочує час, необхідний для пошуку інформації, зрештою прискорюючи наукові дослідження та розробки [14]. Перелічені приклади демонструють трансформаційний вплив семантики на когнітивні веб-сервіси, прокладаючи шлях до більш розумних і ефективних цифрових екосистем.

Однією з основних проблем інтеграції семантики в когнітивні веб-сервіси є вирішення проблем масштабованості та сумісності. Оскільки веб-сервіси ростуть і розвиваються, забезпечення їх ефективного масштабування при збереженні

бездоганної сумісності стає вирішальним. Складність виникає через необхідність керувати величезними масивами даних і різноманітними системами, які повинні взаємодіяти та функціонувати злагоджено [23]. Щоб подолати ці проблеми, важливо розробити інфраструктури, які можуть обробляти великомасштабні дані та забезпечувати сумісність між різними платформами. Мета полягає в тому, щоб полегшити інтеграцію систем, заснованих на реляційних базах даних і нечіткій логіки, забезпечуючи більш згуртоване та ефективне середовище веб-сервісів.

Інша значна проблема полягає в подоланні обмежень у можливостях обробки природної мови (NLP). Хоча NLP досяг значного прогресу, залишаються перешкоди для досягнення тонкого розуміння людської мови, яка є важливою для покращення когнітивних веб-сервісів. Сучасні системи NLP часто борються з контекстом, двозначністю та тонкощами людського спілкування. Це обмеження може перешкоджати ефективності веб-сервісів, які покладаються на точне розуміння мови та взаємодію. Для вирішення цих проблем життєво важливі постійні дослідження та розробки технологій NLP. Удосконалюючи алгоритми NLP та інтегруючи складне семантичне розуміння, когнітивні веб-сервіси можуть досягти більш точної та значущої взаємодії з користувачами.

Заглядаючи вперед, майбутні досягнення в галузі семантики та когнітивних веб-сервісів містять величезний потенціал для трансформації галузі веб-сервісів. Оскільки семантичні технології продовжують розвиватися, вони обіцяють покращити можливості веб-сервісів для обробки та розуміння складних даних, таким чином покращуючи досвід користувачів і процеси прийняття рішень. Інтеграція розширеного семантичного розуміння з новими технологіями, такими як машинне навчання та штучний інтелект, швидше за все, призведе до більш інтуїтивно зрозумілих та інтелектуальних веб-сервісів. Очікується, що ця еволюція суттєво вплине на різні сектори, від охорони здоров'я до електронної комерції, надаючи більш персоналізовані та ефективні сервіси, які задовольняють конкретні потреби та вподобання користувачів.

Отже, семантика відіграє ключову роль у трансформації когнітивних веб-сервісів, надаючи їм здатність розуміти та обробляти дані на рівні, наближеному

до людського мислення. Завдяки онтологіям, машинному навчанню та NLP ці сервіси здатні забезпечувати більш точний пошук, релевантність інформації та інтуїтивно зрозумілу взаємодію з користувачами. Однак існують виклики, пов'язані з масштабованістю, сумісністю та складністю обробки природної мови, які потребують подальшого вдосконалення технологій. Перспективи розвитку семантичних веб-сервісів обіцяють значний вплив на різні галузі, відкриваючи нові можливості для створення більш розумних, адаптивних та ефективних цифрових систем.

2.5. Аналіз існуючих методів оркестровки та хореографії на основі семантики

Оркестровка та хореографія веб-сервісів є двома основними концепціями у сфері веб-сервісів, кожна з яких служить окремим цілям у координації сервісів. Неможливо переоцінити значення семантики в координації веб-сервісів, оскільки вона відіграє вирішальну роль у підвищенні сумісності та ефективності оркестровки та хореографії. Семантика забезпечує спільне розуміння залучених даних і процесів, що важливо, коли різні сервіси, можливо розроблені незалежно, повинні працювати разом [25, с. 46-52]. Використовуючи семантичні анотації, сервіси можуть краще інтерпретувати інформацію, якою обмінюються, зменшуючи ймовірність неправильного спілкування та помилок. Це спільне розуміння також полегшує динамічне виявлення та композицію сервісів, створюючи більш гнучкі та адаптивні системи. У контексті веб-сервісів семантика діє як міст, який з'єднує різні системи, дозволяючи їм функціонувати як єдине ціле, незважаючи на основні відмінності в дизайні та реалізації.

Реалізація оркестровки та хореографії у веб-сервісах містить кілька проблем, які розробники повинні вирішити, щоб забезпечити ефективну координацію сервісів. Однією з головних проблем є управління складністю, яка виникає внаслідок інтеграції кількох сервісів, кожна з яких має власний набір поведінки та взаємодії. Іншим значним завданням є забезпечення надійності та відмовостійкості

в умовах збоїв у мережі або недоступності сервісів, які можуть порушити весь робочий процес.

Семантична анотація веб-сервісів є важливою технікою, яка покращує машиночитаність веб-сервісів, забезпечуючи більш ефективне виявлення, композицію та виконання. Цей процес передбачає додавання метаданих до веб-сервісів для опису їх функціональних можливостей, входів, виходів та інших відповідних характеристик за допомогою формальної мови. Методи семантичної анотації часто використовують онтології для забезпечення спільного розуміння предметної області, сприяючи взаємодії між різними системами. Використовуючи семантичні анотації, сервіси можна легше інтегрувати в складні робочі процеси, забезпечуючи автоматизовану оркестровку та хореографію [25, с. 46-52].

Використання семантичних мов і фреймворків має ключове значення для управління складністю оркестровки веб-сервісів. Такі фреймворки, як OWL-S (мова веб-онтології для сервісів) і WSMO (онтологія моделювання веб-сервісів), пропонують надійну інфраструктуру для визначення семантичних властивостей веб-сервісів. OWL-S надає набір конструкцій для визначення можливостей веб-сервісів, таким чином уможливлуючи автоматичне виявлення та виклик. WSMO, з іншого боку, розширює ці можливості, надаючи комплексну модель, яка включає цілі, посередників та онтології, пропонуючи цілісний підхід до оркестровки семантичного веб-сервісу [25, с. 46-52].

На практиці існує кілька успішних реалізацій семантичної оркестровки, які демонструють потенціал цих підходів. Наприклад, використання OWL-S на платформах електронної комерції дозволяє автоматично створювати веб-сервіси, що спрощує процес пошуку та виконання сервісів, які відповідають конкретним вимогам користувача. Іншим яскравим прикладом є розгортання WSMO в системах управління бізнес-процесами, де семантичні анотації дозволяють динамічно адаптувати робочі процеси відповідно до змін бізнес-умов.

Моделі та протоколи семантичної хореографії є важливими для організації взаємодії між веб-сервісами шляхом визначення порядку та логіки, за якою ці сервіси мають спілкуватися. На відміну від оркестровки, яка зазвичай включає

центрального координатора, хореографія зосереджена на спільній поведінці кількох сервісів, гарантуючи, що вони працюють разом без централізованої контрольної точки. Кожна з існуючих моделей пропонує унікальні підходи до керування взаємодією сервісів на основі семантичної хореографії. Наприклад, структура онтології моделювання веб-сервісів (WSMO) забезпечує міцну основу для розуміння та реалізації хореографії шляхом використання онтологій для визначення взаємодії між веб-сервісами [25, с. 46-52]. Застосування цих моделей та протоколів забезпечує вищий рівень абстракції у визначенні взаємодії сервісів, сприяючи взаємодії та гнучкості в динамічних веб-середовищах.

Онтології відіграють ключову роль у покращенні взаємодії в рамках семантичної хореографії, надаючи спільний словниковий запас і загальне розуміння концепцій, якими обмінюються сервіси. Вони, у свою чергу, сприяють бездоганній інтеграції гетерогенних систем, дозволяючи різним сервісам узгоджено інтерпретувати дані, незалежно від базових технологій реалізації. Використовуючи онтології, семантична хореографія може подолати проблеми, пов'язані з неоднорідністю даних, і забезпечити більш плавну взаємодію між різними веб-сервісами [25, с. 46-52].

Тематичні дослідження, що ілюструють застосування семантичної хореографії на практиці, підкреслюють її потенціал трансформувати складну взаємодію сервісів у більш керовані та ефективні процеси. Успіх цих тематичних досліджень підкреслює цінність семантичної хореографії в додатках реального світу, пропонуючи розуміння найкращих практик і стратегій для оптимізації взаємодії сервісів [58, с. 152-156].

Оркестрація та хореографія веб-сервісів є двома різними, але взаємодоповнюючими концепціями, які дозволяють реалізувати інтеграцію та взаємодію між різними сервісами. Обидві моделі допомагають автоматизувати бізнес-процеси, проте мають свої концептуальні відмінності, які визначають їхні області застосування.

Оркестрація передбачає централізоване управління, де один контрольний процес (диригент) координує виконання та взаємодію між різними веб-сервісами.

Оркестрація забезпечує контроль над потоком даних і викликами, що дозволяє створити єдиний робочий процес.

Моделі оркестрації BPEL (Business Process Execution Language) та WS-BPEL (розширення BPEL, яке включає в себе підтримку семантики через використання онтологій для кращого визначення бізнес-процесів та інтеграції) часто використовуються в середовищах з жорсткими вимогами до контролю та управління бізнес-процесами, таких як фінансові сервіси, логістика та управління ланцюгами постачання.

Хореографія є децентралізованим підходом, де кожен веб-сервіс має знання про своїх сусідів і може діяти відповідно до попередньо визначених правил. У цій моделі не існує єдиного контрольного процесу; замість цього сервіси взаємодіють між собою відповідно до стандартів та протоколів. Моделі хореографії представлені такими основними мовами як WS-CDL (Web Services Choreography Description Language) та SWS (Semantic Web Services). Хореографія підходить для динамічних середовищ, де сервіси можуть змінюватися або оновлюватися, таких як IoT (інтернет речей), де пристрої повинні взаємодіяти автономно.

Концептуальні відмінності оркестрації та хореографії наводимо у табл. 2.5.

Таблиця 2.5

Концептуальні відмінності оркестрації та хореографії

Категорія для порівняння	Оркестрація	Хореографія
Контроль	Центральний контроль (диригент) координує всі сервіси	Децентралізований підхід, де кожен сервіс знає, як взаємодіяти з іншими
Управління потоком	Визначає послідовність і логіку виконання бізнес-процесів	Визначає, як сервіси повинні співпрацювати без конкретного контролю
Гнучкість	Менш гнучка в адаптації до змін; потребує переробки контрольного процесу	Більш гнучка, оскільки зміни в одному сервісі не впливають на інші

Як бачимо, оркестрація та хореографія веб-сервісів є критично важливими для інтеграції та автоматизації бізнес-процесів. Вибір між цими підходами залежить від специфіки застосування, вимог до контролю, гнучкості та швидкості

змін у системі. Використання семантики покращує інтеграцію, забезпечуючи спільне розуміння та оптимізацію взаємодії між сервісами.

Семантичні веб-сервіси (SWS) використовують метадані, онтології та семантичні анотації для забезпечення більш інтелектуальної інтеграції та взаємодії. Завдяки використанню семантики оркестрація та хореографія набувають додаткової гнучкості, автоматизації та можливостей масштабування. Розглянемо області застосування кожного підходу (табл. 2.6).

Таблиця 2.6

Області застосування хореографії та оркестрації семантичних веб-сервісів

Область застосування	Приклад	Семантика
Оркестрація		
Управління складними бізнес-процесами	У банківській сфері автоматизація процесів оформлення кредиту, страхування чи управління рахунками клієнтів. Оркестрація дозволяє централізовано координувати перевірки документів, оцінку ризиків, обробку платежів та комунікацію з клієнтами.	Використання онтологій для стандартизації даних між відділами (наприклад, онтології фінансових транзакцій).
Управління ланцюгами постачання (Supply Chain Management)	Взаємодія між постачальниками, виробниками, логістичними компаніями та роздрібними мережами. Оркестрація забезпечує координацію замовлень, відстеження доставки та оновлення статусів.	Забезпечує узгодженість даних про товари, маршрути та строки поставки.
Інтеграція різнорідних систем	Електронний уряд (e-Government), де оркестрація поєднує різні сервіси для обробки заявок громадян (наприклад, отримання субсидій, оформлення паспортів).	Використання онтологій для синхронізації даних між державними органами.
Підтримка рішень у медицині	Автоматизація процесів у лікарнях, включаючи обробку медичних записів, планування операцій і взаємодію з лабораторними системами.	Стандартизація даних пацієнтів за допомогою медичних онтологій (наприклад, HL7 або SNOMED CT).
Хореографія		
Інтернет речей (IoT)	"Розумний дім", де пристрої (термостати, освітлення, охоронні системи) взаємодіють автономно, обмінюючись даними про стан системи.	Використання онтологій IoT (наприклад, W3C SSN/SOSA) для забезпечення інтероперабельності пристроїв.

Продовження табл. 2.6

Електронна комерція	Автоматичне узгодження між сервісами інвентаризації, оплати, доставки та підтримки клієнтів.	Забезпечення інтелектуального узгодження між різними платформами (наприклад, опис товарів через стандарт GoodRelations).
Управління транспортом і логістикою	Координація автономних транспортних засобів, які динамічно змінюють маршрути залежно від трафіку чи метеорологічних умов.	Використання онтологій для опису дорожньої інфраструктури, вантажів і маршрутів.
Динамічні бізнес-екосистеми	Платформи типу Uber або Airbnb, де постачальники (водії, орендодавці) автономно обслуговують клієнтів.	Забезпечення автоматичного підбору постачальників і узгодження їхніх пропозицій із запитами клієнтів.
Розподілені наукові дослідження	Платформи, які об'єднують наукові лабораторії для спільного проведення експериментів або обміну даними.	Уніфікація даних та моделей для спільного використання в різних наукових галузях.

Представимо результати узагальненого порівняння областей застосування оркестрації та хореографії (табл. 2.7).

Таблиця 2.7

Порівняння областей застосування оркестрації та хореографії

Категорія	Оркестрація	Хореографія
Тип інтеграції	Статична, централізована	Динамічна, децентралізована
Підходить для	Складних і стабільних процесів	Динамічних і змінюваних середовищ
Тип взаємодії	Через єдиний контрольний процес	Прямі взаємодії між сервісами
Приклади	Управління ланцюгами постачання, медицина	ІоТ, автономний транспорт, електронна комерція

Таким чином, на підставі аналізу даних, наведених у табл. 2.5, 2.6, 2.7, формулюємо висновок стосовно того, що оркестрація підходить для середовищ, де необхідний жорсткий контроль і стабільність, тоді як хореографія забезпечує автономність і динамічну взаємодію. Семантика у обох підходах покращує розуміння і сумісність між системами, дозволяючи ефективніше реалізовувати складні сценарії інтеграції та автоматизації.

Отже, веб-сервіси забезпечують критично важливу основу для сучасних програмних систем, забезпечуючи інтеграцію та взаємодію між різними

платформами та технологіями. Оркестровка та хореографія веб-сервісів представляють два важливі підходи до управління цими сервісами, кожен із яких має свої сильні та слабкі сторони. Тоді як семантика є важливим фактором, що підвищує сумісність та ефективність обох підходів, пропонуючи спільне розуміння та можливість автоматизації. Хоча обидва підходи стикаються з різними викликами, зокрема з управлінням складністю та забезпеченням надійності, їх правильна реалізація може призвести до значних переваг для організацій, що прагнуть оптимізувати свої бізнес-процеси.

2.6. Моделі оркестрування та хореографії веб-сервісів у когнітивних системах

У сучасних когнітивних системах важливу роль відіграє ефективна взаємодія між різними веб-сервісами, яка забезпечує виконання складних завдань та досягнення високої продуктивності. Для управління цією взаємодією застосовуються моделі оркестрування та хореографії, які визначають порядок і правила обміну даними між сервісами. Оркестрування зосереджується на централізованому управлінні процесами, тоді як хореографія забезпечує децентралізовану взаємодію, де кожен сервіс виконує свої функції, дотримуючись загальних домовленостей. У контексті когнітивних систем ці моделі стають ключовими елементами для створення адаптивних і гнучких середовищ, які здатні реагувати на зміни та підтримувати складні сценарії використання.

2.6.1. Використання фреймворків для композиції когнітивних сервісів

У сучасних інформаційних системах зростає потреба у створенні комплексних рішень, які поєднують функціональність кількох сервісів, забезпечуючи їх узгоджену роботу в рамках єдиного завдання. Використання фреймворків для композиції когнітивних сервісів дозволяє автоматизувати інтеграцію, знижуючи складність ручного налаштування та підвищуючи ефективність роботи. Особливо актуальними стають підходи, що базуються на

штучному інтелекті, які здатні динамічно адаптуватися до змін та використовувати семантичні моделі для інтерпретації і взаємодії між сервісами. У цьому контексті агентний підхід на основі LLM, реалізований у LangChain, є потужним інструментом, що дозволяє поєднувати сервіси з використанням семантичного підходу, відкриваючи нові можливості для побудови інтелектуальних систем.

Виходячи із зазначеного вище, здійснимо аналіз можливостей агентного підходу на основі LLM (Large Language Models) [78] в LangChain для підтримки композиції сервісів із семантичним підходом. Зважаємо на те, що його можна структурувати основними аспектами, характеристику яких надаємо.

LangChain пропонує основи агентного підходу. LangChain – це сучасний фреймворк, який надає потужні інструменти для побудови агентів, що використовують великі мовні моделі (LLM) як центральний компонент. Основною ідеєю є створення систем, які можуть динамічно взаємодіяти з зовнішніми ресурсами, інтегрувати різні сервіси та вирішувати завдання в реальному часі. Завдяки цьому LangChain дозволяє автоматизувати роботу з даними, сервісами та іншими компонентами інформаційних систем.

Агенти, створені в LangChain, мають низку ключових можливостей. По-перше, вони використовують LLM для обробки тексту, інтерпретації запитів користувача та формування відповідей. Це забезпечує природний і зрозумілий спосіб взаємодії з системою, що наближає її роботу до людського мислення. По-друге, агенти можуть підключатися до зовнішніх сервісів через так звані Tools – спеціальні інструменти, які виконують конкретні функції, наприклад, обробку даних, доступ до API або виконання складних математичних обчислень.

Важливою перевагою такого підходу є природна інтерактивність між агентом і середовищем. Агент здатний отримувати інформацію, обробляти її та використовувати для прийняття подальших рішень. Наприклад, агент може звернутися до бази даних, проаналізувати отримані дані та зробити висновки, які стануть основою для наступних дій.

Агентний підхід у LangChain має дві основні характеристики, які роблять його ідеальним для композиції сервісів. Гнучкість дозволяє агенту адаптувати свою

поведінку залежно від контексту. Наприклад, якщо змінюється запит або доступні сервіси, агент автоматично перебудовує стратегію виконання завдання. Розширюваність означає, що можна легко додавати нові інструменти, інтегрувати їх із вже існуючими та розширювати функціонал системи без необхідності кардинальних змін.

Завдяки цим можливостям LangChain дозволяє створювати системи, які можуть об'єднувати різні сервіси в єдиний робочий процес, виконуючи складні завдання ефективно та динамічно. Це робить його особливо корисним для проєктів, які потребують інтеграції великої кількості сервісів та гнучкого підходу до їхньої композиції.

