Рогушина Ю.В., Гладун А.Я.,
Осадчий В.В., Прийма С.М.

# Онтологічний аналіз у Web

Монографія

Мелітополь 2015

УДК 004.414.2:111.1
ББК 32.973-01
О 59

Рекомендовано до друку вченою радою Мелітопольського державного педагогічного університету ім. Богдана Хмельницького (протокол № 5 від 25.11.2015 р.)

**Рецензенти:**
***Боргест Н.М.*** **–** кандидат технічних наук, професор Самарського державного авіатехнічного університету, м. Самара, РФ, член Міжнародної асоціації з онтологій та їх застосувань IAOA, випусковий редактор наукового журналу "Онтология проектирования»;
***Валькман Ю.Р.*** **–** доктор технічних наук, професор, Міжнародного науково-навчального центру інформаційних технологій і систем НАН та МОН України, м.Київ;
***Зелик Я.І.*** **–** доктор технічних наук, провідний науковий співробітник відділу космічних інформаційних технологій та систем Інституту космічних досліджень НАН України та ДКА України, м.Київ.

**О 59** **Онтологічний аналіз у Web** : монографія / [Ю.В. Рогушина, А.Я. Гладун, В.В. Осадчий, С.М. Прийма]. — Мелітополь: МДПУ ім. Богдана Хмельницького, 2015. — 407 с.

ISBN 978-617-7346-27-1

Монографію присвячено проблематиці розробки, дослідження та використання онтологій в розподілених застосуваннях. Проаналізовано моделі та методи подання онтологій, їх зв'язок з технологіями Semantic Web. В роботі аналізуються питання, що стосуються здобуттям онтологічних знань з відкритих джерел Web, Wiki-ресурсів та природномовних документів. Розглядається роль онтологічного аналізу в інтелектуалізації пошукових систем, Web-сервісів та програмних агентів. Наводяться приклади застосування онтологій в освіті та науці.

Робота орієнтована на дослідників та науковців, які займаються розробками в галузі розподілених інтелектуальних систем.

ISBN 978-617-7346-27-1
УДК 004.414.2:111.1
ББК 32.973-01

## КОРОТКИЙ ЗМІСТ

## ЗМІСТ

# ВСТУП

Сьогодні головні напрями розвитку інформаційних технологій (ІТ) пов'язані зі створенням інтелектуальних інформаційних систем (ІС), які використовують знання з відповідних предметних областей (ПрО). Ці ІС застосовують методи, запозичені зі штучного інтелекту (ШІ), які забезпечують ефективну обробку, аналіз і генерацію нових знань. Крім того, значна частина сучасних ІС орієнтована на розподілену обробку інформації та на функціонування у відкритому інформаційному просторі Web. Одним з найважливіших завдань ШІ нині є побудова загальної комплексної теорії інтелектуальних ІС, яка дала б змогу інтегрувати різні напрями ШІ (теорію подання знань, теорію розв'язання задач – логіки, евристики та стратегії, теорію програм) з архітектурою розподілених інтелектуальних ІС та інтелектуальними користувацькими інтерфейсами.

Сьогодні ШІ фокусується не стільки на розробці окремих напрямів, скільки на їх глибокій семантичній інтеграції, метою якої має стати узагальнена технологія комплексного проектування інтелектуальних ІС на основі семантичних моделей обробки знань.

Через складність здобуття знань важливого значення набуває проблема забезпечення їх інтероперабельності й повторного використання. Перспективним підходом до розв'язання цієї проблеми є *онтологічний аналіз*: онтології базуються на ґрунтовному теоретичному базисі дескриптивних логік, для них уже створені загальноприйняті стандарти опису, мови й програмні засоби.

Проблема здобуття знань з Web є складовою найрізноманітніших ІС і загалом пов'язана з інтелектуальним аналізом даних і розпізнаванням властивостей інформаційних об'єктів, що стосуються задачі, яку розв'язує користувач.

Використання онтологій забезпечує, наприклад, виконання пошуку інформації у Web на семантичному рівні (а подібне завдання часто є складником сучасних інтелектуальних застосувань, що функціонують у відкритому інформаційному середовищі). Такий пошук надає можливість отримувати не просто документи або дані, а *знання* про потрібні користувачам інформаційні об'єкти (ІО), які можуть мати досить складну й певним чином формалізовану структуру. Приміром, окремим випадком такого пошуку є знаходження семантичних Web-сервісів, що потрібні для розв'язання проблеми користувача, або пошук програмного агента, який реалізує відповідні функції.

Актуальними на сьогодні є й питання застосування методів Data Mining для поповнення онтологій з ресурсів Web, які мають власну специфіку: великий обсяг і його швидке зростання, що потребує

високошвидкісної обробки та отримання результатів, динамічність і суперечність наявної інформації, значна гетерогенність різних типів структурованих і напівструктурованих даних.

Інтерес до цього напряму досліджень у нашій країні зростає, підтвердженням чого є щорічні міжнародні конференції, які проводяться в Україні (зокрема «Інтелектуальний аналіз інформації» (НТУУ–КПІ), KDS «Знание – Диалог – Решение», УкрПрог (Інститут програмних систем НАНУ)).

У рамках цієї роботи розглянуто питання щодо створення й удосконалення онтологій, а також використання їх методів у різноманітних інтелектуальних Web-застосуваннях.

У розділі I «*Теоретичні засади онтологічного аналізу»* проаналізовано засоби та методи керування знаннями в сучасних інтелектуальних інформаційних системах, обґрунтовано доцільність використання в них онтологічного аналізу, наведено різні підходи до визначення, класифікації та оцінки онтологій, а також розглянуто використання дескриптивних логік як теоретичного базису для онтологічного подання знань.

У розділі II «*Аналітичний огляд використання онтологій у Semantic Web»* окреслено технології, мови та стандарти Semantic Web, що використовуються для керування знаннями, засоби логічного виведення на онтологіях і програмні засоби для створення й редагування онтологій (Protégé, Fluent Editor).

У розділі III «*Роль онтологій в інтелектуальних Web-застосуваннях»* розглянуто різні сфери застосування онтологічного аналізу у Web, проаналізовано засоби інтелектуалізації поведінки програмних агентів і мультиагентних систем на основі онтологічних моделей, системи доступу до даних на основі онтологій, використання онтологій у корпоративних системах і в Business Intelligence, а також у застосуваннях Semantic Grid.

У розділі IV «*Розробка онтологічних моделей і методів семантичного пошуку у Web»,* який присвячено методам семантичного інформаційного пошуку, проаналізовано сучасні інформаційно-пошукові системи, доведено доцільність використання онтологічних знань у пошуку інформаційних об’єктів, запропоновано методи використання тезаурусів як засобу моделювання знань у пошуку природномовних інформаційних ресурсів та основи семантичної розмітки. Наведено результати аналізу методів і засобів інформаційного пошуку в Semantic Web. Семантичний пошук розглянуто на рівні зіставлення онтологічної моделі інформаційної потреби користувача з онтологічною моделлю інформаційного ресурсу. Для моделювання знань запропоновано використання тезаурусів у пошуку

в природномовних інформаційних ресурсах. З'ясовано призначення та методи семантичної розмітки природномовних текстів.

У розділі V «*Інтелектуалізація Web-сервісів на основі онтологій*» відображено концепція Web-сервісів та сервіс-орієнтована архітектура. Розглянуто основні стандарти, пов'язані з використанням Web-сервісів: SOAP, WSDL і UDDI; проаналізовано використання онтологій у пошуку семантичних Web-сервісів на основі їх семантичної розмітки за допомогою термінів з онтології OWL-S.

У розділі VI «*Використання онтологій для інтелектуального аналізу даних у Web»* наведено технології інтелектуального аналізу даних як засобу здобуття онтологічних знань з інформаційних ресурсів і структури Web, розглянуто використання Text Mining і систем автоматичного реферування для обробки природномовних документів, презентованих у Web, запропоновано алгоритми семантичної розмітки текстів на основі онтологій предметних областей, що цікавлять користувачів, розроблено метод поповнення онтологій на основі такої розмітки.

У розділі VII «*Використання відкритих джерел Web для поповнення онтологічних моделей предметних областей»,* що присвячений технологіям колективного збирання й аналізу інформації на основі технологій Wiki, з'ясовано, як семантичні Wiki можуть стати основою для динамічного поповнення онтологій предметних областей тими знаннями, що містяться в ресурсах Web.

У розділі VIII «*Застосування онтологічного аналізу для дослідження компетенцій в освіті й науці*» запропоновано методи використання онтологій для аналізу компетенцій експертів з наукових досліджень і для контролю за навичками студентів у мультиагентних системах е-навчання; наведено онтологічну модель як основу адаптивності відкритої освіти дорослих, проаналізовано шляхи та методи використання онтологій як підґрунтя для розвитку єдиного європейського освітнього простору, що забезпечує прозорість європейської та національних рамок кваліфікацій.

Додатки містять онтології предметних областей, що використовуються в роботі для демонстрації можливостей програмних систем, відображених за допомогою мови OWL.

