УДК 531.383

С.Ф. Теленик, О.А. Амонс, К.В. Єфремов, В.Т. Лиско

ЛОГІЧНИЙ ПІДХІД ДО ІНТЕГРАЦІЇ ПРОГРАМНИХ ЗАСТОСУВАНЬ ПІДТРИМКИ МІЖДИСЦИПЛІНАРНИХ НАУКОВИХ ДОСЛІДЖЕНЬ

The paper proposes an approach to the solution of software applications integration problem dealing with the support of interdisciplinary research on the example of the functioning of World Data System using the apparatus of mathematical logic and the theory of artificial intelligence. The authors proposed the overall architecture of the system integration, a formal language for description of existing applications and their functionalities. The inference rules were formulated that define the implementation circuit of application methods and data exchange in the users' tasks solution process and the algorithm of the mechanism work dealing with the building of the users' tasks tree solution based on the received output was designed. The proposed approach and the developed models and algorithms are illustrated with the help of the life safety component calculation task and indication of the threat critical values for the analysis of Ukrainian regions sustainable development, and this analysis is carried out annually by the World Data Center for Geoinformatics and Sustainable Development on behalf of the Committee on the systematic analysis of the Ukrainian NAS Presidium.

Вступ

В умовах глобалізації економіки і суспільного життя українська наука повинна інтегруватися в світові і європейські структури, які сприяють консолідації досліджень і тим самим – розвитку наукової діяльності [1]. І це дуже важливий процес, оскільки виникають нагальні потреби у міждисциплінарних наукових дослідженнях, насамперед забезпечення сталого розвитку в глобальному і регіональному контекстах [2]. Але для ефективної реалізації міждисциплінарних наукових досліджень необхідно створити належні умови для інформаційного обміну в процесі розв'язання наукових задач. Науково-технічний прогрес, який свого часу сприяв появі інформаційно-комунікаційних технологій (ІКТ), сьогодні значно пришвидшується завдяки їм. Вони стрімко розвиваються, охоплюючи нові сфери діяльності людини і збільшуючи продуктивність праці. Але раціонально використовувати ІКТ можуть лише фахівці у цій галузі, тоді як потреби ефективного інформаційного обміну в міждисциплінарних дослідженнях можна задовольнити лише за рахунок раціонального використання ІКТ фахівцями, які досконало знають свою предметну сферу [3].

Розвиток галузі інформаційних технологій (ІТ), яка переживає якісні зміни, пов'язані з усталенням ІКТ, створив умови для розподілених обчислень, ефективного використання інформаційних та інших ресурсів. Консолідація ресурсів, впровадження технологій віртуалізації сприяють процесу заміни локальних рішень розподіленими, які дають змогу комплексно використовувати всі об'єднані в глобальну мережу обчислювальні потужності, системи збереження даних і забезпечують доступ до накопичених інформаційних ресурсів. Сервісний підхід, сформований на послугах зв'язку, поширився на інфраструктуру, засоби розроблення, програмні застосування. Поява широкого спектра нових видів сервісів, насамперед контенту, призвела до конвергенції сервісів і формування узагальненого поняття інформаційно-комунікаційних сервісів (ІКС). Швидко розширився склад провайдерів ІКС, з'явилися конвергентні провайдери. Можливості сервісів, які сьогодні надаються провайдерами, високий рівень їх якості за помірної ціни дає змогу підприємствам і організаціям відмовитися від розвитку власної ІТ-інфраструктури і перейти до компонентно-базованого проектування власних інформаційно-телекомунікаційних систем (ІТС) з широким вибором наявних ІКС [4].

Однак ефективне використання переваг розподілених систем і можливостей сервіс-орієнтованих технологій вимагає забезпечення уніфікованого доступу до сервісів. У той же час формування нового демократичного ІТ-середовища, в якому надавати сервіси можуть навіть невеликі підприємства, природно супроводжувалося використанням різноманітних засобів і технологій доступу [5]. Тому історично сформоване ІТ-середовище є гетерогенним, що певною мірою ускладнює доступ користувачів до його ресурсів або, щонайменше, робить його некомфортним.

Зокрема, така ситуація характерна для наукової діяльності. Так, створена в 1956 р. під егідою Міжнародної наукової ради (МНР,

International Council for Science – ICSU) система Світових центрів даних (СЦД) забезпечує збір, збереження, поширення й аналіз даних, отриманих у багатьох галузях науки [1]. За час свого існування система СЦД накопичила багато даних і застосувань, які можна використовувати для вирішення актуальних проблем розвитку суспільства. Вони становлять один із найпотужніших інформаційних ресурсів, яким користуються сотні тисяч науковців, і потреба в ньому зростає в міру того, як зростає потреба в міждисциплінарних дослідженнях для забезпечення сталого розвитку, вирішення нагальних проблем протидії забрудненню природного середовища тощо. Але ефективному використанню цих ресурсів перешкоджають труднощі у взаємодії успадкованих застосувань, зумовлені відмінністю архітектур, різноманіттям форматів представлення даних та іншими чинниками.

У галузі ІКТ розроблюються і поступово реалізуються перспективні архітектурні рішення, наприклад Next Generation Network (NGN) і Next Generation Service Overlay Network (NGSON) [6], які забезпечують взаємодію різноманітних технологій транспортного рівня. Але виникає потреба в комплексній інтеграції ресурсів із різних джерел, і зусилля розробників ІКТ повинні спрямовуватися на те, щоб учені різних галузей знань могли використовувати накопичені ресурси, орієнтуючись на знання предметної сфери, а не на особливості ІТ. До кола джерел, інтегрований доступ до ресурсів яких доцільно забезпечити, належать бази даних, веб-сайти і портали, різноманітні успадковані файлові системи і репозиторії структурованих за різними моделями даних.

Зазвичай інтеграцію даних в інформаційних системах розуміють як забезпечення єдиного уніфікованого інтерфейсу для доступу до певної сукупності, в загальному випадку, неоднорідних незалежних джерел даних [7]. Для користувача інформаційні ресурси всієї сукупності джерел, які інтегруються, повинні представлятися як нове єдине джерело, а система, яка забезпечує такі можливості, називається системою інтеграції даних [5]. Існує кілька підходів до інтеграції даних, серед яких є успішно реалізовані й досить поширені в реальних системах. Їх аналітичний огляд наведено у працях [5, 8]. Останнім часом стійко зміцнює позиції концепція простору даних, яку розглядають як нову абстракцію керування даними. В Україні вона отримала розвиток у працях Н.Б. Шаховської, наприклад [9].

Погляд на інтеграцію як взаємодію застосувань не такий поширений, але дуже важливий з точки зору вирішення комплексної проблеми інтеграції в новому ІТ-середовищі. Загалом він формувався в середовищі фахівців з автоматизації програмування [10], особливо у зв'язку з впровадженням перспективних технологій промислового виготовлення програмного продукту, таких як "збірне програмування" [11]. Фокус боротьби з ускладненням програмних систем перенесли з розроблення гнучких настроюваних і масштабованих структур, наприклад пакетів прикладних програм, на розроблення засобів взаємодії виокремлених програмних систем.

Систематичне дослідження власне інтеграції застосувань розпочалося на зламі тисячоліть, коли різко постала проблема інтеграції численних бізнес-застосувань, насамперед класів CRM, ERP, SSM та ін. У нових умовах господарювання бізнес ставить перед ІТ нові задачі, розв'язання яких за рахунок придбання нових і модернізації наявних бізнес-застосувань уже не забезпечує повернення інвестицій. Тоді й постала потреба в побудові інформаційних систем підприємства на основі інтегрованого комплексу бізнес-застосувань і з'явилися перші спроби формалізованого описання бізнес-застосувань і їх взаємодії для розв'язання задач бізнес-підрозділів. Так з'явилися технології Component Object Model (COM) і Common Object Request Broker Architecture (CORBA), які забезпечували взаємодію програм на одному комп'ютері, а потім ці можливості були перенесені на взаємодію програмних систем, які базувалися на різних комп'ютерах. Реалізовані в технологіях Distributed COM (DCOM), Electronic Access Interchange (EAI) і CORBA зазначені можливості забезпечували інтеграцію виокремлених програмних систем для вирішення більш складних проблем користувачів.

Ще одним важливим для інтеграції застосувань рішенням була сервіс-орієнтована архітектура – Service-oriented Architecture (SOA), поява якої поступово через проблематику інтеграції мультибаз і федеративних баз даних, великих сховищ даних, різноманітних репозиторіїв сприяла перенесенню концепції інтеграції на веб-застосування [12]. Але повна відмова ІТ-підрозділів від вирішення проблем ефективності за рахунок вбудовування бізнес-застосувань із мінімальними втратами в існуючу ІТ-інфраструктуру стала можливою лише після кризи 2008 року з появою і впровадженням тех-

нологій хмарних обчислень. Саме тоді стала зрозумілою необхідність чіткого бачення IT-інфраструктури всієї інформаційної системи підприємства і засобів її побудови, експлуатації і розвитку з урахуванням нових можливостей IT-середовища, яке бурхливо формувалося на засадах консолідації, віртуалізації та мультисервісного обслуговування.

Лідери галузі запропонували рішення на основі інтеграційних платформ, які забезпечували повну функціональність за рахунок розвинених засобів масштабування і настроювання. Але такий централізований підхід вимагає відповідної IT-інфраструктури, і не всі підприємства можуть його дозволити. Крім того, з'являється певна залежність підприємства від компаній-виробників таких інтеграційних рішень.

Тому для більшості підприємств більш прийнятним є підхід до інтеграції, пов'язаний із широким використанням бізнес-застосувань різних розробників за допомогою програм-посередників, наприклад програмних систем класу Message Oriented Middleware (MOM), із застосуванням сучасних підходів до здешевлення володіння IT-інфраструктурою, насамперед засобів використання успадкованих застосувань. Перешкодою на шляху до реалізації децентралізованого підходу є відмінності в архітектурних компонентах бізнес-застосувань від різних виробників, насамперед у моделях даних і бізнес-процесів, технологіях побудови програмної системи та її базових елементах (серверах системи керування базами даних і застосувань), засобах взаємодії компонентів, доступу користувачів тощо.

В умовах хмарних обчислень сервісне обслуговування уможливлює надання програмних сервісів визначеного рівня якості за прийнятну ціну і тим самим обґрунтовує доцільність програмних (SaaS), інфраструктурних (IaaS) і платформних (PaaS) сервісів [13]. Сьогодні цей напрям можна розглядати як теоретико-методологічну основу для перенесення сервісного підходу до створення інформаційних систем і широкого впровадження компонентно-базованого їх розроблення.

Щоб успадковані застосування стали складовою системи на основі сервісного обслуговування, необхідно певним чином узгодити їх моделі даних і бізнес-процесів із прийнятими в новій системі, а також забезпечити взаємодію з базовими елементами програмної системи. Тому інтеграція даних в інформаційних системах та інтеграція як взаємодія застосувань мають комплексуватися. Виникає потреба розроблення технічного рішення, яке дасть можливість:

- створювати механізми ефективного управління інформаційними ресурсами;
- розширювати сфери стандартизації протоколів і форматів обміну даними на основі взаємодії відкритих систем;
- створювати крос-платформні застосування, інтегрувати дані та застосування;
- спрощувати розгортання і розвиток систем за рахунок приховування внутрішніх деталей апаратного і програмного забезпечення.

Для створення рішень, які підтримують зазначені можливості, необхідно вирішити такі проблеми:

1) розробити формальні моделі й методи подання метаданих для описання джерел, їх функціональних можливостей, особливостей доступу до них;

2) розробити моделі й методи автоматизованого проектування та реалізації застосувань з єдиною архітектурою, здатних до ефективної взаємодії для реалізації задач міждисциплінарних досліджень, і створити на їх основі платформні рішення за моделлю PaaS;

3) розробити моделі, методи й засоби реалізації взаємодії застосувань на семантичному рівні;

4) розробити моделі й методи підтримки формування користувачами запитів на необхідні дані в зручних для них термінах предметної сфери (сфер) досліджень;

5) привести IT-інфраструктуру системи СЦД у відповідність до потреб на засадах консолідації, віртуалізації, мультисервісного обслуговування на основі моделей і технологій хмарних обчислень;

6) розробити моделі, методи й засоби ефективного управління IT-інфраструктурою системи СЦД, здатні забезпечити визначений рівень якості інформаційного обслуговування.

