УДК 004.624
DOI https://doi.org/10.36994/2788-5518-2024-02-08-08

ФОРМАЛІЗАЦІЯ ОЗНАК СЕРВІСІВ У СЕРВІСНО-ОРІЄНТОВАНІЙ АРХІТЕКТУРІ

Рогоза В. С., *д.т.н., проф., Західно-Поморський Технологічний Університет, м. Щецин, Польща; Навчально-науковий комплекс «Інститут Прикладного Системного Аналізу» – ННК «ІПСА», Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна. https://orcid.org/0000-0003-2327-156X, wrogoza@wi.zut.edu.pl*

Зарічний Я. С., *Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна. https://orcid.org/0000-0002-7596-5857, yarik120700@gmail.com*

Анотація. *Сервісно-орієнтована архітектура (SOA) є фундаментальною концепцією у сучасних розподілених інформаційних системах, що забезпечує масштабованість, гнучкість та ефективність інтеграції програмних компонентів. Одним із ключових аспектів SOA є чітка формалізація ознак сервісів, що включає їхні характеристики, можливості та обмеження, необхідні для забезпечення сумісності, повторного використання та автономності в розподілених системах.*

У статті розглядаються наявні підходи до формалізації ознак сервісів, зокрема Universal Description, Discovery and Integration (UDDI), а також їхні недоліки, такі як обмеження застарілих стандартів, наприклад SOAP і XML. Представлено новий підхід до формального опису сервісів, який базується на використанні метаданих, поділених на категорії: базові, функціональні, контекстуальні та метадані продуктивності (QoS). Запропоновані моделі метаданих забезпечують можливість ефективного пошуку, інтеграції та управління сервісами в SOA.

Окрему увагу приділено застосуванню моделей машинного навчання та рекомендаційних систем для автоматизації пошуку та вибору сервісів. У статті запропоновано створення профілів користувачів та вендорів, що дозволяють підвищити релевантність вибору сервісів шляхом аналізу історії використання та оцінок. Такий підхід сприяє покращенню процесу прийняття рішень щодо інтеграції сервісів та підвищенню ефективності роботи інформаційних систем.

У підсумку стаття пропонує комплексний підхід до формалізації ознак сервісів, що дозволяє підвищити рівень сумісності, ефективності та автоматизації управління сервісами у сервісно-орієнтованих архітектурах. Запропоновані методи сприяють покращенню інтеграції сервісів, їх повторного використання та адаптації до змінних вимог користувачів та бізнес-процесів.

Ключові слова: *сервісно-орієнтована архітектура, профілі користувачів, профілі сервісів, формалізація ознак, мови опису інтерфейсів.*

FORMALIZATION OF SERVICE CHARACTERISTICS IN SERVICE-ORIENTED ARCHITECTURE

Walery Rogoza, *dr hab. prof. West Pomeranian University of Technology, Szczecin, Poland; Educational and Scientific Complex "Institute of Applied Systems Analysis" – ESC "IASA", The National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. https://orcid.org/0000-0003-2327-156X, wrogoza@wi.zut.edu.pl*

Yaroslav Zarichnyi, *National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. https://orcid.org/0000-0002-7596-5857, yarik120700@gmail.com*

Abstract. *Service-oriented architecture (SOA) is a fundamental concept in modern distributed information systems that provides scalability, flexibility, and efficiency in the integration of software components. One of the key aspects of SOA is the clear formalization of service features, including their characteristics, capabilities, and constraints necessary to ensure interoperability, reuse, and autonomy in distributed systems.*

The article discusses existing approaches to formalizing service features, in particular Universal Description, Discovery, and Integration (UDDI), as well as their shortcomings, such as the limitations of outdated standards, such as SOAP and XML. A new approach to formalizing service descriptions is presented, which is based on the use of metadata divided into categories: basic, functional, contextual, and performance (QoS) metadata. The proposed metadata models provide the ability to effectively search, integrate, and manage services in SOA.