У роботі узагальнено результати авторських теоретичних і прикладних досліджень, що знайшли застосування в національних і міжнародних науково-дослідних проектах і були реалізовані в прикладних інтелектуальних інформаційних системах.

# РОЗДІЛ І.
# ТЕОРЕТИЧНІ ЗАСАДИ ОНТОЛОГІЧНОГО АНАЛІЗУ

## 1. 1. Використання знань в інтелектуальних інформаційних системах

Інтелектуальна система – це система, здатна розв'язувати задачі, які традиційно вважаються творчими і вимагають від людини як інтелектуальних зусиль, так і знань з відповідної ПрО. Саме з вивченням таких систем пов'язана група наук, об'єднана під назвою «штучний інтелект». Інтелектуальна ІС – це складна система, що може сприймати, аналізувати, перетворювати й зберігати моделі об'єктів.

Основною характерною властивістю інтелектуальних ІС є їх здатність використовувати знання для досягнення очікуваних результатів у своїй роботі.

На рівень інтелектуальності системи впливають як здатність до навчання й самонавчання, тобто до використання наявних знань у нових, заздалегідь невідомих ситуаціях, так і широта її ПрО [19]. Поняття інтелектуалізації ІС охоплює такі аспекти, як підвищення «грамотності» ІС, наявність метазнань ПрО та розширення способів здобуття нових знань [2]. Чим вищою є інтелектуальність ІС, тим ефективнішим буде досягнення поставленої мети [58]. Нові знання формуються завдяки застосуванню наявних знань (інформації про процеси розв'язання, логічне виведення, закономірності ПрО тощо) до даних, що породжують нову інформацію [56]. Однак такі властивості, як «широта ПрО» й «нові знання», складно оцінити кількісно.

У [22] рівень інтелектуальності ІС $I(A) = S(s_1,\ldots,s_9), i = \overline{1,9}$ визначається наявністю в неї таких властивостей:

- $s_1$ – автономність роботи системи;
- $s_2$ – взаємодія з іншими об'єктами;
- $s_3$ – сприймання інформації про навколишній світ;
- $s_4$ – застосування абстракції;
- $s_5$ – використання знання ПрО;
- $s_6$ – адаптивність поведінки ІС;
- $s_7$ – здатність до навчання на власному досвіді;
- $s_8$ – толерантність до помилок;
- $s_9$ – здатність до спілкування природною мовою.

У [96] пропонується кількісна оцінка рівня інтелектуальності ІС: $I(A) = \sum_{i=1}^{6} s_i * w_i$, де $w_i$ – вага і–ї властивості, а $s_i$ – значення і–ї властивості:

- $s_1$ – використання моделі навколишнього світу для формування планів власних дій;
- $s_2$ – наявність альтернативних варіантів при плануванні дій;
- $s_3$ – здатність до коригування плану в процесі його реалізації, якщо виконувані дії призводять до небажаних наслідків;
- $s_4$ – використання власного досвіду для розширення й корекції моделі ПрО;
- $s_5$ – спілкування з користувачем природною мовою;
- $s_6$ – допустимість періоду реалізації плану розв'язання задачі.

Більш детально критерії оцінки рівня інтелектуальності ІС і методи їх порівняння проаналізовано в [109, 127].

Слід відрізняти окремі випадки й категорії інтелектуальних ІС – системи керування знаннями; системи, що керуються знаннями; експертні системи; системи підтримки ухвалення рішень тощо.

*Комп'ютерна система, керована знаннями*, – це система, заснована на уніфікованій базі знань, що містить у систематизованому вигляді всю ту інформацію, яку використовує ця інформаційна система [59].

Актуальність ІС, керованих знаннями, зумовлюється тим, що сьогодні важливого значення набувають задачі, для яких апріорі важко локалізувати інформацію, достатню для їх розв'язання. У процесі розв'язання таких задач може знадобитися заздалегідь непередбачена інформація, що задовольнятиме тим чи іншим вимогам. Вихідними даними для таких задач може вважатися вся інформація, до якої така ІС здатна отримати доступ (приміром, через середовище Web).

Приклади задач, які розв'язують ІС, керовані знаннями:

- інформаційно-пошукові задачі, у яких за описом потрібної інформації необхідно здобути з інформаційних ресурсів інші характеристики цієї інформації;
- задачі інтеграції інформації, призначені для побудови нових знань у придатній для користувача формі за наявною інформацією та знаннями щодо цієї інформації;
- задачі, для яких апріорі невідомі способи й програмні засоби їх розв'язання (ці способи потрібно віднайти з множини доступних для ІС програм, скомпонувати, скомбінувати, наприклад, побудувати композицію Web-сервісів);
- задачі логічного виведення.

Уніфікований спосіб кодування різних видів знань є необхідним для забезпечення сумісності різних видів знань і різних мов подання знань, що так само потрібно для інтеграції знань [82], а також для інтеграції різних мов подання знань. Без узгодження знань, що

інтегруються, з якоюсь загальновживаною уніфікованою формою, інтеграція неможлива.

Уніфікація семантичних моделей обробки знань припускає такі їх елементи:

- структуризація баз знань і типологія знань;
- моделі інформаційного пошуку;
- моделі інтеграції знань;
- моделі розв'язання задач;
- уніфікація моделей користувацьких інтерфейсів;
- візуалізація баз знань.

У [59] наводиться кілька визначень системи, керованої знаннями. Система, керована своєю базою знань, розглядається як багатоагентна комп'ютерна система, агенти якої взаємодіють, обмінюються інформацією тільки через її базу знань і керуються її базою знань.

Не слід ототожнювати поняття «*система, керована знаннями» й «система, що базується на знаннях»*. Поняття «система, що базується на знаннях» є більш загальним, оскільки в цьому разі БЗ не обов'язково має бути активною та семантично структурованою. Це значить, що керування в системі, що базується на знаннях, може здійснюватися програмою, а не ситуаціями та подіями в оброблюваній БЗ. А це, так само, означає, що обробка інформації в системі, яка базується на знаннях, не обов'язково здійснюється колективом агентів, що працюють над базою знань взаємодіють через неї.

Системи, керовані знаннями, варто вважати важливим етапом еволюції систем, заснованих на знаннях, що забезпечує більш високий рівень їх гнучкості.

Поняття системи, керованої знаннями, не слід плутати з поняттям *інтелектуальної системи*, тобто системи, здатної розв'язувати інтелектуальні задачі. Хоча очевидно: якщо інтелектуальну систему побудувати як систему, керовану знаннями, то вона набуде здатності до прискореного розвитку.

Інтелектуалізація ІС – це процес підвищення рівня інтелекту ІС. Пропонуємо таке визначення.

Нехай є дві ІС – А і В, здатні знаходити розв'язання $R_A(d)$ і $R_B(d)$ відповідно в ситуаціях d з визначеної ПрО D. Система В отримана внаслідок інтелектуалізації А, $B = Int(A)$, якщо:

- система В завжди одержує розв'язання в ситуаціях, у яких його знаходить і система А, $\exists R_A(d) \Rightarrow \exists R_B(d)$;
- є ситуації, у яких А не знаходить розв'язання, а В знаходить.

Перехід сучасних комп'ютерних систем до систем, що базуються на знаннях, є найважливішою тенденцією переходу до наступного

покоління комп'ютерних систем. При цьому важливо підкреслити, що будь-яку сучасну комп'ютерну систему можна реалізувати в архітектурі систем, керованих знаннями. Така трансформація сучасних комп'ютерних систем суттєво підвищить їх якість і конкурентоздатність. Унаслідок трансформації сучасних комп'ютерних систем у функціонально еквівалентні системи, керовані знаннями, ми отримаємо різні, але семантично сумісні системи, що відрізняються структурною складністю баз знань, функціональною складністю агентів обробки знань і здатні до формування тимчасових колективів комп'ютерних систем за необхідності колективного розв'язання складних задач.

Не варто також плутати поняття «система, керована знаннями» з поняттям «*система керування знаннями*» [21, 190]. Система керування знаннями є системою окремого виду, що базується на знаннях, і являє собою корпоративну систему, яка автоматизує процеси створення, поширення й використання інформації всередині деякого підприємства [70]. Але сучасні інтелектуальні ІС потребують розвитку технологій, що застосовуються в системах керування знаннями, з урахуванням нових розподілених засобів доступу до інформації та знань, а також стандартів і засобів подання й аналізу знань у Web.

Можна класифікувати комп'ютерні системи, керовані знаннями, за такими ознаками:

- рівень інтелектуальності засобів обробки знань;
- рівень розвитку рецепторно-ефекторних засобів взаємодії із зовнішнім середовищем.

## 1. 2. Керування знаннями в сучасних Web-застосуваннях

*Керування знаннями* – це сукупність процесів, пов'язаних з ефективним створенням, збереженням, поширенням і використанням знань для досягнення цілей [2, 71].

Процес керування знаннями поширюється в чотирьох основних напрямах:

1) дослідження знань та їх систематизація;

2) усвідомлення знань і визначення їхньої цінності;

3) планування й виконання дій відповідно до результатів аналізу знань;

4) постійна актуалізація та переосмислення знань.