Однією з найважливіших залишається проблема розроблення математичних моделей, методів і засобів інтеграції різноманітних застосувань, як успадкованих на основі традиційних технологій, так і клієнт-серверних і веб-орієнтованих технологій. У статті пропонується підхід до інтеграції прикладних застосувань із використанням апарату математичної логіки і теорії штучного інтелекту для міждисциплінарних досліджень на прикладі функціонування системи СЦД. Специфіка інтеграції застосувань для міждисциплінарних наукових досліджень полягає в тому, що не вимагається узгод-

ження моделей бізнес-процесів, які по суті замінюються схемами розв'язання задач користувачів. У загальному ж випадку інтеграція застосувань здійснюється на основі погоджених моделей бізнес-процесів, для підтримки яких і призначений інтегрований комплекс бізнес-застосувань.

Постановка задачі інтеграції прикладних застосувань для міждисциплінарних досліджень

Необхідно створити математичні моделі і методи як основу цілісного рішення для забезпечення центрів збереження і оброблення даних, яке надасть гнучкі можливості на рівні інтеграції інформації та доступності сервісів для користувачів. Зазначені моделі й методи повинні бути позбавлені недоліків алгоритмічного підходу, пов'язаних із необхідністю перепрограмування алгоритмів оброблення даних при появі нових типів даних і зміні реалізованих чи появі нових алгоритмів. На рівні оброблення даних таке рішення повинне забезпечити:

- розподіленість обчислень і використання віддалених машинних ресурсів;
- гнучкість і адаптацію до навантаження на систему;
- простоту в використанні системи;
- надання даних за допомогою віддаленого клієнта або сервісу;
- доступність засобів інтелектуального оброблення даних безпосередньо кінцевому користувачу без спеціальних знань і навичок;
- швидку й просту інтеграцію даних і застосувань різноманітних глобальних інформаційних систем.

Особливості інтеграції застосувань

На сьогодні існує понад 50 СЦД, які за більш ніж 50 років вибудували систему накопичення, аналізу, оброблення та міжнародного обміну даними. В потужних системах збереження даних СЦД накопичені величезні обсяги астрономічних, геофізичних та інших наукових даних. На серверах СЦД здійснюється оброблення цих даних, для чого застосовується багато різноманітних прикладних застосувань, побудованих на різних технологіях.

Безумовно, з точки зору вчених, які працюють над вирішенням різноманітних ресурсовимогливих проблем, важливо мати доступ не до великої наданої системи можливостей, а до цілісної сукупності джерел даних і застосувань, налаштованих на задоволення їх конкретних потреб з дружнім інтерфейсом, який не вимагає спеціальних знань у галузі ІТ. Однак система накопичення, аналізу, оброблення і міжнародного обміну даними СЦД не забезпечує такого доступу. Крім того, вона не була розрахована на зростання рівня вимог наукової спільноти і є недостатньо гнучкою для використання в міждисциплінарних дослідженнях. Тому в 2008 р. була створена нова міждисциплінарна структура – Світова система даних (ССД, World Data System – WDS) для напрацювання і впровадження нового, координованого глобального підходу до наукових даних, який гарантує універсальний рівноправний доступ до якісних даних для досліджень, освіти і прийняття рішень. Новій структурі доведеться вирішувати накопичені проблеми, насамперед уніфікації форматів і протоколів передачі даних, забезпечення зручного доступу до даних, організації контролю якості наукових даних [1].

Труднощі, які при цьому виникають, пов'язані з тим, що існуючі СЦД накопичують і пропонують неоднорідні дані, визначені специфікою відповідної галузі науки і країни. Можливості телекомунікаційних мереж постійно зростають, як і можливості сучасних комп'ютерів і сховищ даних, відкриваючи дорогу новим технологіям розподілених обчислень – Grid-системам і хмарним обчисленням. Об'єднання раніше виокремлених систем збору, збереження і оброблення даних СЦД на новій технологічній інтеграційній базі забезпечить значне збільшення їх сумарної ефективності. Створення такої системи надасть ученим зручний централізований доступ до раніше розрізнених ресурсів, одночасно полегшуючи і пришвидшуючи науково-дослідну роботу по всьому світу.

Найнагальнішою із проблем, які вимагають ефективних рішень, є проблема інтеграції. Сьогодні в багатьох СЦД використовуються успадковані технології, побудовані на програмних системах із жорсткою структурою. Більш нові рішення побудовані на клієнт-серверних і веб-орієнтованих технологіях. Сервісний підхід до оброблення й аналізу даних не набув необхідного поширення. Іноді відсутня можливість підключення геоінформаційних систем і навіть найпростіших загальнодоступних сервісів типу Google Maps. Якщо до цього додати відсутність єдиного стандарту, то необхідність інтеграції джерел даних, у якій забезпечуються інтеграція застосувань й інтеграція баз даних, можна вважати достатньо обґрунтованою.

При цьому формування сучасного IT-середовища передбачає інтеграцію у таких зрізах:

технологічний: інтегровані дані та сервіси мають базуватися на єдиному стеку протоколів взаємодії одного з іншим, використовувати спільні формати передачі даних для того, щоб уможливити інтеграцію;

змістовий: повинен підтримуватися єдиний формат описання даних у семантичній структурі джерел даних;

функціональний: використання різнорідних сервісів у єдиній структурі для розв'язання поставлених користувачем задач.

Огляд існуючих рішень з інтеграції прикладних застосувань

Існуючі рішення базуються на технологіях створення, функціонування й розвитку розподілених систем. Незважаючи на відмінності різних технологій створення розподілених сервіс-орієнтованих систем, загальні принципи і теоретико-методологічні підходи завжди залишаються схожими. Необхідно внести до реєстру, описати різнорідні сервіси та забезпечити можливість прозорого спілкування з клієнтами і між собою. Крім того, для мережної взаємодії необхідно визначитися із протоколами усіх рівнів моделі OSI. Для цього можна використовувати структуру корпоративного стандарту ІТС з операціями інтелектуалізації сервісів, запропоновану в праці [14] на основі міжнародних, державних і галузевих стандартів та корпоративних документів, насамперед [15]. Ця структура складається із двох частин, перша з яких формується із чотирьох рівнів стандартів стека протоколів TCP/IP, а другу становлять застосування користувачів відповідно до класу ІТС, призначення й операцій інтелектуалізації сервісів. На рис. 1 наведений варіант структури корпоративного стандарту.

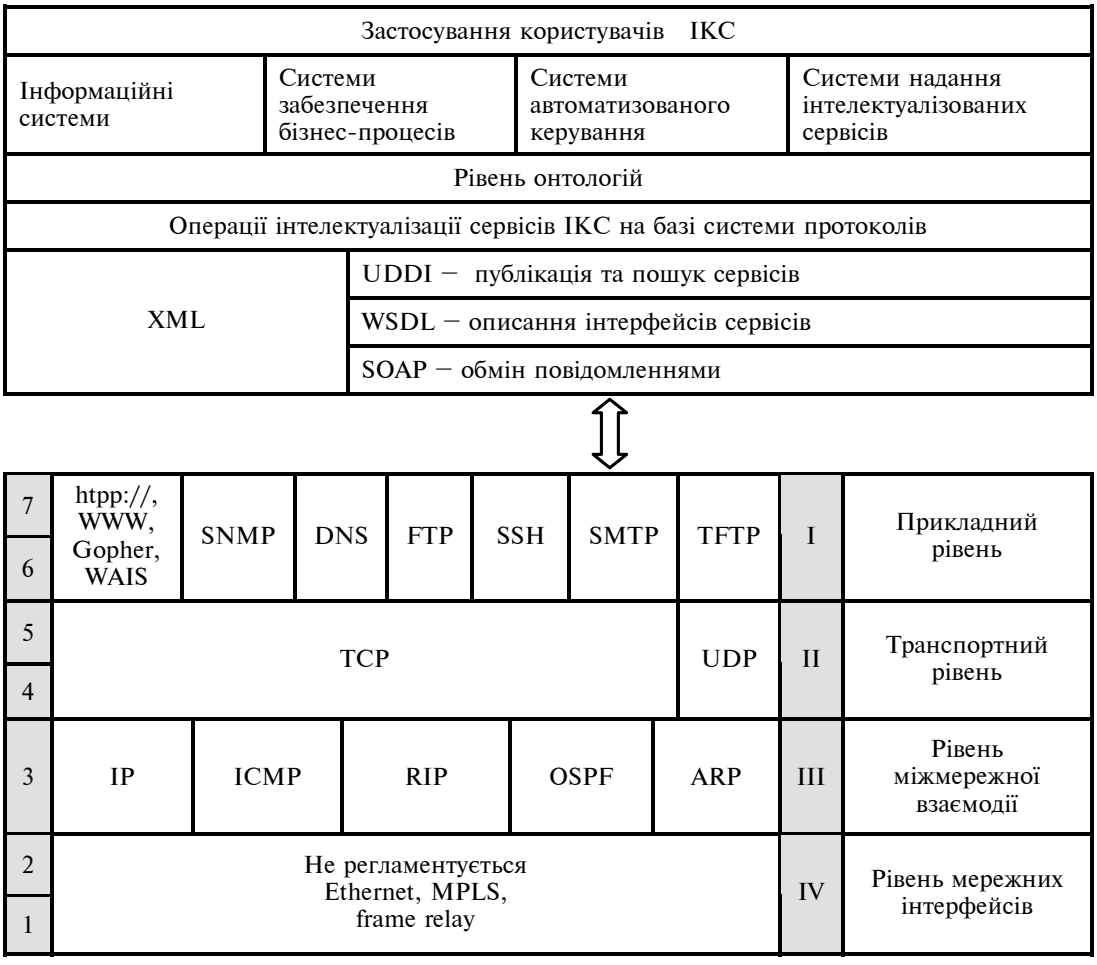

Рис. 1. Стек протоколів для організації взаємодії сервісів

На рис. 1 для зручності наведені рівні міжнародного стандарту взаємодії відкритих систем OSI і їх співвідношення з відповідними рівнями стека протоколів TCP/IP. Використана класифікація ІТС запропонована у праці [14]. Вона дає змогу систематизувати найрізноманітніші ІКС за рівнем вимог і запитів користувачів, їх практичних потреб і професійної підготовки, не накладаючи ніяких обмежень на функціональність ІТС, їх атрибутів і операцій. Введені зміни стосуються використовуваних протоколів першої частини структури і деталізації протоколів, на базі яких здійснюється інтелектуалізована взаємодія застосувань у процесі розв'язання більш складних задач користувачів.

Для ведення реєстру сервісів використовується технологія Universal Description, Discovery and Integration (UDDI) [16], яка забезпечує можливість публікації інформації щодо сервісу і пошуку потрібного сервісу як людиною, так і програмою-клієнтом.

Для уніфікованого описання сервісів, що дає можливість використовувати їх незалежно від мови програмування, застосовується Web Service Description Language (WSDL), який, згідно з визначенням W3C, становить собою формат XML для описання мережних сервісів як набору операцій, що працюють за допомогою повідомлень із документно- або процедурно-орієнтованою інформацією [17]. Документи WSDL, по суті утворюючи уніфікований прошарок, який уможливлює використання сервісів, створених на базі різних платформ, описують інтерфейс сервісу, URL, механізми комунікації, які "розуміє" сервіс, методи, які він надає з відповідними параметрами (тип, назва, розміщення Listener'а сервісу), структуру повідомлень сервісу. Зазначимо, що більшість платформ підтримують формування WSDL-описів сервісів, надаючи розробникам можливість працювати з сервісами як зі звичайними класами.

Для реалізації ключової частини взаємодії – обміну повідомленнями, можна використовувати одну з поширених технологій, обґрунтовуючи вибір відповідністю вимогам системи накопичення, оброблення й обміну даними:

SOAP — проста й зручна у використанні разом із Business Process Execution Language (BPEL) технологія, яка забезпечує взаємодію розподілених систем, незалежно від об'єктної моделі, операційної системи або мови програмування. Дані передаються у вигляді XML-документів особливого формату [18];

CORBA — створений для підтримки розроблення і розгортання складних об'єктно-орієнтованих прикладних систем механізм для інтеграції ізольованих систем, який надає можливість програмам на різних мовах програмування, які виконуються в різних вузлах мережі, взаємодіяти одна з іншою так само просто, немов вони перебувають в адресному просторі одного процесу. Це досягається за рахунок уніфікованого конструювання їх інтерфейсів за допомогою спеціальної декларативної мови Interface Definition Language (IDL). При цьому інтерфейс і його опис також не залежить від операційної системи чи архітектури процесора;

REST (Representational State Transfer) – це стиль архітектури програмного забезпечення для розподілених систем, який використовується для побудови веб-служб. Кожна одиниця інформації однозначно визначається глобальним ідентифікатором URL у строго заданому форматі. Дані передаються без будь-яких прошарків;

WCF (Windows Communication Foundation) – це платформа Майкрософт, призначена для створення застосувань, які обмінюються даними по мережі й не залежать від стилю та протоколу [19].

