УДК 004.41

С.С. Ремарович

СИСТЕМИ ВИЯВЛЕННЯ WEB-СЕРВІСІВ В СЕРВІС-ОРІЄНТОВАНІЙ АРХІТЕКТУРІ: ПРОБЛЕМИ І РІШЕННЯ

З появою парадигми сервіс-орієнтованого обчислення і зростаючою кількістю доступних Web-сервісів в Інтернеті посилюється запит на засоби для виконання виявлення, вибору, композиції і виклику Web-сервісів. На сьогодні запропонована велика кількість підходів щодо виявлення Web-сервісів, яка обумовлена низкою задач та їх можливими рішеннями при побудові систем виявлення Web-сервісів. У цій статті проведено аналітичний огляд задач, які постають при розробці і функціонуванні таких систем, та наведені існуючі підходи до їх вирішення. Показано, що проблема виявлення Web-сервісів може бути висвітлена в дескриптивній логіці на основі «найкращого покриття».

Вступ

Сервіс-орієнтоване обчислення (Service-Oriented Computing – SOC) являється новою обчислювальною парадигмою, головна мета якої полягає у підтримці розробки розподілених застосувань в гетерогенних середовищах [1]. SOC дозволяє компонувати і збирати разом розподілену функціональність для створення програмних систем. Ця функціональність поставляється у вигляді основних будівельних блоків, які називаються сервісами. Сервіс може бути визначений як частина функціональності, яка виконана зовнішнім постачальником. Постачальники публікують свої сервіси в реєстрі сервісів, визначаючи при цьому: умови включення, технічні обмеження, вимоги і семантичну інформацію. З іншого боку, споживачі сервісів виявляють опубліковані сервіси через реєстр, і, в свою чергу, вибирають і укладають контракт з ним. SOC реалізується за допомогою технологій Web-сервісів, оскільки вони призначені для підтримки інтероперабельності взаємодії постачальник-споживач через Інтернет [2, 3].

У зв'язку із великим значенням виявлення сервісів для SOC парадигми, як дослідники, так і практики розробляють системи виявлення сервісів. Останнім часом Web-пошукові системи (наприклад, Google) стали новим джерелом для знаходження Web-сервісів [4], так як описи сервісів, як правило, знаходяться на Web-серверах, які скануються та індексуються пошуковими системами. З іншого боку, шляхом адаптації існуючих інформаційно-пошукових методів (Information Retrieval - IR), деякі дослідники запропонували використовувати описи Web-сервісів як документи, тим самим зводячи проблему виявлення відповідних сервісів до добре відомої проблеми знаходження релевантних документів [5]. Інші підходи запропонували анотувати описи сервісів метаданими, які являють собою однозначні визначення понять із загальної онтології (семантика), що дало початок поняттю семантичних Web-сервісів [6].

Існують істотні відмінності в підходах до виявлення сервісів. Одна з основних відмінностей полягає у тому, як такі підходи розглядають описи сервісів. Наприклад, семантичні підходи залежать від загальних онтологій і анотованих ресурсів, водночас як IR-підходи ґрунтуються на текстових описах. Хоча системи виявлення сервісів прагнуть вирішити одну й ту ж проблему, вони можуть дуже відрізнятися одна від одної, і один з варіантів може бути доречним у певному середовищі, але не в інших. Різноманітність варіантів для виявлення Web-сервісів мотивує необхідність обґрунтованих критеріїв, щоб охарактеризувати їх. Масштабованість, відмовостійкість і відповідність стандартам є важливими аспектами таких систем.

В роботі [5] автори представили комплексне дослідження методів, архітектур і моделей, пов'язаних з виявленням

Web-сервісів, проаналізувавши більше 30 підходів для виявлення Web-сервісів (у тому числі промислових та наукових). Вони розробили таксономію для організації цих підходів, на основі чотирьох основних категорій: архітектура, стандарти, моделі і QoS-дані (Quality of Service – QoS).

Альтернативним підходом до виявлення Web-сервісів є WSIL (Web Services Inspection Language), пропозиція IBM® і Microsoft [7]. Коли ми виявляємо Web-сервіс за допомогою стандарту UDDI (Universal Description, Discovery and Integration), то йдемо до централізованого реєстру. WSIL дозволяє перейти безпосередньо до постачальника сервісів і спитати, які сервіси він надає. Специфікація WSIL будується навколо моделі на основі XML для побудови агрегації посилань на існуючі описи Web-сервісів, які виставлені з використанням стандартної технології Web-сервера. WSIL забезпечує розподілений метод виявлення сервісів, який подає посилання на описи сервісів у точці-пропозиції постачальника сервісів, вказавши, як інспектувати Web-сайт для доступних Web-сервісів. Специфікація WSIL визначає місце розташування на Web-сайті, де можна шукати описи Web-сервісів. Оскільки WSIL фокусується на розподіленому виявленні сервісів, то специфікація WSIL доповнює UDDI, допомагаючи виявляти сервіси, які доступні на Web-сайтах, та які ще не перераховані в реєстрі UDDI.

У цій статті показано, що проблема виявлення Web-сервісів може бути висвітлена в дескриптивній логіці на основі «найкращого покриття». Проведено аналітичний огляд задач, які необхідно вирішувати системам виявлення Web-сервісів, та підходи до їх розв'язання, які існують на сьогодні. Процес виявлення Web-сервісів охоплює весь спектр завдань від запиту на сервіс до виклику сервісу. Вирішення проблем, пов'язаних з виявленням сервісів, залежить від рівня пошуку – синтаксичний, чи семантичний; функціональний, чи процесний; від того, який семантичний формалізм використовується для опису сервісів і як управляти цими описами.

Web-сервіси та їх стандарти

Перед тим, як розглядати задачі систем виявлення Web-сервісів, важливо спочатку уточнити, що саме означає «виявлення Web-сервісів» і розглянути технології, які пов'язані з Web-сервісами. Хоча існують різні пропозиції для виявлення Web-сервісів, розуміння виявлення (discovery) є дуже різним, і часто його плутають з такими термінами, як вибір (selection) і зіставлення (matching) [8]. Серед невеликої кількості визначень виявлення сервісів у літературі, найбільш офіційне визначення є те, що викладено в [9]: «Акт локалізації машинно-оброблюваного опису ресурсу, пов'язаного з Web-сервісом, що, можливо, був раніше невідомий, і, що відповідає деякому функціональному критерію. Він включає в себе зіставлення набору функціональних та інших критеріїв з набором описів ресурсу. Мета полягає в тому, щоб знайти відповідний ресурс, пов'язаний з Web-сервісом». Спочатку основна мета технології Web-сервісів полягала у тому, щоб визначити Web-сервіси в машинно-зрозумілій формі для того, щоб автоматично виявляти їх іншими учасниками (агентами або сервісами). В результаті, знаходження відповідних Web-сервісів розглядається як важливий ключ до композиції сервісів, виклику і виконання.

Виявлення Web-сервісів являє собою процес пошуку Web-сервісів, які задовольняють певним вимогам. Система виявлення сервісів пов'язує між собою постачальників і споживачів сервісів. Провайдери можуть використовувати систему виявлення, щоб рекламувати свої сервіси, а споживачі можуть використовувати її для виявлення сервісів, які відповідають їхнім потребам. Об'єктом пошуку системи виявлення є Web-сервіс, тобто, його опис. Розглянемо цю сутність.

Виявлення сервісів через Інтернет посилається на сукупність технологічних стандартів для модульних програм, які можуть бути опубліковані, виявлені і активізовані через Web [1]. Ця сукупність складається з чотирьох рівнів: зв'язок, серіалізація, опис і виявлення. Вичерпний

опис цих технологічних стандартів може бути знайдений в роботі [10].

Визначення Web-сервісів згідно W3C виглядає наступним чином: «Web-сервіс являється програмною системою, яка визначається URI (Uniform Resource Identifier), чиї загальнодоступні інтерфейси та зв'язки визначені і описані за допомогою XML (Extensible Markup Language). Його визначення може бути виявлене іншими системами програмного забезпечення. Ці системи можуть потім взаємодіяти з Web-сервісом в порядку, встановленому його визначенням, використовуючи XML засновані повідомлення, що передаються через Інтернет-протоколи». Web-сервіси, які посилені додатковими метаданими, що виражають їх семантику, називаються *семантичними Web-сервісами* [11].

Web-сервіси визначаються такими елементами:

• входи *I*, тобто значення, які необхідні сервісу для його виконання;

• виходи *O*, тобто значення, які виробляються після виконання сервісу;

• попередні умови *P*, тобто всі умови, які повинні бути дотримані, щоб гарантувати успішне виконання сервісу;

• ефекти *E*, тобто зміни в стані світу, які відбуваються у разі успішного або неуспішного виконання сервісу.

Мета виявлення (запит) може бути визначена подібним набором елементів: у цьому випадку входи і попередні умови виражають значення та умови, які відомі тому, хто викликає, в той час як виходи і ефекти являють собою очікувані результати виклику сервісу.

Крім функціональних характеристик, Web-сервіси описуються з точки зору не функціональних (QoS) аспектів (вартість, час відгуку, надійність, доступність, готовність, безпечність і т. д.).

В реальному світі наявні сервіси не є атомарними, і не можуть взагалі бути виконані в одному кроці запит-відповідь. Взагалі, кожен компонентний сервіс може бути вказаний як протокол взаємодії, де різні «атомарні» виклики та відповіді були об'єднані в складні шаблони виконання. Хоча деталі точного протоколу, необхідного для взаємодії з існуючим сервісом, не важливі у виявленні, вони стають необхідні, коли ми прагнемо до генерації композитних Web-сервісів, які є виконувані. З цієї причини, композиція виконуваних сервісів повинна мати справу з описами Web-сервісів з точки зору складних процесів, які складаються з довільних комбінацій атомарних взаємодій, в стилі, наприклад, OWL-S моделей процесів [12] або на основі абстрактної машинної моделі, як у WSMO інтерфейсах [13], чи WS-BPEL описів. Проблема композиції сервісів означає побудувати новий сервіс (композитний сервіс), який виконує деяку потрібну функціональність, взаємодіючи з наявними сервісами (компонентні сервіси). Функціональність запиту може покриватись набором сервісів.