Special attention is paid to the use of machine learning models and recommender systems to automate service search and selection. The article proposes the creation of user and vendor profiles that allow increasing the relevance of service selection by analyzing usage history and ratings. This approach

helps to improve the decision-making process regarding service integration and increase the efficiency of information systems.

In conclusion, the article offers a comprehensive approach to formalizing service features, which allows increasing the level of compatibility, efficiency, and automation of service management in service-oriented architectures. The proposed methods help to improve service integration, their reuse, and adaptation to changing user requirements and business processes.

Key words: *service-oriented architecture, user profiles, service profiles, feature formalization, interface description languages.*

Вступ

Сервісно-орієнтована архітектура (SOA) є ключовою концепцією сучасних розподілених інформаційних систем, що забезпечує гнучкість, масштабованість та ефективність інтеграції програмних компонентів. У межах SOA кожен сервіс розглядається як самостійна функціональна одиниця, яка взаємодіє з іншими сервісами через стандартизовані інтерфейси. Проте для ефективного пошуку, вибору, композиції та управління сервісами необхідна їхня чітка формалізація, що включає визначення характеристик, можливостей та обмежень кожного сервісу. Формалізація ознак сервісу відіграє ключову роль у підвищенні сумісності сервісів. Через відсутність чітких підходів до опису характеристик та навичок сервісів веде до складнощів щодо їх повторного перевикористання, композиції тощо.

Вебсервіси забезпечують надання функціональних можливостей, які постачальники послуг пропонують різним споживачам у децентралізованому та розподіленому середовищі. З часом необхідність змін у вебслужбах спричиняє появу їхніх нових версій. Часто для задоволення потреб різних користувачів потрібне масштабне налаштування служб, що змушує постачальників розгортати декілька варіантів вебслужб, адаптованих для кожної групи користувачів [1].

Клієнтам не потрібно знати розташування серверів безпосередньо; їм досить мати доступ до служби пошуку, який може бути забезпечений за допомогою механізмів початкового завантаження, таких як початкові посилання або широкомовні повідомлення. Уся інша необхідна інформація про місцезнаходження отримується через службу пошуку.

Мета статті

У цій статті досліджуються та обґрунтовуються підходи до формалізації ознак сервісу у сервісно-орієнтованій архітектурі (SOA) з метою підвищення ефективності, масштабованості та гнучкості інформаційних систем. Приділено увагу наявним рішенням, таким як UDDI, оглянуто їх недоліки. У статті розглядаються ключові характеристики сервісів, їхня класифікація та принципи формального опису, що забезпечують сумісність, повторне використання та автономність сервісів у розподілених системах. Окрему увагу приділено практичним аспектам впровадження формалізованих ознак у сучасних SOA-платформах, включаючи використання метаданих та моделей машинного навчання для вдосконалення управління сервісами. Крім того, аналізуються перспективи використання семантичного пошуку для автоматизації процесів взаємодії між сервісами та користувачами, що може суттєво підвищити ефективність управління сервісно-орієнтованими системами.

Огляд схожих рішень

На сьогодні були деякі підходи, в яких були спроби сформувати загальні ознаки та характеристики сервісів. Серед таких підходів можна відзначити UDDI (Universal Description Discovery and Integration). Така система була спроєктована як реєстр вебсервісів, що дає можливості щодо опису та пошуку й інтеграції вебсервісів користувачами [2].