У 1986 р. американським ученим Карлом Віїгом на конференції в Швейцарії, яка проводилася Міжнародною організацією праці під егідою ООН, було вперше використано термін «керування знаннями» («knowledge management») за аналогією до таких понять, як

«керування даними» й «керування інформацією». Сьогодні наявні досить суперечливі визначення цього терміна, що відображають різні аспекти, пов'язані з ефективним використанням знань у різних сферах.

Основні проблеми керування знаннями у Web пов'язані з:

- *інтеграцією* знань, отриманих від різних ІР (наприклад, інтеграція онтологій кількох різних ПрО або ІР в одній ПрО);
- пошуком *протиріч* між знаннями, що відображені в контенті різних ІР, і оцінкою їх достовірності та надійності;
- *здобуттям нових знань* з уже наявних і їх поданням у формі, зрозумілій користувачеві;
- *пошуком* знань, потрібних конкретному користувачеві для розв'язання певних задач;
- автоматизацією *створення метаданих*, що коректно відображають контент ІР (як текстових, так і мультимедійних) на рівні змісту, й ефективним пошуком у таких метаописах.

Можна навести ще багато прикладів проблем, що постають у процесі керування знаннями у Web, але всі вони зводяться до таких:

1. Вибір засобів подання знань (досить потужних, щоб задовольнити різноманітні потреби користувачів, але придатних для швидкої обробки й розуміння людиною);
2. Методи створення нових знань за наявною інформацією (приміром, створення метаопису ІР, формування онтології ІР, логічне – індуктивне, традуктивне, дедуктивне – виведення, виконання запитів до БЗ);
3. Методи порівняння різних інформаційних об'єктів на семантичному рівні (наприклад, інтеграція або пошук відмінностей двох онтологій, співвідношення інформаційного запиту й ІР, що відповідають цьому запиту, визначення ПрО за контекстом ІР);
4. Оцінка якості нових знань (достовірність, несуперечливість, актуальність, повнота).

Керування знаннями у сучасному Web має певну специфіку, що визначається як великими обсягами даних, які обробляються, так і динамічністю, гетерогенністю й децентралізованістю даних.

Особливий інтерес становить застосування онтологічного аналізу як основи для обробки розподілених знань. Сьогодні значна кількість досліджень пов'язана з теоретичним базисом онтологій, їх побудовою, удосконаленням, зі здобуттям знань з онтологічних структур, а також з іншими важливими аспектами менеджменту онтологій, які значно різняться за метою та призначенням такого аналізу.

Можна розглянути ще багато напрямів, у яких ведуться такі дослідження, і окреслити безліч проблем, що виникають у процесі використання онтологій у застосовних інформаційних системах [73].

Одним з найбільш популярних є проект Semantic Web, запропонований автором WWW Тімом Бернесом-Лі (інша назва цього проекту – «гігантський глобальний граф» (giant global graph – GGG)). Основна ідея цього проекту полягає в перетворенні Web на глобальну базу знань (рис. 1. 1).

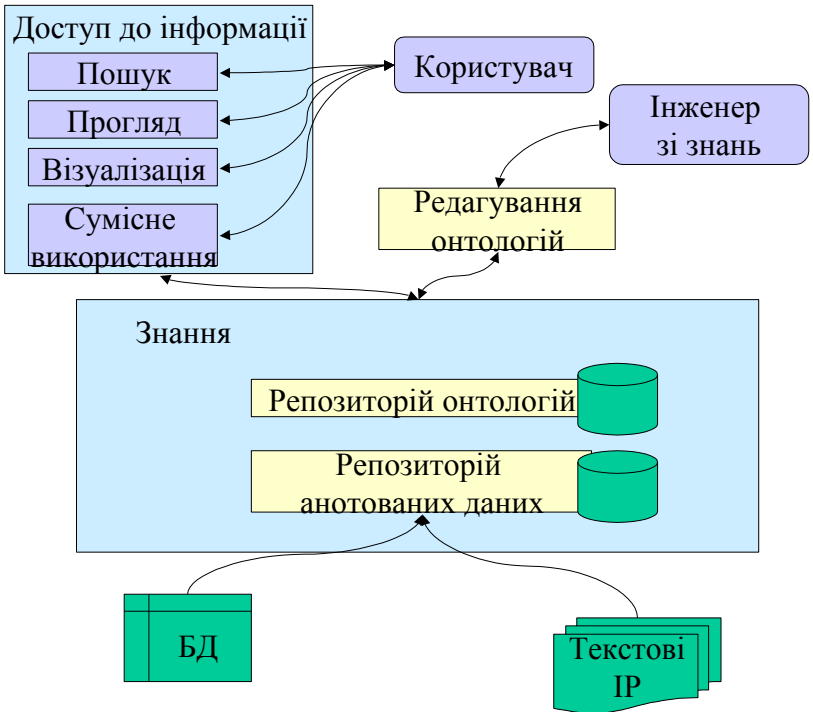



Рис. 1. 1. Архітектура керування знаннями на основі Semantic Web

### 1. 3. Визначення онтології ПрО

Сьогодні в різноманітних наукових дослідженнях, присвячених онтологічному аналізу, пропонується багато різних тлумачень терміна «онтологія предметної області. Однак кожне з них має певні недоліки. Через те, що для розв'язання різноманітних проблем, пов'язаних з онтологіями ПрО, використовуються різні трактування цього поняття, що безпосередньо залежать від специфіки задач, які розв'язуються за допомогою онтологій, можна дійти висновку, що на сьогодні немає загальноприйнятого визначення цього терміна.

Докладніше різні підходи щодо визначення онтології ПрО, їх переваги та недоліки проаналізовано в [86].

У нашій роботі виділяємо такі підходи до визначення онтології ПрО:

- гуманітарний;
- математичний;
- комп'ютерний.

### 1. 3. 1. Гуманітарний підхід

Гуманітарний підхід до визначення поняття «онтологія предметної області» здебільшого передбачає визначення в термінах, що інтуїтивно розуміються, але не формалізовані. Розглянемо приклади таких визначень.

- Онтологія – це консенсус щодо предметної області для визначених цілей, тобто онтологія в цьому трактуванні пояснюється настільки вільно, наскільки це можливо без втрати її сутності [352].
- Онтологія – це теорія про те, які об'єкти та поняття можуть бути у свідомості агента, що володіє знаннями [467]: БЗ це модель певної частини світу, представлена засобами деякої мови, яка має словник і синтаксис. Це дає змогу, задавши механізм виведення, будувати логічні висновки про цей світ, а онтологія, яка визначає словник і структуру тверджень, що виражаються елементами моделі, додатково до обмежень, що накладаються цим синтаксисом, визначає обмеження на те, що можна презентувати в цій моделі.
- Онтологія складається з понять та їх визначень, ієрархічної організації понять, відношень між поняттями (не тільки відношень «is-a» і «part-of»), аксіом для формалізації визначень і відношень [352].
- Онтологія – це фактична специфікація концептуалізації на рівні знань, тобто множина відмінних рис, що важливі для агента [467], де концептуалізація розуміється в змісті [269]: це об'єкти, поняття та інші сутності, що передбачаються наявними в деякій предметній області, а також відношення між ними. Концептуалізація є абстрактним, спрощеним поглядом на світ, презентованим для деяких цілей. Онтологія є інтенсіональним описом знань у деякій предметній області. Вона визначає структуру і словник статичних знань предметної області: знання ПрО відображають фактичну ситуацію в предметній області, а онтологія ПрО задає обмеження щодо структури й змісту знань.
- Онтологія – фактичний опис чи презентація деякої частини концептуалізації [448], яка може мати різні форми, але обов'язково повинна містити словник термінів і деяку специфікацію їхнього змісту, визначення та вказівки стосовно зв'язку понять, що в сукупності накладає структуру на предметну область і обмежує можливі інтерпретації термінів. Онтологія фактично завжди є проявом розподіленого розуміння предметної області, з яким погоджується деяка кількість агентів.

Концептуалізація є поглядом на світ: вона відповідає способу мислення про деяку предметну область і може розглядатися як «множина неформальних правил, що обмежують структуру частини дійсності» [269].

- Онтологія, подібно до концептуалізації, відображає об'єкти й відношення ПрО у вигляді речень формальної мови в термінах об'єктів і відношень, необхідних для визначення семантики мови подання знань [204], де екстенсіональну концептуалізацію слід також розуміти в змісті [269], надаючи точне й недвозначне визначення поняттям, необхідним для подання знань, без посилання на обчислювальні аспекти.
- Онтологія – це фактична специфікація інтенсіональної концептуалізації [297]: знання ПрО презентуються з використанням декларативного формалізму «універсуму області» множини об'єктів і опису співвідношення між ними, що відображаються в словнику подання, де визначення пов'язують імена сутностей в універсумі (наприклад, класи, відношення, функції чи інші об'єкти) зі зрозумілим для людини текстом, який розкриває значення цих імен, а також з формальними аксіомами, що обмежують інтерпретацію та правильне використання цих термінів.
- Онтологія – це формальна (машинно-читабельна), фактична специфікація розподіленої концептуалізації [432], де концептуалізація абстрактна модель деякого явища у світі, задана за допомогою понять, релевантних цьому явищу, а всі використані поняття й обмеження щодо їх застосування явно визначені й прийняті деякою групою. Онтологія містить універсальні обґрунтовані знання предметної області, що не залежать від їх використання. Онтологія претендує на те, щоб відображати певний ступінь консенсусу про знання предметної області. Онтологія забезпечує терміни, їх зміст, їх відношення й обмеження тощо, і в процесі взаємодії всі учасники повинні сприйняти ці визначення.