Стандартним форматом для опису Web-сервісів є мова опису WSDL (Web Service Description Language) в XML-форматі [14], яка представляє опис сервісу у вигляді набору операцій, виклик яких відбувається на основі обміну повідомленнями. В об'єктно-орієнтованих термінах, WSDL-документ описує сервіс як статичний інтерфейс, операцію як метод і повідомлення, як аргументи методу (входи і виходи). Цього може бути досить для Web-сервісів, які беруть участь в обміні повідомленнями без урахування станів. Крім того, WSDL дозволяє провайдерам також описати винятки як повідомлення. Кожна частина WSDL документу може містити документацію у вигляді коментарів. WSDL описи публікуються в реєстрах сервісів.

Домінуючим стандартом для реєстрів Web-сервісів є UDDI [15]. UDDI специфікація забезпечує незалежний від платформи спосіб опису і виявлення Web-сервісів і постачальників Web-сервісів. Структури даних UDDI забезпечують основу для опису базової інформації сервісу і розширюваний механізм для вказівки детальної інформації про доступ до сервісу, використовуючи будь-яку стандартну мову опису. Специфікація UDDI організовує три групи метаданих: білі, зелені та жовті сторінки. Білі сторінки підтримують бізнес-інформацію (наприклад, адреса, контакт та інші відомі ідентифікато-

ри). Жовті сторінки зв'язують сервіс з промисловими класифікаціями, заснованими на стандартних таксономіях, як United Nations Standard Products and Services Code (UNSPSC), Standard Industrial Classification (SIC) або North American Industry Classification System (NAICS) [16]. Зелені сторінки зберігають технічну інформацію про сервіси, виставлених одним або кількома підприємствами, наприклад, WSDL-документ. Ця специфікація складається з чотирьох основних типів даних, а саме businessEntity, businessService, binding Template і tModel (технічна модель), які визначені в XML.

UDDI являє собою основу для інших підходів до виявлення Web-сервісів і не призначений для того, щоб забезпечити потужні можливості для виявлення сервісів, але він призначений для забезпечення стандартизованих форматів для програмного бізнесу та виявлення сервісів, так що пошукові системи можуть бути побудовані на його вершині. UDDI надає тільки перегляд за категоріями, і встановлення відповідності на основі ключових слів при виявленні сервісу. Наскільки нам відомо, існує безліч комерційних реалізацій UDDI і тільки одна реалізація відкрита. Apache Software Foundation підтримує jUDDI [17], альтернатива з відкритим вихідним кодом.

Компонентні і композитні сервіси можуть виражатися за допомогою мови WS-BPEL (Business Process Execution Language) [18], яка є де-факто стандартом для опису поведінки Web-сервісів з урахуванням станів. У WS-BPEL набір атомарних операцій зв'язку (invoke, receive і reply) об'єднується в робочий процес (workflow), який визначає процес, реалізований сервісом зі станами. Атомарні зв'язки відповідають атомарним операціям Web-сервісу і визначаються у WSDL специфікації.

За останні кілька років були проведені дослідження, щоб додати значущі дані до Web-сервісів, породивши таким чином семантичні Web-сервіси [6]. Семантичні Web-сервіси дали початок чотирьом основним напрямкам досліджень: визначення, генерування, управління та використання метаданих сервісу. У майбутньому припускається, що програмні агенти, діючи в інтересах своїх користувачів, будуть шукати відповідні сервіси. Автоматичне укладання контрактів з сервісами залежить від визначення моделі для опису характеристик рівня сервісу, а для виклику сервісу потрібні дані про репутацію провайдерів і платіжна інформація. Другий напрям досліджень стосується того, що описи сервісів повинні бути анотовані метаданими відповідно до визначеної моделі. Третій напрямок досліджень охоплює питання зберігання анотацій та їх ефективного добування з семантичних реєстрів. І на кінець, семантичні реєстри мають забезпечувати механізми виявлення, які використовують переваги від семантичних описів. Для автоматичного виявлення сервісів необхідно, щоб шукачі могли довіряти описам провайдерів. Інтернет є дуже відкритим гетерогенним середовищем, тому виникає серйозна проблема довіри між споживачами і постачальниками сервісів [19]. Це цікава проблема, яка представляє багато можливостей для досліджень, які не тільки містять, як забезпечити класичні не функціональні QoS вимоги, такі як час відгуку або доступність, але й функціональні (визначити функціональність сервісу) та правові (ліцензування і авторські права) аспекти.

Анотації семантичних сервісів відіграють дуже важливу роль у виявленні сервісів і впливають на архітектуру, алгоритми, інструменти та ефективність систем виявлення сервісів.

Існує три основні формалізми для визначення метамоделі, або типу семантичної інформації, для опису Web-сервісів, а саме OWL-S [20], WSMO [13] і WSDL-S [21]. OWL-S, який був прийнятий як представлення W3C в листопаді 2004 року, забезпечує основу для опису як функцій, так і оголошень Web-сервісів за допомогою OWL (Ontology Web Language). OWL є рекомендацією W3C для опису семантичних відношень домену.

OWL-S являється однією з найбільш часто вживаних мов для семантичних анотацій Web-сервісів. Це онтології OWL, які надають конструкції, щоб визначити сенс атомарних і композитних

сервісів. OWL заснована на дескриптивній логіці. Основа OWL-S – клас *Service* з трьома головними компонентами, які представлені наступними класами:

Service Profile визначає, що сервіс вимагає від споживачів і що він пропонує їм. Він виражає те, які перетворення здійснюються сервісом, визначаючи I/O і перед-/пост-умови. З точки зору виявлення і композиції, це найважливіша частина визначення OWL-S сервісу.

Service Model визначає, як працює сервіс з точки зору процесу. Модель визначає сервіс як *атомарний* процес, або *композицію* кількох атомарних сервісів. Для цього процесу вона визначає, які вхідні дані мають бути надані для його виконання, які передумови мають виконуватися, і які вихідні дані і ефекти створюються. Цей опис має узгоджуватись з визначеннями I/O, перед-/пост-умов в *Service Profile*.

Service Grounding визначає, як сервіс використовується. Він визначає протокол зв'язку, формати повідомлень, методи серіаліації, а також інші конкретні деталі сервісу, необхідні для його виклику.

Ще однією ініціативою у цьому напрямку є WSMO (Web Service Modeling Ontology) [13], яка забезпечує поглиблену концептуальну модель для опису семантики сервісів і сприяє автоматизації виявлення сервісу, композиції і виклику. WSMO включає у себе чотири підсистеми: *онтології*, щоб формалізувати знання предметної області; *цілі*, які описують мету користувача; *Web-сервіси* та їх семантичний опис і *посередники* для підтримки сумісності та вирішення неоднорідності. Опис WSMO елементів представлено за допомогою Web Service Modeling Language (WSML) [22]. WSMO також визначає концептуальну модель WSMX [23], середовище виконання семантичних Web-сервісів. Таким чином, WSMO, WSML і WSMX утворюють всеосяжний фреймворк для моделювання, опису та виконання семантичних Web-сервісів.

WSDL-S дозволяє визначити семантику сервісів, які описані у WSDL [14]. Замість того, щоб запропонувати нову мову, він визначає деякі розширення WSDL, щоб семантично анотувати сервіси. WSDL-S являється агностиком стосовно мови визначення семантики. Семантична модель предметної області може бути представлена в OWL, RDF, WSML або UML. WSDL-S пов'язує ці моделі з елементами сервісу, визначених у WSDL. Один Web-сервіс може бути анотований кількома семантичними моделями.

І запит, і оголошення сервісу визначаються на мовах опису сервісів з певною виразністю. Така виразність надзвичайно впливає на якість процедури виявлення. Крім того, повнота описів також впливає і на процес встановлення відповідності між запитами на сервіс і оголошеними сервісами.

Одним з головних внесків OWL-S і WSMO є можливість міркування і посередництва над описами сервісів, яка підтримується мовами, що лежать в їх основі, тобто OWL і WSML.

Ефективність вирішення задач при виявленні Web-сервісів залежить від обраної моделі формалізації сервісу. Потрібен формалізм, який забезпечуватиме гарне семантичне анотування сервісів та підходи для встановлення семантичної відповідності між оголошеним сервісом і запитом на сервіс, а також автоматизованого виявлення підходящих сервісів при побудові композитного сервісу, який реалізує конкретний бізнес-процес. Так як при побудові композитного сервісу поведінка компонентних сервісів впливає на хід бізнес-процесу, то формалізм повинен мати справу з динамічним аспектом (поведінкою) сервісу. Це є одним з ключових аспектів, що робить очевидною доцільність використання дескриптивних логік (Description logic - DL) [24] для опису сервісів, так як DL забезпечує можливість міркувань для сервісів, і таким чином дозволяє описати їх динамічний аспект.

Формалізація виявлення Web-сервісів на основі покриття-виведення

Розглянемо підхід до виявлення сервісів в контексті дескриптивної логіки. Ключовим аспектом дескриптивних логік є їх формальні семантики та підтримка су-

джень. Механізм виявлення сервісів повинен підтримувати гнучке порівняння, оскільки нереально очікувати точного збігу запитів на сервіс та оголошення сервісу. Пропонується використовувати операцію різниці на описах сервісів. Така операція дозволяє з підмножини описів Web-сервісів витягти ту частину, яка є семантично спільною з заданим запитом на сервіс, і ту частину, яка семантично відмінна від запиту. Знаючи першу і останню частини, можна ефективно вибрати відповідні Web-сервіси. Алгоритм порівняння приймає у якості вхідних даних запит на сервіс *Q* та онтологію сервісів *T* і знаходить множину сервісів, що називається «краще покриття» запиту *Q*, описи яких містять якомога більше спільного з інформацією запиту *Q*, наскільки це можливо, і якнайменше лишньої інформації щодо *Q*, наскільки це можливо.