Семантичний підхід до композиції сервісів базується на використанні формалізованих описів функціональності та інтерфейсів сервісів у вигляді, зрозумілому для машин, що дозволяє забезпечити автоматизовану інтеграцію сервісів, полегшуючи їхню взаємодію та підвищуючи ефективність роботи складних систем. Основна ідея полягає в тому, щоб надати системі структуровану інформацію про сервіси, яку вона може аналізувати, інтерпретувати та використовувати для прийняття рішень.

Для реалізації такого підходу часто використовуються онтології або інші подібні формати, серед яких JSON-LD, OWL чи RDF. Вони дозволяють описати:

- функціональність сервісу, тобто які задачі він може виконувати;
- дані, які сервіс приймає на вхід і повертає у відповідь, включаючи їхній формат і тип;
- контекст використання, що визначає, коли і як сервіс може бути застосований, а також залежності між ним і іншими сервісами.

Використання семантичного підходу спрощує інтеграцію навіть складних систем, оскільки забезпечує однозначне трактування можливостей кожного сервісу.

У контексті LangChain великі мовні моделі (LLM) грають ключову роль у реалізації цього підходу (рис. 2.10).

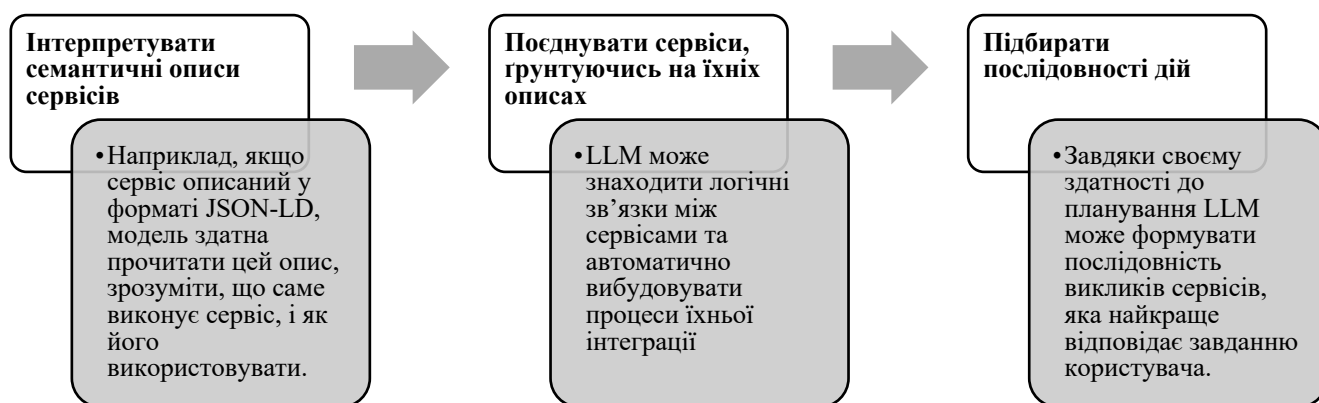


Рис. 2.10. Можливості LLM

Також відзначаємо, що якщо користувач запитує інформацію, яка вимагає обробки даних кількома сервісами, LLM може автоматично визначити, які саме сервіси потрібні, як передати дані між ними та в якому порядку викликати кожен з них. Такий підхід дозволяє значно розширити можливості автоматизації завдань у складних системах, спрощуючи інтеграцію сервісів і підвищуючи їхню взаємодію. Завдяки семантичним описам, агенти можуть працювати в динамічному середовищі, легко адаптуючись до змін та додаючи нові сервіси без значних зусиль.

Додатково зазначимо, що LangChain відкриває нові можливості для автоматизації завдань шляхом інтеграції великих мовних моделей (LLM) з процесом семантичної композиції сервісів, яка забезпечує не лише ефективне поєднання функціональності різних сервісів, але й динамічне управління їхньою взаємодією, що робить систему гнучкою та адаптивною.

Першим кроком у семантичній композиції є інтерпретація семантичних метаданих. LLM здатна зчитувати описи сервісів, представлені у формі метаданих, наприклад, у форматах JSON-LD, Swagger чи OpenAPI. Ці описи містять інформацію про API сервіси, їхні функції, очікувані вхідні та вихідні дані, а також обмеження у використанні. Використовуючи свої можливості розуміння тексту, модель може аналізувати ці метадані, щоб визначити, які сервіси доступні та як їх можна використовувати для вирішення конкретного завдання.

Наступним етапом є автоматичне планування взаємодії сервісів. Ґрунтуючись на завданні, яке поставив користувач, агент у LangChain може побудувати послідовність дій, що включає виклик кількох сервісів. Наприклад, якщо користувач потребує аналітики з кількох джерел, агент визначить, які сервіси необхідно викликати спочатку для збору даних, а потім для їх обробки. Описаний процес планування відбувається автоматично, що значно скорочує час на налаштування системи та робить її максимально зручною для користувача.

Ключовою особливістю такої інтеграції є **ітеративна композиція**. Агент у LangChain може динамічно змінювати свій план дій залежно від проміжних результатів, отриманих на кожному етапі виконання завдання. Наприклад, якщо один із сервісів повернув неочікувані дані, агент може переорієнтувати роботу, скориставшись іншими сервісами для досягнення бажаного результату, що, як наслідок, забезпечує високий рівень гнучкості та адаптивності системи, особливо в умовах невизначеності або зміни вимог.

Інтеграція LLM та LangChain дозволяє створювати інтелектуальні системи, які можуть автоматично поєднувати сервіси, враховуючи їхню семантику та контекст використання. Такий підхід є важливим для складних проєктів, які потребують багатоетапної обробки даних і динамічної взаємодії з різними ресурсами. Він не лише спрощує роботу розробників, але й значно підвищує ефективність роботи систем, роблячи їх більш адаптивними до сучасних викликів.

Інтеграція LLM із LangChain відкриває широкий спектр можливостей для вирішення складних завдань, що потребують взаємодії з кількома сервісами одночасно. Завдяки семантичному підходу та інтелектуальним можливостям LLM, такі системи можуть автоматично визначати оптимальні шляхи обробки запитів, забезпечуючи точність і гнучкість.

LLM може автоматизувати роботу з різноманітними API для виконання комплексних завдань. Процес починається з аналізу текстового запиту користувача. Наприклад, якщо користувач запитує:

«Знайди доступні рейси на завтра з Києва до Варшави та порівняй ціни»

LLM інтерпретує цей запит, розбиває його на частини та визначає, які API потрібні (сервіси для пошуку авіарейсів, порівняння цін або перевірки наявності місць). Отримавши дані з одного API, LLM може передати їх іншому для подальшої обробки. Наприклад, результати пошуку рейсів можуть бути використані для отримання додаткової інформації про знижки чи рекомендації. У підсумку користувач отримує комплексну відповідь, яка об'єднує дані з кількох джерел, автоматизуючи весь процес.

Системи, що базуються на онтологіях або інших семантичних описах сервісів, дозволяють LLM динамічно визначати, як взаємодіяти з різними компонентами. Наприклад, якщо сервіси для обробки даних мають залежності (один сервіс потребує результатів іншого для виконання своєї функції), LLM може побудувати оптимальний граф виконання операцій. Розглянемо завдання, пов'язане з обробкою зображень: користувач просить

«Проаналізуй фото на наявність обличчя і створи текстовий звіт із результатами»

У даному випадку LLM розуміє, що потрібен сервіс для аналізу зображень, а потім сервіс для генерації текстових звітів. Система автоматично вибудовує послідовність:

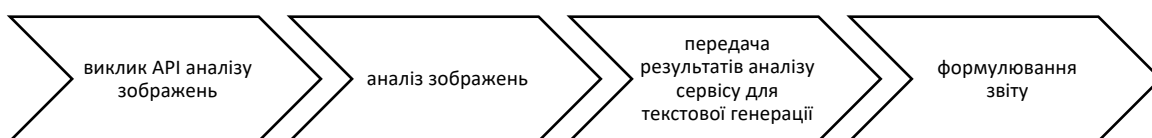


Рис. 2.11. Послідовність побудови оптимального графу виконання операцій

Також звертаємо у контексті нашого аналізу увагу на те, що LLM у складі LangChain може бути ефективним інструментом для створення рекомендаційних систем. Завдяки інтеграції сервісів для пошуку, фільтрації та персоналізації контенту, агент може надавати користувачам індивідуально налаштовані рекомендації. Наприклад, у сфері електронної комерції агент може аналізувати

історію покупок користувача, отриману через один API, і використовувати її для генерації рекомендацій через інший сервіс. При цьому система може враховувати контекст, наприклад, сезонність чи попередні запити користувача. Завдяки семантичному аналізу агент розуміє, як пов'язані різні категорії товарів, і може надавати більш релевантні рекомендації.

Інтеграція агентного підходу в LangChain з використанням великих мовних моделей (LLM) забезпечує значні переваги, особливо в завданнях, пов'язаних із поєднанням різноманітних сервісів (рис. 2.12).

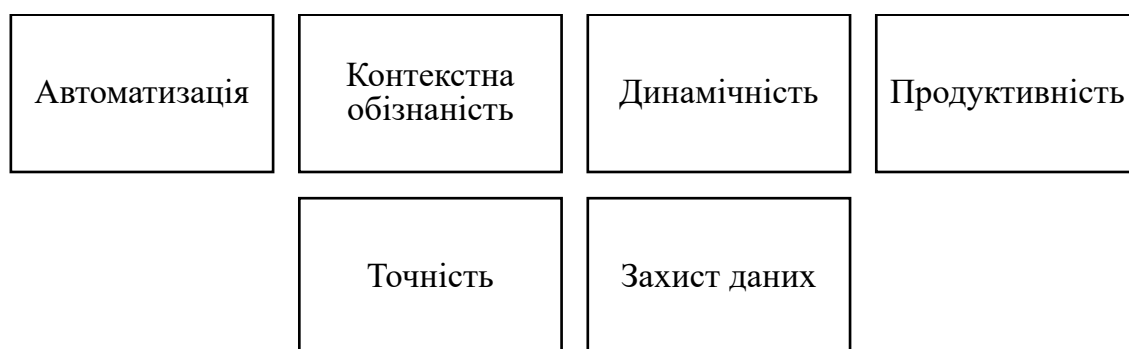


Рис. 2.12. Переваги (ліворуч) та виклики (праворуч) інтеграції агентного підходу в LangChain з використанням великих мовних моделей (LLM)

Однією з основних переваг інтеграції агентного підходу в LangChain з використанням великих мовних моделей (LLM) є зменшення обсягу ручної роботи, яка зазвичай потрібна для інтеграції сервісів. У традиційному підході розробникам доводиться самостійно налаштовувати з'єднання між сервісами, визначати порядок викликів API, а також обробляти отримані дані. Завдяки LLM і LangChain ці процеси автоматизуються. Агент самостійно визначає, які сервіси потрібні, як їх викликати, та які дані передати між ними, завдяки чому значно знижується навантаження на розробників.

Також зважаємо на те, що однією із переваг є те, що LLM здатен утримувати контекст завдання, що є ключовою перевагою агентного підходу. Наприклад, якщо користувач запитує спочатку про прогноз погоди, а потім просить поради щодо місць для відпочинку, агент зможе поєднати ці запити, запропонувавши рекомендації, які відповідають погодним умовам. Таким чином, контекстна

обізнаність дозволяє створювати більш інтелектуальні системи, які краще розуміють потреби користувача, що надзвичайно важливо в складних завданнях, де результати попередніх кроків впливають на наступні дії.

Перевага, яка полягає у динамічності, пояснюється тим, що системи, побудовані на LangChain, є надзвичайно гнучкими й здатними працювати в умовах невизначеності. Агент може адаптувати свої дії залежно від змін у середовищі або доступності сервісів. Наприклад, якщо один із сервісів стає недоступним, агент може знайти альтернативу або переглянути план виконання завдання. Крім того, завдяки можливості інтегрувати нові сервіси без суттєвих змін у логіці роботи системи, LangChain забезпечує простоту масштабування.

Попри значні переваги, агентний підхід із використанням LLM і LangChain має свої виклики та обмеження, які слід враховувати під час розробки і впровадження систем. Основні проблеми стосуються продуктивності, точності інтерпретації та захисту даних.

Робота з великими семантичними описами сервісів вимагає значних обчислювальних ресурсів. Наприклад, якщо опис API включає складну структуру даних, взаємозалежності або великий обсяг метаданих, LLM може витратити багато часу на їх обробку, що у результаті може призвести до затримок у виконанні запитів, особливо в реальному часі. Інтеграція кількох сервісів у рамках одного завдання може додатково ускладнювати систему, збільшуючи час на взаємодію між агентом і сервісами.

Хоча LLM має потужні можливості для обробки тексту, вони можуть включати помилки під час інтерпретації складних або неоднозначних семантичних описів. Наприклад, якщо API характеризується схожими функціями з різними параметрами або неоднозначними назвами, модель може неправильно зрозуміти, який виклик слід використовувати. Також складні запити користувача, що містять багатозначність або неточності, можуть призводити до генерації некоректних послідовностей дій.

Інтеграція з кількома сервісами одночасно створює значні виклики для безпеки даних. Коли агент передає дані між різними API, існує ризик витоку

конфіденційної інформації або неправомірного доступу до даних. Наприклад, якщо один із сервісів має слабкі механізми автентифікації, це може поставити під загрозу всю систему. Також звертаємо увагу на те, що деякі сервіси можуть зберігати або аналізувати отримані дані для власних потреб, що може викликати питання відповідності законодавству про захист даних, наприклад, GDPR.

Здійснивши аналіз переваг та викликів інтеграції агентного підходу в LangChain з використанням великих мовних моделей (LLM) вважаємо, що завдяки перевагам: автоматизації, контекстній обізнаності та динамічності, агентний підхід із LangChain дозволяє створювати інтелектуальні, адаптивні й ефективні системи, які відповідають сучасним вимогам інтеграції сервісів. Такий підхід не лише полегшує процес розробки, але й забезпечує кращий користувацький досвід, знижуючи бар'єри для впровадження нових технологій. Вважаємо, що існуючі виклики продуктивності, точності та безпеки даних є важливими аспектами, які необхідно враховувати при використанні LangChain і LLM для композиції сервісів. Хоча ці обмеження можуть ускладнювати впровадження, їх можна подолати за допомогою правильного проектування системи, регулярного тестування і застосування сучасних підходів до забезпечення безпеки задля створення надійних та ефективних рішень, що максимально використовують потенціал агентного підходу.

Потенційні напрямки покращення агентного підходу в LangChain, на нашу думку, полягають у:

1. Використанні спеціалізованих моделей для обробки онтологій і семантичних описів.
2. Оптимізації роботи агентів через кешування і попереднє індексування.
3. Розширенні функціоналу LangChain для підтримки складних сценаріїв.

Відтак, зважаємо на те, що удосконалення агентного підходу в LangChain через інтеграцію спеціалізованих моделей, оптимізацію продуктивності та розширення функціоналу відкриває нові перспективи для його застосування.

Отже, вважаємо, що LangChain у поєднанні з LLM є перспективним підходом для створення агентів, які здатні ефективно підтримувати композицію сервісів із

семантичним підходом та відкривають нові можливості для автоматизації складних процесів у багатьох галузях, таких як логістика, фінанси, освітні платформи тощо.

Висновки до розділу 2

Технології розвитку семантичного Веб включають XML (eXtensible Markup Language), RDF (Resource Description Framework), SPARQL (протокол SPARQL і мова запитів RDF), OWL (мова веб-онтологій). Формат RDF найбільш корисний у забезпеченні спільного використання інформації, зміст якої може однаково інтерпретуватися різними програмними агентами. Мова веб-онтологій (OWL) є ще одним критичним компонентом семантичної мережі, що забезпечує мову схем або мову представлення знань, яка покращує семантичне багатство веб-даних. OWL має три діалекти (у порядку зростання виразності): Lite, DL та Full. Розуміння можливостей OWL та характеристик його діалектів дозволяє ефективно вибирати відповідний підхід для моделювання когнітивних сервісів, адаптуючи його під конкретні вимоги і завдання. Майбутні перспективи та нові тенденції в розвитку інструментів міркування на основі OWL є багатообіцяючими та відображають зростаючий попит на більш складні та ефективні можливості міркування. SPARQL, що розшифровується як SPARQL Protocol and RDF Query Language, є спеціалізованою мовою запитів, розробленою для доступу до даних RDF та обробки даних. Синтаксис і структура SPARQL є ключовими для його функціональності та ефективності запитів до даних RDF. SQWRL, або мова веб-правил, розширена семантичними запитами, відіграє ключову роль у надсиланні запитів до онтологій у рамках семантичної мережі. Вибір між SPARQL і SQWRL для семантичних веб-проектів часто залежить від конкретних вимог і існуючої інфраструктури проекту. SPARQL зазвичай віддають перевагу для проектів, що включають великі пов'язані дані в різноманітних наборах даних через його широке визнання та надійну продуктивність у обробці даних RDF. Навпаки, SQWRL найкраще підходить для проектів, орієнтованих на онтологію, де міркування над

складними ієрархіями класів і зв'язками є вирішальними. Таким чином, вибір між SPARQL і SQWRL повинен ґрунтуватися на характері даних, складності необхідних запитів і загальних цілях проекту.

Аналіз сучасних методів оркестрації мікросервісів демонструє, що такі інструменти, як Kubernetes, Docker Swarm і Apache Mesos, значно підвищують гнучкість, масштабованість і надійність складних систем, автоматизуючи процеси розгортання, масштабування та управління контейнеризованими застосунками. Оркестрація є ключовим елементом сучасної IT-інфраструктури, забезпечуючи стабільну та ефективну роботу мікросервісних архітектур.

Методи хореографії веб-сервісів відіграють ключову роль у забезпеченні ефективної та скоординованої роботи розподілених систем. Завдяки використанню формальних моделей, таких як UML, BPMN, WS-CDL і BPEL, а також застосуванню різних підходів – від централізованого й децентралізованого до контрактно-орієнтованого та подійного – компанії можуть оптимізувати свої бізнес-процеси, підвищити сумісність між сервісами та мінімізувати ймовірність помилок. Практичне впровадження хореографії веб-сервісів дозволяє організаціям різних галузей, таких як електронна комерція, охорона здоров'я та логістика, досягати нових рівнів операційної ефективності та задоволеності клієнтів. Завдяки гнучкості, масштабованості та автоматизації бізнес-процесів, ці методи стають важливим інструментом у сучасному цифровому середовищі, забезпечуючи інтеграцію технологій із бізнес-цілями.

Семантика відіграє ключову роль у трансформації когнітивних веб-сервісів, надаючи їм здатність розуміти та обробляти дані на рівні, наближеному до людського мислення. Завдяки онтологіям, машинному навчанню та NLP ці сервіси здатні забезпечувати більш точний пошук, релевантність інформації та інтуїтивно зрозумілу взаємодію з користувачами. Однак існують виклики, пов'язані з масштабованістю, сумісністю та складністю обробки природної мови, які потребують подальшого вдосконалення технологій. Перспективи розвитку семантичних веб-сервісів обіцяють значний вплив на різні галузі, відкриваючи нові

можливості для створення більш розумних, адаптивних та ефективних цифрових систем.