Основна перевага гуманітарного підходу до визначення терміна «онтологія предметної області» – це намагання прояснити змістовну сутність цього поняття та інших, пов'язаних з нею понять, зокрема таких, як «концептуалізація», «знання» й «подання знань», а найсуттєвішим недоліком є неспроможність надати визначення, необхідне для розв'язання технічних проблем.

### 1. 3. 2. Комп'ютерний підхід

Комп'ютерний підхід пов'язаний з розробкою комп'ютерних мов для подання онтологій. Приклади мов опису онтологій наведено в роботах [268, 300].

Основна перевага комп'ютерного підходу – це формальність розроблених у його межах засобів для опису онтологій. Але проблема полягає в тому, що немає задовільного визначення різниці між мовами опису онтологій і мовами подання знань, які за своєю семантикою еквівалентні мовам числення предикатів. Тому не зрозуміло, наскільки такі мови спеціалізуються саме на формалізації онтологій предметних областей і як саме така спеціалізація відображається в синтаксисі та семантиці цих мов. Крім того, залишається відкритим питання про змістовну інтерпретацію конструкцій мов для опису онтологій за допомогою термінів предметних областей.

### 1. 3. 3. Математичний підхід

У межах математичного підходу робляться спроби визначити поняття онтологія в математичних термінах і за допомогою математичних конструкцій. Прикладом визначення поняття «онтологія» в математичних термінах може слугувати таке: онтологія – це логічна теорія, що обмежує припустимі моделі логічної мови, надаючи аксіоми, які задовольняють значення нелогічних символів (предикатів і функцій) логічної мови, що використовуються як «примітиви для визначених цілей подання». Концептуалізація розглядається як множина неформальних правил, що обмежують структуру частини дійсності [304].

Основний недолік цього підходу – брак фактичного зв'язку між математичними термінами й змістовними аспектами предметних областей.

У [466] модель концептуалізації предметної області визначається як багатосортна алгебраїчна система S = <U, R, F, C>, де U – множина сортів, R – множина відношень, F – множина функцій, C – множина констант, а база знань визначається як підсистема цієї алгебраїчної системи. База знань може бути задана у вигляді множини аксіом на багатосортній логічній мові, сигнатура якої збігається з сигнатурою моделі концептуалізації. Модель онтології предметної області являє собою алгебраїчну систему O, сигнатура якої – множина метатермінів, що дають змогу задавати алгебраїчні системи. Онтологія предметної області – презентація концептуалізації за допомогою проблемно-незалежних термінів, тобто визначена в такий спосіб онтологія не зв'язана з предметною областю. Інші математичні моделі онтології

ПрО наведено в [218], де світ розглядається як множина не пов’язаних одна з одною ситуацій.

Усі, наведені вище визначення онтології ПрО, доповнюють одне одного. Загалом визначення математичного підходу мають значні переваги порівняно з визначеннями комп’ютерного підходу як завдяки тому, що за такої ж чіткості мають меншу кількість технічних деталей, так і завдяки фактичній спеціалізації щодо формалізації поняття «онтологія предметної області». Однак кожне таке визначення має певні недоліки. Найбільш критичним для розвитку математичного підходу є брак фактичних припущень щодо властивостей ПрО, їх онтологій, концептуалізації і знань, характерних для гуманітарного підходу, і явного зв’язку цих припущень з елементами математичних моделей.

Яке з наведених визначень онтології є більш прийнятним, значною мірою залежить від того, для чого розробляється така онтологія. У [367] визначено такі причини розробки онтологій:

- спільне використання загального розуміння інформаційної структури людьми та/або програмними агентами;
- можливість повторного використання знань ПрО;
- можливість робити явними припущення ПрО;
- відмежування знань ПрО від операційного знання;
- аналіз знань у ПрО.

Онтологія ПрО – це та частина знань ПрО, що обмежує значення її термінів, які не залежать від іншої (змінюваної) частини знань цієї ПрО. Таку онтологію ПрО можна розглядати як набір угод про предметну область, а інша частина знань ПрО є множиною емпіричних та інших законів цієї області. Отже, онтологія визначає ступінь узгодження значень термінів між фахівцями предметної області [86].

Упродовж досить тривалого періоду досліджень і розробок у сфері моделювання ПрО, з урахуванням нових потреб і подальшого розвитку досягнутих результатів був накопичений досвід онтологічного опису ПрО як основи значеннєвого моделювання.

При формулюванні знань, що становлять зміст моделей ПрО, фундаментальну роль відіграють два когнітивні судження про світ: можливість розрізнення дискретних об’єктів та наявність зв’язків між ними [177]. Множина об’єктів, що розглядаються в контексті ситуації або задачі, становить її предметну область. Зв’язки між об’єктами визначають відношення в ПрО: унарні зв’язки інтерпретуються як властивості об’єктів, а зв’язки довільної арності відображають різні асоціації об’єктів. Потужність моделювання, властива бінарним відношенням, дає змогу передавати будь-які асоціативні відношення в ПрО. Це дає можливість презентувати ПрО у вигляді мережі

пов’язаних об’єктів, що в теорії подання знань відома як семантична мережа.

З позиції подання об’єктів ПрО ця мережа розпадається на дві підмережі: «класи» та «екземпляри». У першій підмережі є лише два сорти вершин і дуг: співіснують вершини-класи й вершини-властивості; дуги сорту «є видом» з’єднують вершини-класи, а дуги «є частиною» з’єднують кожну вершину-властивість з однією й тільки з однією вершиною-класом. Саме ця підмережа відображає понятійну структуру ПрО, визначаючи онтологію ПрО: сукупність понять про доступні для відчуття/виміру властивості об’єктів і різновиди об’єктів у контексті доступних властивостей. Підмережа «екземпляри» являє собою денотат онтології. Загалом при розв’язуванні задач онтологія становить теорію, а денотативна модель конкретизує цю теорію стосовно актуальної ситуації в ПрО, що моделюється.

На змістовному рівні онтологія ПрО – це сукупність угод (визначення термінів предметної області, їх тлумачення, твердження, що обмежують можливий зміст цих термінів, а також тлумачення цих тверджень), які є результатом домовленості між членами співтовариства, що працює в цій ПрО. Між такими властивостями ПрО, як онтологія, концептуалізація, знання та дійсність, і елементами цієї математичної конструкції повинна бути встановлена фактична відповідність.

Модель онтології кожної ПрО має містити як формальні елементи, так і їх змістовне тлумачення в термінах, зрозумілих фахівцям цієї предметної області. Більш формальне визначення онтології ПрО наведено в [87].

Будуючи онтологічні системи, намагаються дотримуватися таких правил:

- формалізації, тобто опису об’єктивних елементів дійсності за допомогою єдиних, чітко визначених зразків (термінів, моделей тощо);
- використання обмеженої кількості базових термінів (сутностей), на основі яких конструюються всі інші поняття;
- внутрішньої повноти й логічної несуперечності [207].

Формально ці правила ґрунтуються на таких принципах:

- *повнота*: категорії верхнього рівня мають вичерпно презентувати матерію; за межами цих категорій не повинно бути жодних проявів сущого.
- *науковість і проблемна орієнтація:* усі категорії та концепти онтології мають бути виражені поняттями, які використовуються в природничих і математичних науках при вивченні матеріального світу і є загальноприйнятими; при цьому частина

онтологій представлена концептами, які широко застосовуються в міждисциплінарних текстах з нейтральною, загальновживаною лексикою, а інша частина структурується під конкретну ПрО;

- *взаємозв'язок рівнів*: категорії онтології верхнього рівня розкриваються через набори концептів середнього рівня, концепти нижнього рівня використовуються для визначення термінів словника ПрО, а зв'язки між рівнями організовуються за допомогою відношень (мереологічних і таксономічних, синонімічних і специфічних для ПрО);
- *асоціативність*: концепти онтології нижнього рівня використовуються для індексування термінів ПрО на основі семантичних відношень типу «перебувати в асоціативному зв'язку» [55];
- *відображення антагонізмів:* концепти, що відображають властивості чи поняття, які мають свою протилежність чи додатковість, входять до онтології групами з визначенням антагонізму.

## 1. 4. Класифікація онтологій

На сьогодні створена велика кількість різноманітних онтологій, пов'язаних з найрізноманітнішими ПрО, опис яких здійснюється за допомогою різних мов. Ці онтології різняться за багатьма властивостями – обсягом, виразністю, призначенням, ступенем формалізації знань тощо.