І хоч сьогодні найбільш прийнятною видається SOAP, на основі якої здійснюється взаємодія за схемою, наведеною на рис. 1, всі технології дають можливість інтегрувати сервіси різноманітного походження.

Для врахування особливостей сервісного, процесно-функціонального і компонентно-базованого підходів до створення і підтримки рішень користувачів використовуються такі відомі архітектурні способи організації групової функціональності:

"хореографія" (Web Service Choreography) – підхід, який визначає протоколи взаємодії вебсервісів між собою, коли сервіси кооперуються для реалізації однієї глобальної задачі, виконуючи окремі її частини. Роль, яку перебирає на себе сервіс, визначає його модель обміну повідомленнями з іншими сервісами. Цей метод демонструє високу ефективність для невеликих задач, але одночасно зі зростанням складності задач швидко зростає кількість задіяних сервісів, рішення стають надто громіздкими, і так само швидко зменшується ефективність;

"оркестрація" (Service orchestration) – підхід до об'єднання сервісів за допомогою сервісу-оркестратора в один багатофункціональний

бізнес-сервіс. Роль сервісів-оркестраторів можуть виконувати впроваджені в систему агенти, що дає змогу не змінювати її ієрархічну структуру й таким способом комбінувати переваги оркестрації і МАС. Для реалізації оркестрації розроблено спеціальну мову для описання бізнес-процесів і протоколів їх взаємодії між собою BPEL. Ця мова, яку іноді називають оркестраційною, надає процес(сервіс)-оркестратор для координування роботи підпорядкованих йому сервісів. Координована взаємодія досягається за рахунок того, що web-сервіси обмінюються повідомленнями. Проблема полягає в тому, що для виконання довготривалих бізнес-процесів зі складною структурою важливі не тільки самі повідомлення, а й порядок обміну, врахування стану, результатів, часу та інших аспектів бізнес-процесів. Якщо роботу базових веб-сервісів організовує програмний агент як сервіс-оркестратор більш високого рівня, то отримуємо механізм керування на рівні веб-сервісів, тоді як організаційні питання (послідовність обміну повідомленнями, врахування завантаженості тощо) будуть вирішувати агенти рівнем вище. Для обміну повідомленнями разом із BPEL зазвичай застосовують SOAP, хоча можливі й інші альтернативи [20].

У науковому співтоваристві уже використовуються системи, побудовані на декомпозиції загальної задачі (потоку) в послідовність окремих підзадач (підпотоків) – так звані workflow-системи. Серед діючих міждисциплінарних workflow-середовищ слід назвати Triana Workflows і Kepler. Користувач-дослідник може редагувати сценарій розв'язання задачі, використовуючи для цього засоби автоматизації системи, що надає їй гнучкості і розширює сферу застосування. В архітектурі цих систем виділяються такі рівні:

1) клієнтський (саме він дає користувачу можливість швидко розробляти потоки робіт за допомогою графічного редактора сценарію);

2) проміжний (його сервер потоків забезпечує виконання робочих потоків, їх авторизацію та узгодження із залученими гетерогенними ресурсами);

3) ресурсний (надає ресурси для виконання окремих підпотоків).

Робочий потік оформлюється як окремий компонент, готовий до використання. Компоненти можуть входити до інших компонентів. За рахунок жорсткої типізації (визначаються внутрішні формати для потоків і компонентів) і формування метаданих для кожного компонента, які описують типи його вхідних і вихідних даних, здійснюється верифікація потоків і компонентів на сумісність типів даних на їх входах і виходах. Поширенню цих систем сприяють зручний графічний інтерфейс користувача, багата бібліотека стандартних компонентів для різних дисциплін, ефективний диспетчер виконання сценаріїв та засоби моніторингу.

Останнім часом також широко використовується Taverna Workflow Management System – система для розроблення, проектування і виконання складних потоків робіт, яка дає змогу координовано використовувати різнорідні сервіси WSDL, BioMoby, SoapLab та інших типів [21].

На жаль, ці системи не сумісні на рівнях як диспетчера виконання, так і описання потоків і компонентів. Крім того, їм бракує інтелектуальної складової або така складова недостатньо розвинена, і будь-яку робочу послідовність сервісів необхідно збирати вручну або редагувати, що істотно обмежує гнучкість і сферу застосування цих систем.

На основі наведених технологій запропоновано кілька рішень, які можна використовувати для створення прикладних систем накопичення, оброблення й обміну науковими даними у різних галузях. Найвідомішими з них є системи ЕСИМО і GEOSS. Вони досить поширені, хоча їм властивий ряд недоліків щодо оброблення різнорідної інформації:

- складність об'єднання різнорідних типів даних від різних джерел в одному запиті;
- складність розширення функціональності та збільшення ступеня автоматизації, викликані недоліками архітектури цих систем, яка не передбачає можливості об'єднання в єдину мережу;
- відсутність єдиної моделі метаданих для всіх об'єктів метаданих;
- відсутність єдиного стандарту оформлення й оброблення даних;
- наявність у користувача специфічного набору знань, умінь і навичок для складання запиту.

На основі огляду можна зробити обґрунтований висновок щодо можливості використання для інтеграції прикладних застосувань напрацьованих підходів, моделей, технологій і засобів, насамперед:

схеми реалізації взаємодії застосувань на основі стека протоколів UDDI, SOAP (або REST) & BPEL, WSDL, XML над транспортним рівнем;

технології типу WCF;

підхід до об'єднання сервісів за допомогою сервісу-окрестартора в один багатофункціональний бізнес-сервіс.

Але необхідно підвищувати рівень автоматизації процесів побудови схем взаємодії сервісів для задач, сформульованих користувачем. По суті, мова має йти про розроблення ефективних моделей і методів інтеграції застосувань, на основі яких може бути створена сучасна система інтеграції застосувань, здатна забезпечити користувачеві наведені вище переваги.

Архітектура системи інтеграції застосувань

Принциповим положенням створення розподіленої системи є спосіб організації взаємодії сервісів. Із двох основних відомих способів взаємодії сервісів – оркестрації та хореографії – для системи СЦД більш ефективним буде перший. Дійсно, оркестрація, яка групує базові сервіси в ієрархічно скомплексовані системи, підпорядковуючи їх адміністраторам – сервісам-окрестраторам, дає змогу об'єднати сервіси за зручними для СЦД ознаками (галузь науки, функціональність, регіональне розміщення тощо), а сервісам-окрестраторам надає повноваження відповідно до політик міжнародного обміну даними. Таким чином, можна вибудувати просту й ефективну систему, в якій пошук необхідного сервісу або побудова їх композиції для складних запитів, притаманних міждисциплінарним дослідженням, не буде вимагати великого часу, оскільки кожний сервіс-оркестратор має інформацію щодо функціональності підпорядкованих сервісів і, крім того, вони можуть координувати свої можливості в процесі планування і реалізації запитів користувачів. Включення нових сервісів до системи також не буде становити особливих проблем, оскільки для цього потрібно буде лише скласти опис сервісу, внести його в реєстр і закріпити за певним сервісом-оркестратором [22].

Такий підхід буде працювати, коли користувач точно знає свої інформаційні потреби і має відповідні знання і навички для складання ланцюжків запитів до відомих оркестраторів. Але важливіше те, що підхід дасть змогу працювати із системою користувачам, які не мають можливості ініціювати сервіси, вказуючи порядок їх роботи й уточнюючи параметри виконання і взаємодії. Користувач повинен лише вміти формулювати свої потреби в термінах певної предметної сфери. Але тоді для об'єднання сервісів і організації їх взаємодії в потрібному порядку системі необхідна інтелектуальна складова. Для цього використовуються інтелектуальні агенти, які становлять ядро системи, реалізуючи логіку її функціонування, спрямовану на підтримку формування запитів, планування їх виконання й організацію взаємодії базових сервісів. Включення інтелектуальних агентів до ієрархічної структури як сервісів-оркестраторів приводить до формування дворівневої системи, нижній рівень якої формують сервіси, що розв'язують базові задачі, а верхній – інтелектуальні агенти, які оркеструють базові сервіси. Пов'язані один з одним інтелектуальні агенти, використовуючи реєстри підпорядкованих їм базових сервісів і їх функціональності, за рахунок спілкування можуть реалізувати методи логічного виведення. Результатом виведення і буде композиція дій базових агентів, виконання якої дасть змогу отримати розв'язок поставленої користувачем задачі. Ця композиція дій або вивід передається на нижній рівень, де й здійснюється функціональна реалізація процесу розв'язання задачі. На цьому етапі контроль передається агентам нижнього рівня, які видають лише кінцевий результат для відправлення користувачу.

Власне виконання запиту користувача здійснюється після формування виводу, який містить вказівники на одне або кілька джерел даних, композицію з ідентифікаторів методів, які сервіси системи повинні виконати над цими даними. Деякі запити не вимагають попереднього оброблення даних і задовольняються методами сервісів безпосередньо після отримання інформації з джерела даних. Інші ж вимагають певної послідовності попередньо виконаних над даними дій і, як наслідок, включення методів інших сервісів. Потрібні для цього залежності описуються в аксіомах, які вводяться до загальної бази знань при реєстрації відповідних сервісів у системі. Зазначені операції дають можливість отримати кінцевий продукт роботи системи – дані, які становлять розв'язок задачі користувача. Але початком активної роботи системи можна вважати надходження сформульованого користувачем запиту. Хоча, з огляду на вимоги до інтерфейсу системи, необхідно допомогти користувачеві скласти запит у знайомих йому термінах.

У рішенні, яке пропонується, для створення реєстру сервісів використовується UDDI, для уніфікованого їх описання – WSDL. Ключова ж частина – обмін повідомленнями – реалізується за допомогою SOAP, а оркестрація

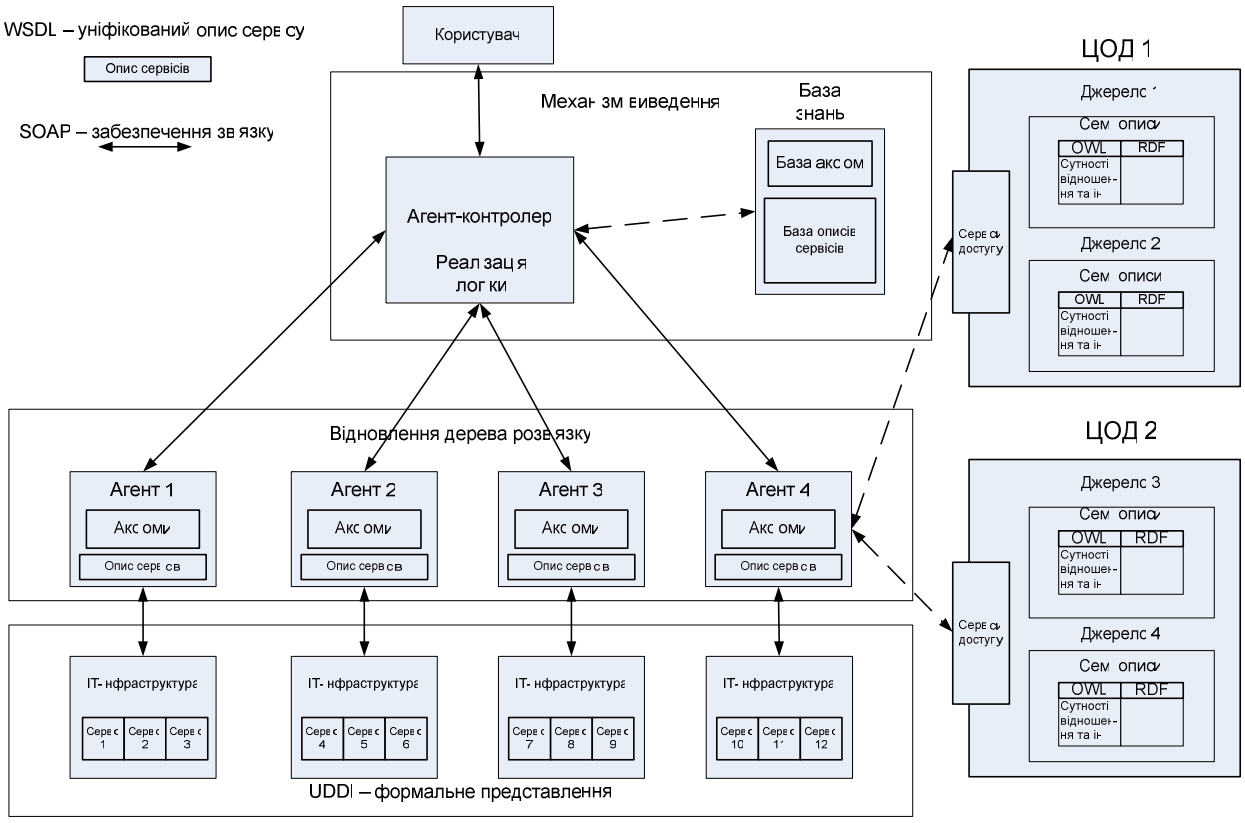

Рис. 2. Схема взаємодії компонентів системи

сервісів – BPEL. Для описання агентів використовується JADE. Схема взаємодії компонентів системи наведена на рис. 2.