Веб-сервіси забезпечують критично важливу основу для сучасних програмних систем, забезпечуючи інтеграцію та взаємодію між різними платформами та технологіями. Оркестровка та хореографія веб-сервісів представляють два важливі підходи до управління цими сервісами, кожен із яких має свої сильні та слабкі сторони. Тоді як семантика є важливим фактором, що підвищує сумісність та ефективність обох підходів, пропонуючи спільне розуміння та можливість автоматизації. Хоча обидва підходи стикаються з різними викликами, зокрема з управлінням складністю та забезпеченням надійності, їх правильна реалізація може призвести до значних переваг для організацій, що прагнуть оптимізувати свої бізнес-процеси.

Завдяки перевагам: автоматизації, контекстній обізнаності та динамічності, агентний підхід із LangChain дозволяє створювати інтелектуальні, адаптивні й ефективні системи, які відповідають сучасним вимогам інтеграції сервісів. Такий підхід не лише полегшує процес розробки, але й забезпечує кращий користувацький досвід, знижуючи бар'єри для впровадження нових технологій. Вважаємо, що існуючі виклики продуктивності, точності та безпеки даних є важливими аспектами, які необхідно враховувати при використанні LangChain і LLM для композиції сервісів. Хоча ці обмеження можуть ускладнювати впровадження, їх можна подолати за допомогою правильного проектування системи, регулярного тестування і застосування сучасних підходів до забезпечення безпеки задля створення надійних та ефективних рішень, що максимально використовують потенціал агентного підходу. Удосконалення агентного підходу в LangChain через інтеграцію спеціалізованих моделей, оптимізацію продуктивності та розширення функціоналу відкриває нові перспективи для його застосування. LangChain у поєднанні з LLM є перспективним підходом для створення агентів, які здатні ефективно підтримувати композицію сервісів із семантичним підходом та відкривають нові можливості для автоматизації складних процесів у багатьох галузях, таких як логістика, фінанси, освітні платформи тощо.

РОЗДІЛ 3. РОЗРОБКА СЕМАНТИЧНОЇ СТРУКТУРИ

3.1. Вимоги до семантичної структури когнітивних веб-сервісів

3.1.1. Функціональні вимоги

Семантична структура когнітивних веб-сервісів повинна бути продуманою та універсальною, щоб забезпечувати високу ефективність, адаптивність і інтеграцію в сучасні цифрові системи.

Функціональні вимоги до семантичної структури когнітивних веб-сервісів включають:

1. Опис когнітивних можливостей.
2. Контекстуальна обізнаність.
3. Обробка обмежень.
4. Автоматизація пошуку й вибору.
5. Забезпечення інтеграції користувачів.

Таблиця 3.1

Функціональні вимоги до семантичної структури когнітивних веб-сервісів

№ з/п	Назва функціональної вимоги	Зміст функціональної вимоги
1	Опис когнітивних можливостей	Семантична структура повинна забезпечувати формалізований опис когнітивних можливостей сервісу, таких як адаптація, навчання, розуміння природної мови чи розпізнавання образів.
		Когнітивні можливості мають бути легко інтегровані в опис сервісу для автоматичного вибору й оркестрації.
2	Контекстуальна обізнаність	Структура повинна враховувати специфіку задачі, включаючи: Тип і формат вхідних/вихідних даних (наприклад, текст, аудіо, зображення). Тип задачі, яку виконує сервіс (наприклад, переклад, розпізнавання, класифікація). Параметри якості обслуговування (QoS), такі як час відповіді, частота помилок і доступність.

Продовження таблиці 3.1

3	Обробка обмежень	Можливість опису обмежень, таких як час виконання, точність результату, мовна підтримка та розмір партії даних.
		Забезпечення врахування цих обмежень під час вибору або композиції сервісів.
4	Автоматизація пошуку й вибору	Структура повинна підтримувати автоматичний пошук сервісів на основі когнітивних можливостей, типів даних, задач і контексту використання.
		Динамічний вибір сервісів у режимі реального часу з урахуванням зміни умов середовища чи вимог.
5	Забезпечення інтеграції користувачів	Семантична структура повинна враховувати особливості користувачів, зокрема їхні ролі та мовні налаштування.
		Надання можливостей для персоналізованої взаємодії сервісів із користувачами.

Здійснюючи аналіз когнітивних можливостей сервісів із табл. 3.1, відзначимо, що вони мають бути здатними виконувати складні завдання, такі як адаптація до змін, навчання, розуміння природної мови чи розпізнавання образів. Семантична структура повинна чітко описувати ці можливості, щоб їх можна було легко використовувати для автоматичного вибору або об'єднання сервісів. Наприклад, якщо сервіс призначений для аналізу зображень, його здатність розпізнавати об'єкти чи класифікувати сцени повинна бути чітко зазначена в описі.

Контекстуальна обізнаність полягає в тому, що сервіси повинні враховувати специфіку задач, які перед ними ставляться, включаючи:

- тип і формат даних, з якими вони працюють (наприклад, текст, аудіо чи зображення);
- конкретні задачі, які вони виконують (переклад, розпізнавання мовлення, класифікація);
- вимоги до якості обслуговування, такі як швидкість роботи, точність результатів і доступність.

Підкреслимо, що семантична структура має забезпечувати динамічну адаптацію сервісів до зміни умов середовища. Наприклад, якщо зростає обсяг даних або змінюється формат запитів, сервіс повинен автоматично підлаштовуватися до цих змін.

Відзначимо, що у реальному світі сервіси працюють в умовах певних обмежень (обмеження за часом виконання, рівнем точності, підтримкою певних мов чи обсягом даних, які можна обробити за один запит), тому семантична структура повинна дозволяти опис цих обмежень і враховувати їх під час вибору сервісу або об'єднання кількох сервісів для виконання задачі.

Автоматизація пошуку й вибору означає, що система повинна автоматично знаходити сервіси, які найкраще відповідають поставленим вимогам, враховуючи їх когнітивні можливості, типи задач, контекст використання й доступні ресурси. Крім того, сервіси повинні бути здатні до динамічного вибору в режимі реального часу, коли змінюються умови середовища або потреби користувача.

Семантична структура повинна враховувати особливості кінцевих користувачів, таким чином реалізуючи інтеграцію із ними. Ці особливості можуть включати їх ролі (наприклад, адміністратор чи звичайний користувач), мовні налаштування і персональні уподобання тощо. Окрім іншого важливо відзначити, що когнітивні веб-сервіси мають пропонувати інтерфейс, який можна адаптувати під потреби кожного користувача, забезпечуючи комфортну та ефективну взаємодію.

Таким чином, функціональні вимоги у вигляді когнітивних можливостей, контекстуальної обізнаності, обробки обмежень, автоматизації пошуку й вибору, забезпечення інтеграції користувачів спрямовані на створення системи, яка є не лише технічно потужною, а й зручною для користувачів. Семантична структура когнітивного веб-сервісу повинна забезпечувати легкість інтеграції нових сервісів, їхню адаптивність до зовнішніх умов і персоналізацію для кожного користувача.

3.1.2. Нефункціональні вимоги

Нефункціональні вимоги забезпечують високу якість роботи веб-сервісу та семантичної структури, яка використовується у ньому. Нefункціональні вимоги до семантичної структури когнітивних веб-сервісів включають: продуктивність, масштабованість, розширюваність, сумісність, надійність, безпеку, гнучкість.

Таблиця 3.2

Нефункціональні вимоги до семантичної структури когнітивних
веб-сервісів

№ з/п	Назва нефункціональної вимоги	Зміст нефункціональної вимоги
1	Продуктивність	Швидкість обробки семантичних запитів (наприклад, SPARQL) повинна бути оптимізованою для роботи з великим обсягом даних та сервісів.
2	Масштабованість	Структура повинна підтримувати ефективну роботу з великим числом сервісів і різноманітних задач, включаючи поточні когнітивні та ті, що можуть з'явитися в майбутньому.
3	Розширюваність	Можливість додавання нових когнітивних можливостей, типів даних, форматів та задач без необхідності значного доопрацювання структури.
4	Сумісність	Можливість додавання нових когнітивних можливостей, типів даних, форматів та задач без необхідності значного доопрацювання структури.
5	Надійність	Забезпечення коректної роботи навіть у випадку відсутності деяких даних або виникнення помилок під час запитів.
6	Безпека	Захист даних і операцій від несанкціонованого доступу, включаючи шифрування запитів і результатів.
7	Гнучкість	Адаптація до нових вимог чи змін у середовищі роботи без потреби внесення значних змін у структуру.

Відзначимо, що нефункціональні вимоги, наведені у таблиці 3.2, створюють основу для створення ефективної, адаптивної та надійної семантичної структури когнітивних веб-сервісів. Вони забезпечують довгострокову функціональність системи, її зручність для користувачів і відповідність вимогам сучасних технологій.

3.2. Формулювання задачі розробленої платформи

Під час створення когнітивного веб-сервісу увагу було зосереджено на розробці платформи, яка здатна автоматизувати процес вибору, компонування та виконання веб-сервісів для вирішення завдань, сформульованих користувачем. Основна ідея полягала у створенні інструменту, що надає можливість користувачам отримувати максимально релевантні результати для їхніх задач, навіть якщо ці задачі є складними або багатоступеневими.

В основі ядра задач лежить семантичний опис і пошук сервісів. Адже для забезпечення ефективного пошуку веб-сервісів було створено систему, що використовує семантичний підхід, тобто коли кожен сервіс описується у вигляді формалізованої моделі, яка включає інформацію про його функціональні можливості, когнітивні властивості, тип задач, з якими він працює, а також параметри вхідних і вихідних даних. Наприклад, якщо сервіс обробляє текст для аналізу змін карти бойових дій, його модель містить опис задачі (аналіз змін карти бойових дій), тип вхідних даних (текст, що містить географічні назви, описи дій, наприклад, «взяли під контроль», «відступили», «атакували»), тип вихідних даних (оновлена карта бойових дій із позначенням змін, таких як нові контрольовані території чи лінії фронту), та обмеження, такі як точність географічних координат, мова тексту чи швидкість обробки. Усе це дозволяє системі швидко знаходити сервіси, які найкраще відповідають вимогам запиту користувача. Платформа є гнучкою та готовою до інтеграції з новими типами сервісів у майбутньому завдяки підтримці пошуку на основі когнітивних властивостей, таких як здатність сервісу навчатися або адаптуватися до нових умов.

Відзначимо, що компонування сервісів для вирішення складних задач означає, що іноді запит користувача вимагає не одного, а кількох взаємопов'язаних сервісів для виконання завдання. Для цього при розробці когнітивного веб-сервісу було розроблено механізм компонування, який автоматично формує послідовність виконання сервісів. Наприклад, якщо користувач хоче отримати аналіз змін у розташуванні військ на основі текстових звітів, система спочатку обирає сервіс

обробки тексту для виділення географічних назв та ключових подій (наприклад, «переміщення військ», «захоплення території»), а потім сервіс геоприв'язки, який перетворює ці дані на карту бойових дій. При цьому забезпечується сумісність між вихідними текстовими даними та географічною інформацією, а також враховуються обмеження, такі як швидкість обробки чи деталізація карти. Логічні зв'язки між сервісами є ключовим елементом цього процесу. Тому до його структури було інтегровано інструменти для перевірки сумісності сервісів, щоб уникнути ситуацій, коли вихід одного сервісу не може бути використаний іншим.

Інтерпретація запитів користувачів є важливою задачею, оскільки вони можуть задавати запити у текстовій або голосовій формі. Наприклад, вони можуть сказати: «Знайди всі міста у цьому тексті, які перебувають у зоні підвищеної небезпеки, та виконай це за 5 секунд». Для обробки таких запитів впроваджено інтеграцію з NLP-моделями (зокрема GPT), які допомагають зрозуміти суть завдання та перетворити його у формалізований запит. Відтак, система аналізує ключові частини запиту: тип задачі, специфічні вимоги (точність, час виконання) та формат даних. Наприклад, якщо військовий оператор зазначив, що вхідні дані знаходяться у форматі аудіо (наприклад, перехоплений радіосигнал або голосова команда), когнітивний веб-сервіс автоматично визначить відповідні сервіси, здатні працювати з цим типом даних. Система може обрати сервіси для розпізнавання мови (SpeechToText) або аналізу аудіо (AudioSignalAnalysis) та побудує оптимальний план виконання задачі, наприклад, спочатку перетворення аудіо в текст, а потім аналіз тексту для виявлення ключових слів, імен чи загроз. Ця частина функціональної системи веб-сервісу є критично важливою, оскільки користувачам не потрібно знати технічні деталі сервісів. Вони просто формулюють задачу у зрозумілій формі, а платформа бере на себе всю технічну роботу.

Верифікація виконання та відповідності результату полягає в тому, що після виконання задачі відбувається перевірка на відповідність результату очікуванням користувача та заданим параметрам. Наприклад, якщо користувач задав завдання знайти список міст у текстовому звіті розвідки за 10 секунд, когнітивний веб-сервіс, наприклад TextAnalysis, автоматично виконає обробку тексту. Платформа оцінює,

чи вдалося виконати завдання в зазначений час і чи коректно ідентифіковано всі географічні об'єкти. Якщо завдання виконано успішно, система надає результат із переліком міст, знайдених у тексті. Якщо ж час виконання або точність не відповідають вимогам, платформа фіксує помилку, оптимізує запит або пропонує інші сервіси для досягнення необхідного результату. Відзначимо, що механізм верифікації також враховує специфічні вимоги, такі як точність або мовна підтримка. Якщо результат не відповідає вимогам, платформа може автоматично повторити задачу, використовуючи інші сервіси, або повідомити користувача про проблему.

Уся система побудована за клієнт-серверною архітектурою. Користувачі взаємодіють із клієнтом – веб-інтерфейсом, через який вони задають запити у текстовій, графічній чи голосовій формі. Запит передається на сервер, де відбувається його аналіз, пошук та компонування сервісів.

Сервер є головним компонентом, що обробляє запит, працює з семантичною структурою, виконує логічні висновування та управляє виконанням сервісів. Після завершення задачі сервер повертає результат клієнту. Така архітектура дозволяє масштабувати систему та забезпечувати високу продуктивність навіть для складних задач.

Отже, розроблена платформа когнітивних веб-сервісів автоматизує вибір, композицію та виконання веб-сервісів для вирішення завдань, визначених користувачем, навіть якщо вони складні або багатоетапні. Він використовує семантичний підхід для опису сервісів, що дозволяє здійснювати ефективний пошук на основі функціональних можливостей, параметрів введення/виведення та когнітивних властивостей. Платформа також інтегрує обробку природної мови для інтерпретації запитів користувачів, гарантуючи, що система може обробляти різні формати даних і перевіряти результати виконання на відповідність очікуванням користувачів, що в кінцевому підсумку забезпечує гнучке та масштабоване рішення для вирішення складних проблем.

3.3. Опис запропонованої моделі оркестрування та хореографії

Здійснюючи опис запропонованої моделі оркестрування та хореографії, відзначаємо, що у центрі оркестрування знаходиться оркестратор – модуль, який відповідає за управління всім процесом виконання задачі (Додаток А).

Перший крок, який він виконує – це прийняття задачі від користувача або іншого компонента системи. Задача може включати детальний опис вимог, таких як тип виконуваної операції (наприклад, розпізнавання образів, переклад тексту), формат вхідних і вихідних даних (текст, аудіо, зображення) та інші обмеження (час виконання, точність тощо).

Наступним кроком функціональної діяльності оркестратора є аналіз цих вимог на підставі використання правил логічного висновування, побудованих на основі онтології. Онтологія є основою для формалізації знань про доступні сервіси, їх функціонал і параметри роботи. Наприклад, якщо задача полягає в обробці тексту, оркестратор виключає сервіси, які працюють лише із зображеннями, та обирає ті, що здатні виконати необхідну операцію, наприклад, аналіз тексту чи його переклад.

Побудова послідовності: планування виконання задачі настає після аналізу вимог і визначення релевантних сервісів, коли оркестратор формує чіткий план дій. Цей план включає послідовність викликів сервісів із визначенням їх порядку виконання. Наприклад, якщо користувач задає завдання перекласти текст радіоперехоплення з іноземної мови та провести аналіз настрою повідомлення, когнітивна платформа автоматично активує оркестратор. Спочатку він обирає сервіс машинного перекладу (MachineTranslation) для перекладу тексту на мову аналітика. Після цього перекладений текст передається до сервісу аналізу настрою (SentimentAnalysis), який визначає емоційне забарвлення повідомлення, наприклад, виявляє ознаки напруги, страху чи агресії. Формування плану відбувається з урахуванням залежностей між сервісами та їхніх вхідних/вихідних даних. Оркестратор враховує не лише функціональну сумісність, а й обмеження, наприклад, максимальний час виконання або необхідний рівень точності.

Останній етап – це управління безпосереднім виконанням задачі. Оркестратор поетапно викликає сервіси, передає між ними необхідні дані та контролює їхнє виконання. Наприклад, після отримання проміжного результату від одного сервісу, оркестратор відправляє його як вхідні дані для наступного. Однією з важливих функцій оркестратора є контроль за дотриманням обмежень. Якщо якийсь сервіс виходить за межі встановленого часу виконання або повертає помилку, оркестратор може перезавпустити операцію, спробувати інший сервіс або повідомити про проблему. Наприклад, якщо сервіс розпізнавання зображень не зміг коректно обробити дані, оркестратор може автоматично переключитися на альтернативний сервіс. Додатково оркестратор може моніторити весь процес виконання задачі, збираючи дані для подальшого аналізу. Це дозволяє виявляти слабкі місця системи, оптимізувати її продуктивність і забезпечувати високу якість виконання запитів.

Оскільки хореографія – це інший підхід до управління сервісами, що відрізняється децентралізованою моделлю взаємодії, у цьому випадку немає єдиного компонента, який би керував усім процесом виконання задачі. Натомість кожен сервіс самостійно вирішує, як діяти далі, враховуючи логіку задачі та поточний контекст.

У моделі хореографії кожен сервіс наділений автономією у прийнятті рішень. Тобто після завершення своєї частини роботи сервіс аналізує, які сервіси можуть отримати його результати, і самостійно вирішує, кому передати дані. Відзначаємо, що цей процес базується на логічному висновуванні та інформації про поточний контекст задачі. Вибір базується на правилах, закладених у систему, та даних онтології, яка описує можливості кожного сервісу. Такий підхід дозволяє уникнути залежності від єдиного оркестратора, що робить систему більш стійкою, оскільки навіть при збоях одного з компонентів інші сервіси можуть продовжувати роботу, приймаючи рішення на основі доступної інформації.

Однією з ключових особливостей хореографії є здатність до динамічного вибору сервісів, коли кожен сервіс під час виконання задачі може аналізувати поточні умови та змінювати свій вибір наступного кроку. Наприклад, якщо

основний сервіс недоступний або перевантажений, система автоматично вибирає альтернативний сервіс із подібним функціоналом. Ця технологія вибору здійснюється на основі заздалегідь визначених правил, які враховують відповідність вхідних і вихідних даних між сервісами. Наприклад, якщо один сервіс генерує текст як вихідні дані, наступний сервіс повинен бути здатний працювати саме з текстом. Додатково враховуються такі параметри, як час виконання, точність та інші критерії якості обслуговування (QoS). Динамічність дозволяє системі адаптуватися до змінних умов роботи, таких як раптове зростання навантаження або поява нових сервісів.