Наведемо низку визначень [25, 26]. Нехай $C, C_1, \ldots, C_n$ і D – це дескрипції концептів:

C поглинається (включається в) D (позначається $C \sqsubseteq D$), якщо $C^I \subseteq D^I$ для будь-якої інтерпретації I.

C еквівалентно D (позначається $C \equiv D$), якщо $C^I = D^I$ для всіх інтерпретацій I.

D є найменшим спільним узагальнювачем для $C_1, C_2, \ldots, C_n$ (позначається $D = lcs(C_1, C_2, \ldots, C_n)$, *lcs* – least common subsummer), якщо $C_i \sqsubseteq D_i$, для всіх i $(1 \le i \le n)$, та D є найменшою дескрипцією концептів, що володіють даною властивістю, тобто, якщо D' є дескрипцією концептів, що задовольняє умові $C_i \sqsubseteq D'$, $1 \le i \le n$, то $D' \sqsubseteq D$ [27].

Інтенсіональні дескрипції, які містяться в базі знань, побудовані з використанням дескриптивних логік, називаються ***термінологією***.

Визначення 1 (термінологія). Нехай A ім'я концепту та C – дескрипція концепту. Тоді $A \doteq C$ є визначенням концепту. Термінологія $\mathcal{T}$ – це кінцева множина визначень концептів таких, що кожне ім'я концепту зустрічається не більше одного разу в лівій частині визначення.

Ім'я концепту A називається концептом, який визначений в термінології $\mathcal{T}$, якщо воно зустрічається в лівій частині визначення концепту в термінології $\mathcal{T}$. Іншими словами, A називається атомарним концептом.

Інтерпретація I задовольняє твердженню $A \doteq C$, якщо $A^I = C^I$. Інтерпретація I є моделлю для термінології $\mathcal{T}$, якщо I задовольняє всім твердженням в $\mathcal{T}$.

Термінологія, яка побудована з використанням конструкторів мови $\mathcal{L}$, називається $\mathcal{L}$ -термінологією. Далі вважаємо, що термінологія $\mathcal{T}$ є ациклічною, тобто не існує циклічних залежностей між визначеннями концептів. Ациклічні термінології можуть бути розгорнуті шляхом заміщення певних імен їх визначеннями до тих пір, поки певні імена все ще будуть зустрічатися в правих частинах визначень. Таким чином, поняття найменшого спільного узагальнювача (least common subsumer – *lcs*) множини дескрипцій може бути розширено до концептів, що містять певні імена. В даному випадку, для позначення найменшого спільного узагальнювача концептів C і D щодо термінології $\mathcal{T}$ ми використовуємо вираз $lcs_{\mathcal{T}}(C, D)$ (тобто, *lcs* застосовується до розгорнутих описів C і D).

Визначення 2 (операція різниці) [26]. Нехай C *і* D – дві дескрипції концептів такі, що $C \sqsubseteq D$. Різниця $C - D$ дескрипцій концептів C *і* D визначається наступним чином: $C - D := \max_{\sqsupseteq} \{B \mid B \sqcap D \equiv C\}$.

Різниця двох дескрипцій C і D визначається як дескрипція, яка містить всю інформацію, що є частиною дескрипції C, але не входить до дескрипції D. Це визначення операції різниці вимагає, щоб другий операнд включав у себе перший. Однак, якщо операнди C і D не порівняні щодо відношення включення, то різниця $C-D$ може бути задана шляхом побудови найменшого спільного узагальнювача C і D, тобто $C - D := C - lcs(C, D)$.

Визначення 3 (скорочена клаузальна форма та структурна еквівалентність). Нехай $\mathcal{L}$ – дескриптивна логіка. Клауза в $\mathcal{L}$ – це дескрипція A з наступною властивістю: $(A \equiv B \sqcap A') \Rightarrow (B \equiv \top) \vee (B \equiv A) \top$. Кожна кон'юнкція $A_1 \sqcap \ldots \sqcap A_n$ – це клауза, яка може бути представлена як множина клауз $\{A_1, \ldots, A_n\}$.

$A = \{A_1, \ldots, A_n\}$ називається скороченою множиною клауз, якщо $n = 1$, або жодна клауза не поглинає кон'юнкцію інших клауз: $\forall i (1 < i < n) : A_i \not\supseteq A \setminus A_i$. Тоді множина A називається скороченою формою клауз (*RCF* – reduced clause form) кожної дескрипції $B \equiv A_1 \sqcap \ldots \sqcap A_n$.

Нехай $A = \{A_1, \ldots, A_n\}$ та $B = \{B_1, \ldots, B_m\}$ будуть скороченими множинами клауз в дескриптивній логіці $\mathcal{L}$. A *і* B є еквівалентними структурами (позначається $A \cong B$), якщо: $n = m \wedge \forall i (1 < i < n) \exists 1 \le j, k \le n : A_i \equiv B_j \wedge B_i \equiv A_k$.

Якщо в дескриптивній логіці для кожної дескрипції всі його RCF мають еквівалентні структури, ми кажемо, що RCF форми – структурно унікальні в цій логіці.

Операція структурної різниці визначається як операція різниці множин, що застосована до множин клауз, де порівняння клозів виконується на основі відношення еквівалентності.

Тепер введемо поняття структурного включення, так як воно визначене в [26].

Визначення 4 (структурне включення). Кажуть, що відношення включення в дескриптивній логіці $\mathcal{L}$ буде структурним, якщо для будь-якої клаузи $A \in \mathcal{L}$ та будь-якої дескрипції $B = B_1 \sqcap \ldots \sqcap B_m \in \mathcal{L}$, яка задана *RCF* формою, виконується: $A \sqsupseteq B \Leftrightarrow \exists 1 \le i \le m : A \sqsupseteq B_i$.

Формалізація задачі найкращого покриття. Введемо деякі основні визначення, що необхідні, щоб формально визначити задачу найкращого покриття. Нехай $\mathcal{L}$ – DL зі структурним включенням, $\mathcal{T}$-$\mathcal{L}$ -термінологія, і $Q \not\equiv \bot$ – когерентна дескрипція $\mathcal{L}$-концепту. Множина визначених концептів, які зустрічаються в $\mathcal{T}$, позначається як $S_{\mathcal{T}} = \{S_i, i \in [1, n]\}$, де $S_i \not\equiv \bot$, $\forall i \in [1, n]$. Далі ми вважаємо, що дескрипції концептів $S_i, i \in [1, n]$ представляються своїми RCF-формами.

Визначення 5 (покриття). Покриттям Q, що використовує термінологію $\mathcal{T}$, є кон'юнкція E деяких імен S_i із $\mathcal{T}$ таких, що: $Q - lcs_{\mathcal{T}}(Q, E) \not\equiv Q$.

Отже, покриття концепту Q, що використовує термінологію $\mathcal{T}$, визначається як довільна кон'юнкція концептів, що зустрічаються в $\mathcal{T}$, які розділяють з Q деяку загальну інформацію. Слід зауважити, що покриття E концепту Q завжди сумісне з Q (тобто, $Q \sqcap E \not\equiv \bot$), так як $\mathcal{L}$ – це DL з структурно унікальними RCF та $Q \not\equiv \bot$, та $S_i \not\equiv \bot$, $\forall i \in [1, n]$.

Щоб визначити поняття найкращого покриття, необхідно охарактеризувати частину опису покриття E, що не міститься в описі запиту Q, і частини запиту Q, яка не міститься в описі його покриття E.

Визначення 6 (залишок та недостача) (rest and miss). Нехай Q – дескрипція $\mathcal{L}$-концепта та E – покриття Q, яке використовує термінологію $\mathcal{T}$. Залишок Q щодо E, позначається *RestE(Q)*, визначається як: $RestE(Q) \equiv Q - lcs_{\mathcal{T}}(Q, E)$.

Недостатня інформація запиту Q щодо E, позначається *MissE(Q)*, визначається як: $MissE(Q) \equiv E - lcs_{\mathcal{T}}(Q, E)$.

Визначення 7 (найкраще покриття). Дескрипція концепту E називається найкращим покриттям запиту Q, яка використовує термінологію $\mathcal{T}$, якщо:

E – це покриття Q, що використовує $\mathcal{T}$, та не існує жодного покриття E' запиту Q, що використовує $\mathcal{T}$, такого, що $(|Rest_{E'}(Q)|, |Miss_{E'}(Q)|) < (|Rest_E(Q)|, |Miss_E(Q)|)$, де $<$ – це лексикографічний порядковий оператор.

Найкраще покриття визначається як покриття, яке має, по-перше, найменший залишок, по-друге, найменшу кількість недостатньої інформації.

Задача найкращого покриття, позначається $BCOV(\mathcal{T}, Q)$, – це задача обчислення всіх найкращих покриттів запиту Q,

що використовують термінологію $\mathcal{T}$. Алгоритм найкращого покриття надано в [28].

Семантичне міркування для виявлення Web-сервісів. Розглянемо запропонований механізм суджень, що може використовуватися для автоматизації виявлення Web-сервісів у контексті DAML-S онтологій. Більш докладний опис цих аспектів можна знайти в [29]. Як згадувалось вище, структура онтології сервісів має три основні частини: *ServiceProfile*, *ServiceModel* та *ServiceGrounding*. Профайл сервісу забезпечує інформацію про сервіс, яка може використовуватися агентом, щоб визначити, чи задовольняє сервіс його потребам.

Як пропонується в [30], відповідність між запитом (виражається за допомогою профайлу сервісу) і оголошенням сервісу визначається шляхом порівняння всіх виходів запиту з виходами оголошення сервісу та всіх входів оголошення сервісу з вхідними даними запиту. Адаптуємо цю ідею, але використаємо алгоритм обчислення найкращого покриття замість алгоритму порівняння, заданого в [30]. Механізм виявлення сервісу на основі заданого запиту на сервіс Q і онтології $\mathcal{T}$. $\mathcal{T}$ має обчислити найкращу комбінацію Web-сервісів, яка максимально, на скільки можливо, задовольняє вихідним даним запиту Q і яка вимагає, на скільки це можливо, мінімального числа вхідної інформації, не наданої в описі Q. Таку комбінацію Web-сервісів ми назвемо найкращим покриттям профайлу Q з використанням $\mathcal{T}$. Для вирішення даного завдання необхідно розширити методи найкращого покриття, представлені вище, щоб врахувати описи профайлів, представлені далі.