Опис інтерфейсу (WSDL) кожного зареєстрованого в системі сервісу нижнього рівня і пов'язаних із ним аксіом заноситься до бази знань системи і безпосередньо до бази знань агента-контролера, якому новий сервіс буде підпорядкований. Вибір агента-контролера здійснюють згідно з політиками, прийнятими в системі СЦД, наприклад за критерієм географічного розміщення розгорнутого агента, з метою мінімізації обміну даними.

На рівні агентів агент-оркестратор (або кілька), отримавши запит користувача, опитує агентів з метою пошуку потрібних сервісів і вільних ресурсів. Запит користувача обробляється з урахуванням сервісів системи, їх функціональності (описаної аксіомами системи) і правил виведення, визначених логічним формалізмом. Отриманий вивід передається на нижній рівень для реалізації.

На рівні сервісів кожний агент виконує виклики необхідних сервісів зі свого набору і відправляє оброблені дані відповідно до ланцюжка дій. Це відбувається згідно з деревом розв'язання задачі, відновленим механізмом побудови дерева розв'язання на основі виводу.

Для реалізації такої взаємодії сервісів у системі СЦД найбільш прийнятним видається логічний підхід, який не тільки дає змогу вийти на рівень сформульованих вище вимог, але й позбавлений недоліків традиційного алгоритмічного підходу. Дійсно, необхідно лише розробити метод виведення і механізм побудови дерева розв'язання, які будуть підтримувати процеси формування запиту, планування його виконання, відновлювати схему розв'язання задачі і реалізувати її. Для створення і описання цих складових підходу найбільш прийнятним є логічний підхід. А для реалізації логічного підходу необхідно:

– описати існуючі застосування і їх функціональні можливості на формальній мові;

– сформулювати правила виведення;

– визначитися з методом виведення;

– розробити алгоритм роботи механізму побудови дерева виконання запиту користувача на основі виводу;

– реалізувати зазначені методи і механізми в агентах системи.

Формальна логічна система

Опишемо формалізм, на базі якого буде побудована програмна система, що забезпечить вирішення поставленої проблеми. За основу візьмемо клаузальну логіку першого порядку й опишемо мову формальної системи, дотримуючись запропонованих у праці [3] визначень її конструктивних елементів.

Символи:

***службові*:** (,); [,]; {, }; ::; ⟨,⟩; ,;

***константи*:**

1) *індивідні*: простих типів (int, real, char, bool) – a_1^1, a_2^1, ..., a_1^2, a_2^2, ..., причому кожна константа a_i^k належить сорту (простому типу) k; структурного типу (construct) – c_1, c_2, ...; процедурного типу (method) – d_1, d_2, ...; об'єктного типу (problem, entity, relation) – e_1, e_2, ...;

2) *функціональні і-місні*: для індивідів типу k – h_1^1, h_2^1, ..., h_1^2, h_2^2, ...;

3) *предикатні і-місні*: для індивідів типу k – A_1^1, A_2^1, ..., A_1^2, A_2^2, ... (до цього класу включаються таксономічні, реляційні та інші предикати, а також традиційні відношення, щонайменше рівності "=" і порядку "≥");

***змінні*:** для індивідів типу k – x_1^1, x_2^1, ..., x_1^2, x_2^2, ..., причому кожна змінна x_i^k належить типу k;

***логічні*:** $\neg, \wedge, \vee, \leftarrow, \exists, \forall, \Leftrightarrow$.

Індивідні терми типу *k*:

1) кожна індивідна константа a_i^k типу k є індивідним термом типу k;

2) кожна вільна змінна x_i^k для індивідів типу k є індивідним термом типу k;

3) якщо h_i^j – деяка функціональна константа для індивідів типу k, τ_1, ..., τ_j – терми для індивідів типу k, то h_i^j (τ_1, ..., τ_j) є індивідним термом типу k;

4) інших індивідних термів типу k немає.

Терми, отримані застосуванням правил побудови 1 або 2 визначення, назвемо простими, а всі інші – складними.

Формули для індивідів:

1) якщо A_i^j – предикатна константа для індивідів, τ_1, ..., τ_j – терми для них, то A_i^j (τ_1, ..., τ_j) – атомарна формула для індивідів;

2) атомарна формула для індивідів – формула для них;

3) інших формул для індивідів немає.

Надалі вважаємо, що в системі є універсальний перетворювач формул до традиційного для клаузальної форми (мова йде про клаузи Хорна) вигляду з єдиним символом →, атомарними формулами ліворуч і праворуч від нього, неявним квантором ∀.

Специфікатори – це конструкції вигляду τ_1, де τ_1 – терм для індивідного об'єкта. Визначимо специфікатори конструктів:

1) якщо – індивідні терми типу entity, $e_1^r, ..., e_i^r$ – індивідні терми типу relation, $A_1^j(a_1^1, ..., a_j^1), ..., A_i^j(a_1^k, ..., a_j^k)$ – атомарні формули для індивідів простих типів, τ – індивідний терм типу construct, то τ: $(e_1^e, ..., e_i^e, e_1^r, ..., e_i^r, A_1^j(a_1^1, ..., a_j^1), ..., A_i^j(a_1^k, ..., a_j^k))$ – специфікатор конструкта.

2) інших специфікаторів конструктів немає.

Визначимо передумови:

1) якщо c_1 – специфікатор конструкта, П – послідовність атомарних формул, то $\langle \tau_1 : (c_1, П) \rangle$ – елементарна передумова;

2) елементарна передумова – передумова;

3) якщо $\langle \tau_1 \rangle$ – передумова, а τ_2 – елементарна передумова, то $\langle \tau_1, \tau_2 \rangle$ – передумова;

4) інших передумов немає.

Визначимо постумови:

1) якщо τ_1 – специфікатор конструкта, П – послідовність атомарних формул, то $\langle \tau_1 : (c_1, П) \rangle$ – елементарна постумова;

2) елементарна постумова – постумова;

3) якщо $\langle \tau_1 \rangle$ – постумова, а τ_2 – елементарна постумова, то $\langle \tau_1, \tau_2 \rangle$ – постумова;

4) інших постумов немає.

Визначимо специфікатори методів:

1) якщо τ – індивідний терм типу method, $\langle \tau_1 \rangle$ – передумова, $\langle \tau_1 \rangle$ – постумова, то τ: $(\langle \tau_1, \tau_2 \rangle)$ – специфікатор методу;

2) арність по конструктах передумови методу має бути не меншою арності по конструктах постумови методу;

3) інших специфікаторів методів немає.

Визначимо специфікатори проблем:

1) якщо $\langle \tau_1 \rangle$ – передумова, а $\langle \tau_2 \rangle$ – постумова, τ – терм для індивідного об'єкта типу

Problem, то τ: $\langle\langle\tau_1\rangle,\langle\tau_2\rangle\rangle$ – специфікатор проблеми;

2) інших специфікаторів проблем немає.

Клауза – це вираз вигляду $\Pi \to \Lambda$, де Π – послідовність атомарних формул; Λ – єдина атомарна формула. Залежно від вигляду атомарних формул будемо виділяти клаузи:

1) індивідні (містять тільки атомарні формули для індивідів);

2) типізовані (містять тільки атомарні формули для типів);

3) загального вигляду (містять атомарні формули для індивідів і типів).

Базу знань системи становить онтологія аксіом, які зображують методи сервісів системи, й онтології джерел даних, описаних на мові OWL на базі RDF. Також до бази знань належать правила виведення, необхідні для отримання потрібного результату у випадках, коли необхідно комбінувати методи як одного, так і різних сервісів.

Правила виведення:

1. Якщо $d_1:(\langle\tau_1\rangle,\langle\tau_2\rangle)$ і $d_2:(\langle\tau_3\rangle,\langle\tau_1\rangle)$, то $d_1:(\langle d_2\rangle,\langle\tau_2\rangle)$.

2. Якщо $d_1:(\langle\tau_1\rangle,\langle\tau_2\rangle)$ і $d_2:(\langle\tau_3\wedge\tau_4\rangle,\langle\tau_1\rangle)$, то $d_1:(\langle d_2\rangle,\langle\tau_2\rangle)$.

3. Якщо $d_1:(\langle\tau_1\wedge\tau_2\rangle,\langle\tau_3\rangle)$, d_2: $(\langle\tau_4\rangle,\langle\tau_1\rangle)$ і d_3: $(\langle\tau_5\rangle,\langle\tau_2\rangle)$, то d_1: $(\langle d_2\wedge d_3\rangle,\langle\tau_3\rangle)$.

4. Якщо d_1: $(\langle\tau_1\rangle,\langle\tau_2\rangle)$ і $d_2:(\langle\tau_3\vee\tau_4\rangle,\langle\tau_1\rangle)$, то $d_1:(\langle d_2\rangle,\langle\tau_2\rangle)$.

5. Якщо $d_1:(\langle\tau_2\wedge\tau_2\rangle,\langle\tau_1\rangle)$, то d_1: $(\langle\tau_2\rangle,\langle\tau_1\rangle)$.

6. Якщо $d_1:(\langle\tau_1,\tau_2\rangle,\langle\tau_3\rangle)$, $d_2:(\langle\tau_4\rangle,\langle\tau_1\rangle)$ і d_3: $(\langle\tau_5\rangle,\langle\tau_2\rangle)$, то d_1: $(\langle d_2,d_3\rangle,\langle\tau_2\rangle)$.

7. Якщо $d_1:(\langle\tau_1\rangle,\langle\tau_2,\tau_3\rangle)$, $d_2:(\langle\tau_2\rangle,\langle\tau_4\rangle)$ і d_3: $(\langle\tau_3\rangle,\langle\tau_5\rangle)$, то $d_2:(\langle d_1\rangle,\langle\tau_4\rangle)$ і $d_3:(\langle d_1\rangle,\langle\tau_5\rangle)$.

Метод виведення

Будемо використовувати запропонований у праці [1] метод, побудований на типізації тверджень і аналогії. Його ідея полягає в побудові виводу в абстрактному просторі з подальшим його використанням для керування процесом виведення у вихідному просторі пошуку розв'язків. Це дасть можливість підвищити ефективність виведення за рахунок відсікання більшої частини безперспективних гілок виводу у вихідному просторі.

Надалі під мультиклаузою (*m*-клаузою) будемо розуміти мультимножину літералів (атомарних формул) і будемо записувати в *m*-клаузі літерал L стільки разів, скільки разів він повторюється. Звичайну клаузу будемо вважати *m*-клаузою, кратність кожного літерала в якій дорівнює одиниці. На *m*-клаузи переносяться операції $\cup$ (об'єднання), $\cap$ (перетин), – (різниця), $\cdot$ (конкатенація) і відношення $\subseteq$ (входження) для мультимножин.