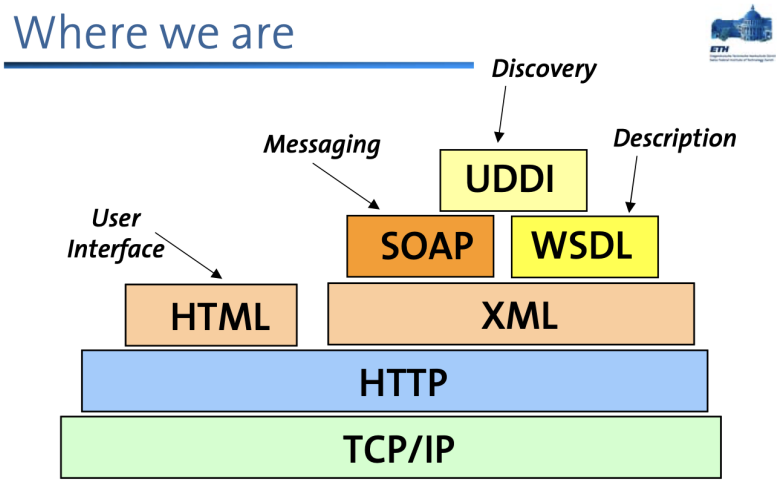

Рис. 1. Загальна структура UDDI [3]

UDDI базується на таких протоколах, як SOAP, XML, WSDL. SOAP використовується як основа для обміну повідомленнями між клієнтами та реєстром. Такий протокол забезпечує стандартний формат XML повідомлень для пошуку та реєстрації й отримання інформації про вебсервіси. Дані, які описують вебсервіс, представляються у форматі XML, також може використовуватися мова опису WSDL як формат опису інтерфейсів вебсервісів [3].

Така технологія доволі застаріла й не набула високого попиту серед користувачів. Серед недоліків можна відзначити підтримку застарілих протоколів комунікації в мережі – SOAP та XML. А також неможливість використання більш сучасних протоколів обміну та передачі даних.

Формалізація ознак сервісу

У Service-Oriented Architecture (SOA) будь-який сервіс може бути описаний за допомогою різних типів метаданих та типів даних. Ознаки сервісів можна поділити на деякі категорії: базові метадані, функціональні метадані, контекстуальні дані, дані про продуктивність і якість обслуговування тощо.

Метою базових метаданих є надання важливої інформації високого рівня про послугу, яка полегшує її ідентифікацію, виявлення та категоризацію. Він утворює фундаментальний рівень опису сервісу у сервісно-орієнтованій архітектурі (SOA).

Таблиця 1

Профіль сервісу

Елемент метаданих	**Ціль**	**Приклад**
Name	Унікально ідентифікує службу, дозволяючи посилатися на неї, отримати доступ і відрізняти її від інших служб	Платіжний шлюз
Description	Пояснення того, що робить служба	Обробляє платежі кредитними картками
Service ID/Endpoint	Знаходження та доступність до сервісу	https://api.example.com/v1/pay
Category/Tags	Дозволяє групувати службу з подібними послугами, полегшуючи організацію та навігацію каталогом послуг	Платежі, фінанси
Version	Відстежування та керування оновленнями або сумісністю	v1.2.3

Функціональні метадані описують робочі аспекти служби, такі як її входи, виходи, методи та протоколи. Його основна мета – надання докладної інформації про те, як взаємодіяти зі службою, що дозволяє розробникам і системам ефективно інтегрувати, споживати та використовувати службу в робочих процесах або програмах.

Таблиця 2

Профіль сервісу, який описує функціональні метадані

Елемент метаданих	Ціль	Приклад
Input Specifications	Визначає дані, необхідні для виклику служби, включаючи типи, формати та структури.	JSON object: {“user_id”: “string”}
Output Specifications	Описує дані, які повертає служба, включаючи типи, формати та структури.	JSON object: {“status”: “success”}
Methods/Operations	Перелічує конкретні функції або дії, які може виконувати служба.	createUser, getUserDetails
Service Protocol	Вказує протокол зв'язку, який використовується для доступу до служби.	HTTP, gRPC, SOAP
Supported Formats	Визначає формати даних, які підтримуються для введення/виведення.	JSON, XML
Authentication Requirements	Описує будь-які механізми автентифікації, необхідні для доступу до служби.	API Key, OAuth, JWT

Метадані продуктивності та якості обслуговування (QoS) надають показники та характеристики, які описують, наскільки добре сервіс працює за певних умов. Його основна мета полягає в тому, щоб користувачі та системи могли оцінити надійність, ефективність і масштабованість служби, допомагаючи встановити очікування та підтримувати прийняття обґрунтованих рішень під час інтеграції або вибору послуг.