Нехай $\mathcal{T} = \{S_i, i \in [1,n]\}^{\mathcal{T}}$ – обмежена онтологія і $E \equiv S_l \sqcap ... \sqcap S_p$, де $l, p \in [1,n]$, є кон'юнкцією декількох сервісів, присутніх в $\mathcal{T}$. Позначимо $I(E)$ (відповідно $O(E)$) концепт, визначений за допомогою кон'юнкції всіх входів (відповідно виходів), що зустрічаються в секції профайлу всіх сервісів S_i, для всіх $i \in [l, p]$. Таким самим чином ми будемо використовувати $I(Q)$ (відповідно $O(Q)$) для позначення концепту, визначеного за допомогою кон'юнкції всіх входів (відповідно виходів), що зустрічаються в секції профайлу в заданому запиті Q.

Розширимо поняття покриття, залишку та недостачі до профайлів сервісів наступним чином.

Визначення 8 (покриття профайлу (Pcover)). Покриття профайлу запиту Q, називається *Pcover*, що використовує термінологію $\mathcal{T}$, є кон'юнкцією E деяких сервісів S_i з $\mathcal{T}$ таке, що

$$O(Q) - lcs_{\mathcal{T}}(O(Q), O(E)) \not\equiv O(Q).$$

Отже, покриття *Pcover* запиту Q, що використовує термінологію $\mathcal{T}$, визначається як довільна кон'юнкція Web-сервісів, які присутні в $\mathcal{T}$, що розділяє з Q деякі виходи.

Визначення 9 (залишок профайлу (*Prest*) та недостача профайлу (*Pmiss*)). Нехай Q запит на сервіс та E – покриття запиту Q з використанням термінології $\mathcal{T}$. *Prest* запиту Q щодо E, що позначається $Prest_E(Q)$, визначається наступним чином:

$$Prest_E(Q) \doteq O(Q) - lcs_{\mathcal{T}}(O(Q), O(E)).$$

Недостатня інформація профайла про Q щодо E позначається $Pmiss_E(Q)$ та визначається наступним чином: $Pmiss_E(Q) \doteq I(E) - lcs_{\mathcal{T}}(I(Q), I(E))$.

Нарешті, поняття найкращого покриття може бути розширене до профайлів шляхом відповідної заміни у визначенні 7 [31] *rest* та *miss* на *Prest* та *Pmiss* профайлів. Розроблений алгоритм, названий *computeBProfileCov* [28], обирає комбінації сервісів, які найкраще збігаються із заданим запитом, та ефективно обчислює виходи запиту, які не можуть бути задоволені наявними сервісами (тобто, *Prest*), а також входи, які потребуються обраними сервісами та не надаються у запиті (тобто, *Pmiss*).

Розглянемо приклад, який показує, як може використовуватися поняття найкращого покриття профайлу для встановлення відповідності між запитом на сервіс та оголошеннями сервісів. Розглянемо онтологію Web-сервісів, що містить наступні три сервіси:

ToTravel дозволяє зарезервувати подорож, задану маршрутом (тобто, пунктом відправлення та пунктом прибуття), датою і часом прибуття. Вхідні параметри: *Itinerary, Arrival*. Вихідні параметри: *TripReservation.*

FromTravel дозволяє зарезервувати подорож, задану маршрутом та датою і часом відправлення. Вхідні параметри: *Itinerary, Departure*. Вихідні параметри: *TripReservation.*

Hotel дозволяє зарезервувати готель по заданим пункту призначення і періоду часу, який виражається термінами – дата в'їзду та дата виїзду. Вхідні параметри: *Destination, StayDuration*. Вихідні параметри: *HotelReservation.*

Приклад онтології туризму:

$$\textbf{Itinerary} \equiv (\geq 1\ \textbf{departurePlace}) \sqcap (\textbf{departurePlace.Location}) \sqcap (\geq 1\ \textbf{arrivalPlace}) \sqcap (\forall \textbf{arrivalPlace.Location})$$

$$\textbf{Arrival} \equiv (\geq 1\ \textbf{arrivalDate}) \sqcap (\forall \textbf{arrivalDate.Date}) \sqcap (\geq 1\ \textbf{arrivalTime}) \sqcap (\forall \textbf{arrivalTime.Time})$$

$$\textbf{Departure} \equiv (\geq 1 \textbf{departureDate}) \sqcap (\forall \textbf{departureDate.Date}) \sqcap (\geq 1 \textbf{departureTime}) \sqcap (\forall \textbf{departureTime.Time})$$

$$\textbf{Destination} \equiv (\geq 1 \textbf{DestinationPlace}) \sqcap (\forall \textbf{destinationPlace.Location})$$

$$\textbf{StayDuration} \equiv (\geq 1\ \textbf{checkIn}) \sqcap (\forall \textbf{checkIn.Date}) \sqcap (\geq 1 \textbf{checkOut}) \sqcap (\forall \textbf{checkOut.Date})$$

$$\textbf{TripReservation} \equiv \ldots$$

$$\textbf{HotelReservation} \equiv \ldots$$

$$\textbf{CarRental} \equiv \ldots$$

Профайли сервісів звертаються до концептів, що визначені в онтології туризму, для опису якої використовується синтаксис звичайної дескриптивної логіки. Дескрипція концепту *Itinerary* позначає клас екземплярів, пункти відправлення яких (відповідно пункти прибуття) є екземплярами концепту *Location*. Більше того, екземпляри, які належать цьому класу, повинні мати хоча б один пункт відправлення (обмеження $(\geq 1\ \textbf{departurePlace})$) та один пункт прибуття (обмеження $(\geq 1\ \textbf{arrivalPlace})$). Вхідний параметр сервісу *ToTravel* будується з використанням кон'юнкції всіх його вхідних параметрів наступним чином: $\boldsymbol{I}(\textbf{ToTravel}) \equiv \equiv \textbf{Itinerary} \sqcap \textbf{Arrival}$. За допомогою заміни концептів ***Itinerary*** та ***Arrival*** їх описами, ми отримуємо наступний еквівалентний опис:

$$\boldsymbol{I}(\textbf{ToTravel}) \equiv (\geq 1 \textbf{departurePlace}) \sqcap (\forall \textbf{destinationPlace.Location}) \sqcap (\geq 1\ \textbf{arrivalPlace}) \sqcap (\forall \textbf{arrivalPlace.Location}) \sqcap (\geq 1 \textbf{arrivalDate}) \sqcap (\forall \textbf{arrivalDate.Date}) \sqcap (\geq 1 \textbf{arrivalTime}) \sqcap (\forall \textbf{arrivalTime.Time})$$

Входи і виходи інших Web-сервісів обчислюються аналогічним чином. Розглянемо запит на сервіс *Q*, який шукає турпакет, що поєднує у собі подорож з готелем і прокатом автомобілів, враховуючи місце відправлення, місце прибуття, дату відправлення, пункт призначення (готель), а також дати в'їзду та виїзду. Входи і виходи запиту *Q* можуть бути виражені наступними описами:

$$\boldsymbol{I}(Q) \equiv (\geq 1 \textbf{departurePlace}) \sqcap (\forall \textbf{departurePlace.Location}) \sqcap (\geq 1\ \textbf{arrivalPlace}) \sqcap (\forall \textbf{arrivalPlace.Location}) \sqcap (\geq 1 \textbf{departureDate}) \sqcap (\forall \textbf{departureDate.Date}) \sqcap (\geq 1 \textbf{destinationPlace}) \sqcap (\forall \textbf{destinationPlace.Location}) \sqcap (\geq 1 \textbf{checkIn}) \sqcap (\forall \textbf{checkIn.Date}) \sqcap (\geq 1 \textbf{checkOut}) \sqcap (\forall \textbf{checkOut.Date})$$

$$\boldsymbol{O}(Q) \equiv \textbf{TripReservation} \sqcap \textbf{HotelReservation} \sqcap \textbf{CarRental}$$

Відповідність між запитом на сервіс *Q* і трьома оголошеними сервісами, що задані вище, може бути досягнуто шляхом обчислення найкращого покриття профайлу для запиту *Q* з використанням цих сервісів. Вийде наступний результат: найкраще покриття профайлу – *FromTravel*, *Hotel*; Prest – *CarRental*; Pmiss – *departureTime*.

У даному прикладі, існує єдине найкраще покриття профайлу для *Q*, що відповідає дескрипції $E \equiv Hotel \sqcap \sqcap FromTravel$. Вибрані сервіси генерують концепти *TripReservation* і *HotelReservation*, які є частиною вихідних параметрів, які вимагаються запитом *Q*. З описів сервісів ми бачимо, що жоден з Web-сервісів не забезпечує концепт *CarRental*. Отже, найкращі покриття профайлу запиту *Q* матимуть у точності наступний залишок профайлу: $Rest_E(Q) \equiv carRental$. Залишок відповідає вихідному параметру запита, який не може бути згенерований жодним оголошеним сервісом. Більше того, *Pmiss* відображає інформацію (*departureTime*), яка потрібна на вхід вибраним сервісам, але не надається у вхідних параметрах запиту. Точніше, найкраще покриття профайлу запиту *Q* матиме в точності наступну недостатню інформацію: $Pmiss_E(Q) \equiv (\geq 1 departureTime) \sqcap (\forall departureTime.Time)$. Слід зазначити що, хоча рішення $E' \equiv Hotel \sqcap ToTravel$ генерує такі ж вихідні параметри (тобто, концепти *TripReservation* і *HotelReservation*), воно не буде обрано, так як *Pmiss* більше, ніж при першому рішенні (воно містить відношення *arrivalTime* і *arrivalDate*).