Нехай $A_1\in C_1, A_2\in C_2$, а α_1 і α_2 – такі підстановки, що для деякого літерала L у правій частині A_1 і лівій частині A_2, $A_1,\alpha_1=\{L\}$, $A_2,\alpha_2=\{L\}$, причому якщо $|A_i|>1, i=1,2$, то відповідна підстановка α_i є найбільш загальним уніфікатором літералів із A_1. Тоді клауза, отримана з об'єднання клауз $C_1\alpha_1$ і $C_2\alpha_2$ вилученням L з правої і лівої частин називається *m*-резольвентою *m*-клауз C_1 і C_2. Нехай всі *m*-клаузи вихідної множини S упорядковані, а C_1 і C_2 – дві з них. Нехай $A_1\in C_1, A_2\in C_2$, а α_1 і α_2 – такі підстановки, що для деякого літерала L у правій частині A_1 і лівій частині A_2, $A_1,\alpha_1=\{L\}$, $A_2,\alpha_2=\{L\}$, причому якщо $|A_i|>1$, $i=1,2$, то відповідна підстановка α_i є найбільш загальним уніфікатором літералів із A_i. Тоді клауза, отримана з об'єднання клауз $C_1\alpha_1$ і $C_2\alpha_2$ вилученням L з правої частини і останнього L лівої частини і викреслюванням будь-якого непідкресленого літерала, за яким немає жодного іншого літерала, називається упорядкованою лінійною *m*-резольвентою упорядкованих *m*-клауз C_1 і C_2. У випадку, коли останній літерал у лівій частині упорядкованої *m*-клаузи уніфікується з підкресленим літералом у правій частині цієї ж клаузи, упорядкована лінійна *m*-резольвента може бути отримана вилученням останнього літерала лівої частини упорядкованої *m*-клаузи, тобто редукцією *m*-клаузи.

Назвемо *m*-резолюційним виводом Tm пару, перший компонент якої – множина вершин виводу, а другий – множина трійок вершин (для виділення компонентів будемо використовувати пару функцій-селекторів: $s\text{-}N(Tm)$ і $s\text{-}M(Tm)$ відповідно). Кожна вершина *m*-резолюційного виводу характеризується позначкою і значенням глибини у виводі таким чином, що

якщо $n \in s\text{-}N(Tm)$, то $s\text{-}L(n)$ – позначка вершини n і $s\text{-}D(n)$ – значення її глибини у виводі. Якщо $\langle n_1, n_2, n_3 \rangle \in s\text{-}M(Tm)$, то $s\text{-}L(n_3)$ є m-резольвентою $s\text{-}L(n_1)$ і $s\text{-}L(n_2)$. Кожна трійка такого вигляду називається m-резолюцією. Якщо $n \in s\text{-}N(Tm)$ і вершина n не є третьою складовою жодної з трійок з $s\text{-}M(Tm)$, то n називається початковою вершиною виводу Tm. Позначка такої вершини – початкова m-клауза. Якщо $n \in s\text{-}N(Tm)$ і вершина n не є ні першою, ні другою складовою жодної з трійок з $s\text{-}M(Tm)$, то n називається термінальною вершиною виводу, а її позначка – термінальною m-клаузою. Загалом Tm має задовольняти таке обмеження: якщо $\langle n_1, n_2, n_3 \rangle \in s\text{-}M(Tm)$, то $\langle n_2, n_1, n_3 \rangle \in s\text{-}M(Tm)$. Значення глибини (у виводі) кожної вершини n, $n \in s\text{-}N(Tm)$, за визначенням: 1) $s\text{-}D(n) = 0$ для будь-якої початкової вершини n; 2) $s\text{-}D(n) = 1 + \min\{\max\{s\text{-}D(n_1), \{s\text{-}D(n_1) | \langle n_1, n_2, n_3 \rangle \in s\text{-}M(Tm)\}$ для будь-якої непочаткової вершини n.

Назвемо m-резолюційний вивід Tm виводом із S, якщо позначки початкових вершин Tm належать множині m-клауз S. Із S виводиться C, якщо Tm є виводом із S, а C – позначкою однієї з вершин Tm. Нехай значення функції Result() визначене для Tm і дорівнює C, якщо термінальна клауза C виводу Tm є єдиною, тобто Result(Tm) = C. Нехай Tm_1 і Tm_2 – m-резолюційні виводи. Назвемо Tm_1 субвиводом Tm_2 (позначимо $Tm_1 \subseteq Tm_2$), якщо виконуються такі умови: $s\text{-}N(Tm_1) \subseteq s\text{-}N(Tm_2)$ і $s\text{-}M(Tm_1) \subseteq s\text{-}M(Tm_2)$. При цьому, якщо всі початкові вершини Tm_1 – початкові вершини Tm_2, то Tm_1 назвемо початковим субвиводом Tm_2.

Нехай Tm – m-резолюційний вивід із S, всі m-клаузи якого упорядковані. Якщо для будь-якої трійки $\langle n_1, n_2, n_3 \rangle$ із $s\text{-}M(Tm)$ $s\text{-}L(n_3)$ є упорядкованою лінійною m-резольвентою $s\text{-}L(n_1)$ і $s\text{-}L(n_2)$, то Tm назвемо упорядкованим лінійним виводом із S. Тоді, за визначенням m-резолюційного виводу: $\max\{s\text{-}D(n_1), s\text{-}D(n_2)\} = s\text{-}D(n_3) - 1$.

Крім загальних властивостей m-резолюційних виводів, упорядковані лінійні m-резолюційні виводи мають особливості. Так, для упорядкованого лінійного m-резолюційного виводу Tm справедливі такі умови: 1) якщо $\langle n_1, n_2, n_3 \rangle \subseteq s\text{-}M(Tm)$, то $\langle n_2, n_1, n_3 \rangle \notin s\text{-}M(Tm)$; 2) для кожної непочаткової вершини n_3 глибини r існує трійка $\langle n_1, n_2, n_3 \rangle$, у якій n_1 має глибину $r-1$, n_2 відповідає початковій вершині або відсутня; 3) Tm містить єдину термінальну клаузу.

Для керування впорядкованим лінійним виводом у праці [7] використовується абстракція типізації. Нехай f – відображення із множини m-клауз у множину підмножин m-клауз, таке, що: 1) якщо m-клауза C_3 є m-резольвентою m-клауз C_1 і C_2, а $D_3 \in f(C_3)$, то існують $D_1 \in f(C_1)$ і $D_2 \in f(C_2)$ такі, що результат підстановки деякої m-резольвенти D_1 і D_2 належить D_3; 2) $f(\varnothing) = \{\varnothing\}$; 3) якщо результат деякої підстановки m-клаузи C_1 належить m-клаузі C_2, то для будь-якої абстракції для C_2 є абстракція D_1 для C_1 така, що результат підстановки D_1 належить D_2. Назвемо таке відображення f m-абстрактним відображенням, а будь-яке D із $f(C)$ – m-абстракцією. Під відображенням типізації будемо розуміти деяке відображення φ із літералів у літерали, яке відображає кожну атомарну формулу у формулу, терми якої мають тип, найближчий за ієрархією до базових типів. У праці [7] встановлено, що відображення типізації φ є m-відображенням.

Упорядковані лінійні виводи Tm і Um перебувають у відношенні $\rightarrow_f$ (позначимо $Tm \rightarrow_f Um$), де f – m-відображення, якщо вершини виводів Tm і Um перебувають у такому відношенні відповідності R, що: 1) одночасно виконується $\forall n(n \in s-N(Tm))\ \exists n'(n \in s-N(Um))\ (nRn')$ і $\forall n(n \in s-N(Um))\ \exists n'(n \in s-N(Tm))\ (nRn')$; 2) якщо nRn', де $n \in s-N(Tm)$, $n' \in s-N(Um)$, то n і n' можуть бути початковими або термінальними вершинами тільки одночасно; 3) для термінальних вершин Tm і Um відношення R є відношенням один до одного; 4) якщо $\langle n_1, n_2, n_3 \rangle \in s\text{-}M(Tm)$, $\langle n_1, n_2, n_3 \rangle \in s\text{-}M(Um)$ і $n_3 R n_3'$, то $n_1 R n_1'$ і $n_2 R n_2'$; 5) якщо n і n' – початкові вершини виводів Tm і Um відповідно і nRn', то $s\text{-}L(n') \in f(s\text{-}L(n))$; 6) як-

що n і n' – непочаткові вершини виводів Tm і Um відповідно і nRn', то приклад $s\text{-}L(n')$ належить $f(s\text{-}L(n))$.

Отже, упорядковані лінійні виводи мають властивість мінімальності, тобто точно одну термінальну вершину, і якщо трійка $\langle n_1, n_2, n_3\rangle$ належить мінімальному упорядкованому лінійному виводу, то жодна інша трійка не може містити вершину n_3 як третю компоненту, за виключенням трійки $\langle n_1, n_2, n_3\rangle$.

Використаємо ці властивості впорядкованих лінійних виводів для побудови механізмів виведення, які реалізуються агентами. Після того як користувач сформулював задачу, система будує вивід – по суті перевіряє можливість задоволення запиту за рахунок наявних ресурсів. Це завдання виконують агенти-оркестратори як важлива складова інтелектуального ядра системи. Маючи доступ до переліку всіх сервісів, їх описів і аксіом, які й специфікують можливості застосування сервісів, вони запускають процес формування виводу. Тоді й відбувається побудова ланцюжка сервісів, здатного забезпечити потрібний користувачу результат, розпочинаючи з отримання даних із відповідних їх джерел.

Упорядкований лінійний m-резолютивний вивід подаватимемо парою множин вершин $V = \{k_1, k_2, k_3, ..., k_n\}$ і трійок вершин $T\{\langle k_1, k_2, k_3\rangle, \langle k_3, k_4, k_5\rangle, ..., \langle k_{n-2}, k_{n-1}, k_n\rangle\}$, де k_i – вершини дерева, k_n – термінальна вершина, результат роботи алгоритму. Вивід формується при покладанні за початкові вершини виводу постумови сформульованої проблеми та як бічної вершини – сумісної аксіоми з бази знань. Аксіома є сумісною, якщо послідовність атомарних формул, які відповідають цій вершині виводу, або будь-яка її частина є постумовою аксіоми. Третю вершину цієї трійки отримуємо застосуванням аксіоми до формули постумови з урахуванням правил виведення. Третя вершина першої трійки стає першою вершиною другої трійки, і процес повторюється доти, поки не буде досягнуто термінальної вершини – передумови, вказаної в проблемі. Якщо атомарну формулу, що відповідає третій вершині трійки, може бути спрощено застосуванням одного з правил виведення, вона спрощується в наступній трійці, яка матиме всього дві вершини – вершину зі спрощуваною формулою і вершину зі спрощеною формулою. Пошук сумісних аксіом відбувається зіставленням конструкта оброблюваної в дійсний момент постумови з конструктами аксіом з онтології. Таким чином, з онтології вибирається множина аксіом, описані якими сервіси оброблюють необхідний нам тип і формат об'єктів. Підбір і перебір аксіом під час роботи механізму виведення відбувається лише на цій множині. Якщо під час пошуку аксіом із бази знань для чергової вершини сумісних аксіом знаходиться декілька, то кожна з них використовується для подальшої побудови своєї "паралельної" версії виводу. Таким чином, під час роботи механізму виведення може формуватися

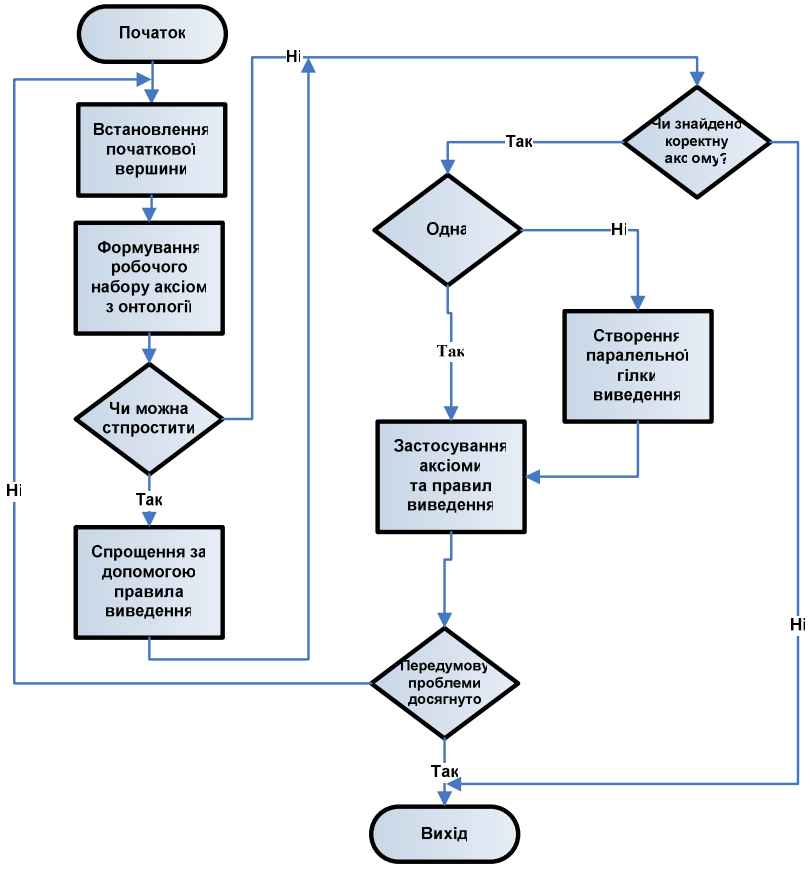

Рис. 3. Алгоритм роботи механізму виведення

деяка кількість виводів різних довжини і складності залежно від кількості аксіом у базі знань і комбінацій їх застосування до кожної вершини.