Також, будуючи таку систему, корисно знати профілі вендорів-постачальних цих сервісів і користувачів. Профіль користувача має містити історію використання вебсервісів.

Таблиця 3
Профіль метаданих сервісу

QoS Element	Purpose	Example
Latency	Вимірює час, потрібний службі для відповіді на запит.	200 ms
Throughput	Вказує кількість запитів, які сервіс може обробити за секунду.	500 requests/second
Uptime/Availability	Показує відсоток часу, протягом якого послуга працює та доступна.	99.99% uptime
Error Rate	Надає відсоток невдалих або помилкових запитів за певний період.	0.02% error rate
Response Time Distribution	Надає детальні показники часу відгуку, включаючи середні значення та процентилі (наприклад, P90, P99).	P99: 300 ms

Таблиця 4
Профіль вендора сервісу

Used Services	Purpose	Example
Service Identifier	Ідентифікатор сервісу.	UUID-string
Category	Категорія, до якої належить сервіс.	«Фінанси»
Last used	Дата у вигляді UNIX timestamp, коли останній раз користувач звертався до сервісу або шукав його.	1738615470
Rating	Оцінка сервісу, яку поставив користувач.	4

Профілі користувачів можуть стати основою для побудови рекомендаційної моделі, яка допомагає автоматизувати пошук, вибір і використання сервісів у сервісно-орієнтованій архітектурі (SOA). Профілі користувачів також допомагають створювати рекомендаційні системи, які дозволяють автоматично підбирати сервіси на основі попереднього досвіду користувача та його подібності до інших споживачів. Це значно скорочує час на пошук потрібного сервісу, підвищує ефективність роботи та сприяє глибшій інтеграції сервісно-орієнтованих рішень у бізнес-процеси. Оцінку сервісу можна буде збирати таким чином, що після використання сервісу або його знаходженням питати у користувача, чи був сервіс корисний. Таким чином, можна буде закрити повний цикл рекомендації та пошукової видачі сервісу [4; 5].

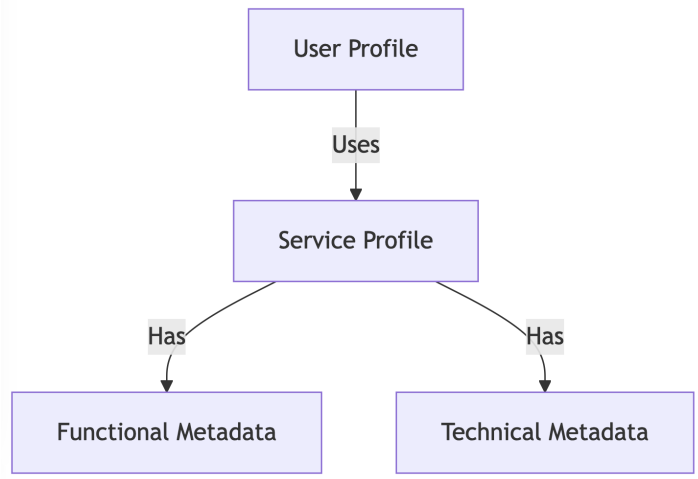

Рис. 2. Загальний вид реєстру

Також пропонується використання цієї системи як системи для виявлення нових та пошуку потрібних сервісів. Комунікація між сервісом і клієнтом може здійснюватися за допомогою будь-яких протоколів та форматів передачі й представлення даних. На відміну UDDI використовує SOAP (Simple Object Access Protocol) та WSDL (Web Services Description Language), що є досить важкими та складними для налаштування і підтримки. У випадку прямої комунікації користувач і сервіс можуть взаємодіяти через будь-які доступні технології, такі як REST, GraphQL, WebSockets, gRPC, MQTT тощо.