Ключовою складовою хореографії є передача контексту між сервісами. Контекст – це набір даних, який включає результати попереднього кроку, статус виконання задачі, мету, яку потрібно досягти, та будь-яку іншу важливу інформацію. Передача контексту дозволяє кожному наступному сервісу розуміти, що вже виконано, і яка частина роботи ще залишилася. Наприклад, якщо сервіс аналізу тексту виконав свою частину роботи, він передає результат іншому сервісу разом із метаданими, такими як мова тексту, обсяг даних та вимоги до часу виконання наступної операції. Це гарантує, що кожен сервіс має повну картину задачі і може коректно виконати свою частину. Передача контексту також дозволяє уникнути втрати інформації. Якщо один із сервісів не може виконати задачу, наступний сервіс отримає всі необхідні дані для продовження роботи, що як наслідок, підвищує надійність системи та дозволяє забезпечити безперервність виконання задач навіть у разі помилок чи збоїв.

Отже, запропонована модель оркестровки та хореографії зосереджена навколо оркестровця, який керує виконанням завдань, приймаючи завдання, визначені користувачем, аналізуючи вимоги та плануючи послідовність виконання відповідних сервісів. Оркестровник використовує правила логічного висновку на основі онтології для вибору відповідних сервісів, гарантуючи дотримання таких обмежень, як час виконання та точність. Навпаки, модель хореографії дозволяє сервісам працювати автономно, приймаючи незалежні рішення на основі контексту

завдання, що підвищує стабільність і адаптивність системи, забезпечуючи динамічний вибір і передачу контексту між сервісами.

3.4. Зміст та структура онтології для опису когнітивних веб-сервісів

Онтологія когнітивного веб-сервісу є формальною моделлю, що описує взаємозв'язки між основними компонентами системи, такими як:

- користувачі;
- когнітивні сервіси;
- типи даних;
- задачі та обмеження.

Вона дозволяє визначити, як веб-сервіси можуть обробляти різноманітні типи даних для виконання завдань користувачів, враховуючи їхні потреби, технічні вимоги та параметри якості.

На рис. 3.1. наведена схематична структура онтологічної моделі когнітивного веб-сервісу.

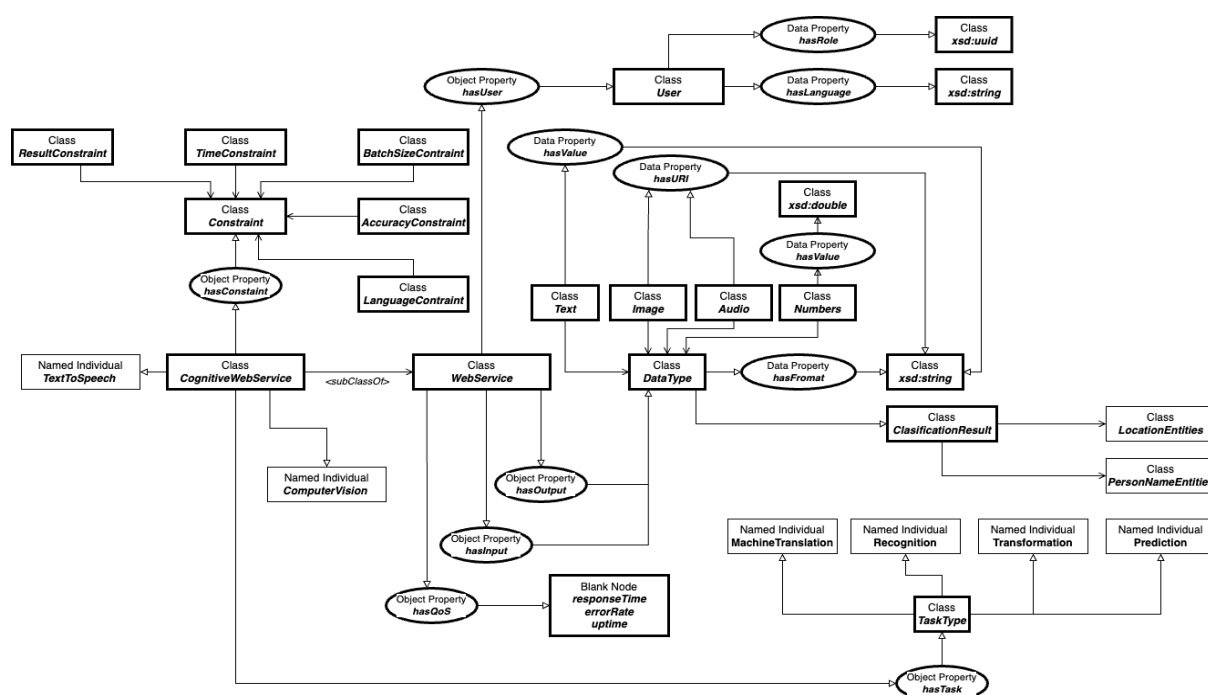


Рис. 3.1. Схематична структура онтологічної моделі когнітивного веб-сервісу [26]

Наведена на рисунку схема ілюструє онтологічну модель когнітивного веб-сервісу, яка описує взаємозв'язок між користувачами, веб-сервісами, типами даних, задачами, а також обмеженнями та параметрами виконання сервісів.

Будь-який когнітивний веб-сервіс орієнтований на взаємодію з користувачами, які мають різні ролі, потреби та технічні характеристики. Для забезпечення персоналізованого досвіду та ефективності роботи система повинна враховувати індивідуальні параметри кожного користувача. Ця частина моделі дозволяє зрозуміти, як описуються користувачі в рамках системи. Наводимо класи користувачів та їх властивості:

- **Class User** представляє користувачів, які взаємодіють із системою.
 - **Data Property hasRole** визначає роль користувача у системі (наприклад, «адміністратор», «користувач»).
 - **Data Property hasLanguage** вказує мову користувача, що дозволяє адаптувати сервіси для його зручності.
 - **Data Property hasUuid** унікальний ідентифікатор для кожного користувача, що дозволяє ідентифікувати його в системі.

Когнітивні сервіси – це основа платформи, яка вирішує задачі користувача. Вони дозволяють обробляти різні типи даних, виконувати інтелектуальні завдання та забезпечують гнучкість у використанні. У цій моделі класи сервісів детально описують їх функціонал, взаємодію з даними та якісні характеристики:

- **Class WebService** представляє когнітивні веб-сервіси, що виконують конкретні завдання.
 - **Object Property hasInput** вказує, які типи даних можуть бути вхідними для сервісу.
 - **Object Property hasOutput** визначає типи даних, які сервіс повертає як результат.
 - **Object Property hasQoS** описує параметри якості обслуговування (QoS), такі як: **responseTime** (час відповіді сервісу), **errorRate** (рівень помилок), **uptime** (час безвідмовної роботи сервісу).

– **Class CognitiveWebService** – підклас **WebService**, що включає специфічні когнітивні сервіси: **TextToSpeech** (сервіс для перетворення тексту в мову), **ComputerVision** (сервіс для аналізу зображень).

Когнітивні сервіси працюють із широким спектром даних, таких як текст, зображення, аудіо чи числа. Ця частина моделі дає змогу описати формати, типи та специфіку даних, які використовуються для обробки та аналізу. Вона також забезпечує гнучкість у роботі з різноманітними задачами.

– **Class DataType** описує різні типи даних, з якими працюють сервіси: **text**: Текстові дані), **image** (зображення), **audio** (аудіофайли), **numbers** (числові дані), **data Property hasFormat** (визначає формат даних (наприклад, .jpg, .wav)), **data Property hasValue** (вказує конкретне значення даних (наприклад, текстовий рядок чи числове значення)).

Кожен користувач має конкретну задачу, яку необхідно вирішити за допомогою когнітивних сервісів. Наведена модель дозволяє описати типи задач, які підтримує система, та їх класифікацію, забезпечуючи адаптивність платформи під різні сценарії використання.

- **Class TaskType** описує типи задач, які можуть виконувати сервіси:
 - **MachineTranslation** (машинний переклад тексту).
 - **Recognition** (розпізнавання, наприклад, тексту чи об'єктів).
 - **Transformation** (перетворення даних, наприклад, зміна формату чи структури).
 - **Prediction** (передбачення на основі вхідних даних).

Результати роботи сервісів повинні мати чітку структуру та відповідати очікуванням користувача. Відтак саме ця частина моделі описує, як представити вихідні дані після виконання задач. Вона також дозволяє зберігати додаткову інформацію, таку як метадані чи ідентифіковані об'єкти.

- **Class ClassificationResult** – результати, отримані після виконання когнітивного завдання:
 - **Class LocationEntities** (географічні об'єкти, знайдені у даних).

- **Class PersonNameEntities** (імена людей, ідентифіковані у текстах чи інших даних).

Будь-який веб-сервіс чи задача має певні обмеження, такі як час виконання, точність чи мова. Для успішного виконання завдань важливо враховувати ці фактори. Тому частина моделі, яка за це відповідає дає змогу визначити конкретні обмеження, пов'язані із завданням чи сервісом:

– **Class Constraint**: Загальний клас для визначення обмежень на виконання задач:

- **Class TimeConstraint** відображає обмеження за часом виконання.
- **Class BatchSizeConstraint** відображає обмеження на кількість оброблюваних даних у пакеті.
- **Class AccuracyConstraint** відображає вимоги до точності виконання.
- **Class LanguageConstraint** відображає обмеження, пов'язані з підтримкою конкретних мов.
- **Object Property hasConstraint** вказує, яке обмеження застосовується до сервісу чи задачі.

Отже, онтологія не лише структуровано описує елементи когнітивного веб-сервісу, але й забезпечує основи для його адаптації, ефективності та персоналізації під індивідуальні потреби користувачів.

3.5. Реалізація додатку для обробки запитів на основі семантичної структури

Реалізація веб-сервісу для обробки запитів на основі семантичної структури полягає у створенні інтерактивного інтерфейсу, який дозволяє користувачам легко формулювати запити природною мовою та отримувати результати у зручному форматі. Основним компонентом цього веб-сервісу є семантичний аналіз, що

здійснюється на серверній стороні, де запити узгоджуються з концепціями онтології.

У веб-сервісі реалізовано логічне висновування, яке формує послідовність сервісів для виконання запиту. Користувачі мають можливість завантажувати файли для аналізу, а система автоматично обробляє дані, передаючи їх між сервісами у визначеній послідовності. Результати виконання запитів виводяться у зрозумілому та візуально привабливому форматі, що забезпечує зручність у роботі з веб-сервісом.

3.5.1. Концептуальна архітектура додатку

Концептуальна архітектура веб-сервісу для обробки запитів на основі семантичної структури є основою для реалізації функціоналу, який забезпечує ефективну взаємодію між користувачами та сервісами. Основними компонентами архітектури є користувацький інтерфейс, серверна частина, семантична онтологія та механізм обробки запитів.

Користувацький інтерфейс (UI) розроблено з урахуванням зручності та інтуїтивності, що дозволяє користувачам легко вводити запити та отримувати результати. Він містить форми для введення тексту, завантаження файлів і перегляду результатів у зручному форматі, що сприяє швидкій обробці запитів.

Серверна частина відповідає за обробку запитів, здійснюючи семантичний аналіз, логічне висновування та виконання запитів. Сервер сприймає запити у форматі JSON, узгоджує їх із концепціями онтології та визначає відповідні сервіси, що підвищує точність і ефективність обробки. Механізм обробки запитів організовує виклик сервісів у визначеній послідовності та керує передачею даних між ними.

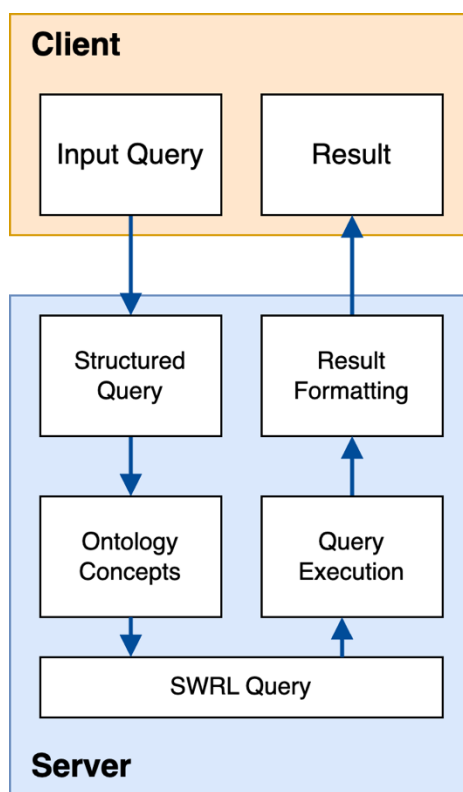


Рис. 3.2. Послідовність дій обробки запиту в клієнт-сервісній архітектурі [26]

Семантична онтологія слугує основою для семантичного аналізу, забезпечуючи структуру знань, що описує взаємозв'язки між поняттями, що, як наслідок, дозволяє системі точно розуміти запити користувачів і виконувати їх з максимальною ефективністю.

Відтак, представлена концептуальна архітектура створює гнучку і масштабовану систему, яка здатна адаптуватися до змінюваних вимог користувачів і розширюватися новими сервісами, зберігаючи при цьому високу продуктивність і точність обробки запитів.

Процес обробки запиту в системі когнітивного веб-сервісу полягає в тому, що на початковому етапі система взаємодіє з користувачем, який формулює свій запит у природній мові, що може бути представлене в текстовій або голосовій формі. Клієнтська частина веб-сервісу відповідальна за збір цього запиту і його передачу на сервер для подальшої обробки, забезпечуючи гнучкість, дозволяючи

користувачам зручно взаємодіяти з системою, не вимагаючи від них спеціальних знань чи навичок.

Після отримання запиту сервер виконує інтеграцію з GPT (Generative Pre-trained Transformer), що дозволяє інтерпретувати запит і перетворити його у формалізований вигляд, відповідно до специфікацій системи. Розуміння та формалізація вхідного запиту є складним завданням. При налаштуванні OpenAI GPT надається підказка:

Твоя задача – перетворити запити природною мовою в структурований JSON-формат. Визначай:

- Тип вхідних даних (``inputType``) – що потрібно обробити.
- Тип вихідних даних (``outputType``) – що очікується отримати.
- Обмеження (``constraints``) – наприклад, максимальний час виконання або інші специфікації (необов'язково).

Приклад. Користувач формує запит у вигляді:

«Перетворити голосовий запис на список локацій не довше ніж за 10 секунд».

За допомогою API OpenAI GPT запит перетворюється у формалізований вигляд:

```
{
  "inputType": "Audio", "outputType": "LocationEntities", "constraints": {
    "maxExecutionTime": 10
  }
}
```

Так як ***LocationEntities*** є підтипом класу результатів класифікації, цей

```
{
  "inputType": "Audio", "outputType": "ClasificationResult", "constraints": {
    "maxExecutionTime": 10, "resultType": "LocationEntities"
  }
}
```

потрібно нормалізувати і привести до вигляду:

Оброблений запит далі аналізується через онтологію та застосовуються SWRL-правила (Semantic Web Rule Language), що визначають логіку виконання

запиту. Сервер також координує виконання різних сервісів, необхідних для реалізації запиту, що забезпечує ефективність і швидкість обробки.

Онтологія є центральним елементом системи, який описує ключові концепції, включаючи сервіси, дані та обмеження. Вона формує структуру знань, на основі якої відбувається обробка запиту. Використання SWRL дозволяє виконувати логічне висновування, що перетворює початковий запит у послідовність викликів сервісів; забезпечує узгодженість і точність виконання запитів, а також дозволяє системі адаптуватися до різних сценаріїв використання.

Перетворення запиту в послідовність викликів сервісів складається із таких етапів:

1. Встановлення зв'язків між сервісами: SWRL-правила знаходять сервіси, вихідні дані яких (hasOutput) відповідають вхідним даним наступного сервісу (hasInput):

```
Service(?s1) ^ hasOutput(?s1, ?o) ^ Service(?s2) ^ hasInput(?s2, ?o) ^
maxExecutionTime(?s1, ?t1) ^ maxExecutionTime(?s2, ?t2) ^ swrlb:add(?totalTime,
?t1, ?t2) ^ swrlb:lessThanOrEqual(?totalTime, 10) -> Connected(?s1, ?s2)
```

2. Визначення початкового та кінцевого сервісів:

```
Service(?s) ^ hasInput(?s, Audio) -> StartService(?s) .
Service(?s) ^ hasOutput(?s, ClasificationResult) -> EndService(?s) .
```

3. Побудова ланцюга сервісів: логічний вивід формує послідовність, що відповідає запиту:

```
StartService(?s1) ^ Connected(?s1, ?s2) ^ EndService(?s2) -> ServiceChain(?s1,
```

Джерело: [26]

Отже, концептуальна архітектура веб-сервісу для обробки запитів розроблена для сприяння ефективній взаємодії між користувачем і сервісом через його основні компоненти: інтерфейс користувача, серверну частину, семантичну онтологію та механізм обробки запитів. Інтерфейс користувача надає перевагу зручності використання, дозволяючи користувачам легко надсилати запити та

переглядати результати, тоді як сервер обробляє ці запити за допомогою семантичного аналізу та логічного висновку, підвищуючи точність та ефективність. Семантична онтологія лежить в основі розуміння системою запитів користувачів, дозволяючи їй адаптуватися до мінливих вимог і розширюватися за допомогою нових сервісів, що в кінцевому підсумку забезпечує високу продуктивність обробки запитів.

3.5.2. Алгоритмічна модель обробки запитів

Система когнітивного веб-сервісу реалізує комплексний підхід до обробки запитів користувачів, перетворюючи їх із природної мови в структуровану форму та забезпечуючи точне виконання задач. Усі дії системи розділені на п'ять послідовних етапів, кожен із яких відповідає за певний аспект роботи: від аналізу запиту до формування кінцевого результату. Такий підхід дозволяє забезпечити високу точність, ефективність і відповідність отриманого результату початковим потребам користувача. Алгоритмічна модель обробки запитів відображена на рис. 3.3.

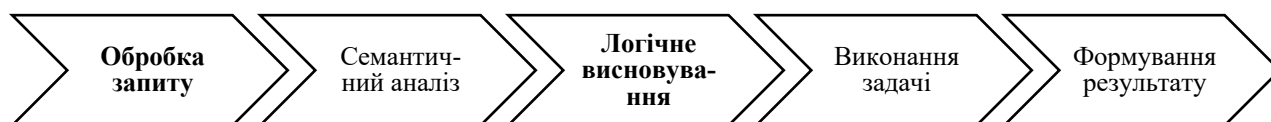


Рис. 3.3. Алгоритмічна модель обробки запитів

Обґрунтовуючи алгоритмічну модель обробки запитів, відзначаємо, що на початковому етапі система приймає запит користувача, сформульований природною мовою. Цей запит автоматично перетворюється у формат JSON, де чітко зазначаються параметри задачі: тип вхідних і вихідних даних, а також можливі обмеження чи специфічні вимоги. Це забезпечує структурованість запиту, необхідну для подальшої обробки.

Далі, на етапі системного аналізу, запит аналізується з урахуванням онтології – бази знань, яка описує концепції та зв'язки між ними в рамках системи. У цьому процесі визначається, які сервіси найкраще відповідають потребам користувача.

Семантичний аналіз дозволяє співставити запит із доступними можливостями, забезпечуючи релевантність обробки.