Після завершення роботи механізму виведення з множини отриманих виводів вибирається вивід із найменшою кількістю трійок вершин і найменшою кількістю застосованих аксіом. Довші, циклічні та тупикові виводи відкидаються.

Множина формул і клауз, які відповідають вершинам трійок виводу і відображають застосування аксіом і правил при їх побудові, є виходом механізму виведення.

Алгоритм роботи механізму виведення відображено на рис. 3.

Приклад роботи механізму виведення та побудови дерева вирішення для проблеми P типу Problem P: $\langle\langle A\rangle,\langle K\rangle\rangle$, де A і K – передумова та постумова.

Аксіоми:

$S1$: $(\langle A\rangle,\langle K\rangle)$;

$S2$: $(\langle A\rangle,\langle C\rangle)$;

$S3$: $(\langle A\rangle,\langle G\rangle)$;

$S4$: $(\langle A\rangle,\langle H\rangle)$;

$S5$: $(\langle B\wedge C\rangle,\langle E\rangle)$;

$S6$: $(\langle E\wedge C\rangle,\langle F\rangle)$;

$S7$: $(\langle B\rangle,\langle D\rangle)$;

$S8$: $(\langle E\wedge G\wedge H\rangle,\langle T\rangle)$;

S9: $(\langle F\wedge D\rangle,\langle K\rangle)$.

Механізм виведення виконує пошук рішення, починаючи від постумови проблеми P, використовуючи наявні аксіоми та правила:

$S9$: $F\wedge D$;

$S6$: $E\wedge C\wedge D$;

$S7$: $E\wedge C\wedge B$;

$S5$: $B\wedge C\wedge C\wedge B$.

Спрощення за правилом 5: $B\wedge C\wedge C\wedge B\Leftrightarrow$ $\Leftrightarrow B\wedge C$;

$S2$: $B\wedge A$;

$S1$: $A\wedge A$.

Спрощення за правилом 5: $A\wedge A\Leftrightarrow A$.

Вивід, зображений на рис. 4, становить множину трійок вершин: $\langle A, A\wedge A\rangle$, $\langle A\wedge A, B\leftarrow A,$ $B\wedge A\rangle$, $\langle B\wedge A, C\leftarrow A,\ B\wedge C\rangle$, $\langle B\wedge C, B\wedge C\wedge B\rangle$, $\langle B\wedge C\wedge B, B\wedge C\wedge C\wedge B\rangle$, $\langle B\wedge C\wedge C\wedge B, E\leftarrow B\wedge C,$ $E\wedge C\wedge B\rangle$, $\langle E\wedge C\wedge B, D\leftarrow B, E\wedge C\wedge D\rangle$, $\langle E\wedge C\wedge D,$ $F\leftarrow E\wedge C, F\wedge D\rangle$, $\langle F\wedge D, K\leftarrow F\wedge D, K\rangle$.

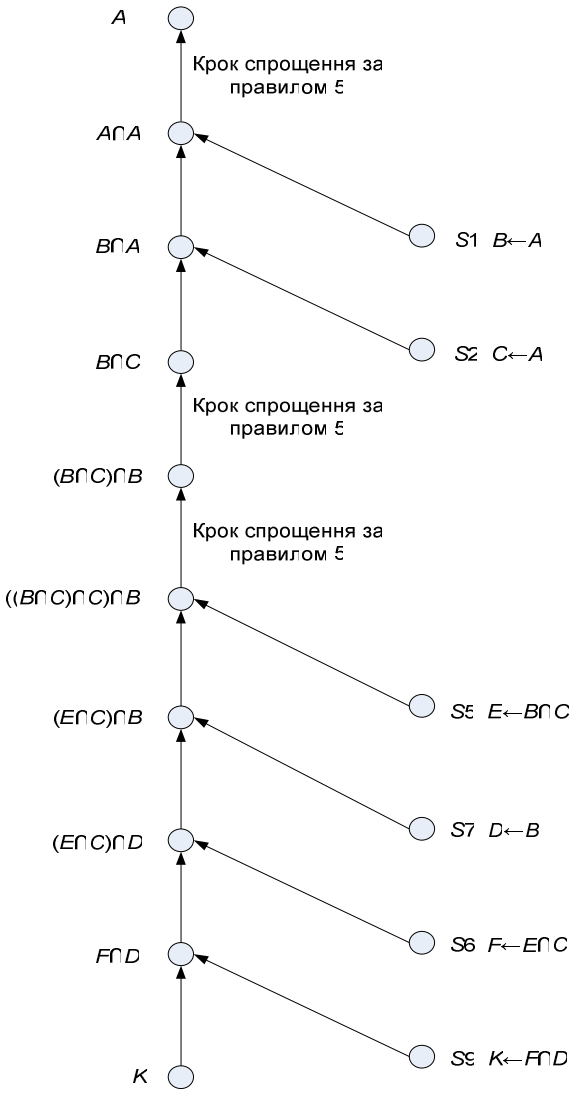

Рис. 4. Вивід для проблеми $\langle\langle A\rangle,\langle K\rangle\rangle$

Механізм відновлення схеми виконання запиту

Для отримання з виходу механізму виведення функціональної послідовності дій, яку і буде виконувати наша система, з врахуванням особливостей і характеру сервісів, необхідно задіяти механізм побудови схеми виконання запиту.

Схема виконання запиту становить зв'язний орієнтований граф без орієнтованих циклів з паралельними орієнтованими шляхами з кореня у вершини, який має три типи вершин, і задається трійкою $G=\langle V, E, \Theta\rangle$, де $V=$ $=V_1\cup V_2\cup V_3$, V_1 – множина вершин-методів, V_2 – множина вершин-передумов і вершин-постумов, V_3 – множина вершин даних, у яких

відбувається об'єднання або розщеплення даних; E – множина ребер; Θ – підмножина декартового добутку $E \times V \times V$, яка визначає співвідношнння ребер і пар вершин. Схема визначає послідовність і взаємозвя'зок за даними дій, які будуть виконані виконавчими механізмами системи з оброблення даних для отримання потрібного користувачу результату.

Алгоритм побудови схеми виконання запиту. Дано: вивід Result $= G = \langle V, T \rangle$. Знайти: $G = \langle V, E, \Theta \rangle$.

Крок 1. Визначення термінальної трійки – такої трійки, яка містить вершину k, що не є ані першою, ані другою складовою жодної з трійок множини T.

Крок 2. У вибраній трійці початкова вершина є вершиною постумови, друга – вершиною методу, термінальна – вершиною передумови. Метод, що перетворює передумову на постумову, визначається за однозначною відповідністю формулам вершин постумови та передумови аксіоми методу. Вершини з'єднуються ребрами передумова–метод, метод–постумова. У випадку, якщо передумова аксіоми методу складається з об'єднання або перетину кількох елементів, між вершинами передумов і методу вставляється вершина даних, що групує передумови згідно з відповідним правилом виведення. Якщо в термінальній трійці лише дві вершини, то було застосоване правило виведення для перетворення клауз вершин і місце вершини методу в цій гілці древа займає вершина даних перетворення, що відповідає застосованому правилу виведення, з відповідними їй вершинами постумов.

Крок 3. Вершини, ребра та їх співвідношення заносяться до G.

Крок 4. Вилучивши оброблену трійку з множини трійок вершин T, виконуємо крок 2. Якщо множина пуста – побудову дерева виконання запиту завершено.

Після завершення побудови дерева виконання запиту воно подається на вхід виконавчого механізму. Виконання конкретного методу, вказаного в дереві, представлене конструктом, що описує вхідні дані (сутності, зв'язки, відношення між ними), передумови методу, постумови методу і вихідні дані. Гілки дерева виконання запиту, що знаходяться після вершини розщеплювання даних, можуть виконуватися паралельно доти, доки не дійдуть до вершини об'єднання даних, де після завершення всіх паралельних гілок, залучених до об'єднання, продовжують виконуватися в послідовному режимі. Алгоритм роботи механізму побудови дерева виконання запиту відображено на рис. 5.

Приклад роботи механізму побудови дерева вирішення для проблеми P. Дерево, побудоване механізмом за трійками виводу для проблеми $\langle\langle A \rangle, \langle K \rangle\rangle$, наведено на рис. 6.