Ця технологічна нейтральність надає більшу гнучкість у виборі інструментів для розробки та розгортання сервісів. Розробники можуть самостійно вибирати найкращі способи інтеграції без обмежень, пов'язаних із застарілими стандартами чи необхідністю підтримки централізованих реєстрів сервісів.

Таким чином, пряма взаємодія між користувачем і сервісом не тільки підвищує продуктивність та безпеку, а й дозволяє вибирати будь-які сучасні технології для передачі даних, що робить систему більш гнучкою, адаптивною та ефективною.

Уніфікація контракту взаємодії між компонентами в розподілених системах

Розподілена система складається з безлічі компонентів, розташованих на різних машинах, які взаємодіють, обмінюючись даними та узгоджуючи свої дії, щоб для кінцевого користувача система виглядала як єдине ціле.

Для відображення інформації, що використовується у запущених програмах, застосовуються структури даних. Інформація представляється у вигляді послідовності байтів у повідомленнях, що передаються між компонентами розподіленої системи. Таким чином, перед передаванням даних потрібно перетворити структуру даних на послідовність байтів. Після отримання повідомлення ці дані також повинні бути здатні перетворюватися назад у початкову структуру даних [6].

Комп'ютери обробляють різні типи даних, які можуть відрізнятися залежно від того, де саме ці дані передаються. Примітивні типи даних можуть мати різні формати зберігання та представлення. Наприклад, цілі числа можуть впорядковуватися у двох способах: **big-endian**, де старший байт (Most Significant Byte) знаходиться першим, та **little-endian**, де старший байт (Most Significant Byte) знаходиться останнім, а молодший байт (Least Significant Byte) – першим. Архітектури комп'ютерів також відрізняються у способі представлення чисел із плаваючою комою [7].

Ще одна проблема пов'язана із кодуванням символів. У більшості UNIX-систем використовується кодування ASCII, яке використовує один байт на символ. Натомість стандарт Unicode застосовує два байти на символ, дозволяючи підтримувати тексти багатьма мовами.

Щоб забезпечити успішну передачу даних між комп'ютерами, необхідно конвертувати їх у стандартний формат. Якщо обидва комп'ютери мають однакову архітектуру, конвертація може бути пропущена. У всіх інших випадках дані перетворюються на стандартний зовнішній формат перед передачею та відновлюються у локальний формат після отримання. Для цього дані передаються у форматі відправника разом із описом цього формату, щоб одержувач міг виконати необхідну конвертацію.

Варто зауважити, що самі байти залишаються незмінними під час передачі. Усі типи даних, які передаються як параметри чи результати, повинні бути здатні до перетворення у прийнятний формат для підтримки механізмів Remote Procedure Call (RPC) або Remote Method Invocation (RMI). Таким чином, стандарт зовнішнього представлення даних визначає узгоджений спосіб представлення структур даних і примітивних значень [7].

Питання використання IDL для опису профілей сервісу

Мова опису інтерфейсу або мова визначення інтерфейсу (IDL) – це загальний термін для мови, яка дозволяє програмі чи об'єкту, написаним однією мовою, спілкуватися з іншою програмою, написаною невідомою мовою. IDL зазвичай використовуються для опису типів даних та інтерфейсів у незалежний від мови спосіб, наприклад, між написаними на C++ і написаними на Java. IDL зазвичай використовуються в програмному забезпеченні для віддаленого виклику процедур. У цих випадках машини на обох кінцях зв'язку можуть використовувати різні операційні системи та комп'ютерні мови. IDL пропонують міст між двома різними системами [8].