Після аналізу система використовує логічні правила SWRL (Semantic Web Rule Language) для побудови послідовності виконання сервісів. Ці правила визначають, які сервіси слід викликати та в якому порядку, щоб досягти результату. Таким чином, формується оптимальна схема виконання задачі.

Сервіси запускаються у визначеній послідовності, відповідно до збудованого логічного ланцюжка. Дані, отримані від одного сервісу, автоматично передаються наступному, забезпечуючи безперервний процес обробки. Цей етап гарантує коректне виконання кожного елемента задачі.

На завершальному етапі система формує фінальний результат, який передається користувачу. Результат може бути представлений у різних форматах (текст, граф, зображення тощо) залежно від поставленої задачі. Таким чином, користувач отримує розв'язання, адаптоване до його початкових вимог.

На основі наданого фрагмента коду client.py (Додаток Б), алгоритмічна модель обробки запитів може бути детально описана через реалізацію кожного з ключових етапів, що складають цю модель.

1. Обробка запиту

На початку роботи програми виконується обробка запиту, яка представлена у формі користувацького інтерфейсу за допомогою Streamlit. Користувач вводить свій запит у текстове поле, а також може завантажити файл для обробки. Код для цього виглядає так:

```
query = st.text_area("☁️ Ваш запит:", placeholder="Опишіть, що потрібно зробити")  
file = st.file_uploader("📎 Файл (якщо потрібно):")
```

Після натискання кнопки «Виконати», запит та вміст файлу кодуються у форматі JSON, готові для відправки на сервер:

```
data = {  
    "query": query,  
    "file": encode_file_content(file) if file else None  
}
```

2. Семантичний аналіз

Коли запит надіслано на сервер, сервер виконує семантичний аналіз запиту, узгоджуючи його з концепціями онтології. У коді цей процес не детально представлений, але важливо відзначити, що запит обробляється сервером через REST API, який, як видно з запиту:

```
response = requests.post("http://localhost:8083/process", json=data)
```

3. Логічне висновування

Логічне висновування в системі базується на обробці даних, що отримуються від сервісів. Код містить функції для візуалізації ланцюга сервісів:

```
st.graphviz_chart(render_service_chain(result['workflow']))
```

Ця частина коду формує візуалізацію ланцюга сервісів, які були задіяні у виконанні запиту, а також демонструє логіку, з якою ці сервіси взаємодіють один з одним.

4. Виконання задачі

Виконання задачі включає в себе виклик відповідних сервісів у визначеній послідовності. Сервер обробляє запит і викликає необхідні сервіси, передаючи дані між ними. Хоча сам процес виклику сервісів не відображено безпосередньо в коді клієнта, але вважається, що він реалізується на стороні сервера, до якого направляється запит.

5. Формування результату

Після виконання запиту та обробки даних, результат генерується для користувача. У коді є частина, яка відповідає за вивід результатів:

```
st.subheader("✅ Результати")
st.json(result['results'])
```

Ця секція виводить фінальний результат запиту у зрозумілому для користувача форматі.

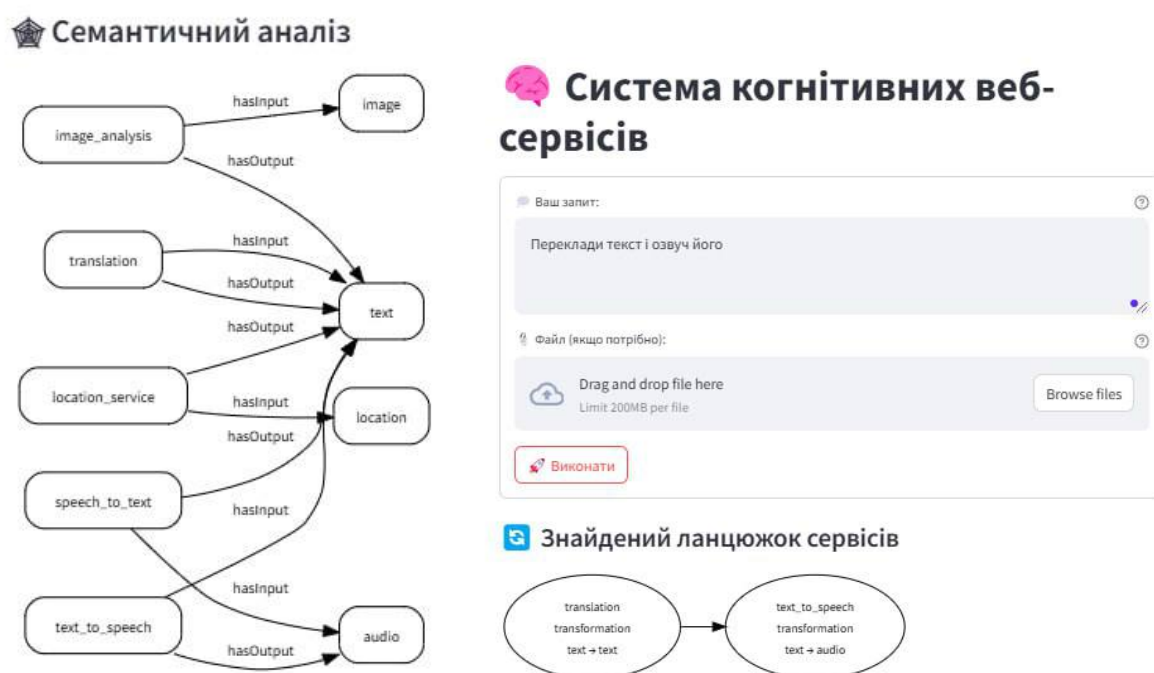


Рис. 3.4. Інтерфейс користувача для взаємодії з платформою

Отже, фрагмент коду `client.py` (Додаток А) демонструє, як алгоритмічна модель обробки запитів реалізується у практичному контексті через інтерактивний веб-сервіс. Вона включає в себе всі основні етапи, такі як обробка запиту, семантичний аналіз, логічне висновування, виконання задачі та формування результату, що забезпечує зручність і ефективність у використанні когнітивних веб-сервісів.

Висновки до розділу 3

Оркестрування є потужним інструментом для централізованого управління виконанням задач у когнітивних веб-сервісах. Воно забезпечує чітку координацію між сервісами, ефективне використання ресурсів і адаптивність до змінних умов чи вимог. Хореографія забезпечує високу гнучкість і адаптивність системи, дозволяючи сервісам взаємодіяти без централізованого контролю, що робить систему більш масштабованою, оскільки нові сервіси можуть легко інтегруватися, просто дотримуючись правил взаємодії. Такий підхід ідеально підходить для динамічних середовищ, де вимоги та доступні сервіси постійно змінюються.

Онтологія не лише структуровано описує елементи когнітивного веб-сервісу, але й забезпечує основи для його адаптації, ефективності та персоналізації під індивідуальні потреби користувачів. Основним елементом онтології є клас користувачів (User), що взаємодіють із системою. Користувачі описуються через такі властивості: роль у системі (hasRole), мова взаємодії (hasLanguage), ідентифікатор (hasUuid). Клас WebService описує веб-сервіси, які виконують інтелектуальні задачі. У нього входять підкласи, наприклад, CognitiveWebService, що включають спеціалізовані сервіси, такі як TextToSpeech (перетворення тексту в мову) та ComputerVision (аналіз зображень). Властивості сервісів включають вхідні та вихідні типи даних (hasInput, hasOutput) і параметри якості обслуговування (hasQoS). Клас DataType описує типи даних, які обробляють сервіси: текст, зображення, аудіо, числа. Властивості даних включають формат (hasFormat) та значення (hasValue), що забезпечує гнучкість у роботі з різноманітними даними. Клас TaskType описує типи задач, які підтримує система: машинний переклад (MachineTranslation), розпізнавання (Recognition), перетворення даних (Transformation), прогнозування (Prediction). Клас ClassificationResult описує вихідні дані, отримані після виконання задач. Підкласи, такі як LocationEntities (географічні об'єкти) чи PersonNameEntities (імена людей), деталізують специфіку результатів. Клас Constraint дозволяє визначити обмеження, пов'язані із сервісами чи задачами: час виконання (TimeConstraint), точність (AccuracyConstraint), підтримувані мови (LanguageConstraint).

Концептуальна архітектура веб-сервісу для обробки запитів розроблена для сприяння ефективній взаємодії між користувачем і сервісом через його основні компоненти: інтерфейс користувача, серверну частину, семантичну онтологію та механізм обробки запитів. Інтерфейс користувача надає перевагу зручності використання, дозволяючи користувачам легко надсилати запити та переглядати результати, тоді як сервер обробляє ці запити за допомогою семантичного аналізу та логічного висновку, підвищуючи точність та ефективність. Семантична онтологія лежить в основі розуміння системою запитів користувачів, дозволяючи їй адаптуватися до мінливих вимог і розширюватися за допомогою нових веб-сервісів, що в кінцевому підсумку забезпечує високу продуктивність обробки запитів.

Система когнітивного веб-сервісу використовує структуровану алгоритмічну модель обробки запитів користувачів. Спочатку запити користувача природною мовою перетворюються в структурований формат JSON, що дозволяє визначити точні параметри завдання. Потім система аналізує запит із базою знань, визначає оптимальну послідовність виконання веб-сервісом завдань за допомогою правил SWRL і, зрештою, представляє кінцевий результат користувачеві в різних форматах, гарантуючи, що вихідні дані відповідають початковим вимогам користувача.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОЇ СЕМАНТИЧНОЇ СТРУКТУРИ

4.1. Методологія оцінки ефективності семантичної структури

Оцінка ефективності семантичної структури проводиться на основі метрик, що характеризують продуктивність пошуку, релевантність знайдених сервісів та відповідність вимогам якості обслуговування (QoS).

Критерії оцінки продуктивності семантичної структури визначають, наскільки ефективно вона виконує пошук і композицію сервісів.

Запропоновані наступні критерії для порівняння:

- **Час виконання** – середній час виконання запитів на пошук і композицію сервісів;
- **Точність пошуку (Precision):**

$$\text{Precision} = \frac{\text{Релевантні сервіси, знайдені системою}}{\text{Всі знайдені сервіси}} \quad (4.1),$$

- **Повнота пошуку (Recall):**

$$\text{Recall} = \frac{\text{Релевантні сервіси, знайдені системою}}{\text{Всі релевантні сервіси}} \quad (4.2),$$

- **Ступінь відповідності QoS (Fitness)** – відображає, наскільки послідовність сервісів відповідає заданим обмеженням запиту (час виконання, вартість):

$$\text{Fitness} = 0.5 * \text{TimeScore} + 0.5 * \text{CostScore} \quad (4.3),$$

$$\text{де TimeScore} = \frac{\text{MaxTime} - \text{ActualTime}}{\text{MaxTime}}, \text{CostScore} = \frac{\text{MaxCost} - \text{ActualCost}}{\text{MaxCost}}$$

У дослідженні «QoS-aware Service Composition Using Fuzzy Set Theory and Genetic Algorithms» представлено підхід до складання веб-сервісів з урахуванням якості обслуговування (QoS), який використовує теорію нечітких множин та генетичні алгоритми. У цьому підході функція пристосованості (fitness function) використовується для вибору парето-оптимальних наборів сервісів та усунення менш ефективних, порівнюючи їхні QoS-показники. Мета полягає в максимізації

таких атрибутів QoS, як надійність та доступність, одночасно мінімізуючи інші, наприклад, вартість та затримку [75].

Формула функції пристосованості за Jiajun Xu враховує різні QoS-атрибути та їхні вагові коефіцієнти, що відображають пріоритети користувача. Загальна структура такої функції може бути представлена як зважена сума нормалізованих QoS-атрибутів:

$$Fitness = w1 * QoS1 + w2 * QoS2 + ... + wn * QoS_n \quad (4.4),$$

де $w1, w2, ..., wn$ – вагові коефіцієнти, що відповідають важливості кожного QoS-атрибути, а $QoS1, QoS2, ..., QoS_n$ – нормалізовані значення відповідних атрибутів якості обслуговування.

Використовуючи цей підхід можна ефективно оцінювати та обирати оптимальні комбінації сервісів, забезпечуючи баланс між різними аспектами QoS відповідно до вимог користувача.

Отже, ефективність семантичних структур у пошуку та композиції сервісів оцінюється за допомогою різних показників продуктивності, включаючи час виконання, точність пошуку, повноту пошуку та ступінь відповідності QoS.

4.2. Оцінка функціональних можливостей

Для визначення переваг запропонованого методу було проведено порівняння ключових характеристик (*семантична точність, здатність до автоматичної композиції, врахування QoS-параметрів, адаптивність до змін*) між існуючими підходами (*пошук на основі ключових слів, семантичний пошук із фіксованими правилами, композиція на основі шаблонів*) та розробленим методом автоматизації інтеграції та оркестрації когнітивних веб-сервісів. Результати порівняння відображено у табл. 4.1.

Таблиця 4.1

Результати порівняння ключових характеристик когнітивних веб-сервісів

Критерій	Існуючі методи	Запропонований метод
Семантична точність	Використання семантичних описів сервісів обмежене фіксованими правилами.	Динамічне планування та логічні висновки дозволяють точніше обробляти складні запити.
Адаптивність до змін	Обмежена, оскільки використовуються статичні шаблони композиції.	Автоматичне оновлення ланцюгів сервісів при зміні доступності чи появи нових сервісів.
QoS-параметри	Часткове врахування (лише ранжування результатів за QoS).	Інтеграція QoS у процеси планування та виконання робочих процесів.
Масштабованість	Ефективність знижується при збільшенні кількості сервісів чи складності запитів.	Висока ефективність у масштабованих середовищах завдяки семантичним правилам і логіці.

Із результатів відображених у табл. 4.1 бачимо, що семантична точність – у традиційних методах використовується як чітко та точно заданий набір правил для визначення відповідності сервісів запиту, який, відповідно, обмежує можливість обробки складних запитів. Відзначимо, що запропонований підхід використовує динамічне планування та логічні висновки, що дозволяє точніше розпізнавати зміст запиту та підбирати найбільш релевантні сервіси.

Адаптивність до змін характеризується тим, що у традиційних системах композиція сервісів базується на статичних шаблонах, що ускладнює оновлення системи при зміні доступності сервісів. Відтак, у новому підході система автоматично адаптується, оновлюючи ланцюги сервісів відповідно до змін у середовищі.

QoS-параметри демонструють те, що більшість існуючих методів враховують якість обслуговування (QoS) лише на етапі ранжування знайдених сервісів. Запропонований метод інтегрує QoS-показники (наприклад, час виконання, вартість) безпосередньо в процеси планування та виконання, що дозволяє оптимізувати вибір сервісів ще на етапі формування запиту.

Масштабованість як критерій означає, що традиційні системи можуть втрачати ефективність із ростом кількості сервісів та складності запитів.

Запропонований підхід, завдяки використанню семантичних правил і логіки, підтримує високу продуктивність навіть у масштабованих середовищах, забезпечуючи швидкий пошук та обробку запитів.

Отже, із здійсненого порівняння видно, що за розробленим методом автоматизації інтеграції та оркестрації когнітивних веб-сервісів забезпечується гнучкість, точність і адаптивність системи, які роблять її більш ефективною в умовах динамічного середовища.

4.3. Експериментальне дослідження ефективності пошуку та композиції сервісів

Для оцінки ефективності методу було згенеровано описи для різних когнітивних сервісів: SpeechToText, TextToSpeech, ImageClassification, ObjectDetection, SentimentAnalysis, NamedEntityRecognition, MachineTranslation, FaceRecognition, KeywordExtraction, AudioClassification, TextSummarization, ImageCaptioning, VideoClassification, AnomalyDetection, DocumentClassification тощо. Реалізовано сервіси SpeechToText, TextToSpeech, TextClassification, MachineTranslation та визначено запити для виконання цих сервісів та їх композиції.

Наводимо приклади запитів для військової сфери, які можуть бути оброблені розробленим когнітивним веб-сервісом, із зазначенням відповідних послідовностей сервісів:

1. Розпізнати військову техніку на фото або відео

Запит: «Визначити тип військової техніки на зображенні»

*Послідовність сервісів: **ImageClassification** > **ObjectDetection***

В умовах бойових дій важливо швидко ідентифікувати тип військової техніки, що знаходиться в зоні спостереження. За допомогою сервісів **ImageClassification** та **ObjectDetection** система аналізує зображення або відео,

визначаючи, чи містить воно військові об'єкти, і класифікує їх (наприклад, танк, бронетранспортер або артилерійська установка), що дозволяє військовим оперативно реагувати на загрози або скоригувати свої дії.

2. Аналіз перехопленого аудіо повідомлення

Запит: «Перетворити голосове повідомлення на текст, визначити його тональність і виділити ключові об'єкти»

*Послідовність сервісів: **SpeechToText** > **SentimentAnalysis** > **NamedEntityRecognition***

При роботі з перехопленими розмовами критично важливо швидко отримати зміст і зрозуміти, яку інформацію вони містять. Використовуючи **SpeechToText**, система перетворює аудіо в текст, після чого **SentimentAnalysis** аналізує емоційне забарвлення повідомлення (наприклад, чи воно панічне або впевнене), а **NamedEntityRecognition** виділяє ключові імена, місця або події.

3. Моніторинг ризиків у повідомленнях розвідки

Запит: «Перекласти та проаналізувати іноземний текстовий звіт на предмет загроз»

*Послідовність сервісів: **MachineTranslation** > **TextClassification** > **AnomalyDetection***

Оскільки досить часто інформація з відкритих джерел або агентурних повідомлень часто потребує перекладу та аналізу, когнітивний веб-сервіс у разі необхідності спочатку застосовує **MachineTranslation** для автоматичного перекладу тексту, потім **TextClassification** для його тематичного аналізу (наприклад, чи міститься загроза або заклики до активних дій), а на завершення **AnomalyDetection** виявляє підозрілі патерни або нестандартні вирази.

4. Виявлення незвичних активностей у відеопотоці

Запит: «Аналізувати відео з дрону та виявити підозрілу активність»

*Послідовність сервісів: **VideoClassification** > **AnomalyDetection***

Автоматичний аналіз відеопотоку, наприклад, з безпілотників, дозволяє виявляти підозрілу активність у реальному часі. Спочатку **VideoClassification** аналізує загальну сцену, визначаючи її контекст (наприклад, військовий табір або скупчення транспорту), а **AnomalyDetection** фіксує нестандартну поведінку – наприклад, незвичне пересування людей або техніки, що може свідчити про підготовку до атаки або евакуацію.

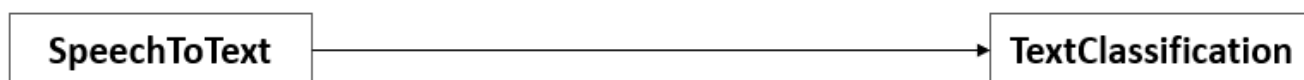
5. Збір ключової інформації з радіоперехоплення

Запит: «Розпізнати мову в аудіо, витягти ключові слова та класифікувати за рівнем загрози»

*Послідовність сервісів: **AudioClassification** > **SpeechToText** > **KeywordExtraction** > **TextClassification***

Військові операції часто супроводжуються перехопленням аудіоповідомлень противника. Щоб пришвидшити їх обробку, система застосовує **AudioClassification** для визначення типу запису (переговори, радіопередача, сигнали), потім **SpeechToText** переводить мовлення у текст, **KeywordExtraction** виділяє основні поняття (імена, місця, команди), а **TextClassification** визначає зміст повідомлення (наприклад, наказ, запит про підкріплення або розвіддані).