Задачі систем виявлення семантичних Web-сервісів

Розглянемо процес виявлення Web-сервісів, який схематично показано на рисунку, та його основні задачі на основі стандартного фреймворку системи виявлення, представленого в роботі [32]. Постачальники сервісів публікують оголошення про Web-сервіси в централізованому сховищі через Web-сайт системи або на власних Web-сайтах. Web-робот (Web crawler) може зібрати всі ці семантичні описи з окремих сайтів публікації і зберігати їх в централізованому сховищі без відома провайдерів. Запитувачі сервісів, що шукають сервіси, які відповідають певним вимогам, представляють свої запити певною мовою. Тут запитувачі сервісів і постачальники є загальними термінами, що використовуються, наприклад, для позначення окремих агентів, які беруть участь у транзакціях. Репозиторій/реєстр сервісів агрегує оголошені сервіси після вирішення проблем неоднорідності (якщо такі є) за допомогою посередника. Коли ініціюється запит на сервіс, то він спочатку порівнюється з сервісами в репозиторії за допомогою механізму метчмейкінгу (matchmaking engine). Після цього отримується набір підібраних сервісів, далі відбувається узгодження сервісів за допомогою взаємодії з провайдерами для доступу до динамічної інформації, якщо така є. Вибір сервісу проводиться з урахуванням критеріїв, яким надає перевагу запитувач, та/або на основі не функціональних (QoS) характеристик сервісу. Нарешті, вибраний сервіс викликається і повертаються результати.

Як бачимо, процес виявлення Web-сервісів охоплює весь спектр завдань від запиту на сервіс до виклику сервісу. Вирішення проблем, пов'язаних з виявленням сервісів, залежить від того, як описати сервіси і як управляти цими описами. Розглянемо основні задачі, які мають бути вирішені при розробці систем виявлення: публікація, зберігання сервісів, створення запитів на сервіс, метчмейкінг сервісу, узгодження і вибір сервісу, інтеграція виявлення і композиції сервісів та інші.

Публікація сервісу. Постачальники сервісів проектують і розробляють сервіси з певною бізнес-метою. Сервіси потім розгортають на серверах, так що вони можуть бути викликані іншими сервісами або агентами. Для того, щоб бути викликаними, вони мають бути описані мовами опису, такими як, наприклад, OWL-S, WSMO, DL. Більшість інструментів у даний час підтримують генерацію WSDL автоматично, коли сервіси розгортаються. Файли WSDL можуть бути анотовані WSDL-S або SAWSDL, щоб вставити семантику. У деяких випадках провайдери можуть також створювати свої власні мови семантичного опису, такі як в DIANE [33]. Модуль публікації надає користувачу програмний інтерфейс для додавання, видалення або

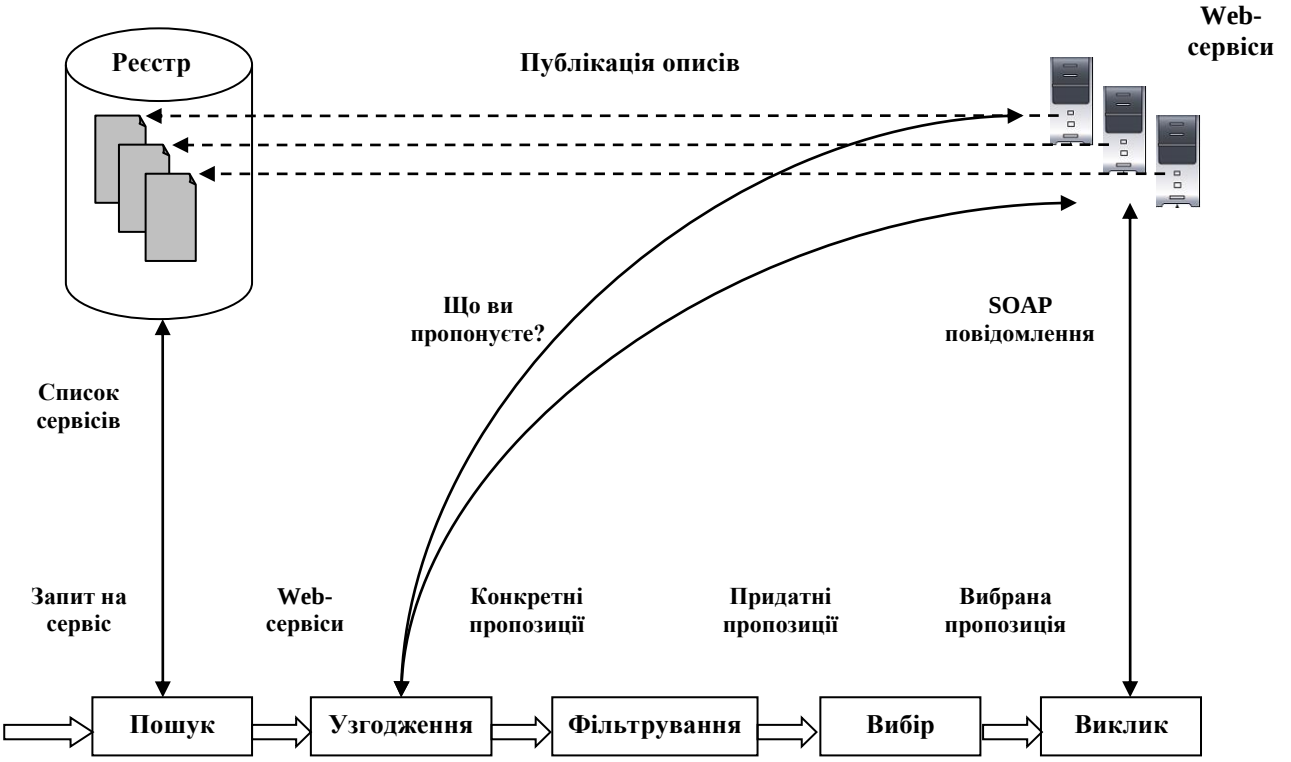

Рисунок. Процес виявлення Web-сервісів

оновлення описів Web-сервісів, а також додавання, видалення або оновлення описів провайдерів. Крім аутентифікації, запит на публікацію, який ініціює процес публікації, може включати в себе наступні параметри: ідентифікатор постачальника сервісу, URL (Uniform Resource Locator), який вказує на документ, що описує сервіс, довільне ім'я сервісу і необов'язковий текстовий опис. Далі, на основі цих вхідних даних відбувається процес, який включає кілька етапів, як це представлено, наприклад, у проекті FUSION [34]. Відбувається розбір документа, що представляє сервіс, витягується семантична інформація про входи і виходи сервісу. Наступним кроком у процесі публікації є відображення отриманої інформації в UDDI-оголошення сервісу. Створюється функціональний профіль оголошення, який далі класифікується щодо усіх відомих функціональних профілів запитів. Мета цієї процедури класифікації полягає у тому, щоб визначити профілі запитів, які нове додане оголошення сервісу може легко задовольнити. Такий підхід збільшує час, необхідний для завершення публікації оголошення про сервіс, але суттєво скорочує час, необхідний для виконання зіставлення під час виявлення. Виразність мови опису надзвичайно впливає на якість процедури виявлення. Крім того повнота описів також впливає на процес зіставлення. Мова опису має забезпечувати також можливість представлення QoS-характеристик сервісу.

В роботах [35, 36] запропоновані підходи для опису характеристик QoS за рахунок розширення поточної моделі UDDI. Ряд інших робіт висвітлюють підходи до використання зібраної QoS-інформації. Так, в роботі [37] пропонують нову мову запитів, що дозволяє шукачам заявити вимоги QoS. Для оцінки подібності між запитом, на основі UML (Unified Modeling Language), і доступними сервісами використовується процес у два етапи. На першому етапі витягуються сервіси з операціями, які задовольняють не функціональні вимоги запиту. Другий крок використовує евристики подібності, заснованої на граф-відповідності, для знаходження операцій, які найкращим чином відповідають запиту. Автори роботи [38] представляють функцію ранжування Web-сервісів відповідно до їх QoS-характеристик, які були зібрані за допомогою брокера. Функція обчислює матрицю, яка представляє собою Web-сервіс у рядку

і кожен окремий параметр QoS в стовпці. У зв'язку з тим, що параметри QoS варіюються в одиницях і величинах, значення нормуються. Потім шукач визначає вектор ваг, що вказує рівень важливості, призначений кожному параметру QoS. Нарешті, ці ваги вводяться в матрицю як фактори і всі значення перераховуються. Рядок, який максимізує суму його зважених параметрів, являє собою перший за рейтингом Web-сервіс і так далі.

Посередник. Описи сервісів і запити можуть бути описані на різних мовах, що призводить до проблем неоднорідності. Система виявлення має підтримувати посередництво між мовами або вирівняти описи до стандартної мови. Крім того, можливе синтаксичне посередництво, яке долає синонімію (слова, які представляють одне і те ж поняття, але синтаксично відрізняються один від одного), долає різні WSDL-стилі для визначення типів даних. Описи семантичних Web-сервісів і запити можуть використовувати різні онтології. Система виявлення має забезпечити посередника між двома або більше онтологіями для їх вирівнювання, або ж зобов'язати постачальників і споживачів погодитися на спільну онтологію.

Зберігання сервісів. Описи сервісів будуть зберігатися в реєстрі сервісів або базі даних. Онтології домену охоплюють термінологію, яку використовують провайдери і замовники для опису сервісів. Потрібна відповідна технологія індексації для розширеної обробки запитів. Різні мови опису можуть вимагати різних методів для індексації та зберігання сервісів, залежно від вимог застосувань. Реєстр може класифікувати сервіси відповідно до їх домену застосування, провайдера сервісів і т. д. Важливо зазначити, що реєстр сервісів не містить фактичних екземплярів сервісів, а тільки прив'язку описів сервісів до фактичних екземплярів сервісів. Структура реєстру має забезпечувати достатньо інформації для процесу виявлення. Задача зберігання сервісів тісно пов'язана з видом архітектури, на яку буде покладена система, тобто централізована або децентралізована. Централізовані системи виявлення покладаються на один єдиний глобальний реєстр, а децентралізовані – на розподілені технології, такі як Peer-to-Peer (P2P), щоб публікувати описи сервісів на різних вузлах. UDDI-реєстр з його синтаксичним встановленням відповідності служить основою для багатьох систем виявлення сервісів. Деякі підходи [39] розширюють модель даних UDDI семантичними описами OWL-S, комбінуючи синтаксичне і семантичне встановлення відповідності (matchmaking).