З огляду на це, є різні види класифікації онтологій, що залежать від параметрів, покладених в основу класифікації. Загалом усі такі класифікації онтологій можна поділити на дві 40, 118].

*Семантичні* класифікації групують онтології за параметрами, пов'язаними зі змістом інформації, як-от: ступінь формальності презентованих знань [178]; рівень виразності [230] і рівень деталізації подання інформації [369].

За *рівнем виразності* онтології поділяють на великовагові та легковагові. *Великовагові* онтології мають високий рівень аксіоматизації, що дає змогу уникнути термінологічної та концептуальної неоднозначності через неправильну інтерпретацію й підтримувати процеси побудови складних логічних виводів. *Легковагові* онтології – це відносно прості таксономічні структури термінів з відповідними визначеннями. Вони менш аксіоматизовані. Такі онтології мають менш виразні можливості, але їх значно легше та швидше обробляти. Для великовагової онтології можна створити її легковагову версію.

За *ступенем формальності* онтології поділяють на неформальні, частково формалізовані та сильно формалізовані.

Опис неформальних онтологій здійснюється природною мовою, яка припускає неоднозначне розуміння.

Більш формалізованими є онтології, що базуються на термінах (прикладом такої онтології може слугувати проста таксономія з єдиним відношенням «клас-підклас»).

Сильно формалізовані онтології застосовуються для розв'язання складних інженерних задач і мають задавати формальну семантику термінів та їх значень (приміром, таких, як кількість і одиниця виміру) в точних і несуперечливих виразах [302].

За *рівнем детальності подання* онтології оцінюються за такими кількісними параметрами, як кількість класів та екземплярів, середня глибина структури підкласів, середня кількість підкласів класу, середня кількість екземплярів класів, кількість аксіом тощо.

У *прагматичних* класифікаціях онтологій використовуються параметри класифікації, пов'язані з можливостями застосування цих онтологій в ІС: ступінь залежності онтологій від конкретної задачі чи прикладної області; предметна область; мова подання онтологічних знань [40]; мета створення онтології та її наповнення [178].

На практиці конкретна онтологія може поєднувати властивості онтологій, орієнтованих на задачу й на ПрО, й містити терміни, характерні як для онтології задач, так і для онтології домену.

## 1. 5. Онтології як засіб подання знань у Web

### 1. 5. 1. Онтологічне подання знань

Поняття «онтологія» нині активно використовується в інформатиці та штучному інтелекті. Цей термін прийшов з філософії, де позначав частину метафізики – вчення про все суще, про його найзагальніші філософські категорії, такі, як буття, субстанція, причина, дія, явище. При цьому онтологія як наука претендувала на повне пояснення причин усіх явищ.

В інженерії знань під онтологією розуміється детальний опис деякої проблемної області, що використовується для формального й декларативного визначення її концептуалізації [299].

> Часто онтологією називають базу знань спеціального виду, яку можна розділяти, відчужувати й самостійно використовувати в рамках розглянутої ПрО [41].

Можна сказати, що онтологія – це точна специфікація певної області, яка містить у собі словник термінів цієї області та множину

логічних зв'язків (типу «елемент-клас», «частина-ціле») [46], що відображають співвідношення цих термінів між собою.

Зауважимо, що за такого підходу поняття онтології перетинається з давно прийнятим в інформатиці й лінгвістиці поняттям тезауруса. Онтології дають змогу подати поняття в такому вигляді, що вони стають придатними для машинної обробки. Нерідко онтології використовуються як посередники між користувачем та інформаційною системою, вони уможливлюють формалізацію домовленості про термінологію між членами співтовариства, наприклад, між користувачами деякого корпоративного сховища даних [30].

На відміну від звичайного словника, для онтологічних систем характерні внутрішня єдність, логічний взаємозв'язок і несуперечливість використовуваних понять [16].

Класифікувати онтології можна за різними параметрами (залежно від мети класифікації), наприклад: за ступенем залежності від конкретної задачі чи прикладної області; за мовою подання онтологічних знань та її виразних можливостей; за рівнем деталізації аксіоматизації; за предметною областю.

До цих характеристик можна додати й класифікації онтології, пов'язані з розробкою, реалізацією та супроводом онтології, але така типізація більш доречна під час обговорення питань реалізації онтологічних систем.

Класифікуючи онтології за рівнем, виділяють такі їх види [73]:

- *онтології верхнього рівня*. Такі онтології відображають найбільш загальні концепти (простір, час, матерія, об'єкт, подія, дія тощо), які не залежать від конкретної проблеми чи області. Тому вважаємо за доцільне (принаймні в теорії) уніфікувати їх для великих співтовариств користувачів.

Прикладом такої загальної онтології є комерційний проект онтології CYC [330]. Це база знань, що містить усі загальні поняття навколишнього світу, які можуть використовувати найрізноманітніші програмні засоби.

- *орієнтовані на предметну область*. Для багатьох дисциплін сьогодні розробляються стандартні онтології, що можуть застосовуватися експертами предметних областей (доменів) для спільного використання й анотування інформації у своїй сфері.

Наприклад, у сфері медицини створені великі стандартні, структуровані словники, такі, як SNOMED і семантична мережа Системи Уніфікованої Медичної Мови (the Unified Medical Language System) [48]. Також з'являються великі загальноцільові онтології. Так, програма ООН з розвитку (United Nations Development Program)

і компанія Dun&Bradstreet об'єднали зусилля для розробки онтології UNSPSC, що пропонує термінологію товарів і послуг (www.unspsc.org).

- *орієнтовані на задачу*. Це онтологія, що використовується конкретною прикладною програмою й містить терміни, що застосовуються в процесі розробки програмного забезпечення, яке розв'язує конкретну задачу.

Вона відображає специфіку програмного продукту, але може також містити деякі загальні терміни (наприклад, у графічному редакторі будуть і специфічні терміни – палітра, тип заливання, накладення шарів тощо, і загальні – зберегти й завантажити файл).

- *онтології ПрО й онтології задач* відображають, відповідно, словники, що належать визначеній ПрО (наприклад, медицина, дистанційне навчання, Інтернет-технології) чи типовій задачі (наприклад, діагностика, продаж). При цьому вони використовують спеціалізацію термінів, поданих в онтологіях верхнього рівня.
- *прикладні онтології* відображають концепти, що залежать як від онтології задач, так і від онтології домену.

Прикладом може слугувати онтологія автомобілів, будівельних матеріалів, обчислювальної техніки. Онтологія ПрО узагальнює поняття, що використовуються в деяких задачах домену, абстрагуючи їх від самих задач (так, приміром, онтологія автомобілів не залежить від будь-яких особливостей конкретних марок машин) [86].

### 1. 5. 2. Формальні моделі онтологій

Аналіз публікацій доводить, що саме онтології є адекватним та ефективним засобом для моделювання уявлень про різноманітні ПрО і дають змогу формально відображати їх семантику.

Різні наукові джерела містять різноманітні формальні моделі подання онтологій [125, 303,360], проте схожими рисами в них є такі:

- множина термінів (понять, концептів), що може поділятися на множину класів і множину екземплярів;
- множина відношень між поняттями, у якій можна фактично виділити відношення «клас-підклас», ієрархічні (таксономічні) відношення й відношення синонімії (подібності), а також функції окремий приклад відношень, для яких n-й елемент відношення однозначно визначається n-1 попередніми елементами;
- аксіоми та функції інтерпретації понять і відношень.

Формально онтологію можна представити у вигляді трійки <X,R,F>, де X – множина концептів, R – множина відношень

між концептами, F – функції інтерпретації концептів з множини X і відношень з R.

Ця модель є узагальненою, а на практиці використовують більш точні моделі. Наприклад, онтологія може визначатися як структура $O = \langle C, \leq_c, R, \sigma_R, \leq_R, A, \sigma_A, T \rangle$, що складається з:

- чотирьох множин C, R, A і T, які не перетинаються й елементи яких називають, відповідно, ідентифікаторами концептів, ідентифікаторами відношень, ідентифікаторами атрибутів і типами даних;
- структури часткового впорядкування $\leq_c$ над C з верхнім елементом $root_C$, яку називають ієрархією концептів або таксономією;
- функції $\sigma_R : R \to C^+$, яку називають ідентифікатором (сигнатурою) відношення;
- часткового впорядкування $\leq_R$ над R, яку називають ієрархією відношень, де $r_1 \leq_R r_2$ означає, що $|\sigma_R(r_1)| = |\sigma_R(r_1)|$ і $\pi_i(\sigma_R(r_1)) \leq_R \pi_i(\sigma_R(r_2))$ для $\forall i : 1 \leq i \leq |\sigma_R(r_1)|$ (тут $\pi_i(t)$ це і-й компонент кортежу t);
- функції $\sigma_A : A \to C \times T$, що називають ідентифікатором (сигнатурою) атрибута;
- множини типів даних T (наприклад, рядок, ціле).