Розглянемо приклад реалізації одного із запитів. До прикладу, користувач надав запит: «Перетворити голосовий запис на список міст не довше ніж за 10 секунд» та вхідні дані (голосові записи тексту). Запит генерує послідовність з 2 сервісів відповідно до вхідних/вихідних даних та умов класифікації:



```

<hasInput>
  <owl:NamedIndividual rdf:about="cg:AudioInput">
    <rdf:type rdf:resource="cg:Audio"/>
    <format rdf:dataType="http://www.w3.org/2001/XMLSchema#string">wav</format>
    <URI rdf:dataType="http://www.w3.org/2001/XMLSchema#string">http://localhost:8080/uploads/voice.wav</URI>
  </owl:NamedIndividual>
</hasInput>
<hasConstraint>
  <owl:NamedIndividual rdf:about="cg:STTConstraint">
    <rdf:type rdf:resource="cg:Constraint"/>
    <maxExecutionTime rdf:dataType="http://www.w3.org/2001/XMLSchema#integer">10</maxExecutionTime>
    <resultType rdf:dataType="http://www.w3.org/2001/XMLSchema#string">LocationEntities</resultType>
  </owl:NamedIndividual>
</hasConstraint>

<hasOutput>
  <owl:NamedIndividual rdf:about="cg:TextOutput">
    <rdf:type rdf:resource="cg:Text"/>
    <language rdf:dataType="http://www.w3.org/2001/XMLSchema#string">en</language>
    <value rdf:dataType="http://www.w3.org/2001/XMLSchema#string">
      Sample transcription mentioning Paris and New York
    </value>
  </owl:NamedIndividual>
</hasOutput>

```

```

<hasInput>
  <owl:NamedIndividual rdf:about="cg:TextOutput">
    <rdf:type rdf:resource="cg:Text"/>
    <language rdf:dataType="http://www.w3.org/2001/XMLSchema#string">en</language>
    <content rdf:dataType="http://www.w3.org/2001/XMLSchema#string">
      Sample transcription mentioning Paris and New York
    </content>
  </owl:NamedIndividual>
</hasInput>
<hasConstraint>
  <owl:NamedIndividual rdf:about="cg:TCSConstraint">
    <rdf:type rdf:resource="cg:Constraint"/>
    <maxExecutionTime rdf:dataType="http://www.w3.org/2001/XMLSchema#integer">10</maxExecutionTime>
    <resultType rdf:dataType="http://www.w3.org/2001/XMLSchema#string">LocationEntities</resultType>
  </owl:NamedIndividual>
</hasConstraint>

<hasOutput>
  <owl:NamedIndividual rdf:about="cg:LocationEntitiesOutput">
    <rdf:type rdf:resource="cg:LocationEntities"/>
    <entities>
      <Entity>
        <Name rdf:dataType="http://www.w3.org/2001/XMLSchema#string">Paris</Name>
        <Type rdf:dataType="http://www.w3.org/2001/XMLSchema#string">City</Type>
      </Entity>
      <Entity>
        <Name rdf:dataType="http://www.w3.org/2001/XMLSchema#string">New York</Name>
        <Type rdf:dataType="http://www.w3.org/2001/XMLSchema#string">City</Type>
      </Entity>
    </entities>
  </owl:NamedIndividual>
</hasOutput>

```

Ри. 4.1. Приклад реалізації одного із запитів

Таким чином бачимо, що наведені приклади запитів ілюструють, як когнітивний веб-сервіс може допомогти у військовій сфері, автоматизуючи аналіз даних, підвищуючи швидкість обробки інформації та забезпечуючи більш точне прийняття рішень.

4.4. Порівняльний аналіз із альтернативними підходами

Альтернативні підходи до семантичного пошуку та композиції веб-сервісів значно відрізняються за методологією та ефективністю. Розглянемо три основні підходи, представлені у наукових доробках А. Bhuvaneswari, К. Sumathi, Velliangiri Sarveshwaran & А. Sivasangari [62], Luo, G., Tan, W., Fan, J., & Zhou, Q. [79], Klusch, M., Fries, B., & Sycara, K. [76].

Відтак, у публікації А. Bhuvaneswari, К. Sumathi, Velliangiri Sarveshwaran & А. Sivasangari «Hybrid deep learning and similarity measures for semantic web service discovery» (2024) у досліджуваному підході поєднуються методи глибокого навчання та подібності для покращення пошуку семантичних веб-сервісів. Із вивчених матеріалів бачимо, що використання гібридного підходу дозволяє зменшити проблему синтаксичного та семантичного невідповідності між запитом

користувачів і метаданими сервісів. Основна перевага – покращена точність пошуку за рахунок використання глибоких нейронних мереж [62].

Метод представлений колективом авторів в особі Luo, G., Tan, W., Fan, J., & Zhou, Q. зосереджується на оптимізації композиції веб-сервісів з урахуванням параметрів якості обслуговування (QoS). Використовуються генетичні алгоритми та нечітка логіка для підбору найбільш підходящих сервісів, що дозволяє покращити баланс між продуктивністю та відповідністю користувацьким вимогам [79].

Система iSeM використовує адаптивне гібридне вибіркове моделювання та наближене логічне виведення для ефективного вибору семантичних сервісів. Вона дозволяє зменшити витрати на обчислення та підвищити продуктивність за рахунок використання методів нечіткої логіки та евристичних алгоритмів [77].

Здійснимо узагальнення запропонованого методу та альтернативних підходів інших авторів (табл. 4.2).

Таблиця 4.2

Порівняльний аналіз запропонованого підходу із альтернативними

Критерій	Hybrid DL & Similarity (2024)	QoS-Aware Composition (2011)	iSeM (2009)
Семантична точність	Висока	Середня	Висока
Адаптивність до змін	Обмежена	Висока	Висока
Врахування QoS	Часткове	Повне	Часткове
Масштабованість	Середня	Середня	Висока
Обчислювальна ефективність	Середня	Висока	Висока

Виходячи із наведених у табл. 4.2 даних, бачимо, що запропонований метод забезпечує високу адаптивність до змін завдяки використанню онтологічного підходу та динамічного планування сервісів. Він також повністю інтегрує QoS-параметри, що дозволяє оптимізувати процеси пошуку та композиції сервісів у реальному часі.

Також здійснимо графічне порівняння результатів (рис. 4.2).

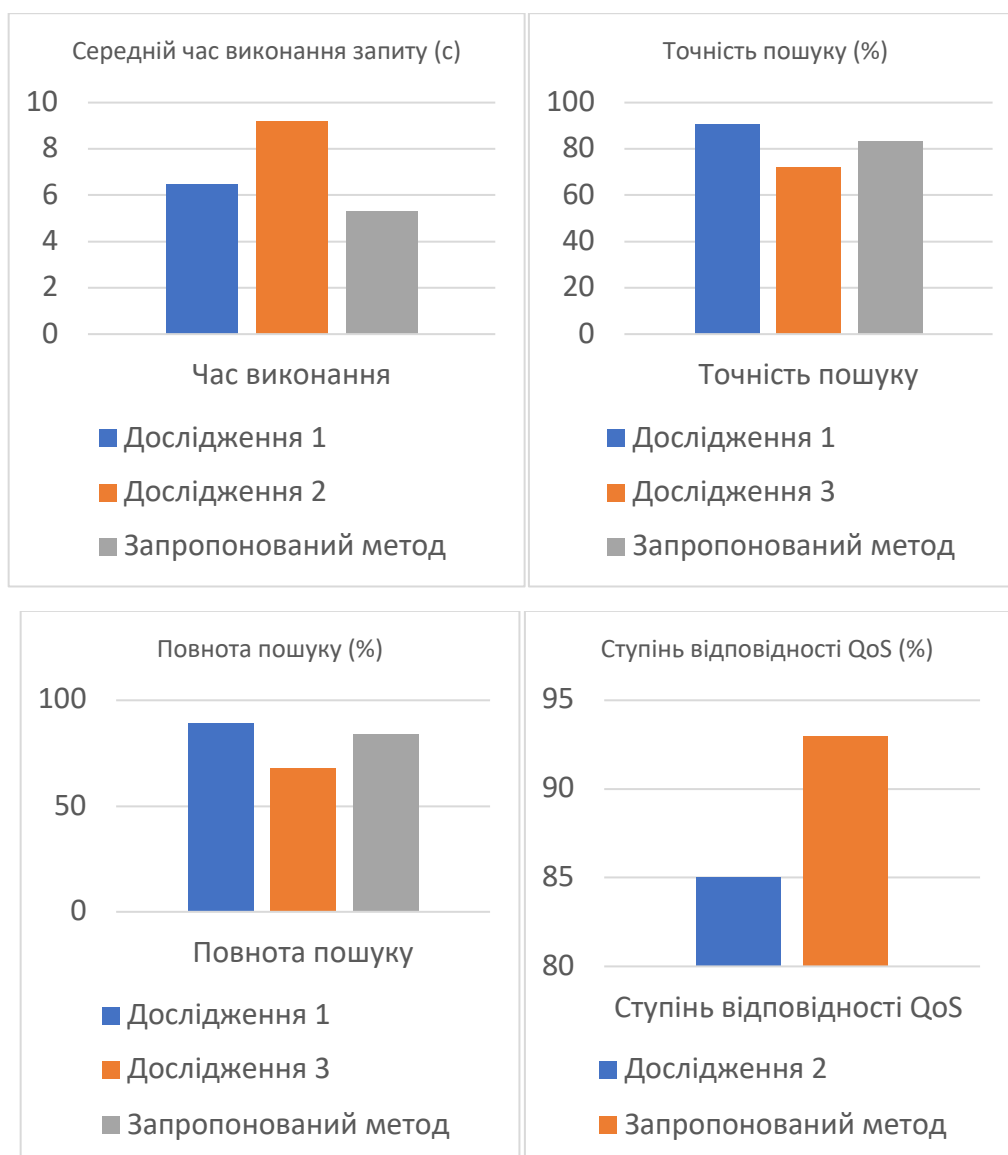


Рис. 4.2. Графічне порівняння результатів [26]

Рисунок 4.1 ілюструє продуктивність фреймворку за такими показниками, як час виконання, точність, повнота та відповідність QoS. Динамічні висновки, що здійснюються за допомогою правил SWRL, суттєво сприяють досягненню цих результатів шляхом:

- скорочення часу виконання завдяки ефективному вибору сервісів;
- підвищення точності через забезпечення семантичної відповідності між сервісами та запитами;
- покращення повноти шляхом врахування всіх сумісних сервісів у межах заданих обмежень.

Запропонований фреймворк для створення когнітивних веб-сервісів має низку переваг над традиційними підходами, зокрема з точки зору семантичної точності, адаптивності та автоматизації. У таблиці 4.3 наведено порівняльний аналіз нашого підходу з іншими відомими парадигмами створення сервісів, включаючи системи на основі ключових слів, сервіс-орієнтовані архітектури (SOA), семантичні веб-сервіси (OWL-S) та платформи когнітивних обчислень (наприклад, IBM Watson).

Таблиця 4.3

Порівняння запропонованої системи з існуючими підходами

Особливість	Запропонована структура	Keyword-Based	SOA (BPEL, WS-CDL)	Semantic Web Services (OWL-S)	Cognitive AI (IBM Watson, LangChain)
Семантична обізнаність	Високий	Низький	Середній	Високий	Високий
Виявлення сервісів	Семантична відповідність на основі правил (SWRL)	Відповідність за ключовими словами	Відповідність за статичним інтерфейсом	Відповідність на основі онтології	Пошук за допомогою штучного інтелекту
Композиція сервісів	Автоматизований, з урахуванням QoS, на основі правил	Статичний, заздалегідь визначений	Оркестрування через BPEL	Семантичний опис в OWL-S	На основі ШІ, але з обмеженою гнучкістю
Адаптивність	Висока (динамічне виведення)	Низька (ручне оновлення)	Обмежена (потребує ручного відновлення)	Помірна (оновлення онтології)	Висока (моделі ШІ, що самонавчаються)
Поінформованість про QoS	Інтегровано в правила та онтологію	Ні	Часто є зовнішнім по відношенню до системи	Обмежена підтримка	Залежить від платформи
Масштабованість	Від середнього до високого	Високий, але неточний	Високий, але складний	Середній (накладні витрати на обробку онтології)	Високий (залежить від ефективності моделі ШІ)
Рівень автоматизації	Високий (на основі міркувань)	Низький	Середній (на основі робочих процесів)	Середній	Високий

На основі цього порівняння, ключові переваги запропонованого методу підсумовуємо нижче:

- висока семантична точність – підхід, заснований на онтології, забезпечує структуроване і контекстно-орієнтоване представлення сервісів, що дозволяє більш точне виявлення і композицію в порівнянні з системами, заснованими на ключових словах або правилах.

- динамічна адаптивність – фреймворк у поєднанні з правилами SWRL дозволяє в режимі реального часу адаптуватися до мінливих вимог, динамічно виводячи нові композиції сервісів.

- усвідомлення QoS – інтегруючи обмеження QoS (наприклад, час виконання, точність) в семантичну модель, система оптимізує вибір сервісів для кращої продуктивності та надійності.

- автоматизоване виявлення та компонування сервісів – логічний висновок на основі SWRL зменшує ручні зусилля, забезпечуючи ефективну та інтелектуальну оркестровку сервісів.

- розширюваність і повторне використання - модульна структура онтології дозволяє безперешкодно інтегрувати нові сервіси і робочі процеси з мінімальною кастомізацією.

- формальне представлення знань – використання онтології забезпечує структуровані міркування та обмін знаннями, покращуючи інтероперабельність.

- оптимізовано для складних робочих процесів – фреймворк ефективно працює з мультисервісними робочими процесами, де традиційні підходи не справляються через свою статичність.

У той же час, фреймворк має певні обмеження:

- накладні витрати на розробку онтології – вимагає знань предметної області та ретельного проектування, що робить створення онтології трудомістким процесом.

- складність правил SWRL – управління великим набором правил SWRL стає складним завданням, оскільки кількість сервісів і взаємозв'язків зростає.

- обчислювальні витрати – міркування над великими онтологіями зі складними правилами можуть вимагати високої продуктивності, що впливає на масштабованість.

- проблеми інтеграції даних – робота з різнорідними джерелами даних з різними форматами і семантикою залишається проблемою для інтероперабельності.

- постійне обслуговування – оновлення онтологій і правил SWRL вимагає постійних зусиль, що збільшує довгострокові витрати.

- відсутність стандартизації – відсутність широко прийнятих стандартів для когнітивних веб-сервісів може перешкоджати міжсистемній сумісності.

- ризик надмірного інжинірингу – для простих додатків складність семантичної структури може переважати її переваги [26].

Узагальнюючи викладене вище, відзначимо, що запропонований метод має найкращий середній час виконання завдяки динамічному вибору сервісів із врахуванням QoS-параметрів, ефективному семантичному плануванню, а також меншій кількості доступних сервісів, що зменшує затримки на пошук і композицію. Також зважаємо на те, що запропонований метод демонструє високу точність та повноту завдяки ефективному семантичному зіставленню та оптимізації вибору сервісів, забезпечуючи конкурентоспроможність із сучасними підходами; досягає найвищих показників завдяки інтеграції глибокого навчання, яке дозволяє краще адаптуватися до контексту запитів; має гірші показники, оскільки його підхід базується на статичному гібридному зіставленні, яке не враховує складні семантичні зв'язки та контекстуальні особливості запитів, що обмежує його здатність знаходити всі релевантні сервіси.

Запропонований метод має показник ступіня відповідності, що є конкурентним у порівнянні з завдяки врахуванню QoS-параметрів, однак результат може бути знижений у сценаріях із високою динамічністю та складними обмеженнями запитів.

Висновки до розділу 4

Методологія оцінки ефективності семантичної структури зосереджена на показниках, які оцінюють продуктивність пошуку, релевантність сервісу та відповідність вимогам якості обслуговування (QoS). Ключові критерії оцінки включають час виконання, точність пошуку, повноту пошуку та ступінь відповідності QoS, який розраховується за допомогою функції відповідності, яка балансує час виконання та вартість. Порівняльний аналіз підкреслює переваги запропонованого методу автоматизації інтеграції та оркестровки когнітивних веб-сервісів, демонструючи покращену семантичну точність, адаптивність до змін та кращу інтеграцію QoS порівняно з традиційними підходами. Експериментальне дослідження демонструє різні когнітивні сервіси та їх застосування у військових умовах, підкреслюючи здатність системи ефективно обробляти складні запити. Загалом, запропонована структура перевершує існуючі методи з точки зору семантичної точності, адаптивності та автоматизованої композиції сервісу, а також вирішує такі проблеми, як розробка онтології та інтеграція даних.

ВИСНОВКИ

WSMO є одним з повноцінних доробок у сфері семантичних описів веб-сервісів, можна дійти висновку що інші фреймворки так само не мають здатності адаптації свого опису до реактивних систем або потоків даних.

Класифікація семантичних структур та моделей допомагає зрозуміти, які інструменти обрати для конкретного завдання, забезпечуючи гнучкість і ефективність роботи з даними.

Веб-сервіси та когнітивні веб-сервіси – це два типи сервісів, що надаються через Інтернет, але вони відрізняються за своїми можливостями та функціями. Ключові характеристики, які використовують когнітивні веб-сервіси, включають здатність до самонавчання, адаптивність до змін умов та інтелектуальну взаємодію з користувачами.

Когнітивні веб-сервіси використовують штучний інтелект для моделювання процесів людського мислення, дозволяючи їм розуміти, вивчати та передбачати потреби користувачів для персоналізованого досвіду. На відміну від традиційних веб-сервісів, які дотримуються статичних правил, когнітивні веб-сервіси адаптуються та розвиваються з часом, використовуючи машинне навчання та обробку природної мови для покращення взаємодії та задоволення користувачів. Ці сервіси значно підвищують ефективність роботи, автоматизуючи рутинні завдання та надаючи динамічні, залежні від контексту відповіді, остаточно змінюючи бізнес-практику та стимулюючи інновації в різних галузях.

Когнітивні веб-сервіси представляють передову еволюцію традиційних веб-сервісів, які включають штучний інтелект для забезпечення більш розумної, адаптивної та персоналізованої взаємодії з користувачем. На відміну від звичайних веб-сервісів, які зосереджуються на обміні даними на основі попередньо визначених правил, когнітивні веб-сервіси використовують такі технології, як обробка природної мови та машинне навчання, щоб аналізувати неструктуровані дані, розуміти контекст і автоматизувати складні завдання. Когнітивні веб-сервіси пропонують значні переваги, включаючи підвищену точність, персоналізацію та

здатність обробляти великі обсяги даних, а також стикаються з проблемами, такими як складність, пов'язана з розробкою та підтримкою когнітивних сервісів, проблема конфіденційності та безпеки даних, ризик упередженості в процесах прийняття рішень. Проте навіть попри це мають значні перспективи у майбутньому.