Різні мови опису інтерфейсів (IDL), такі як OAS, WSDL, Protocol Buffers (Protobuf), CORBA та GraphQL Schema Definition Language (SDL), відіграють важливу роль у розробці програмного забезпечення. Вони забезпечують стандартизовані засоби для опису та визначення інтерфейсів, слугуючи мостом між різними мовами програмування та системами. Використовуючи IDL як проміжне представлення, ці мови стають ключовими інструментами для автоматизованого створення API. Вони охоплюють структурні та поведінкові аспекти програмних компонентів, виступаючи основою для створення API у стандартизованому та нейтральному до мови програмування форматі. Такий підхід спрощує процес розробки, забезпечує узгодженість, зменшує кількість помилок і сприяє створенню API, що відповідають галузевим стандартам і найкращим практикам. У результаті ці мови опису інтерфейсів підвищують ефективність і надійність розробки програмного забезпечення, надаючи єдину основу для визначення та створення інтерфейсів [9].

Використання мови визначення інтерфейсу (IDL) для опису профілів послуг у сервісно-орієнтованій архітектурі (SOA) або подібних середовищах забезпечує структурований і стандартизований спосіб визначення послуг, забезпечуючи плавний зв'язок, інтеграцію та автоматизацію.

Використання спільних об'єктів у розподілених системах створює унікальну низку викликів:

Створення та видалення об'єктів: У розподілених системах об'єкти можуть створюватися та видалятися динамічно. Для керування їхнім життєвим циклом можна використовувати фабрики об'єктів, що забезпечують ефективний розподіл і вивільнення ресурсів.

Збереження стану об'єкта: Розподілені об'єкти часто потребують зберігати свій стан упродовж часу. Для цього можна використовувати об'єктні бази даних або механізми серіалізації, що забезпечують довготривале збереження стану.

Пошук об'єктів: Клієнтам необхідні механізми для пошуку об'єктів у розподіленому середовищі. Це може включати служби імен або реєстраційні служби, які відстежують місцеперебування об'єктів.

Керування версіями об'єктів: Забезпечення сумісності між клієнтами та різними версіями об'єктів є важливим завданням. Стратегії керування версіями дозволяють клієнтам взаємодіяти з відповідною версією об'єкта без збоїв.

Розглянемо варіант використання Protobuf [10] як IDL для опису сервісів.

Protobuf або Protocol Buffers – це бінарний формат серіалізації, розроблений Google для ефективного обміну даними між сервісами.

Розглянемо структуру ***.proto*** файлів.

Насамперед файл .proto – це схема даних для Protobuf. Це місце, де описується структура даних, яку планується серіалізувати або десеріалізувати.

Все починається з указання версії синтаксису. Наприклад, *syntax = "proto3"*. Це говорить компілятору, які правила використовувати під час аналізу вмісту. Потім іде оголошення пакета *package mypackage*. Це допомагає уникнути конфліктів, а також організувати код.

Серце *.proto* файлу – це оголошення повідомлення. Кожне повідомлення – це структура даних, яку можна серіалізувати.

Повідомлення визначаються за допомогою синтаксису повідомлення *MyMessage {}*. Всередині фігурних скобок ви описуєте поля даних. Поля можуть бути стандартними типами даних, такими як *int32*, *float*, *double*, *string* або *bool*. Також можуть бути типи користувачів або інші повідомлення.

У Protobuf є три типи правил для полів: *singular* для одиничних значень, *repeated* для масивів значень і карта для асоціативних масивів. Кожному полю присвоюється унікальний номер. Ці номери використовуються в бінарному представленні та дуже важливі для сумісності даних.

Починаючи номера полів з 1, будьте обережні, не перестрибуючи через десятки і сотні, це може бути корисно для майбутніх версій вашого протоколу. Якщо поле більше не потрібне, позначте його як *reserved*.