Для структури $\leq$ виконуються такі умови: рефлексивність; антисиметричність; транзитивність; наявність верхнього елемента; супремум.

Семантика в металогіці є розділом, що вивчає відношення мовних виразів до об'єктів, які вони позначають, і їх змісту. Сучасна логічна семантика базується на роботах Г. Фреге, Б. Рассела, С. Лесневського, Р. Карнапа, А. Тарського, А. Черча [440]. Семантику онтологічних мов, зазвичай, подають через теорію моделей [198]. Вона, зокрема, визначає функцію інтерпретації, що проектує кожен елемент онтології на множину D, що називають областю інтерпретації.

*Інтерпретацією* онтології $O = \langle C, I, R, T, V, \leq, \perp, \in, = \rangle$ є пара $\langle I, D \rangle$, у якій D – область інтерпретації, а I – функція інтерпретації, така, що:

- $\forall c \in C, I(C) \subseteq D$;
- $\forall r \in R, I(r) \subseteq D \times (D \cup V)$;
- $\forall i \in I, I(i) \subseteq D$;
- $\forall t \in T, I(t) \subseteq V$;
- $\forall v \in V, I(v) \subseteq V$.

Про твердження, виражене онтологічною мовою, говорять, що воно *задовольняється інтерпретацією*, якщо інтерпретація узгоджується з цим твердженням [299].

Формула $\sigma$ в онтології $O = \langle C, I, R, T, V, \leq, \perp, \in, = \rangle$ задовольняє інтерпретації $\langle I, D \rangle$ (що позначається як $I \models \sigma$), якщо виконуються такі умови:

- $I \models c \leq c'$ тоді й тільки тоді, коли $I(c) \leq I(c')$
- $I \models r \leq r'$ тоді й тільки тоді, коли $I(r) \leq I(r')$
- $I \models c \leq c'$ тоді й тільки тоді, коли $I(c) \leq I(c')$
- $I \models c \perp c'$ тоді й тільки тоді, коли $I(r) \cap I(r') = \varnothing$
- $I \models r \perp r'$ тоді й тільки тоді, коли $I(c) \cap I(c') = \varnothing$
- $I \models t \perp t'$ тоді й тільки тоді, коли $I(t) \cap I(t') = \varnothing$
- $I \models c \in c'$ тоді й тільки тоді, коли $I(i) \in I(t)$
- $I \models i.r \in v$ тоді й тільки тоді, коли $\langle I(i), I(v) \rangle \in I(r)$
- $I \models i.r \in i'$ тоді й тільки тоді, коли $\langle I(i), I(i') \rangle \in I(r)$.

Онтологічні формули можуть містити не тільки ці твердження, наприклад, кванторизовані твердження або твердження, що містять логічні оператори. Можна обмежитися відношеннями між сутностями. Онтологія – це набір тверджень, що вибирає множину інтерпретацій, що задовольняють цим твердженням. Такі інтерпретації називають моделями. Вони становлять можливі інтерпретації онтології.

Для онтології $O = \langle C, I, R, T, V, \leq, \perp, \in, = \rangle$ її *моделлю* є інтерпретація $m = \langle I, D \rangle$, що задовольняє усім твердженням онтології о: $\forall \sigma \in o, m \models \sigma$.

Проаналізувавши виразну здатність різних засобів подання онтологій і формальних моделей онтологій, а також різні засоби опису онтологій, що пропонують технології Semantic Web, зазначимо, що вони суттєво відрізняються за своїми виразними можливостями та за своєю складністю: RDF Schemas пропонує найпростіший рівень для подання онтологій, а OWL Full – найскладніший. Вибір засобу подання онтології залежить від специфіки проблеми, для розв'язання якої вона розробляється.

## 1. 6. Дескриптивні логіки як теоретичний базис для онтологічного подання знань

Дескриптивні логіки (DL) виникли як розширення фреймів і семантичних мереж завдяки механізмам формальної логіки. Сьогодні DL використовуються в Semantic Web для побудови онтологій. DL – це сімейство мов подання знань, що слугує для опису понять предметної

області у формалізованому вигляді. Будь-яка логіка DL є логікою першого порядку, але не навпаки. Базові елементи DL – це множина класів NC; множина індивідуумів NI; множина відношень NR.

Дескриптивні логіки дають змогу здійснювати опис понять ПрО в недвозначному, формалізованому вигляді. Вони поєднують у собі, з одного боку, досить багаті виражальні можливості, а з іншого – задовільні обчислювальні властивості, такі, як можливість розв'язання й відносно невисока обчислювальна складність основних логічних проблем, що дає змогу використовувати їх на практиці. Отже, DL є компромісом між виразністю й можливістю розв'язання. DL можна розглядати як розв'язувані фрагменти логіки предикатів, синтаксично ж вони близькі до модальних логік [81].

У математичній логіці кожна мова характеризується своїм *синтаксисом*, тобто правилами побудови виразів цієї мови, і *семантикою*, тобто способом надання цим виразам деякого формального значення, наприклад, вказівкою, які вирази вважаються правдивими, а які – хибними. Щоб сформувати синтаксис якої-небудь DL, необхідно задати непорожні (і, звичайно, скінчені) множини символів, так званих атомарних концептів і атомарних ролей, з яких будуватимуться вирази мови цієї логіки. Конкретна DL характеризується набором конструкторів та індуктивним правилом, за допомогою якого складені концепти цієї логіки будуються з атомарних концептів і атомарних ролей, використовуючи ці конструктори [140].

### 1. 6. 1. Конструктори DL

Типовими конструкторами для побудови складених концептів є:

- перетин (чи кон'юнкція) концептів, позначається як *C and D*;
- об'єднання (чи диз'юнкція) концептів, позначається як *C or D*;
- доповнення (чи заперечення) концепту, позначається як *not C*;
- обмеження щодо значення ролі (чи обмеження квантором загальності), позначається як *forall R.C*;
- екзистенціальне обмеження (чи обмеження квантором існування), позначається як *exists R.C*;
- обмеження щодо значення ролі.

Ключова відмінність логік (наприклад, логіки предикатів першого порядку FOL, DL, а також OWL) від інших формальних мов – це наявність формальної семантики, для опису якої використовується мова, що відрізняється від мови опису синтаксису. Ця мова позбавлена неоднозначностей природної мови, за допомогою якої здійснюється опис семантики, наприклад, мов програмування, а тому значення логічних тверджень завжди суворо визначені (на відміну

від неформального опису семантики). Крім того, наявність другої мови дає змогу чітко розрізняти власне логічні твердження й твердження про їх істинність. Фрейми, семантичні мережі та багато інших поширених засобів подання знань позбавлені формальної семантики, а виражені в них знання орієнтовані на людину, а не на машинну обробку (багато знань про предметну область не є фактичними, а лише припускаються) [189].

У логіках предикатів першого порядку є формальні засоби опису семантики, але немає зручних засобів подання знань про предметну область – не зручно підтримувати ієрархії й відображати структуровані класи.

Спроба об'єднання семантичних мереж з FOL, започаткована Р. Брахман (у системі керування знаннями KL-ONE), сприяла створенню термінологічної логіки, що згодом перетворилася на сімейство логік DL. Важливо пам'ятати, що DL – це не логіка, а *сімейство* логік, що мають різні можливості. Відповідно, різні застосування можуть вибирати для своїх цілей різні логіки сімейства DL. Наприклад, OWL Lite заснований на DL за назвою SHIF(D), а OWL-DL – на SHOIN(D) [197].

Будь-яка DL є *підмножиною* FOL. Це означає, що будь-яке твердження на DL можна подати у вигляді формули FOL, але не навпаки. При цьому вони семантично сумісні, тобто якщо перетворити базу знань DL на базу знань FOL, то з неї можна буде зробити такі ж логічні виводи, як і до перетворення.

У синтаксисі DL перемінні та квантори явно не використовуються, що дуже зручно. Наприклад, твердження A \subset B у DL – це те саме, що й FOL-формула \forall x A(x) -> B(x), але без перемінних.

Більшість логік DL проектуються так, щоб вони були підмножиною не тільки FOL, а і його спеціального фрагмента, що має назву – guarded fragment. Це означає, що DL, зазвичай, мають властивість розв'язання, що досягається через урізання деяких можливостей FOL (зокрема, використання перемінних). Тому OWL DL розв'язний і для нього є логічні процесори, а OWL Full, що не базується на DL, – нерозв'язний і для нього логічних процесорів немає.

### 1. 6. 2. Логіка ALC

Розглянемо позначення деяких DL. ALC (Attributive Language with Complements) – це підмножина DL, на якій базується OWL; при цьому для багатьох реальних онтологій цілком достатньо ALC. Логіка ALC досить проста, але містить багато ключових властивостей OWL [119].

Синтаксис ALC має алфавіт (набір базових символів), що складається з трьох компонентів:

- набір імен базових класів (NC) і два спеціальні класи (універсальний клас top і порожній клас bottom);
- набір імен властивостей (NR);
- набір імен об'єктів (NI).