Технології розвитку семантичного Веб включають XML (eXtensible Markup Language), RDF (Resource Description Framework), SPARQL (протокол SPARQL і мова запитів RDF), OWL (мова веб-онтологій). Формат RDF найбільш корисний у забезпеченні спільного використання інформації, зміст якої може однаково інтерпретуватися різними програмними агентами. Мова веб-онтологій (OWL) є ще одним критичним компонентом семантичної мережі, що забезпечує мову схем або мову представлення знань, яка покращує семантичне багатство веб-даних. OWL має три діалекти (у порядку зростання виразності): Lite, DL та Full. Розуміння можливостей OWL та характеристик його діалектів дозволяє ефективно вибирати відповідний підхід для моделювання когнітивних сервісів, адаптуючи його під конкретні вимоги і завдання. Майбутні перспективи та нові тенденції в розвитку інструментів міркування на основі OWL є багатообіцяючими та відображають зростаючий попит на більш складні та ефективні можливості міркування. SPARQL, що розшифровується як SPARQL Protocol and RDF Query Language, є спеціалізованою мовою запитів, розробленою для доступу до даних RDF та обробки даних. Синтаксис і структура SPARQL є ключовими для його функціональності та ефективності запитів до даних RDF. SQWRL, або мова веб-правил, розширена семантичними запитами, відіграє ключову роль у надсиланні запитів до онтологій у рамках семантичної мережі. Вибір між SPARQL і SQWRL для семантичних веб-проектів часто залежить від конкретних вимог і існуючої інфраструктури проекту. SPARQL зазвичай віддають перевагу для проектів, що включають великі пов'язані дані в різноманітних наборах даних через його широке визнання та надійну продуктивність у обробці даних RDF. Навпаки, SQWRL найкраще підходить для проектів, орієнтованих на онтологію, де міркування над складними ієрархіями класів і зв'язками є вирішальними. Таким чином, вибір між

SPARQL і SQWRL повинен ґрунтуватися на характері даних, складності необхідних запитів і загальних цілях проекту.

Аналіз сучасних методів оркестрації мікросервісів демонструє, що такі інструменти, як Kubernetes, Docker Swarm і Apache Mesos, значно підвищують гнучкість, масштабованість і надійність складних систем, автоматизуючи процеси розгортання, масштабування та управління контейнеризованими застосунками. Оркестрація є ключовим елементом сучасної IT-інфраструктури, забезпечуючи стабільну та ефективну роботу мікросервісних архітектур.

Методи хореографії веб-сервісів відіграють ключову роль у забезпеченні ефективної та скоординованої роботи розподілених систем. Завдяки використанню формальних моделей, таких як UML, BPMN, WS-CDL і BPEL, а також застосуванню різних підходів – від централізованого й децентралізованого до контрактно-орієнтованого та подійного – компанії можуть оптимізувати свої бізнес-процеси, підвищити сумісність між сервісами та мінімізувати ймовірність помилок. Практичне впровадження хореографії веб-сервісів дозволяє організаціям різних галузей, таких як електронна комерція, охорона здоров'я та логістика, досягати нових рівнів операційної ефективності та задоволеності клієнтів. Завдяки гнучкості, масштабованості та автоматизації бізнес-процесів, ці методи стають важливим інструментом у сучасному цифровому середовищі, забезпечуючи інтеграцію технологій із бізнес-цілями.

Семантика відіграє ключову роль у трансформації когнітивних веб-сервісів, надаючи їм здатність розуміти та обробляти дані на рівні, наближеному до людського мислення. Завдяки онтологіям, машинному навчанню та NLP ці сервіси здатні забезпечувати більш точний пошук, релевантність інформації та інтуїтивно зрозумілу взаємодію з користувачами. Однак існують виклики, пов'язані з масштабованістю, сумісністю та складністю обробки природної мови, які потребують подальшого вдосконалення технологій. Перспективи розвитку семантичних веб-сервісів обіцяють значний вплив на різні галузі, відкриваючи нові можливості для створення більш розумних, адаптивних та ефективних цифрових систем.

Веб-сервіси забезпечують критично важливу основу для сучасних програмних систем, забезпечуючи інтеграцію та взаємодію між різними платформами та технологіями. Оркестровка та хореографія веб-сервісів представляють два важливі підходи до управління цими сервісами, кожен із яких має свої сильні та слабкі сторони. Тоді як семантика є важливим фактором, що підвищує сумісність та ефективність обох підходів, пропонуючи спільне розуміння та можливість автоматизації. Хоча обидва підходи стикаються з різними викликами, зокрема з управлінням складністю та забезпеченням надійності, їх правильна реалізація може призвести до значних переваг для організацій, що прагнуть оптимізувати свої бізнес-процеси.

Завдяки перевагам: автоматизації, контекстній обізнаності та динамічності, агентний підхід із LangChain дозволяє створювати інтелектуальні, адаптивні й ефективні системи, які відповідають сучасним вимогам інтеграції сервісів. Такий підхід не лише полегшує процес розробки, але й забезпечує кращий користувацький досвід, знижуючи бар'єри для впровадження нових технологій. Вважаємо, що існуючі виклики продуктивності, точності та безпеки даних є важливими аспектами, які необхідно враховувати при використанні LangChain і LLM для композиції сервісів. Хоча ці обмеження можуть ускладнювати впровадження, їх можна подолати за допомогою правильного проектування системи, регулярного тестування і застосування сучасних підходів до забезпечення безпеки задля створення надійних та ефективних рішень, що максимально використовують потенціал агентного підходу. Удосконалення агентного підходу в LangChain через інтеграцію спеціалізованих моделей, оптимізацію продуктивності та розширення функціоналу відкриває нові перспективи для його застосування. LangChain у поєднанні з LLM є перспективним підходом для створення агентів, які здатні ефективно підтримувати композицію сервісів із семантичним підходом та відкривають нові можливості для автоматизації складних процесів у багатьох галузях, таких як логістика, фінанси, освітні платформи тощо.

Оркестрування є потужним інструментом для централізованого управління виконанням задач у когнітивних веб-сервісах. Воно забезпечує чітку координацію

між сервісами, ефективне використання ресурсів і адаптивність до змінних умов чи вимог. Хореографія забезпечує високу гнучкість і адаптивність системи, дозволяючи сервісам взаємодіяти без централізованого контролю, що робить систему більш масштабованою, оскільки нові сервіси можуть легко інтегруватися, просто дотримуючись правил взаємодії. Такий підхід ідеально підходить для динамічних середовищ, де вимоги та доступні сервіси постійно змінюються.

Онтологія не лише структуровано описує елементи когнітивного веб-сервісу, але й забезпечує основи для його адаптації, ефективності та персоналізації під індивідуальні потреби користувачів. Основним елементом онтології є клас користувачів (User), що взаємодіють із системою. Користувачі описуються через такі властивості: роль у системі (hasRole), мова взаємодії (hasLanguage), ідентифікатор (hasUuid). Клас WebService описує веб-сервіси, які виконують інтелектуальні задачі. У нього входять підкласи, наприклад, CognitiveWebService, що включають спеціалізовані сервіси, такі як TextToSpeech (перетворення тексту в мову) та ComputerVision (аналіз зображень). Властивості сервісів включають вхідні та вихідні типи даних (hasInput, hasOutput) і параметри якості обслуговування (hasQoS). Клас DataType описує типи даних, які обробляють сервіси: текст, зображення, аудіо, числа. Властивості даних включають формат (hasFormat) та значення (hasValue), що забезпечує гнучкість у роботі з різноманітними даними. Клас TaskType описує типи задач, які підтримує система: машинний переклад (MachineTranslation), розпізнавання (Recognition), перетворення даних (Transformation), прогнозування (Prediction). Клас ClassificationResult описує вихідні дані, отримані після виконання задач. Підкласи, такі як LocationEntities (географічні об'єкти) чи PersonNameEntities (імена людей), деталізують специфіку результатів. Клас Constraint дозволяє визначити обмеження, пов'язані із сервісами чи задачами: час виконання (TimeConstraint), точність (AccuracyConstraint), підтримувані мови (LanguageConstraint).

Концептуальна архітектура веб-сервісу для обробки запитів розроблена для сприяння ефективній взаємодії між користувачем і сервісом через його основні компоненти: інтерфейс користувача, серверну частину, семантичну онтологію та

механізм обробки запитів. Інтерфейс користувача надає перевагу зручності використання, дозволяючи користувачам легко надсилати запити та переглядати результати, тоді як сервер обробляє ці запити за допомогою семантичного аналізу та логічного висновку, підвищуючи точність та ефективність. Семантична онтологія лежить в основі розуміння системою запитів користувачів, дозволяючи їй адаптуватися до мінливих вимог і розширюватися за допомогою нових веб-сервісів, що в кінцевому підсумку забезпечує високу продуктивність обробки запитів.

Система когнітивного веб-сервісу використовує структуровану алгоритмічну модель обробки запитів користувачів. Спочатку запити користувача природною мовою перетворюються в структурований формат JSON, що дозволяє визначити точні параметри завдання. Потім система аналізує запит із базою знань, визначає оптимальну послідовність виконання веб-сервісом завдань за допомогою правил SWRL і, зрештою, представляє кінцевий результат користувачеві в різних форматах, гарантуючи, що вихідні дані відповідають початковим вимогам користувача.

Методологія оцінки ефективності семантичної структури зосереджена на показниках, які оцінюють продуктивність пошуку, релевантність сервісу та відповідність вимогам якості обслуговування (QoS). Ключові критерії оцінки включають час виконання, точність пошуку, повноту пошуку та ступінь відповідності QoS, який розраховується за допомогою функції відповідності, яка балансує час виконання та вартість. Порівняльний аналіз підкреслює переваги запропонованого методу автоматизації інтеграції та оркестровки когнітивних веб-сервісів, демонструючи покращену семантичну точність, адаптивність до змін та кращу інтеграцію QoS порівняно з традиційними підходами. Експериментальне дослідження демонструє різні когнітивні сервіси та їх застосування у військових умовах, підкреслюючи здатність системи ефективно обробляти складні запити. Загалом, запропонована структура перевершує існуючі методи з точки зору семантичної точності, адаптивності та автоматизованої композиції сервісу, а також вирішує такі проблеми, як розробка онтології та інтеграція даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Антоненко В. М., Мамченко С. Д., Рогушина Ю. В. Сучасні інформаційні системи і технології: управління знаннями : навчальний посібник. Ірпінь : Національний університет ДПС України, 2016. 212 с.
2. Архітектури та технології WEB-сервісів. URL: <https://eir.zp.edu.ua/server/api/core/bitstreams/5589382e-e03c-4ad7-a17f-e783c5889f89/content>
3. Барташевська Ю. Семантичні технології та семантичний веб. URL: https://duan.edu.ua/images/staff/departments/IT/Files/Conferences/IT_conf_15_May_2020.pdf#page=48
4. Веб-інтелект Web 3.0 (Semantic Web) – реальність сьогодення чи перспектива? URL: https://learn.ztu.edu.ua/pluginfile.php/274229/mod_resource/content/1/%D0%A8%D0%86_%D0%9A%D0%86%D0%9C_%D0%9B-16_%D0%92%D0%B5%D0%B1-%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82.pdf
5. Вовнянка Р.В. Методи та засоби планування дій спеціалізованих інтелектуальних агентів на основі онтологічного підходу. Дисертація на здобуття вченого ступеня кандидата технічних наук. 2017. 230 с. URL: <https://lpnu.ua/sites/default/files/2020/dissertation/1431/dysvovnianskarv.pdf>
6. Вступ до зв'язаних даних (Linked Data). URL: <https://diia.data.gov.ua/info-center/linkedata>
7. Гета А. В., Заїка В. М., Коваленко В. В. та ін. Сучасні засоби ІКТ підтримки інклюзивного навчання : навчальний посібник / за заг. ред. Ю. Г. Носенко. Полтава : ПУЕТ, 2018. 261 с.
8. Гончар Л., Бондарець А. Основні принципи побудови семантичної моделі. URL: https://elartu.tntu.edu.ua/bitstream/123456789/10544/2/ConfTNTU_2011_Honchar_L-Osnovni_pryntsypy_pobudovy_semantychnoi_66.pdf

9. Гриценко В.І., Гладун А.Я., Рогушина Ю.В. Моделі та методи використання семантичних wiki-ресурсів як джерела знань для поповнення формальних онтологій предметних областей. 2018. №2 (192). С. 23-43.

10. Давидовський М. В. Модель та метод міграції екземплярів онтологій у децентралізованих системах. Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня кандидата фізико-математичних наук за спеціальністю 01.05.02 – «Математичне моделювання та обчислювальні методи» (01 – фізико-математичні науки). Запорізький національний університет Міністерства освіти і науки України, Запоріжжя. Запорізький національний університет Міністерства освіти і науки України, Запоріжжя, 2021. 232 с.

11. Дерещкий В.А. Підходи до композиції сервісів в семантичному web середовищі. URL: http://dspace.nbuu.gov.ua/bitstream/handle/123456789/290/%D0%94%D0%B5%D1%80%D0%B5%D1%86%D0%BA%D0%B8%D0%B9_%231.pdf?sequence=1

12. Дзямудич М., Шматковська Т. Вплив сучасних інформаційних систем і технологій на формування цифрової економіки. *Економічний форум*. 2/2022. С. 3-8.

13. Добровольський Г.А., Кеберле Н. Г. Організація баз даних та баз знань: бази знань, засновані на онтологіях. Запоріжжя : ЗНУ, 2023. 126 с.

14. Документно-інформаційні комунікації в умовах глобалізації: стан, проблеми та перспективи : *матеріали IX Міжнародної наук.-практ. конф., м. Полтава, 21 листопада 2024 р./* редкол. І. Г. Передерій, О. Є. Гомотюк та ін. Полтава, 2024. 348 с.

15. Домарацький А.С., Терновой М.Ю. Підхід до побудови зв'язних SPARQL запитів. URL: <https://ela.kpi.ua/server/api/core/bitstreams/a0ab24b3-eda3-496e-bda3-fe443b185506/content>

16. Жаріков Е.В. Інфраструктура інформаційних систем: курс лекцій [Електронний ресурс]: навч. посіб. для студ. освітньої програми «Інженерія програмного забезпечення інформаційних систем» спеціальності 121 «Інженерія програмного забезпечення». Київ : КПІ ім. Ігоря Сікорського, 2022. 151 с.

17. Захарова О.В. Специфікація процесу семантичної анотації веб-сервісів.
URL: <https://ceur-ws.org/Vol-2139/204-213.pdf>
18. Збірник статей здобувачів вищої освіти другого (магістерського) рівня Навчально-наукового інституту інформаційних технологій Університету ДФС України. Ірпінь, 2019. 228 с.
19. Звенігородський О.С., Зінченко О.В., Чичкарьов Є.А., Кисіль Т.М.. Штучний інтелект. Вступний курс. 2022. 193 с.
20. Іванов О. В. Класичний контент-аналіз та аналіз тексту: термінологічні та методологічні відмінності. *Вісник Харківського національного університету імені В. Н. Каразіна*, Харків: Видавничий центр ХНУ імені В. Н. Каразіна, 2013. № 1045. С.72.
21. Інноваційна педагогіка. *Науковий журнал*. Випуск 70. Том 1.
http://www.innovpedagogy.od.ua/archives/2024/70/part_1/70-1_2024.pdf
22. Інноваційні наукові дослідження: теорія, методологія, практика : *Матеріали VIII Міжнародної науково-практичної конференції (м. Київ, 28–29 лютого 2024 р.)* / ГО «Інститут інноваційної освіти»; Науково-навчальний центр прикладної інформатики НАН України. 2-е вид., випр. і доп. Запоріжжя : АА Тандем, 2024. 260 с.
23. Інтелектуальні системи автоматизації : монографія / Аврунін О. Г., Владов С. І., Петченко М. В., Семенець В. В., Татарінов В. В., Тельнова Г. В., Філатов В. О., Шмельов Ю. М., Шушляпіна Н. О. Кременчук : Видавництво «НОВАБУК», 2021. 322 с.
24. Каргаманова Ю. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. URL: <https://dou.ua/forums/topic/40575/>
25. Касьянчук І. В. Огляд хореографії веб-сервісів WSMO для виконання синхронних та асинхронних запитів. *Таврійський науковий вісник. Серія: Технічні науки*. 2024. № (2). С. 46-52. URL: <https://journals.ksauniv.ks.ua/index.php/tech/article/view/574/536>

26. Касьянчук І. В., Петренко А. І. Розробка семантичної структури для композиції когнітивних веб-сервісів. *Technology Audit and Production Reserves*. URL: <https://tarp.net.ua/uk/>
27. Касьянчук І. В., Петренко А. І. Розробка семантичної структури для композиції когнітивних веб-сервісів. URL:
28. Коваленко О.Є. Моделі і методи побудови конвергентних систем ситуаційного управління. Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.05 – комп'ютерні системи та компоненти. Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, Київ, 2021. 364 с.
29. Колесник Л. В., Бакланов О. М. Аналіз способів застосування когнітивних сервісів для seo-оптимізації сайту. *Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління*. С. 52. URL: <https://openarchive.nure.ua/server/api/core/bitstreams/07d2dda4-7265-4654-89ba-83b5eb21f245/content>
30. Комп'ютерні онтології та їх використання у навчальному процесі. Теорія і практика. : Монографія / С. О. Довгий, В. Ю. Велічко, Л. С. Глоба, О. Є. Стрижак., Т. І. Андрущенко, С. А. Гальченко, А. В. Гончар, К. Д. Гуляєв, В. М. Кудляк, К. В. Ляшук, О. В. Палагін, М. Г. Петренко, М. А. Попова, В. І. Сидоренко, О. О. Слюсаренко, Д. В. Стус, М. Ю. Терновой. Київ : Інститут обдарованої дитини, 2013. 310 с.
31. Кравчук І.А. Технологія semantic web в системах електронного навчання. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/29478/1/Kravchuk_SEMANTIC%20WEB.pdf;jsessionid=17D43EB94B4EFB6D0C227F4DD1BD7A0D
32. Ланде Д.В., Субач І.Ю., Бояринова Ю.Є. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки: навчальний посібник. Київ: ІСЗІ КПІ ім. Ігоря Сікорського», 2018. 300 с.
33. Лебідь О.Ю. Побудова когнітивної моделі для аналізу діяльності електронних магазинів. URL: <http://www.economy.nayka.com.ua/?op=1&z=4563>

34. Майстерність штучного інтелекту в маркетингу: вигоди та приклади використання. URL: <https://webpromoexperts.net/ua/blog/maysternist-shtuchnogo-intelektu-v-marketingu-vigodi-ta-prikladi-vikoristannya/>

35. Матеріали Міжнародної наукової конференції «Штучний інтелект: досягнення, виклики та ризики» м. Київ, Україна 15-16 березня 2024 р. URL: https://www.ipai.net.ua/docs/ai_conf_15.03.2024.pdf

36. Мельник І.Г. Використання програм зі штучним інтелектом у сегменті загальної середньої освіти: потенціал і виклики. *Освітня аналітика України*. 2024. № 2 (28). С. 31-44.

37. Міхалевський В.Ц., Міхалевська Г.І. Розвиток інформаційного середовища для формування інтелектуальної надбудови інформаційного суспільства. *Вісник Хмельницького національного університету*. №1. 2020 (281). С. 135-142.