Протокол Protobuf є бінарним форматом серіалізації, який має кілька ключових переваг, що роблять його ідеальним вибором для сучасних розподілених систем. Protobuf є значно компактнішим і швидшим порівняно з текстовими форматами, такими як JSON або XML. Завдяки використанню бінарного формату дані можуть бути передані з мінімальними накладними витратами, що робить Protobuf ідеальним для високопродуктивних систем, де важлива швидкість передачі даних. Це особливо важливо в умовах великих обсягів даних, де зменшення обсягу передаваних повідомлень може суттєво покращити загальну ефективність системи.

Також Protobuf забезпечує строгий контроль над типами даних завдяки типізованим структурам, що дозволяє уникнути багатьох помилок, характерних для текстових форматів. Це дозволяє забезпечити високу надійність під час передачі даних між різними компонентами, що особливо важливо в умовах розподілених систем, де взаємодіють різні технології та мови програмування.

Використання Protobuf як IDL для опису профілів сервісів дозволяє отримати значну перевагу в плані ефективності, надійності та масштабованості, що робить його оптимальним вибором для розподілених систем та сервісно-орієнтованих архітектур (SOA). Крім того, його використання сприяє автоматизації створення API, підвищуючи узгодженість і зменшуючи кількість помилок, що часто виникають під час використання менш строгих форматів.

Висновки

У статті було розглянуто підходи до формалізації ознак сервісів у сервісно-орієнтованій архітектурі (SOA) й наявні напрацювання в цій галузі досліджень. Було проаналізовано структуру метаданих, що описують сервіси, а також визначено їхню роль у забезпеченні сумісності, повторного використання.

Окрему увагу приділено профілям користувачів та постачальників сервісів, оскільки вони відіграють ключову роль у вдосконаленні інтеграційних процесів. Зокрема, профілі користувачів можуть слугувати основою для побудови рекомендаційних моделей, що автоматизують вибір сервісів, підвищуючи ефективність та гнучкість роботи системи.

На відміну від UDDI технології, запропонований варіант не має обмежень щодо протоколів комунікації між споживачем та вебсервісом. Відсутність такого обмеження дає можливість використовувати різні протоколи та формати передачі даних, а це призводить до більшої гнучкості, якої не було раніше.

У цьому контексті важливим є питання використання мови опису інтерфейсу (IDL), яка дозволяє створювати стандартизовані інтерфейси для різних компонентів системи, незалежно від мов програмування, що використовуються. Використання IDL забезпечує чітке визначення типів даних та методів, що робить взаємодію між компонентами більш передбачуваною та ефективною. Одним з найефективніших інструментів у цьому напрямі є використання Protocol Buffers (Protobuf) як IDL для опису профілів сервісів. Протокол Protobuf є бінарним форматом серіалізації, який, на відміну від JSON, є компактнішим, швидшим і типізованим, що дозволяє зменшити накладні витрати на передавання даних у розподілених системах. Використання Protobuf як IDL сприяє більш ефективному опису структур даних та забезпечує сумісність під час передачі даних між різними системами. Крім того, Protobuf забезпечує високу гнучкість для динамічного оновлення контрактів без порушення сумісності, що особливо важливо для розвитку сервісно-орієнтованих архітектур.

Важливим напрямом розвитку цієї галузі є вдосконалення методів семантичного пошуку, зокрема через використання більш складних моделей для кращого розуміння запитів користувачів та точнішого підбору сервісів. Застосування технологій, таких як семантичний веб та контекстний аналіз, може значно покращити точність і швидкість пошуку відповідних сервісів, що базуються на більш глибокому розумінні запитів та потреб користувачів. Таким чином, запропонована модель опису сервісів може слугувати підґрунтям для інтелектуалізації пошуку й рекомендацій сервісів у системі, що включає механізми семантичного аналізу, адаптивні рекомендації та вдосконалене управління сервісами на основі формалізованих ознак.

Література

1. Marwaha P., Banati H., Bedi P. UDDI Extensions for Temporally Customized Web Services. *International Conference on Computational Intelligence: Modeling Techniques and Applications,* 2013. P. 184–190.