ALC дає змогу здійснювати опис складних понять за допомогою таких конструкторів класів, як перетин, об'єднання, доповнення класу, універсальне обмеження властивості та екзистенціальне обмеження властивості.

Логічні формули в ALC (аксіоми) бувають трьох типів:

1) Відношення типу «дочірній клас». Ці аксіоми мають вигляд «C \subset D», де C і D – довільні (можливо складні) класи.

2) Відношення типу «екземпляр класу». Вони мають вигляд «a:C», де «a» позначає об'єкт, а C – довільний клас.

3) Відношення типу «екземпляр властивості». Вони мають вигляд «(a,b):P», де «a,b» позначають два об'єкти, а P – довільна властивість.

Набір аксіом першого типу називається *TBox* (terminological box), а набір аксіом другого й третього типу – *ABox* (assertional box).TBox – це опис ієрархії класів (понять) предметної області. ABox – це набір фактів про конкретні об'єкти, до яких класів вони належать і які мають властивості. *База знань* (чи ***онтологія***) в ALC – це сукупність TBox і ABox.

### 1. 6. 3. Інтерпретація логіки

Якщо інтерпретація I задовольняє деякій аксіомі A, то вона називається *моделлю* A (звідси й назва семантики – *модельно-теоретична*). Наприклад, якщо ми говоримо про аксіоми типу I, то відповідність означає, що зміст класів C і D не *суперечить* змісту аксіоми.

Інтерпретація I *задовольняє* TBox (чи *є моделлю* TBox), якщо вона є моделлю всіх аксіом TBox. Аналогічно визначається модель ABox. Нарешті, інтерпретація *є моделлю* онтології, якщо вона є моделлю її TBox і ABox.

Логічне виведення дає змогу автоматично здобувати нові знання з явно заданих аксіом. Для ALC визначають такі основні задачі логічного виводу:

- *погодженість* (чи несуперечність) онтології: онтологія називається *несуперечливою*, якщо в ній є хоча б одна модель, тобто такий спосіб інтерпретації її класів, об'єктів і властивостей, що не суперечить жодній заданій аксіомі (TBox чи ABox);

- *когерентність* класу: клас С називається *когерентним* в онтології ПрО, якщо хоча б одна модель ПрО інтерпретує його як непорожню множину;
- *виведення аксіоми*: з онтології ПрО виводиться аксіома А, якщо кожна модель ПрО також є моделлю А, тобто за будь-якої інтерпретації класів, об'єктів і властивостей у ПрО, якщо інтерпретація не суперечить аксіомам у ПрО, то вона не суперечить і А.

Дві останні задачі зводяться до задачі погодженості:

- клас С *когерентний* в онтології ПрО тоді й тільки тоді, коли додавання нової аксіоми a:C (де об'єкт «a» раніше не траплявся) *не призводить* до втрати погодженості, тобто питання когерентності розв'язується шляхом додавання нової аксіоми до О й перевірки погодженості;
- аксіома А *виводиться* з онтології ПрО тоді й тільки тоді, коли додавання заперечення аксіоми А до онтології ПрО *призводить* до втрати погодженості (для аксіоми «C subset D» запереченням є аксіома a:(C and not D), а запереченням для a:C – a:not C).

Це метод доказу «від протилежного»: якщо додавання заперечення твердження призводить до протиріччя, то твердження правдиве (тобто воно логічно виводиться з бази знань – онтології). З огляду на це, єдине, що потрібно для всіх задач логічного виводу в ALC, – це алгоритм перевірки погодженості онтології.

Те саме справедливо й для інших DL, і для OWL. Процесори логічного виведення роблять саме це – перевіряють погодженість онтології. Усе інше – це робота препроцесора, яка полягає в додаванні заперечень аксіом (та деякі інші несуттєві моменти). Лише алгоритми *класифікації* (тобто обчислення всіх можливих відношень типу «підклас» між усіма парами класів, що дає змогу побудувати закінчену ієрархію класів) заслуговують на окремий розгляд.

Практично всі принципи (синтаксичні й семантичні) ALC спрацьовують у більш потужних DL, зокрема й тих, на яких засновані OWL Lite, OWL DL і новий OWL 2.0 (його логіка називає SROIQ(D)). Для характеристики інших DL використовують такі позначення:

- F – функціональні властивості;
- E – повна екзистенціальна кваліфікація (екзистенціальні обмеження, крім owl:Thing);
- U – об'єднання понять;
- C – складне заперечення поняття;
- S – логіка ALC з транзитивністю ролей;
- H – ієрархія ролей (підвластивості rdfs:subPropertyOf),

- R – обмежені складні аксіоми включення ролей; рефлексивність і антирефлексивність; диз'юнктивність ролей;
- O – номінали (перелічувані класи обмежень значення об'єкта owl:oneOf, owl:hasValue);
- I – зворотні (інверсні) властивості;
- N – обмеження потужності (restrictions (owl:cardinality, owl:maxCardinality);
- Q – кваліфіковані обмеження потужності (наявні в OWL 2.0, обмеження потужності, відмінні від owl:Thing);
- (D) – використання типів даних властивостей, значень даних або типів даних.

### 1. 6. 4. Найпоширеніші типи DL

DL S – це ALC з транзитивними властивостями (її іноді називають ALCR+). Транзитивність означає, що з R(a,b) і R(b,c) постає R(a,c) для будь-яких a,b,c.

SH – це S з ієрархією властивостей (а не тільки класів, як в ALC).

SHI – це SH зі зворотними властивостями (R – зворотне щодо R, якщо R(x,y) = R(y,x)).

SHIF – це SHI з функціональними властивостями, тобто можна оголошувати деякі властивості як функції (наприклад, «мати» – це функціональна властивість, оскільки вона в людини тільки одна).

SHIN – це SHI з обмеженнями стосовно обсягу області значень властивостей (т. зв. non-qualified number restrictions).

SHIQ – це SHI з обмеженнями стосовно обсягу області значень властивостей і класу значень властивостей.

SHOIQ – це SHIQ з номіналами, які дають змогу здійснювати описи класів у вигляді переліку об'єктів.

SROIQ – це SHOIQ, у якому ієрархії властивостей розширені до композицій властивостей (одне з головних нововведень OWL 2.0).

SROIQ(D) – це SROIQ з «типами даних» і властивостями, що поєднують об'єкти з типами даних.

## 1. 7. Методи узгодження онтологічних знань

Інструменти об'єднання онтологій дають змогу користувачам знайти подібність і несхожість між двома онтологіями й створюють результативну онтологію, що містить елементи обох початкових онтологій. Для досягнення цієї мети потрібно автоматично визначити відповідності між концептами в онтологіях чи забезпечити середовище, у якому користувач зможе легко знайти й позначити ці відповідності. Такі інструменти відомі як інструменти відображення, вирівнювання

й об’єднання онтологій, оскільки вони виконують подібні операції для процесів відображення, вирівнювання й об’єднання.

*Відображення* (зіставлення) онтологій (ontology mapping) – діяльність, спрямована на встановлення відповідності між кількома онтологіями.

*Вирівнювання* онтологій (ontology alignment) полягає в тому, щоб встановити різні види відповідності (чи зв’язку) між двома онтологіями, а потім повторно зберегти початкові онтології й надалі використовувати інформацію цих онтологій.

*Інтеграція* онтологій (ontology merging) – діяльність зі створення нової погодженої онтології чи фрагмента онтології з двох або більше наявних онтологій [364]. Інший термін, що використовують на позначення цієї операції, – об’єднання онтологій.

Наголосимо, що відображення онтологій є необхідним початковим етапом для інтеграції онтологій.

Відображення онтологій полягає у виявленні семантичних зв’язків між подібними елементами з різних онтологій. Вирівнювання онтологій передбачає встановлення різних видів відповідності (чи зв’язку) між двома онтологіями, а потім повторне збереження початкових онтологій з тим, щоб надалі використовувати інформацію з обох онтологій.

Однак і автоматичне об’єднання онтологій, і створення інструментальних засобів, які б керували користувачем у цьому процесі, ще лише починають формуватися.

На сьогодні розроблено багато методів і програмних засобів, що підтримують ці методи, які реалізують в онтологіях операції зіставлення, інтеграції та вирівнювання [316].

Метою відображення (matching) онтологій є зменшення їх неоднорідності. Неоднорідність полягає не лише в різниці між цілями застосувань, відповідно до яких вони розроблялися, а й між формалізмами, за допомогою яких створювалися онтології.

Розглянемо деякі найбільші види неоднорідності [95].

*Синтаксична неоднорідність* виникає тоді, коли дві онтології побудовані різними мовами подання або якщо вони змодельовані з використанням різних формалізмів презентації знань, наприклад, за допомогою OWL та F-логіки. Такий тип невідповідності (mismatch), зазвичай, припускається на теоретичному рівні при встановленні еквівалентності між конструкціями в різних мовах. Отже, іноді можна перекласти онтології з однієї мови на іншу, зберігаючи їх значення.