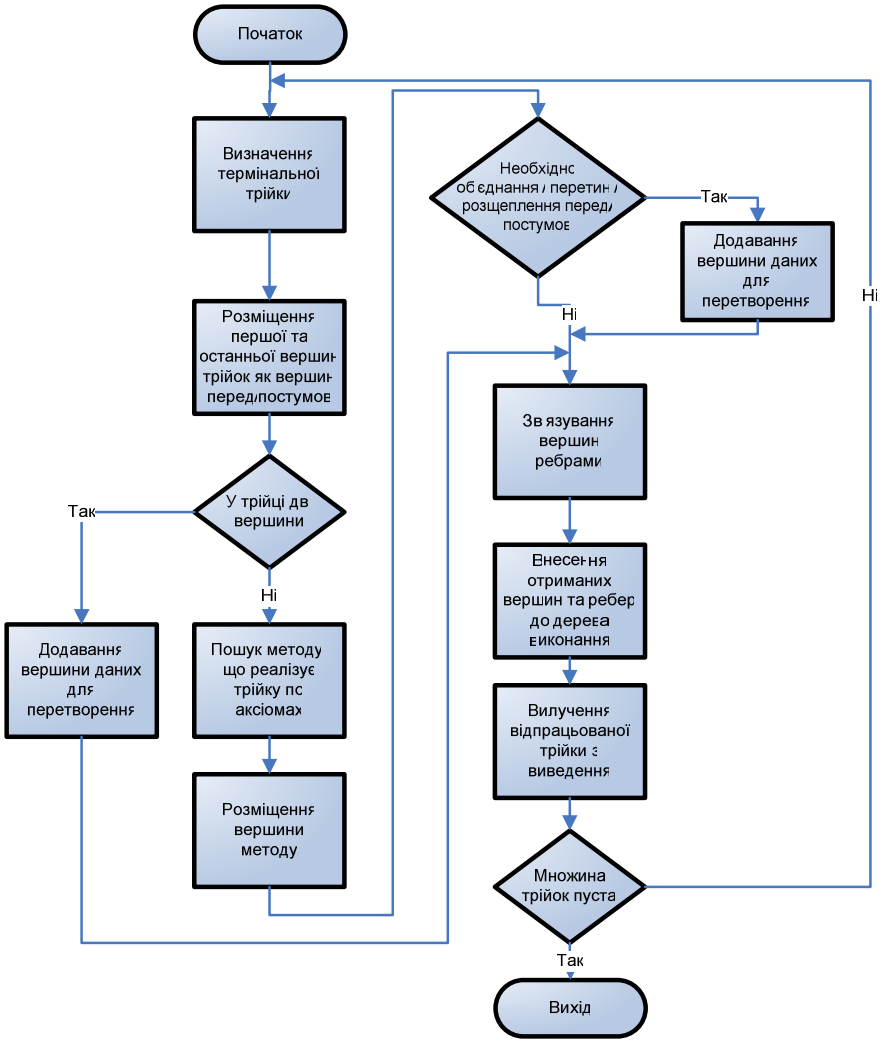

Рис. 5. Алгоритм роботи механізму побудови дерева виконання запиту

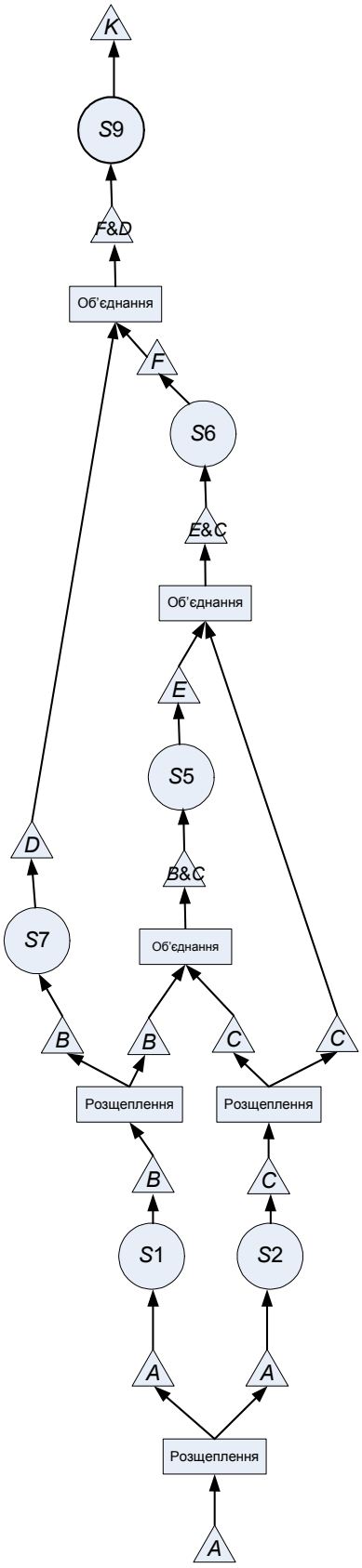

Рис. 6. Дерево вирішення проблеми $\langle\langle A\rangle,\langle K\rangle\rangle$

Визначена послідовність виконання сервісів:

1. $S1$: $(\langle A\rangle,\langle B\rangle)$;
2. $S2$: $(\langle A\rangle,\langle C\rangle)$;
3. $S5$: $(\langle C \wedge B\rangle,\langle E\rangle)$;
4. $S7$: $(\langle B\rangle,\langle D\rangle)$;
5. $S6$: $(\langle E \wedge C\rangle,\langle F\rangle)$;
6. $S9$: $(\langle F \wedge D\rangle,\langle K\rangle)$.

Застосування логічного підходу для розв'язку задач

Продемонструємо працездатність підходу на реальній науковій задачі розрахунку компоненти безпеки життя та індикації критичних значень показників загроз для аналізу сталого розвитку регіонів України [14].

Формула обчислення компоненти безпеки життя має вигляд

$$Csl = \sqrt[3]{\sum_0^n \mathrm{Threat}^3},$$

де значення загрози нормується за формулами (2) або (3).

Мета індикації критичних значень показників загроз полягає у визначенні їх пріоритетів у сенсі першочергової уваги, яку необхідно приділяти ним при прийнятті управлінських рішень на рівні окремого регіону та країни в цілому з метою зменшення впливу загроз на сталий розвиток.

Нехай для кожної адміністративної одиниці $i = \overline{1,n}$ відомий набір значень $\langle x_{i,1}, x_{i,2}, ..., x_{i,m}\rangle$ показників X_j, $j = \overline{1,m}$, які характеризують негативний вплив деяких явищ в економічній, соціальній та екологічній сферах на процеси сталого розвитку. Такі показники, сутність і склад яких визначаються експертним шляхом, будемо називати показниками загроз.

Виходячи зі змісту задачі індикації критичних значень, необхідно визначити таку характеристичну функцію:

$$\Psi(x_{i,j}) = \begin{cases} 0, \text{ якщо значення } x \text{ не є критичним;} \\ 1 \text{ навпаки,} \end{cases} \quad (1)$$

де $i = \overline{1,n}$, $j = \overline{1,m}$.

Зрозуміло, що визначення функції $\Psi(x_{i,j})$ має спиратися на певні критерії, які враховують перевищення $x_{i,j}$ небезпечного порога, від-

носне положення регіону i в рейтингу за показниками X_j, який складено для групи порівняння та для країни в цілому, ступінь "небезпечності" значення $x_{i,j}$ відносно значень інших показників для регіону i.

Для врахування відносного положення регіону в рейтингах для країни в цілому використовується критерій

$$R_{i,j} = \left(1 + e^{\frac{a - x_{i,j}}{b}}\right)^{-1} \quad (2)$$

у випадку, коли більші значення показника X_j відповідають більшому впливу відповідної загрози на сталий розвиток, та

$$R_{i,j} = 1 - \left(1 + e^{\frac{a - x_{i,j}}{b}}\right)^{-1} \quad (3)$$

у випадку, коли менші значення показника X_j відповідають більшому впливу. У формулах (2), (3) параметри a і b обчислюються за формулами

$$a = \bar{X}_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j},\ b = \sigma(X_j) = \sqrt{\frac{\sum_{i=1}^{n}(x_{i,j} - \bar{X}_j)^2}{n}}. \quad (4)$$

Критерій $R_{i,j}$ є безрозмірною величиною та набуває значення в діапазоні $[0,1]$. При цьому значення $R_{i,j}$, близькі до 0,5, відповідають середнім значенням X_j у вибірці, а значення, більші від 0,75, відповідають значенням, які перевищують середні більш ніж на стандартне відхилення. В цьому випадку характеристична функція (1) з урахуванням одного критерію $R_{i,j}$ може бути виражена таким чином:

$$\Psi_R(x_{i,j}) = \begin{cases} 0, R_{i,j} < 0,75; \\ 1, R_{i,j} \geq 0,75. \end{cases}$$

Обчислення критерію $P_{i,j}$, який враховує відносне положення регіону в групі порівняння, може бути здійснено згідно з формулами (2), (3) з урахуванням того, що параметри a і b обчислюються за формулою (4) для кожної групи порівняння окремо.

Критерії $R_{i,j}$ і $P_{i,j}$ є безрозмірними величинами та мають подібну природу, тому можуть бути агреговані за допомогою зваженої суми:

$$K_{i,j} = w_R R_{i,j} + w_P P_{i,j};\ w_R + w_P = 1,$$

у якій вагові коефіцієнти w_R та w_P визначаються експертним шляхом.

Таким чином, для регіону $i = \overline{1,n}$ маємо набір $\langle K_{i,1}, K_{i,2}, ..., K_{i,m}\rangle$ значень агрегованого критерію, який враховує відносне положення регіону в рейтингах, складених для груп порівняння та для країни в цілому. Тепер серед значень $K_{i,j}$, $j = \overline{1,m}$, треба визначити найгірші значення, що можна зробити також за допомогою формули (2):

$$I_{i,j} = \left(1 + e^{\frac{a - K_{i,j}}{b}}\right)^{-1},$$

у якій параметри a і b обчислюються на вибірці $K_{i,j}$, $j = \overline{1,m}$.

Для значень критерію $I_{i,j}$ справедливі всі зауваження, що й для критерію $R_{i,j}$. Тому характеристична функція (1) може бути виражена як

$$\Psi_I(x_{i,j}) = \begin{cases} 0, I_{i,j} < 0,75; \\ 1, I_{i,j} \geq 0,75. \end{cases}$$

Таким чином, значення $\Psi_I(x_{i,j}) = 1$ відповідають найвищому пріоритету уваги, яка повинна приділятися значенню показника X_j при прийнятті управлінських рішень на рівні окремого регіону i.

Приклад. Завдання: розрахувати індикацію критичних значень показників загроз для країни в цілому, групи порівняння макрорайону та відносно окремого регіону.

Дано: $X_{i,j}$ – набір значень показників загроз.

Проблема: $\langle\langle X_{i,j}\rangle, \langle R_{i,j}, R'_{i,j}, P_{i,j}, P'_{i,j}, I_{i,j}, I'_{i,j}\rangle\rangle$, де $R'_{i,j}, P'_{i,j}, I'_{i,j}$ – критерії, які пройшли індикацію.

Аксіоми:

1. Лінійне нормування для країни в цілому: $(\langle D_{i,j}\rangle, \langle R_{i,j}\rangle)$.

2. Лінійне нормування для макрорайону: $(\langle D_{i,j}\rangle, \langle P_{i,j}\rangle)$.

3. Агрегування критеріїв $R_{i,j}, P_{i,j}$: $(\langle R_{i,j}, P_{i,j}\rangle, \langle K_{i,j}\rangle)$.

4. Лінійне нормування для окремого регіону: $(\langle K_{i,j}\rangle, \langle I_{i,j}\rangle)$.

5. Індикація критичних значень загроз для країни: $(\langle R_{i,j}\rangle, \langle R'_{i,j}\rangle)$.

6. Індикація критичних значень загроз для макрорайону: $(\langle P_{i,j}\rangle, \langle P'_{i,j}\rangle)$.

7. Індикація критичних значень загроз для регіону: $(\langle I_{i,j}\rangle, \langle I'_{i,j}\rangle)$.

Пошук рішення:

Аксіома 7: $(\langle I_{i,j}\rangle, \langle I'_{i,j}\rangle)$.

Спрощення за правилом 5: $R_{i,j}, R'_{i,j}, P_{i,j}, P'_{i,j}, I_{i,j}, I_{i,j} \Leftrightarrow R_{i,j}, R'_{i,j}, P_{i,j}, P'_{i,j}, I_{i,j}$.

Аксіома 4: $(\langle K_{i,j}\rangle, \langle I_{i,j}\rangle)$.

Аксіома 3: $(\langle R_{i,j}, P_{i,j}\rangle, \langle K_{i,j}\rangle)$.

Спрощення за правилом 5: $R_{i,j}, R'_{i,j}, P_{i,j}, P'_{i,j}, R_{i,j}, P_{i,j} \Leftrightarrow R_{i,j}, R'_{i,j}, P_{i,j}, P'_{i,j}, P_{i,j}$.

Спрощення за правилом 5: $R_{i,j}, R'_{i,j}, P_{i,j}, P'_{i,j}, P_{i,j} \Leftrightarrow R_{i,j}, R'_{i,j}, P_{i,j}, P'_{i,j}$.

Аксіома 5: $(\langle R_{i,j}\rangle, \langle R'_{i,j}\rangle)$.

Спрощення за правилом 5: $R_{i,j}, R_{i,j}, P_{i,j}, P'_{i,j} \Leftrightarrow R_{i,j}, P_{i,j}, P'_{i,j}$.

Аксіома 1: $(\langle D_{i,j}\rangle, \langle R_{i,j}\rangle)$.

Аксіома 6: $(\langle P_{i,j}\rangle, \langle P'_{i,j}\rangle)$.

Спрощення за правилом 5: $D_{i,j}, P_{i,j}, P_{i,j} \Leftrightarrow$ $\Leftrightarrow D_{i,j}, P_{i,j}$.

Аксіома 2: $(\langle D_{i,j}\rangle, \langle P_{i,j}\rangle)$.

Спрощення за правилом 5: $D_{i,j}, D_{i,j} \Leftrightarrow D_{i,j}$.

Рішення знайдено.