2. UDDI (Universal Description, Discovery and Integration), 2023. URL: https://www.techtarget.com/whatis/definition/UDDI-Universal-Description-Discovery-and-Integration (дата звернення: 01.02.2025).

3. Pautasso C. Distributed Systems UDDI and beyond, Information and Communication System. URL: https://vs.inf.ethz.ch/edu/WS0405/VS/VS-050131.pdf (дата звернення: 01.02.2025).

4. Relationship between UDDI and WSDL, 2021. URL: https://www.ibm.com/docs/en/rsas/7.5.0?topic=uddi-relationship-between-wsdl (дата звернення: 01.02.2025).

5. Лобур М. В., Шварц М. Є., Стех Ю. В., Моделі і методи прогнозування рекомендацій для колаборативних рекомендаційних систем. *Інформаційні системи та мережі*, 2018. № 901. С. 68–75.

6. Мелешко Є. Проблеми сучасних рекомендаційних систем та методи їх рішення. *Системи управління, навігації та зв'язку*, 2018, № 4(50). С. 120–125.

7. Marinov M., Valova I. Component Interaction in Distributed Knowledge-Based Systems. *TEM Journal*. Volume 8. Issue 3, 2019. P. 721–727.

8. Marshaling in Distributed System 2024. URL: https://www.geeksforgeeks.org/marshalling-in-distributed-system (дата звернення: 03.02.2025).

9. Interface Description Language. URL: https://en.wikipedia.org/wiki/Interface_description_language (дата звернення: 03.02.2025).

10. What is IDL in Computing? URL: https://60sec.site/terms/what-is-idl-in-computing-interface-definition-language (дата звернення: 03.02.2025).

11. Protobuf 3.0 specification. URL: https://protobuf.dev/reference/protobuf/proto3-spec/ (дата звернення: 03.02.2025).

References

1. Marwaha P., Banati H., Bedi P. (2013). UDDI Extensions for Temporally Customized Web Services. *International Conference on Computational Intelligence: Modeling Techniques and Applications*. P. 184–190.

2. UDDI (Universal Description, Discovery and Integration), (2023). [Online]. Retrieved from: https://www.techtarget.com/whatis/definition/UDDI-Universal-Description-Discovery-and-Integration.

3. Pautasso, C. Distributed Systems UDDI and beyond, Information and Communication System. [Online]. Retrieved from: https://vs.inf.ethz.ch/edu/WS0405/VS/VS-050131.pdf.

4. Relationship between UDDI and WSDL. (2021). [Online]. Retrieved from: https://www.ibm.com/docs/en/rsas/7.5.0?topic=uddi-relationship-between-wsdl.

5. Lobur, M. V., Schwartz, M. E., Stekh, Y. V. (2018). Models and methods of forecasting recommendations for collaborative recommender systems. *Information Systems and Networks*. No. 901. P. 68–75 [in Ukrainian].

6. Meleshko, E. (2018). Problems of modern recommender systems and methods for their solution. *Control, navigation and communication systems*, No. 4(50). P. 120–125 [in Ukrainian].

7. Marinov, M., Valova, I. (2019). Component Interaction in Distributed Knowledge-Based Systems. *TEM Journal*. Volume 8. Issue 3. P. 721–727.

8. Marshaling in Distributed System. (2024). [Online]. Retrieved from: https://www.geeksforgeeks.org/marshalling-in-distributed-system.

9. Interface Description Language. [Online]. Retrieved from: https://en.wikipedia.org/wiki/Interface_description_language.

10. What is IDL in Computing? [Online]. Retrieved from: https://60sec.site/terms/what-is-idl-in-computing-interface-definition-language.

11. Protobuf 3.0 specification. [Online]. Retrieved from: https://protobuf.dev/reference/protobuf/proto3-spec/.