38. Нельсон Д. Що таке NLP (обробка природної мови)? URL: <https://www.unite.ai/uk/what-is-natural-language-processing/>

39. Новицький О.В. Основні підходи до композиції веб-сервісів електронної бібліотеки. *Інформаційні системи*. URL: http://dspace.nbuiv.gov.ua/bitstream/handle/123456789/1473/%E2%84%962-3_2008_Novytskiy.pdf?sequence=1

40. Освіта для XXI століття: виклики, проблеми, перспективи: матеріали III Міжнародної науково-практичної конференції (16–17 листопада 2021 року, м. Суми). Том 2. Суми: Вид-во СумДПУ імені А. С. Макаренка, 2021. 168 с.

41. Петренко А.І. Проектування сучасних систем сервісів на прикладі мобільної медичної системи для мешканців прифронтових селищ в зоні АТО. URL: https://report.kpi.ua/files/2019_2022.PDF

42. Петренко І.А., Петренко О.О., Автоматизовані методи пошуку і відкриття необхідних сервісів. URL: https://cad.kpi.ua/attachments/043_2017_11_p.pdf

43. Петренко О.О. Стратегії розвитку сервіс-орієнтованих систем у хмарному середовищі. Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук (доктора філософії) за спеціальністю 05.13.12 «Системи автоматизації проектних робіт». Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2018. 239 с.

44. Писарчук О.О., Грінченко О.О., Мельник О.В., Чередник В.М., Байрамова О.В. Когнітивні науки в інженерії програмного забезпечення. С. 202-210. URL:

https://www.researchgate.net/publication/329720437_KOGNITIVNI_NAUKI_V_INZENIERII_PROGRAMNOGO_ZABEZPECENNA

45. Плєсканич В.Л. Інформаційні системи і технології на підприємствах: підручник. Київ : Знання, 2011. 718 с.

46. Плєскач В.Л. , Рогушина Ю.В. Агентні технології: Монографія. Київ: Київ. нац. торг.-екон. ун-т, 2005. 344 с.

47. Попов Д. Короткий посібник з цифрової доступності. URL: https://www.undp.org/sites/g/files/zskgke326/files/2023-06/korotkiy_posibnik_z_cifrovoi_dostupnosti_-_ukr.pdf

48. Пригода А.Я. Аналіз сучасних методів оркестрації мікросервісів. URL: <https://www.economy-confer.com.ua/full-article/5648/>

49. Приходько О.М., Скоробогатова А.О., Семенова Л.Ю. Роль штучного інтелекту у вивченні поведінки сучасних споживачів. *Економічна наука. Інвестиції: практика та досвід*. № 22. 2024. С. 149-156.

50. Про затвердження форматів електронних повідомлень та обміну даними системи електронної взаємодії державних електронних інформаційних ресурсів. URL: <https://zakon.rada.gov.ua/laws/show/z1087-18#Text>

51. Рижов О.А. Онтологічний підхід до викладання фармацевтичного менеджменту та маркетингу майбутнім провізорам. *Адаптивні технології управління навчанням: збірник матеріалів сьомої міжнародної конференції. Одеса-Київ, 28–30 вересня 2021 р.* Київ: ІТЗН НАПН України, 2021. 114 с. С. 37-40.

52. Рогушина Ю.В. Використання систем організації знань на основі онтологій у wiki-ресурсах. *Проблеми програмування*. 2022. № 1. С. 23-33.

53. Рогушина Ю.В., Гладун А.Я., Осадчий В.В., Прийма С.М. «Онтологічний аналіз у Web». 2015. 407 с.

54. Романишин Ю. Л. Теоретичні і методичні засади проєктування веб-базованого освітнього середовища університету: монографія. Івано-Франківськ: НАІР, 2022. 506 с.

55. Сілагін О., Сілагін Є., Денисюк В., Денисюк А. Розробка онтологічної моделі бази знань «Бібліотека» на базі середовища Protégé. *Інформаційні технології та комп'ютерна інженерія*. 2023. № 3.С. 12-21.

56. Спасітелева С.О., Чичкань І.В., Шевченко С.М., Жданова Ю.Д. Розробка безпечних контейнерних застосунків з мікросервісною архітектурою. *Кібербезпека: освіта, наука, техніка*. № 1 (21). 2023. URL: <https://csecurity.kubg.edu.ua/index.php/journal/article/download/506/398/1746>

57. Спирін О.М. Використання системи EPRINTS як засобу інформаційно-комунікаційної підтримки наукової діяльності в галузі педагогічних наук. Дисертація на здобуття наукового ступеня кандидата педагогічних наук. 317 с. URL:

https://lib.iitta.gov.ua/id/eprint/8634/1/%D0%B4%D0%B8%D1%81%D1%81%D0%B5%D1%800_.2015_%D0%A1%D0%B2%D0%B5%D1%82%D0%B0.pdf

58. Танцюра О. Б. Аналіз методів комплексування різноспектральних зображень з використанням універсального показника якості. *Наука і техніка Повітряних Сил Збройних Сил України*. 2016. № 4. С. 152-156.

59. Технології автоматизації системних процесів: Навч. посібник. Меркотан Д.Ю, Сова О.Я., Троцько О.О., Симоненко О.А., Гаман О.В., Степаненко О.Є., Мягих Г.Г., Величко В.П.. Київ: ВІТІ, 2021. 231 с.

60. Цифрова трансформація відкритих освітніх середовищ: колективна монографія / за ред. В.Ю. Бикова, О.П. Пінчук. Київ: ФОП Ямчинський О.В., 2019. 186 с.

61. Що таке Kubernetes, як він працює та де використовується: веб-сайт. URL: <https://blog.colobridge.net/uk/2023/12/kubernetes-what-is-it-ua/>

62. Bhuvaneswari K. Sumathi, Sarveshwaran V., Sivasangari A. Hybrid deep learning and similarity measures for semantic web service discovery. *Springer Journal of Knowledge and Information Systems*. 2024. <https://doi.org/10.1007/s10115-024-02244-x>
63. Charif Y. Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs. URL : <https://perso.limsi.fr/sabouret/ps/PhD/slides-yasmine.pdf>
64. Chithambaramani R., Jayashree K., Krishna B. V., Prakash Mohan, Tayfun Dede. An Efficient Ontology-Based Semantic Interoperability Using MSGO-RNN in Cloud Computing. *Journal of Optimization*. 2024. Vol 2024. P.1. URL: https://openurl.ebsco.com/EPDB%3Agcd%3A12%3A9305772/detailv2?sid=ebsco%3Aplink%3Ascholar&id=ebsco%3Agcd%3A179671547&crl=c&link_origin=scholar.google.com.ua
65. Diulher V., Sorokin A. Аналіз методів інтеграції та узгодження мікросервісів в хмарній архітектурі. *Системи управління, навігації та зв'язку: зб. наук. праць*. Полтава: ПНТУ, 2024. Т. 1 (75). С. 58–60.
66. Docker Orchestration using Swarm. URL: <https://www.linkedin.com/pulse/docker-orchestration-using-swarm-dinesh-balaji-govindan>
67. Docker Swarm – definition & overview: веб-сайт. URL: <https://www.sumologic.com/glossary/docker-swarm/>
68. Dragoni N. et al. Microservices: yesterday, today, and tomorrow. *Communications of the ACM*. 2017. №60 (6). Pp. 85–93.
69. Farid Humaira (2024) *A Tableau-based Algebraic Calculus for Description Logic SHOIQ*. PhD thesis, Concordia University. URL: <https://spectrum.library.concordia.ca/id/eprint/993923/>
70. Fdhila W. Optimized Decentralization and synchronization of Inter-Organizaion Business Processes. URL: https://www.researchgate.net/publication/281658851_Optimized_Decentralization_and_synchronization_of_Inter-Organizaion_Business_Processes
71. Freeman M. Semantic Data Models and Data Contracts: A True Dream Team. Gable. 2024. URL: <https://www.gable.ai/blog/semantic-data-models>

72. Hamilton Mark. Large-Scale Intelligent Microservices. 2020. URL: https://www.researchgate.net/figure/End-to-end-architecture-diagram-for-the-Cognitive-Services-for-Big-Data-Using-this_fig1_344294726
73. Huaxin Li; Zhaoxin Zhang; Yongdong Xu. Web Page Classification Method Based on Semantics and Structure. *2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. 2018 URL: https://www.researchgate.net/publication/335861620_Web_Page_Classification_Method_Based_on_Semantics_and_Structure
74. Ibrahim Yahmadi, Youcef Baghdadi and Zuhoor Al-Khanjari. Graphical description of WS-CDL. *International Conference on Innovations in Information Technology*. URL: https://www.researchgate.net/profile/Zuhoor-Al-Khanjari/publication/236308937_Graphical_description_of_WS-CDL_Towards_extending_SOA_vision_to_standardize_cross-organizational_business_process_modeling/links/584025f708ae61f75dced8ee/Graphical-description-of-WS-CDL-Towards-extending-SOA-vision-to-standardize-cross-organizational-business-process-modeling.pdf
75. Jiajun Xu, Lin Guo, Ruxia Zhang, Hualang Hu, Fei Wang & Zhiyuan Pei. QoS-aware Service Composition Using Fuzzy Set Theory and Genetic Algorithm. *Wireless Personal Communications*. 2018. Volume 102. P. 1009–1028. URL: <https://dl.acm.org/doi/abs/10.1007/s11277-017-5129-8>
76. Klusch M., Fries B., Sycara K. iSeM: Approximated reasoning for adaptive hybrid selection of semantic services. *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*. 2009. P. 30–45. URL: https://doi.org/10.1007/978-3-642-04930-9_3
77. Konrad Abicht. OWL Reasoners still useable in 2023. URL: <https://arxiv.org/pdf/2309.06888>
78. LangChain. URL: <https://www.langchain.com/>

79. Luo G., Tan W., Fan J., Zhou Q. Optimizing QoS-Aware Semantic Web Service Composition. *IEEE International Conference on Web Services (ICWS)*. 2011. P. 439–446. URL: https://link.springer.com/chapter/10.1007/978-3-642-04930-9_24
80. Mesos Architecture. URL: <https://mesos.apache.org/documentation/latest/architecture/>
81. Palagin Oleksandr, Petrenko Mykola, Kryvyi Sergii, Boyko Mykola, Malakhov Kyrylo. Ontology-driven processing of transdisciplinary domain knowledge: Monograph. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine. Taras Shevchenko National University of Kyiv. Published by Iowa State University Digital Press. 2023. 190 p.
82. Pay-as-you-go consequence-based reasoning for the description logic. David Tena Cucala, Bernardo Cuenca Grau, Ian Horrocks. *Artificial Intelligence*. Volume 298, September 2021. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0004370221000692>
83. Roman D. Web Service Modeling Ontology. URL: https://www.researchgate.net/publication/220438679_Web_Service_Modeling_Ontology
84. Sebastian Rudolph. Foundations of semantic web technologies. *Hypertableau II*. URL: <https://iccl.inf.tu-dresden.de/w/images/7/75/FSWT-L15-Hypertableau-II.pdf>
85. SPARQL 1.1 Query Language. URL: <https://www.w3.org/TR/sparql11-query/>
86. SPARQL. URL: <https://vue.gov.ua/SPARQL>
87. SQWRL a query language for OWL. URL: https://ceur-ws.org/Vol-529/owled2009_submission_42.pdf
88. Waterman S. Structural Methods for Lexical/Semantic Patterns. URL: <https://aclanthology.org/W93-0112.pdf>
89. Wenyin L. Object-process diagrams as explicit graphic tool for Web Service composition. 2004. Vol. 8. No. 1, P. 113-127.

90. What is Kubernetes? A Comprehensive Guide to Container Orchestration.
URL: <https://konghq.com/blog/learning-center/what-is-kubernetes>
91. Yeromina N., Shapa L., Budko A. Using of Global Positioning System navigation services in automated fare collection systems. *Третя міжнародна науково-технічна конференція «Комп'ютерні та інформаційні системи і технології»*. Збірник наукових праць. Харків: ХНУРЕ. 2019. С. 51-52.

ДОДАТКИ

Додаток А

Код реалізації оркестратора когнітивних веб-сервісів

```

class ServiceOrchestrator:
    """Оркестратор сервісів"""
    def __init__(self):
        self.ontology = CognitiveOntology()
        self.services = self._init_services()
        self.intent_analyzer = IntentAnalyzer()
        self.metrics = ServiceMetrics()

    def _init_services(self) -> Dict[str, WebService]:
        services = {
            "text_to_speech": TextToSpeechService(),
            "translation": TranslationService(),
            "speech_to_text": SpeechToTextService(),
            "image_analysis": ImageAnalysisService(),
            "location_service": LocationService() # Додали новий сервіс
        }
        for service in services.values():
            self.ontology.add_service(service)
        return services

    def process_request(self, query: str, file_data: Any = None, user: User = None) -> Dict:
        context = ExecutionContext(user)

        try:
            # Аналіз намірів
            analysis = self.intent_analyzer.analyze(query, has_file=file_data is not None)
            logger.info(f"Результат аналізу намірів: {analysis}")

            # Пошук ланцюжка сервісів з урахуванням аналізу намірів
            services = self.ontology.find_service_chain(
                analysis['input_type'],
                analysis['output_type'],
                analysis.get('services') # Передаємо список запитаних сервісів
            )

            if not services:
                raise ValueError("Не знайдено підходящих сервісів")

            # Валідація ланцюжка
            workflow = self.compose_workflow(services)

            # Виконання
            results = self.execute_workflow(workflow, file_data, context)

```

```

# Збереження метрик
for service_name in services:
    self.metrics.add_execution_time(
        service_name,
        context.metrics[service_name][-1]['duration']
    )

if user:
    user.add_request({
        "query": query,
        "services": services,
        "status": "success"
    })

return {
    "workflow": workflow,
    "results": results,
    "context": context.get_summary(),
    "metrics": {
        name: self.metrics.get_service_stats(name)
        for name in services
    },
    "ontology": self.ontology.g.serialize(format='turtle')
}

except Exception as e:
    logger.error(f"Помилка обробки запиту: {str(e)}")
    context.add_error("orchestrator", str(e))
    context.status = "error"
    raise

```


Код реалізації алгоритмічної моделі обробки запиту

```

# client.py
import streamlit as st
import requests
import json
import graphviz
from rdflib import Graph
from typing import Dict, List, Any
import base64

def render_ontology_graph(ttl_data: str) -> graphviz.Digraph:
    """Візуалізація семантичного графу (спрощена версія)"""
    g = Graph().parse(data=ttl_data, format='turtle')
    dot = graphviz.Digraph(comment='Семантичний граф сервісів')

    # Налаштування стилю графу
    dot.attr(rankdir='LR')
    dot.attr('node', shape='box', style='rounded')

    # Відображаємо тільки основні зв'язки між сервісами та типами даних
    important_predicates = [
        'hasInput',
        'hasOutput',
        'type'
    ]

    # Множини для відстеження вже доданих елементів
    added_nodes = set()
    added_edges = set()

    for s, p, o in g:
        # Отримуємо короткі імена
        pred = str(p).split('/')[-1]

        if pred in important_predicates:
            s_label = str(s).split('/')[-1]
            o_label = str(o).split('/')[-1]

            # Фільтруємо технічні класи
            if not any(tech in str(o) for tech in ['Class', 'Property']):
                # Додаємо вузли, якщо ще не додані
                if s_label not in added_nodes:
                    dot.node(s_label, s_label)
                    added_nodes.add(s_label)

                if o_label not in added_nodes and '#' not in str(o):
                    dot.node(o_label, o_label)
                    added_nodes.add(o_label)

```

```

        # Додаємо ребро, якщо ще не додане
        edge_key = (s_label, o_label, pred)
        if edge_key not in added_edges and '#' not in str(o):
            dot.edge(s_label, o_label, pred)
            added_edges.add(edge_key)

    return dot

def render_service_chain(workflow: Dict) -> graphviz.Digraph:
    """Візуалізація ланцюжка сервісів"""
    dot = graphviz.Digraph(comment='Ланцюжок сервісів')

    # Налаштування стилю графу
    dot.attr(rankdir='LR')

    # Додаємо сервіси
    for service in workflow['services']:
        # Створюємо мітку з інформацією про сервіс
        label = f'{service["name"]}\n{service["type"]}'
        if 'input_type' in service and 'output_type' in service:
            label += f'\n{service["input_type"]} → {service["output_type"]}'

        dot.node(service['id'], label)

    # Додаємо зв'язки між сервісами
    for conn in workflow['connections']:
        dot.edge(conn['from'], conn['to'])

    return dot

def encode_file_content(file) -> str:
    """Кодування вмісту файлу в base64"""
    if file is not None:
        content = file.read()
        if isinstance(content, bytes):
            return base64.b64encode(content).decode()
            return base64.b64encode(content.encode()).decode()
    return None

def main():
    st.title("🧠 Система когнітивних веб-сервісів")

    # Бічна панель з прикладами запитів
    with st.sidebar:
        st.header("Приклади запитів")
        st.markdown("""
        - Переклади текст і озвуч його
        - Проаналізуй це зображення
        - Знайди відстань між Києвом та Львовом
        - Переведи аудіо в текст
        """)

```

```

# Інформація про доступні сервіси
st.header("Доступні сервіси")
try:
    response = requests.get("http://localhost:8083/services")
    services = response.json()
    for name, info in services.items():
        st.markdown(f"""
        **{name}**
        - Вхід: {info['input_type']}
        - Вихід: {info['output_type']}
        - Тип: {info['task_type']}
        """)
except:
    st.error("Не вдалося отримати список сервісів")

# Форма запиту
with st.form("request_form"):
    query = st.text_area("🗨️ Ваш запит:",
                        placeholder="Опишіть, що потрібно зробити",
                        help="Опишіть своїми словами, яку задачу потрібно виконати")

    file = st.file_uploader("📎 Файл (якщо потрібно):",
                            help="Завантажте файл для обробки (аудіо, зображення, текст)")

    submitted = st.form_submit_button("🚀 Виконати")

if submitted:
    with st.spinner("⌚ Аналіз запиту..."):
        # Готуємо дані для відправки
        data = {
            "query": query,
            "file": encode_file_content(file) if file else None
        }

    try:
        # Відправляємо запит на сервер
        response = requests.post(
            "http://localhost:8083/process",
            json=data
        )
        result = response.json()

        if "error" in result:
            st.error(f"❌ Помилка: {result['error']}")
        else:
            # Показуємо знайдений ланцюжок сервісів
            st.subheader("🔍 Знайдений ланцюжок сервісів")
            st.graphviz_chart(render_service_chain(result['workflow']))

            # Показуємо семантичний граф

```

```

st.subheader("🏠 Семантичний аналіз")
st.graphviz_chart(render_ontology_graph(result['ontology']))

# Показуємо метрики виконання
st.subheader("🇺🇦 Метрики виконання")
for service, metrics in result.get('metrics', {}).items():
    st.metric(
        label=f"Сервіс {service}",
        value=f"{metrics.get('avg_duration', 0):.2f} сек",
        delta=f"Успішність: {(1 - metrics.get('error_rate', 0)) * 100:.1f}%"
    )

# Результати виконання
st.subheader("✅ Результати")
st.json(result['results'])

except requests.exceptions.RequestException as e:
    st.error(f"❌ Помилка з'єднання з сервером: {str(e)}")
except Exception as e:
    st.error(f"❌ Несподівана помилка: {str(e)}")

if __name__ == "__main__":
    main()

```