Децентралізовані архітектури є прийнятною альтернативою для досягнення належного рівня масштабованості, надійності та відмовостійкості. Зокрема, з Peer-to-Peer (P2P) [40] розподіленими архітектурами, всі вузли-учасники передбачають клієнтські і серверні завдання, розділяють свої можливості і використовують накопичені можливості учасників мережі. Теоретично, ідеальна P2P підтримує нескінченну кількість вузлів, кумулятивні можливості також нескінченні, і відмова партнера не впливає на всю мережу. UDDI специфікація визначає інтегрований реєстр як один або більше децентралізованих вузлів UDDI. Через децентралізований характер розподілених архітектур ефективне опитування таких розподілених реєстрів означає більш складні завдання [41].

Запит на сервіс оголошує намір замовника (людина, агент, сервіс) щодо необхідної функціональності сервісу. Це – формальний чи неформальний опис функціональних і технічних можливостей та/або обмежень, які мають бути задоволені виявленим сервісом. Запит визначається на мові, виразність якої дуже впливає на якість процедури виявлення. І запит, і оголошення сервісу, зазвичай, визначаються на мовах опису сервісів з певною виразністю. У системах виявлення сервісів на основі синтаксичних підходів запити формулюють на основі ключових слів і в текстовій формі. Система Woogle [42], наприклад, очікує WSDL-документ як запит.

У системі FUSION [34] функціональний профіль запиту й оголошення на сервіс моделюються за єдиною онтологією. Профіль запиту може бути попередньо зареєстрований і проіндексований в

системі. При появі нового оголошення на сервіс встановлюється відповідність між існуючими профілями запитів.

Різні мови запитів дозволяють досягти різних ступенів точності в результатах пошуку. Результати пошуку, отримані за допомогою семантичних анотацій, якісно кращі, ніж результати, досягнуті підходами за ключовими словами, бо мови запитів на основі семантики дозволяють шукачам точно сформулювати свої потреби. У разі запитів на основі обмежень і на основі семантики, моделювання мети може вимагати від розробників вивчати нову мову запитів.

Так в роботі [43] запропонована мова EAGLE для вираження мети, яку має задовольняти шуканий сервіс. В реальних бізнес-сценаріях обмежень на типи вхідних і вихідних параметрів, шаблони взаємодії і нефункціональні властивості Web-сервісів буває недостатньо. В роботі [44] запропонували мову специфікації мети, яка є комбінацією μ-числення темпоральної логіки та виразної дескриптивної логіки SHIQ(D). У системах на основі WSMO запити описуються в термінах цілей, тобто результатів, очікуваних запитувачем.

Метчмейкінг сервісу (*matchmaking*) – це метод зіставлення запитаного сервісу з оголошеними сервісами. Результатом метчмейкінгу є список оголошених сервісів разом з їх ступенем відповідності (DoM – Degree of Match), який представляє, як підходить оголошений сервіс. Він являється найбільш важливим і складним процесом у всьому процесі виявлення. Більшість підходів метчмейкінгу сервісів використовують атрибути IOPE (Inputs, Outputs, Preconditions and Effects), описані в профілі сервісу. Однак, незалежно від того, на яких елементах застосований відповідний алгоритм, найважливіша проблема метчмейкінгу полягає у тому, що нереально очікувати, що оголошення і запити будуть в ідеальній парі. Поняття DoM має справу з такими проблемами і може бути неформально визначено як значення з упорядкованого набору значень, яке виражає, наскільки два об'єкти є подібними щодо деякої метрики подібності. Такі об'єкти можуть бути сервісами, атрибутами IOPE або певними операціями сервісу. Алгоритм метчмейкінгу сервісу, який обчислює DoM, може використовуватися для того, щоб оцінити виявлені сервіси згідно їх релевантності до сформованого запиту.

Нехай $S=(I_s, O_s)$ – сервіс та $Q=(I_q, O_q)$ – запит на сервіс, де:

- I_s, I_q – кінцеві множини входів оголошення та запиту на сервіс,
- O_s, O_q – кінцеві множини виходів оголошення та запиту на сервіс.

Ступінь відповідності між двома входами або двома виходами залежить від відношення між концептами, які пов'язані з цими входами або виходами, а саме, мінімальною відстанню між цими поняттями у дереві таксономії. Розрізняють наступні ступені відповідності [30]: *Exact, PlugIn, Subsumes, Fail*. *Exact* – точне збігання – можливе, коли $O_q=O_s$, або O_q є підкласом O_s. *PlugIn* – більш слабкий зв'язок, якщо O_s включає O_q. *Subsumes* – якщо O_q включає O_s, то провайдер не повністю задовольняє запит. Це означає, що провайдер частково задовольняє мету, тому запитувачу прийдеться виконати інші запити, щоб досягти своєї мети. *Fail* – невдача, якщо не знайдено будь-якого відношення категоризації між O_q та O_s.

Ми можемо класифікувати підходи метчмейкінгу за різними критеріями. Одна можлива класифікація наступна [45]:

• *прямий*: запит на сервіс узгоджується з одиничними оголошеннями сервісів;

• *непрямий*: запит на сервіс узгоджується з композитними сервісами, які побудовані з простих або складних робочих процесів з оголошень одиночних сервісів.

Крім того, можна було б також класифікувати метчмейкінг як: підтримка DoM чи ні, використання тільки профілю сервісу або іншої інформації про сервіс, а також, оцінка подібності сервісів за допомогою заснованого на логіці міркування або інших методів. Більшість підходів засновані на онтологіях анотації сервісів, виражених у дескриптивних логіках і, таким чином, виконують метчінг на основі DL [45, 46].

Виділяють наступні групи підходів встановлення відповідності:

Семантичний метчінг здібностей. Основна ідея полягає у тому, що «оголошення відповідає запиту, якщо всі виходи запиту відповідають виходам оголошення, і всі входи оголошення відповідають входам запиту». Таким чином, цей метод приймає до уваги тільки входи і виходи профілів сервісів під час метчмейкінгу. Ступінь відповідності між двома виходами або двома входами залежить від відношення між концептами онтології домену, пов'язаними з цими входами і виходами.

Багаторівневий метчінг. Процес метчмейкінгу виконується на багатьох рівнях, тобто, між входами/виходами, категоріями сервісу і іншими параметрами сервісу (наприклад, QoS). Крім того, він представляє поняття *агрегації DoM*, яке може забезпечити більш модульне та ефективне ранжування відповідних сервісів, за допомогою призначення різних ваг різним рівням метчінгу.

DL метчмейкінг на основі профілю сервісу. Метчмейкінг сервісів виконується через онтологію профілю сервісу. В цій онтології кожен сервіс представлено складним DL виразом концепту, який описує всі обмеження сервісу (входи, виходи і т. д.). Таку онтологію можна розглядати як заснований на логіці реєстр оголошень сервісів. Як тільки така онтологія визначена і містить всі доступні оголошення сервісів, запити на сервіси можуть бути виражені через концепти DL і вставлені в неї. Потім може використовуватись механізм міркувань (reasoner) DL, щоб класифікувати запит-концепт в онтології.

Підхід на основі графа. Опис сервісу (запит або оголошення) представляють у вигляді направленого графа (RDF-граф), вершини якого – екземпляри концептів (тобто, індивіди) і дуги – властивості (тобто, ролі концепту), які зв'язують такі екземпляри. Корінь кожного графа – індивід, який представляє оголошення/запит сервісу. Метчмейкінг між двома графами, один – представляє запит на сервіс, а інший – представляє оголошення сервісу, виконується рекурсивним алгоритмом, використовуючи метчінг стандартної категоризації через оператор «subsumes» [47].

Непрямий метчінг на основі графа. Непрямий метчінг відноситься до ідентифікації композицій сервісів, які найбільш релевантні запиту користувача. У випадку простої композиції, ми можемо думати про них як про потоки робіт, подібних графу, які можуть бути скорочені до «ланцюжків сервісів» (тобто, нециклічні шляхи в графі потоку робіт) у самих простих випадках [48]. Основна ідея побудови простих композитних сервісів (тобто, ланцюжка сервісів) базується на наступному:

входи кожного сервісу, включеного в композицію, мають відповідати входам, отриманим із запиту на сервіс, або виходам попереднього сервісу в ланцюжку;

кожен вихід запиту на сервіс має відповідати виходу останнього сервісу в ланцюжку.

Непрямий метчінг на основі зворотної побудови ланцюжка. Зворотна побудова ланцюжка – цільова процедура міркування, подібна способу, яким працює мова Пролог. Основна ідея підходу полягає у тому, що починаючи з сервісів, які відповідають виходам запиту на сервіс, а не його входам, ми рекурсивно пробуємо з'єднати їх з іншими сервісами, поки не знайдемо сервіс з усіма його входами, що відповідають входам даного запиту на сервіс [49].

Як згадувалось вище, реальні Web-сервіси виступають як складні бізнес-процеси і потребують опису у відповідних мовах, як наприклад WS-BPEL, і моделювання їх поведінки з урахуванням станів. Виявлення Web-сервісів на функціональному рівні може не задовольняти користувача, якщо він має вимоги на поведінку сервісу. Щоб здійснювати пошук Web-сервісів на процесному рівні, вони можуть бути описані як системи переходів із стану в стан (state transition system), при цьому вимога пошуку описана в темпоральній логіці. В роботі [50] пропонується алгоритм виявлення сервісів, який поєднує в собі міркування про процедурні поведінки, які мають задовольняти темпоральні умови, а також онтологічні міркування про семантику даних. Встановлення від-

повідності між запитом і сервісом зводиться до проблеми перевірки моделі (model checking).