Вивід, зображений на рис. 7, становить множину трійок вершин: $\langle D; D, D\rangle$, $\langle D, D; (\langle D\rangle, \langle P\rangle); D, P\rangle$, $\langle D, P; D, P, P\rangle$, $\langle D, P, P; (\langle P\rangle, \langle P'\rangle); D, P, P'\rangle$, $\langle D, P, P'; (\langle D\rangle, \langle R\rangle); R, P, P'\rangle$, $\langle R, P, P'; R, R, P, P'\rangle$, $\langle R, R, P, P'; (\langle R\rangle, \langle R'\rangle); R, R', P, P'\rangle$, $\langle R, R', P, P', R, R', P, P', P\rangle$, $\langle R, R', P, P', R, P\rangle$, $\langle R, R', P, P', R, P; (\langle R, P\rangle, \langle K\rangle); R, R', P, P', K\rangle$, $\langle R, R', P, P', K; (\langle K\rangle, \langle I\rangle); R, R', P, P', I\rangle$, $\langle R, R', P, P', I; R, R', P, P', I, I\rangle$, $\langle R, R', P, P', I, I; (\langle I\rangle, \langle I'\rangle); R, R', P, P', I, I'\rangle$

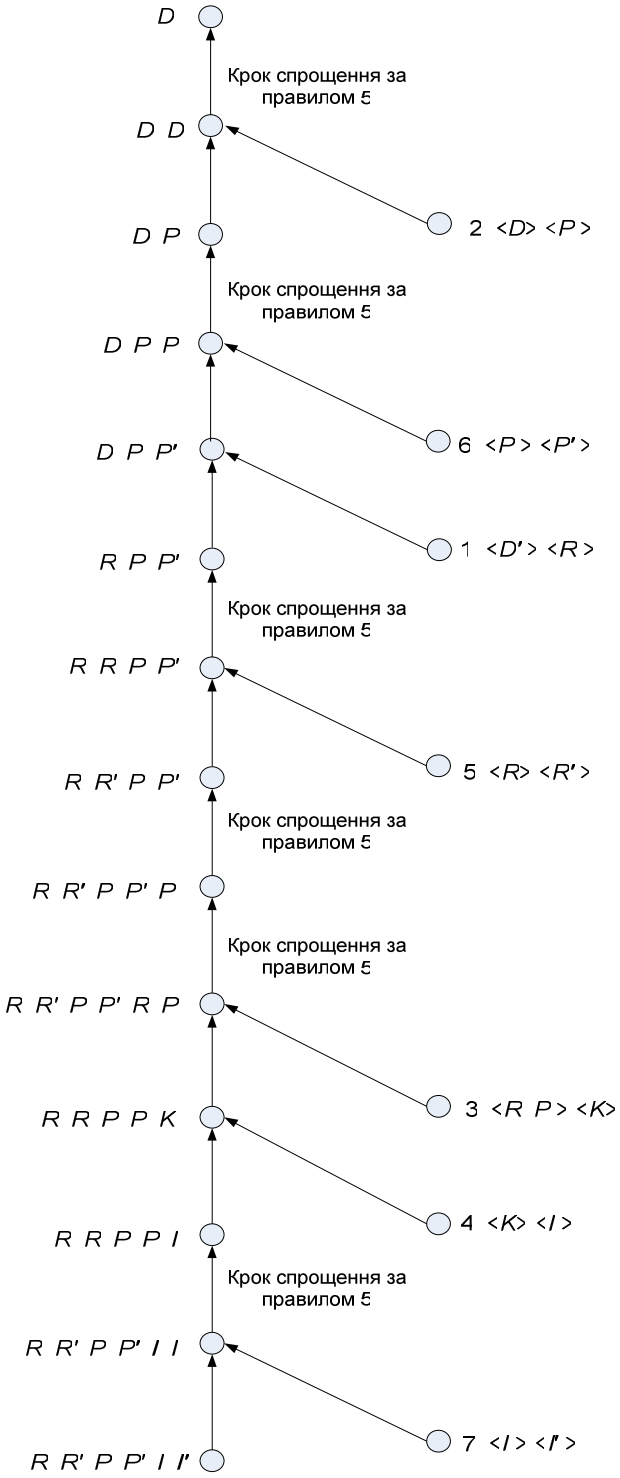

Рис. 7. Вивід для завдання розрахунку показників загроз

Побудоване на основі виводу дерево розв'язання завдання розрахунку показників загроз наведено на рис. 8.

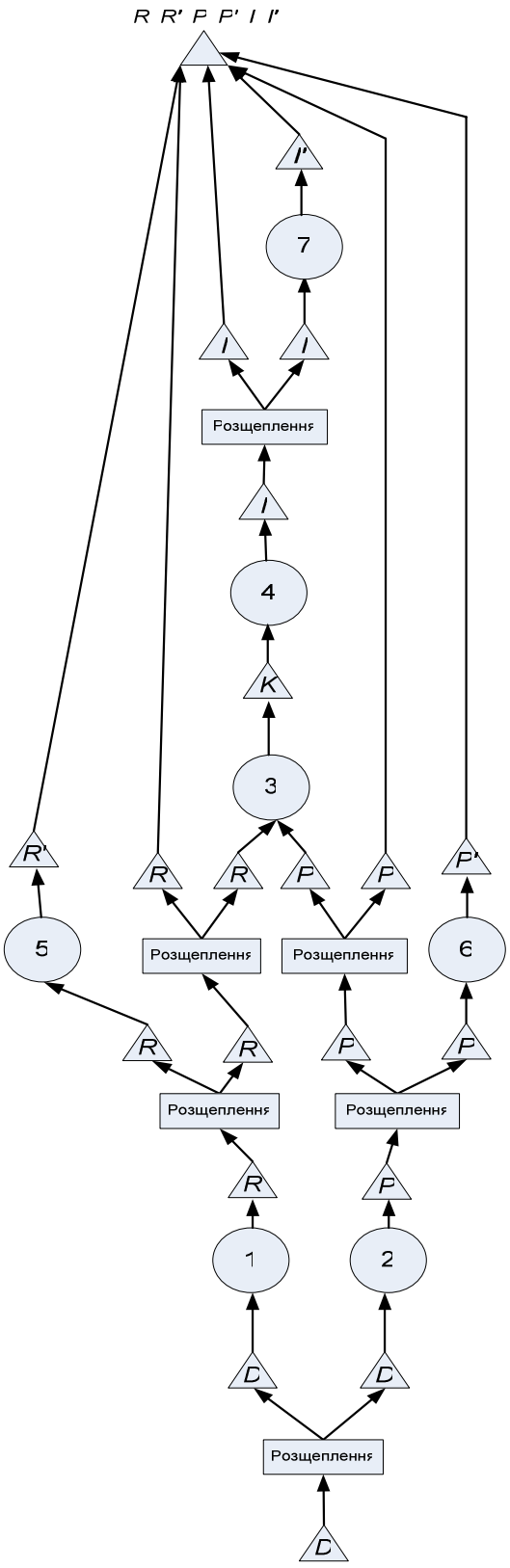

Рис. 8. Дерево розв'язання завдання розрахунку показників загроз

Знайдена послідовність виконання:

1. Аксіома 2: $(\langle D_{i,j}\rangle, \langle P_{i,j}\rangle)$.
2. Аксіома 6: $(\langle P_{i,j}\rangle, \langle P'_{i,j}\rangle)$.
3. Аксіома 1: $(\langle D_{i,j}\rangle, \langle R_{i,j}\rangle)$.
4. Аксіома 5: $(\langle R_{i,j}\rangle, \langle R'_{i,j}\rangle)$.
5. Аксіома 3: $(\langle R_{i,j}, P_{i,j}\rangle, \langle K_{i,j}\rangle)$.
6. Аксіома 4: $(\langle K_{i,j}\rangle, \langle I_{i,j}\rangle)$.
7. Аксіома 7: $(\langle I_{i,j}\rangle, \langle I'_{i,j}\rangle)$.

Висновки

На основі аналізу існуючих центрів даних, їх обладнання та програмного забезпечення запропоновано якісне рішення, що надає простий і гнучкий спосіб інтегрування різнорідних інформаційних систем і їх сервісів у Світову систему даних. Однією з ключових особливостей запропонованого рішення є автоматизація побудови ланцюжка дій, що виконує запит користувача. Створено логічний формалізм як основу рішення і на його базі розроблені метод виведення і механізм побудови дерева виконання запиту користувача.

Аналіз доступних технологій, які використовуються в реалізації розподілених систем, дав можливість для практичної імплементації рішення використовувати сучасний комплекс технологій: UDDI для створення реєстру внесених до системи сервісів; WSDL для уніфікованого опису сервісів; SOAP для обміну повідомленнями між сервісами; BPEL для реалізації загальної координації роботи сервісів. Інтелектуальні агенти можуть реалізовуватися за допомогою JADE.

Реалізація запропонованого рішення надасть можливість комплексно використовувати всі об'єднані обчислювальні потужності та системи зберігання даних Світової системи даних. Таким чином, користувачі легко зможуть отримати доступ до всіх необхідних ресурсів і сервісів, які є у розпорядженні системи.

1. *M.Z. Zgurovsky et al.,* "Integration of the Ukrainian science into the world data system", Cybernetics and Systems Analysis, vol. 46, is. 2, pp. 211–219, 2010.
2. *Аналіз* сталого розвитку – глобальний і регіональний контексти / М.З. Згуровський, А.О. Болдак, К.В. Єфремов та ін. // Міжнар. рада з науки (ICSU). – К.: НТУУ "КПІ", 2012. – Ч. 2. Україна в індикаторах сталого розвитку (2011–2012). – 232 с.
3. *Павлов А.О., Теленик С.Ф.* Алгоритмизация и ИТ в управлении. – К.: Техника, 2002. – 320 с.
4. *Системи* управління хмарними IT-структурами / С.Ф. Теленик, О.І. Ролік, М.В. Ясочка, О.С. Квітко // Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту: Матеріали Міжнар. наук. конф., 16–20 травня 2011 р., м. Євпаторія. – Херсон: ХНТУ, 2011. – **1**. – С. 124–127.
5. *Когаловский М.Р.* Методы интеграции данных в информационных системах. – Режим доступу: http://www.cemi.rssi.ru/mei//articles/kogalov10-05.pdf
6. *IEEE* P1903TM/D1 Draft White Paper for Next Generation Service Overlay Network, NGSON Working Group of the IEEE Standards Committee. New York, May 2008, 28 p.
7. *A.Y. Levy,* "Logic-Based Techniques in Data Integration. Logic-based Techniques in Data Integration", in Logic Based Artificial Intelligence, J. Minker, Ed. Dordrecht, Netherlands: Kluwer Publishers, 2000.
8. *Шаховська Н.Б.* Структура та задачі простору даних // Складні системи і процеси. – 2008. – № 1. – С. 73–86.
9. *Шаховська Н.Б.* Методи опрацювання консолідованих даних за допомогою просторів даних // Проблеми програмування. – 2011. – № 4. – С. 72–84.
10. *Вельбицкий Н.В., Ходаковский В.Н., Шолмов Л.И.* Технологический комплекс производства программ на машинах ЕС ЭВМ и БЭСМ-6. – М.: Статистика, 1980. – 263 с.
11. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. – К.: Наук. думка, 1991. – 213 с.
12. *G. Wiederhold,* Mediators in the Architecture of Future Information Systems, IEEE Computer, vol. 25, no. 3, pp. 38–49, 1992.
13. *Вейдж Д., Миллер Х.Г.* Вычисления в облаке: будут ли полезны открытые сервисы? // Открытые системы. – 2010. – № 1. – С. 68–75.
14. *Маслянко П.П.* Системне проектування інтелектуалізованих інформаційно-комунікаційних систем // Таврический вестник информатики и математики. – 2008. – № 2. – С. 62–67.
15. *OMG* Systems Modeling language (OMG SysML) V1.0 [Online]. Avaliable: http://www.omg.org/docs/formal/07-09-01.pdf
16. *UDDI* Version 3.0.2 Specification [Online]. Avaliable: http://uddi.org/pubs/uddi_v3.htm
17. *Web* Services Description Language (WSDL) 1.1 [Online]. Avaliable: http://www.w3.org/TR/wsdl
18. *SOAP* Version 1.2 Part 1: Messaging Framework (Second Edition) [Online]. Avaliable: http://www.w3.org/TR/soap12-part1/
19. *Windows* Communication Foundation Architecture Overview [Online]. Avaliable: http://msdn.microsoft.com/en-us/library/aa480210.aspx
20. *Matjaz* Juric. BPEL and Java [Online]. Avaliable: http://www.theserverside.com/news/1364554/BPEL-and-Java
21. *Workflow* Management System Taverna [Online]. Available: http://www.taverna.org.uk/
22. *Каталогизация* и интеграция разнородных информационных ресурсов / С.Ф. Теленик, С.В. Жук, В.Т. Лыско, К.В. Ефремов // Молодой ученый. – 2013. – № 5. – С. 176–179.

Рекомендована Радою
факультету інформатики
та обчислювальної техніки
НТУУ "КПІ"

Надійшла до редакції
30 вересня 2013 року