Узгодження сервісу. Вирішення цієї задачі необхідно після того, як замовник отримує перелік сервісів, які відповідають його запиту. Щоб зробити оптимальний вибір, між провайдером сервісу і замовником можуть знадобитися переговори. Для цього існують різні механізми, починаючи від підходів простого вибору на основі якісних властивостей сервісу (QoS) [51] до складних переговорів або схем аукціону. Укладаються контракти між відповідними бізнес-партнерами і вони формалізуються способом, зрозумілим машині, щоб дозволити автоматичне виконання та моніторинг.

Вибір сервісу являє собою етап визначення того, який сервіс (сервіси) використати для доставки потрібного сервісу. Це остаточне рішення залежить від результатів метчмейкінгу і узгодження.

При наявності декількох Web-сервісів з ідентичною функціональністю, користувачі будуть розрізняти їх на основі QoS, поняття, яке охоплює цілий ряд нефункціональних властивостей таких як: ціна, доступність, надійність і репутація [52]. Ці властивості застосовуються як до атомарних Web-сервісів, так і до композитних Web-сервісів. Для того, щоб міркувати про QoS-властивості Web-сервісів, необхідна модель, яка охоплює описи Web-сервісів з точки зору користувача. Така модель має брати до уваги той факт, що QoS включає у себе кілька розмірностей, і той факт, що QoS композитних сервісів визначається на основні QoS компонентних сервісів. Крім того, варто зазначити, що сервіси зазвичай розповсюджуються через Інтернет і що деякі з їх QoS-характеристик (наприклад, доступність і коефіцієнт успішного виконання) залежать від каналу зв'язку і мають бути оцінені з точки зору запитувача, а не постачальника. Таким чином, за наявності, щонайменше, двох сервісів, призначених для виконання однієї і тієї ж задачі, опис запропонованого QoS стає важливим фактором у розрізненні постачальників сервісів.

Web-робот. Система виявлення Web-сервісів крім можливості публікації сервісів може наповнювати реєстр сервісів за допомогою Web-робота, метою якого є збір семантичних описів Web-сервісів, шляхом відвідування Інтернету, знаходження головних Web-сайтів і розбір описів сервісів. Ці описи можуть бути виражені в деякій мові, наприклад, OWL-S або WSDL-S, або на будь-якій іншій мові розмітки. Очевидно, що якщо видавець сервісу не пропонує семантичний опис Web-сервісу, опубліковані Web-сервіси не будуть збиратися в реєстри семантичних Web-сервісів. Цей Web-робот є прикладом цілеспрямованого шукача: він не просто збирає все, з чим він стикається, він збирає тільки описи семантичних Web-сервісів. Очевидна проблема, яка виникає, це «розсіяність» описів сервісів. Іншими словами, не так багато семантичних описів Web-сервісів в Інтернеті для збору і більшість сторінок, відвіданих Web-роботом, не мають нічого спільного з семантичними Web-сервісами. Ця задача системи виявлення Web-сервісів спонукає до розробки так званих спеціалізованих механізмів пошуку (search engine), спрямованих на виявлення описів Web-сервісів, як семантичних (OWL-S, SAWSDL), так і не семантичних (WSDL) у Інтернеті. Такі пошукові механізми були запропоновані останнім часом: Woogle [53], VUSE [54], SWSE [55], OSSE [56], але немає доступної їх реалізації.

Інтеграція пошуку і композиції Web-сервісів. Механізм виявлення сервісів має включати можливості композиції сервісів. Це робить композицію сервісів частиною виявлення сервісів і навпаки. Задача інтеграції пошуку і композиції виникає тоді, коли єдиний сервіс не може виконати цілісну задачу (запит на сервіс). Запит може бути задоволений композицією сервісів на функціональному рівні, тобто композиція шляхом зіставлення передумов і ефектів сервісів, описаних як атомарні компоненти, які, враховуючи деякі входи, повертають деякі виходи [57, 58].

Тим не менш, побудова композиції не є тривіальним завданням. Треба мати справу з різними умовами, вимогами і ре-

зультатами, отриманими від кожного окремого сервісу. Ця проблема може стати ще більш складною через синтаксичний опис сервісів і необхідність розуміння семантики операцій. Крім того, процес виявлення повторюється під час процесу композиції, тобто, водночас як композиція будується, нові сервіси мають бути виявлені й об'єднані з композицією. Цей процес продовжується до тих пір, поки не буде отримано необхідну функціональність, не буде більше знайдено збігів або досягається критерій зупинки процесу композиції (наприклад, максимальна кількість сервісів у композиції). В роботі [59] запропоновано підхід, який дозволяє автоматичне виявлення і композицію під час обробки запиту. Він характеризується використанням семантичних анотацій на основі SAWSDL, щоб побудувати граф композиції.

Підхід до наскрізної (end-to-end) інтеграції виявлення і композиції, запропонованої у дослідженні [60], працює в два етапи. На першому етапі, тобто в момент виявлення та протягом композиції на функціональному рівні, необхідно визначити набір Web-сервісів, що взаємодіючи один з одним, у змозі відповідати запиту композиції. У центрі уваги є необхідні входи і надавані виходи сервісів, щоб отримати виходи, необхідні користувачу. Наприклад, на цьому рівні ми виявляємо, що сервіс «бронювання готелів» і сервіс «бронювання авіаквитків», необхідні, щоб задовольнити запит на відпустку користувача. На другому етапі, враховуючи набір обраних Web-сервісів і з урахуванням мети композиції, композиція рівня процесу відповідає за генерування автоматично виконуваного композитного Web-сервісу. Наприклад, дано моделі процесів двох доступних Web-сервісів для «бронювання готелів» і «бронювання польоту», ми прагнемо генерувати виконуваний композитний сервіс, скажімо, «віртуальна туристична фірма». Взаємодіючи з сервісами «бронювання готелів» і «бронювання польоту», композитний сервіс бронює готельні номери та місця для польоту відповідно до заданої мети.

Підхід до композиції сервісу функціонального рівня, запропонований в [61], заснований на прямій побудові ланцюжка. Неформально, ідея прямої побудови ланцюжка полягає у тому, щоб ітеративно вибрати можливий сервіс *S* і застосувати його до набору вхідних параметрів, які надаються метою *G* (тобто, всі входи, необхідні для *S*, мають бути доступні). Якщо застосування *S* не вирішує цю проблему (тобто, ще не всі виходи, необхідні для мети *G*, доступні), то нова мета *G′* може бути обчислена з *G* і з виходів, породжених *S*, і в цілому процес повторюється.

Деталізація результатів пошуку системи виявлення. Це завдання полягає у здатності системи виявлення сервісів повертати описи сервісів на рівні описів операцій. Деталізація результатів має важливе значення для виклику виявлених сервісів. Операція визначається як найменша частина програмного забезпечення, яку шукач може викликати за допомогою Web-сервісів. Розглянемо, наприклад, Web-сервіс, який надає класичні операції електронної пошти, такі як перевірка достовірності адреси, фільтрування спаму і відправка анонімних повідомлень. Якщо цей сервіс повертається при пошуку сервісу фільтру спаму, то споживач має проаналізувати опис Web-сервісу, щоб отримати відповідну операцію. В ідеалі, виявлення операцій дозволяє споживачам безпосередньо викликати їх. І навпаки, якщо деталізація результатів є «сервіс», то шукач має розібрати опис знайденого сервісу, щоб проаналізувати його операції. Тим не менш, деякі сервіси представляють впорядковані залежності між їх операціями. Наприклад, платні Web-сервіси часто вимагають виклик операції, яка перевіряє обліковий запис, наприклад, «log-in», перед викликом будь-якої іншої запропонованої операції. У цьому контексті важливо, щоб шукач отримав також весь опис сервісу.

Висновки

Парадигма сервіс-орієнтованого обчислення замінює розробку конкретних програмних компонентів комбінацією виявлення сервісів, відбору та виклику. Оскільки ця парадигма може бути реалізована відповідно до різних масштабів, які варі-

ються від всередині програми до всередині компанії, а також різні рівні автоматизації, то виявлення сервісів створює багато проблем. Ці проблеми ініціюють ряд альтернатив для виявлення сервісів. У цій роботі подана формалізація виявлення Web-сервісів на основі покриття-виведення в контексті дескриптивної логіки. Представлено огляд основних задач і існуючих підходів до їх вирішення при створенні систем виявлення Web-сервісів. Виявлення Web-сервісів залежить від семантичного формалізму, який використовують для опису сервісів. Рівень опису сервісу – функціональний, чи рівень процесу – впливає на методи моделювання сервісів та алгоритми і методи встановлення відповідності при виявленні сервісів. Середовище функціонування системи виявлення потребує різних архітектур побудови систем. Взаємозалежність і різні підходи до вирішення задач систем виявлення Web-сервісів вплинули на появу цілого ряду систем, про що свідчить велика кількість оглядових статей у цьому напрямку. Ця робота дозволить зацікавленим особам не тільки вибрати найбільш підходящий існуючий підхід для виявлення Web-сервісів, а й мотивувати дослідження методів отримання Web-сервісів, які більш прийнятні для виявлення.

1. *Erickson J., & Siau K.* Web Service, Service-Oriented Computing, and Service-Oriented Architecture: Separating hype from reality // Journal of Database Management. – 2008. 19(3). – P. 42–54.
2. *Дерецкий В.* Разработка приложений в сервис-ориентированной архитектуре семантического Веб // Пробдеми програмування. – 2010. – № 1. – С. 66–78.
3. *Li S.-H., Huang S.-M., Yen D. C., & Chang C. C.* Migrating legacy information systems to Web Services architecture // Journal of Database Management, 2007. – 18(4). – P. 1–25.
4. *Song H., Cheng D., Messer A., & Kalasapur S.* Web Service discovery using general-purpose search engines // In Proceedings of the IEEE International Conference on Web Services. – 2007 – P. 265–271.
5. *Garofalakis J.D., Panagis Y., Sakkopoulos E., & Tsakalidis, A.K.* Contemporary Web Service Discovery Mechanisms // Journal of Web Engineering. – 2006. – 5(3). – P. 265 – 290.
6. *Paolucci M., & Sycara K.* Autonomous semantic Web Services // IEEE Internet Computing. – 2003. – 7(5). – P. 34–41.
7. *http*://help.eclipse.org/juno/topic/org.eclipse.jst.ws.doc.user/concepts/cwsil.html?cp=66_3_0_0_3
8. *Xia Wang and Wolfgang A. Halang.* Discovery and Selection of Semantic Web Services, volume 453, 2013. Springer-Verlag Berlin Heidelberg, studies in computational intelligence edition, 2013.
9. *WS-Gloss*: http://www.w3.org/tr/ws-gloss/, 2004.
10. *Yu Q., Liu X., Bouguettaya A., & Medjahed B.* Deploying and managing Web Services: issues, solutions, and directions // The International Journal on Very Large Data Bases. – 2008. – 17(3). – P. 537–572.
11. *Studer R., Grimm S., and Abecker A.* Semantic Web Services: Concepts, Technologies, and Applications. Springer, 2007.
12. *The* OWL Services Coalition. OWL-S: Semantic Markup forWeb Services. In Technical White paper (OWL-S version 1.0), 2003.
13. *http*://www.w3.org/Submission/WSMO/.
14. *WSDL*, http://www.w3.org/TR/wsdl.
15. *UDDI*, http://uddi.xml.org/.
16. *NAICS*, http://www.naics.com.
17. *jUDDI*, http://ws.apache.org/juddi/.
18. *http*://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf
19. *Wang S., Zhang L., & Ma N.* A quantitative measurement for reputation of Web Service and providers based on cloud model // In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation. Los Alamitos, CA: IEEE Computer Society. – 2008. – P. 500–505.
20. *Martin D., Burstein M., Mcdermott D., Mcilraith S., Paolucci M., & Sycara K.* Bringing semantics to Web Services with OWL-S. World Wide Web (Bussum). – 2007. 10(3). – P. 243–277.
21. *Sivashanmugam K., Verma K., Sheth A. P., & Miller J. A.* Adding semantics to Web Services standards. In Proceedings of the 2003 International Conference on Web Services, Las Vegas, NV CSREA Press. – 2003. – P. 395–401.
22. *Lausen H, et al.* WSML – a language framework for semantic web services. In:

Position paper for the W3C rules workshop. Washington DC, USA. – 2005.
23. *Sapkota B., et al* D21.v0.1 WSMX Triple-Space Computing. In: Sapkota B., Martin-Recuerda F. (eds) WSMO working draft. – 2005.
24. *S. Staab, R. Studer.* Handbook on Ontologies. Second edition.
25. *E. by F. Baader, D. McGuinness, D. Nardi, and P. F. Patel-Schneider.* Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2002.
26. *Teege G.* Making the difference: a subtraction operation for description logics. In: Doyle J, Sandewall E, Torasso P(eds) Proceedings of KR'94, location, day month 1994. Morgan Kaufmann, San Francisco. – 1994.
27. *Baader F., KËusters R., Molitor R.* Computing least common subsumer in description logics with existential restrictions. In: Dean T (ed) Proceedings of the 16th international joint conference on AI, Stockholm, Sweden, 31 July–6 August 1999, P. 96–103.
28. *Benatallah B., Hacid M.-S. et al.* On automating web services discovery // VLDB J. – 2005. 14(1). – P. 84–96.
29. *Benatallah B., Hacid M-S, Rey C., Toumani F.* Request rewriting-based Web service discovery. In: Fensel D, Sycara K, Mylopoulos J (eds) Proceedings of the international Semantic Web conference (ISWC 2003), Sanibel Island, FL, October 2003. Lecture notes in computer science, vol 2870. Springer, Berlin Heidelberg NewYork, P. 242–257.
30. *Paolucci M., Kawamura T., Payne T.R., & Sycara K.P.* Semantic matching of Web services capabilities. In I. Horrocks & J. Hendler (Eds.), First International Semantic Web Conference on the Semantic Web, Sardinia. Springer-Verlag. – 2002. – P. 333–347.
31. *Benatallah B., Hacid M.-S., Rey C., Toumani F.* Semantic reasoning for Web services discovery. In: Proceedings of the WWW workshop on e-services and the SemanticWeb, Budapest, Hungary, May 2003.
32. *Le Duy Ngan, Rajaraman Kanagasabai* Semantic Web service discovery: state-of-the-art and research challenges, Pers Ubiquit Comput. Springer-Verlag. – 2013. – 17. – P. 1741–1752.
33. *Kuster U. et al* DIANE: a matchmaking-centered framework for automated service discovery, composition, binding, and invocation on the web // Int J Electron Commer. – 12(2). – P. 41–68.
34. *http*://www.seerc.org/fusion/semanticregistry.
35. *Ran S.* A model for Web Service discovery with QoS. SIGecom Exchanges. 4(1). – P. 1–10.
36. *Overhage S., & Thomas P.* Ws-specification: Specifying Web Services using uddi improvements // In Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems, London, P. 100–119.
37. *Kozlenkov A., Spanoudakis G., Zisman A., Fasoulas V., & Sanchez Cid, F.* Architecture-driven service discovery for service centric systems // International Journal of Web Services Research. – 2007. – 4(2). P. 82–113.
38. *Al-Masri E., & Mahmoud Q.H.* Qos-based discovery and ranking of Web Services // In Proceedings of the International Conference on Computer Communications and Networks Los Alamitos. – 2007. – P. 529–534.
39. *Q. Qiu Q. Xiong Y. Yang and F. Luo,* "Study on ontology-based web service discovery" // In IEEE Internationl Conference on Computer Supported Cooperative Work in Design. – 2007. – Vol. 11, P. 641–645.
40. *Gotthelf P., Zunino A., & Campo M. A.* Peer-To-Peer communication infrastructure for groupware applications // International Journal of Cooperative Information Systems. – 2008. – 17(4). – P. 523–554.
41. *Firat A., Wu L., & Madnick S.* General strategy for querying web sources in a data federation environment // Journal of Database Management. – 2009. – 20. P. 1–18.
42. *Dong Z., Halevy A. Y., Madhavan J., Nemes E., & Zhang J.* Similarity search for Web Services // In Proceedings of the Thirtieth International Conference on VeryLarge Data Bases, Toronto, ON, Canada. – 2004. – P. 372–383.
43. *U. Dal Lago M. Pistore and P.Traverso.* Planning with a Language for Extended Goals // In Proc. AAI'02, 2002.
44. *Sudhir Agarwal.* A Goal Specification Language for Automated Discovery and Composition of Web Services. In Tsau Young Lin, Laura Haas, Janusz Kacprzyk, Rajeev Motwani, Andrei Broder, Howard Po, International Conference on Web Intelligence (WI `07), Silicon Valley, California, USA, November, 2007. P. 528–534.
45. *Tsetsos V., Anagnostopoulos C., and Hadjiefthymiades S.* Semantic web service discovery: methods, algorithms and tools. In:

Cardoso, J. (ed.) Semantic Web Services: Theory, Tools and Applications. Information Science Reference, Hershey, PA. – 2007.
46. *Baader F., Calvanese D., McGiuness D., Nardi D., & Patel-Schneider P.* The description logic handbook: Theory, implementation, and applications. Cambridge: Cambridge University Press. – 2003.
47. *Trastour D., Bartolini C., & Gonzalez-Castillo J.* A Semantic Web approach to service description for matchmaking of services // Paper presented at the Semantic Web Working Symposium, Stanford, California. –2001.
48. *Giv R.D., Kalali B., Zhang S., & Zhong N.* Algorithms for direct and indirect dynamic matching of Web services (Tech. Rep.). Waterloo, Ontario, Canada: University of Waterloo, School of Computer Science. – 2004.
49. *Brachman R., & Levesque H.* Knowledge representation and reasoning. San Francisco: Morgan Kaufmann. – 2004.
50. *Pagliarecci F., Pistore M., Spalazzi L., Traverso P.* Web service discovery at process-level based on semantic annotation. In Proceedings of the Fifteenth Italian Symposium on Advanced Database Systems (SEBD 2007), 17–20 June, Torre Canne, BR, Italy, 2007.
51. *Hung PCK, Zhang L-J.* Behind the scenes of web services negotiation and agreement. Int J Web Serv Res (JWSR). – 2004. – 1(2). – P. 37–57.
52. *O'Sullivan J., Edmond D., and Hofstede A.t.*, "What's in a Service?" Distributed and Parallel Databases, 2002. – Vol. 12, nos. 2-3. – P. 117–133.
53. *Dong X., Halevy A., Madhavan J., Nemes E., Zhang J.* Similarity search for web services, Proceedings of VLDB, 2004.
54. *Platzer C., Dustdar S.* A Vector Space Search Engine for Web Services // In Proceedings of the 3rd European IEEE Conference on Web Services, 2005.
55. *Ma C., Song M., Xu K., Zhang X.*, Web Service discovery research and implementation based on semantic search engine // 2nd Symposium on Web Society, 2010.
56. *Dan G.* New ideas for Web service discovery-ontology-based prototype system of service search engine, 2nd International Conference on Software Technology and Engineering, 2010.
57. *Paolucci M., Sycara K., Kawamura T.* Delivering Semantic Web Services. In Proc. WWW2003, 2002.
58. *I. Constantinescu, B. Faltings, W. Binder.* Typed Based Service Composition. In Proc. WWW2004, 2004.
59. *Hobold G.C., Siqueira F.* Discovery of semantic web services compositions based on SAWSDL annotations. In: IEEE 19th international conference on web services. Hawaii, USA. – 2012.
60. *http*://knowledgeweb.semanticweb.org/semanticportal/deliverables/D2.4.6.1v2.pdf
61. *Ruben Lara.* Definition of semantics for web service discovery and composition. Knowledge Web Deliverable D2.4.2, 2004.

Одержано 07.05.2015

Про автора:

Ремарович Світлана Станіславівна,
науковий співробітник.

Місце роботи автора:

Інститут програмних
систем НАН України,
Проспект Академіка Глушкова, 40.
Тел.: 526 6249.
E-mail: svitlana.remarovich@gmail.com