*Термінологічна неоднорідність* виникає через розходження в іменах, що посилаються на ті самі сутності в різних онтологіях. Це

може бути зумовлено використанням різних природних мов, різних технічних підмов, синонімів тощо.

*Концептуальна (семантична) неоднорідність* і логічна невідповідність підтримуються розходженнями в моделюванні однієї й тієї ж ПрО. Це зумовлене використанням різних (хоча, можливо, еквівалентних) аксіом для визначення концептів або внаслідок використання зовсім різних понять (наприклад, геометрія на основі елементарного об'єкта – точки й геометрія на основі елементарного об'єкта – сфери). Є різниця між неоднорідністю концептуалізації, що ґрунтується на розходженні концептів, які моделюються, і неоднорідністю розгортання, що базується на способах презентації концептів.

Проблема відображення онтологій полягає в тому, що:

- сутності (класи, властивості, зв'язки, об'єкти), що мають однакові імена, можуть мати різну семантику;
- сутності (класи, властивості, зв'язки, об'єкти), що мають однакову семантику, можуть мати різні імена.

Відображення онтологій передбачає дві окремі задачі:

- локальне відображення сутностей, що припускає незалежне встановлення відповідностей між двома сутностями розглянутих онтологій;
- глобальне відображення сутностей перегляд локальних відображень з урахуванням відображень усіх інших елементів.

Наприклад, локальне відображення деяких сутностей A1 і A2 може бути встановлено на основі відповідності між їх родовими сутностями A01 і A02, що спочатку може бути ймовірним або недовизначеним. У процесі формування глобального відображення сутностей відповідність між A1 і A2 переглядається з урахуванням додаткової інформації щодо відповідності між A01 і A02.

Сучасні методи встановлення відображення онтологій носять міждисциплінарний характер. Серед них можна виділити такі:

- лінгвістичні (термінологічні, лексичні);
- структурні;
- статистичні (екстенсіональні);
- логічні (формальні, семантичні).

*Лінгвістичний (лексичний) аналіз* визначає подібність між сутностями на основі порівняння імен сутностей (оцінка кількості символів, що збігаються, спільні частини слів, наприклад, «Цілі» та «Цільові настанови») або ж шляхом аналізу синонімії термінів онтологій. Для того, щоб виявити терміни-синоніми, можна використовувати різноманітні словники загальної та професійної

лексики й тезауруси. Такий вид аналізу є початковим етапом встановлення відповідностей між сутностями онтологій.

*Структурний аналіз* поєднує в собі аналіз внутрішньої та зовнішньої структури й аналіз подібності між зв'язками (зокрема ієрархічними).

Аналіз внутрішньої структури передбачає вироблення оцінки подібності між термінами онтології на основі дослідження доменів і областей припустимих значень для атрибутів і зв'язків. Методи аналізу внутрішньої структури іноді називають методами на основі обмежень. Сутностей з такою внутрішньою структурою, а також властивостей зі схожим доменом і областю значень може бути досить багато, тому такі методи застосовуються тільки для формування кластерів подібних понять і потребують подальшого уточнення іншими методами.

Аналіз зовнішньої структури пов'язаний з пошуком подібних посилань на одні й ті самі терміни зовнішніх онтологій.

Аналіз подібності за ієрархічними зв'язками припускає, що оцінка схожості двох сутностей двох онтологій може базуватися на позиціях цих сутностей в ієрархії класів: якщо підкласи й надклас двох сутностей двох онтологій подібні, то самі такі дві сутності теж можуть бути подібними. Це твердження може розглядатися по-різному, а тому породжує ряд можливих критеріїв (ознак) щодо подібності двох сутностей:

- їх безпосередні надкласи вже є схожими;
- усі підкласи їх надкласу (сутності-брати) уже є схожими;
- їх прямі сутності-нащадки вже є схожими;
- усі екземпляри цих класів (сутності-листи) уже є схожими;
- усі (чи більшість) сутностей на шляху від кореня до розглянутої сутності вже є схожими.

Звичайно, застосування таких критеріїв повинно супроводжуватися іншими критеріями.

Аналіз подібності за перехресними зв'язками для визначення подібності між сутностями проводиться так: якщо клас A1 пов'язаний з класом B1 зв'язком типу R1 в одній онтології, а клас A2 пов'язаний з B2 зв'язком типу R2 в іншій онтології, і якщо відомо, що B1 і B2 – схожі та R1 і R2 – схожі, то можна припустити схожість A1 і A2. Так само можна говорити й про подібності типів зв'язків – R1 і R2, якщо відомо, що A1 і A2 – схожі й B1 і B2 – схожі.

Наприклад, якщо відомо, що класи «Людина» та «Особа» є схожими й відношення «працювати» й «бути співробітником» теж схожі, причому в першій онтології клас «Людина» пов'язаний відношенням «працювати» з класом «Організація», а в другій – клас «Особа» пов'язаний відношенням «бути співробітником» з класом

«Компанія» та «працювати», тоді класи різних онтологій «Організація» й «Компанія» є схожими.

*Екстенсіональний* (статистичний) аналіз базується на використанні наявних екземплярів двох класів для оцінки екстенсіональної відповідності між цими класами. Для знаходження відповідності між сутностями використовуються такі діагностичні правила:

- клас C1 є *еквівалентним* класу C2, якщо неможливо знайти такий екземпляр O1 класу C1, який би не належав класу C2, і навпаки;
- клас C1 є *підкласом* класу C2, якщо неможливо знайти такий екземпляр O1 класу C1, який би не належав класу C2, і клас C1 не є еквівалентним класу C2.

*Логічний* аналіз передбачає виявлення родових класів для тих класів, що зіставляються, і вивчення накладених на них обмежень.

Наприклад, в одній онтології може бути клас «Дитина», що є видовим класом для класу «Людина» з накладеним обмеженням на властивість «Вік» <16.

В іншій онтології може бути клас «Неповнолітня особа», що є видовим класом для класу «Особа» з накладеним обмеженням на «Вік особи» <18. При аналізі відповідності між класами «Дитина» й «Неповнолітня особа» виявляються родові класи «Людина» й «Особа». За наявності інформації про відповідність між цими класами відбувається порівняння обмежень, накладених на ці родові класи. Для цього порівнюються властивості класів «Вік» і «Вік особи»: якщо ці властивості схожі, то порівнюються накладені обмеження «<16» і «<10». Унаслідок такого порівняння доходять висновку, що клас «Дитина» є підкласом класу «Неповнолітня особа».

Обмеженням цього методу є потреба в сутностях з обох онтологій щодо яких апріорно відомо, що вони є еквівалентними в двох онтологіях, які зіставляються.

Після одержання локальних відповідностей між сутностями можна визначати глобальну відповідність між ними.

Зауважимо: за наявності баз знань, що містять екземпляри відображуваних онтологій, пріоритетного значення набувають результати екстенсіонального аналізу, однак на практиці результати будь-якого аналізу варто узгоджувати з результатами, одержаними внаслідок інших видів аналізу.

Інструментальні засоби, що використовуються для пошуку відповідності між онтологіями, класифікують за призначенням:

- для об’єднання двох онтологій з метою створення однієї нової (PROMPT, Chimaera, OntoMerge);

- для визначення функції перетворення з однієї онтології на іншу (OntoMorph);
- для визначення відображення між концептами в двох онтологіях, тобто знаходження пари концептів, подібних один до одного (наприклад, OBSERVER, FCA-Merge);
- для встановлення правил відображення з тим, щоб зв'язувати тільки релевантні частини наявних онтологій (ONION).

Прикладами таких засобів є: PROMPT – плагін до системи Protege, який забезпечує об'єднання й групування онтологій; Chimaera – інтерактивний інструмент для об'єднання онтологій на основі редактора онтологій Ontolingua; OntoMerge, що забезпечує побудову об'єднаної онтології як об'єднання двох наявних онтологій і набору аксіом з'єднання; OBSERVER, що використовує DL для відповідей на запити відразу до кількох онтологій тощо. Більш докладно ці засоби проаналізовано в [120].

Крім зіставлення вмісту онтологій, важливим є питання зіставлення інших параметрів онтологій, які впливають на можливість і ефективність їх застосування в різних ІС.

Можна виділити такі основні напрями зіставлення онтологій [415]:

- оцінка якості великомасштабних онтологій;
- експлуатаційні показники технологій зіставлення онтологій;
- дослідження фонових знань, яких бракує;
- неточність у зіставленні онтологій;
- вибір засобу зіставлення та його конфігурації;
- включення користувача в процес зіставлення онтологій;
- пояснення результатів зіставлення;
- соціальне й колаборативне зіставлення онтологій;
- управління вирівнюванням: інформаційна структура й супровід;
- логічне виведення з вирівнюванням.

Розглянемо ці напрями детальніше.

*1. Оцінка якості великомасштабних онтологій* – швидкий розвиток різноманітних підходів до зіставлення онтологій зумовив появу більш суворих вимог до оцінки їх якості та порівняння. У зв'язку з цим, у 2005 році з'явилася Ініціатива з оцінки якості вирівнювання (Alignment Evaluation Initiative – OAEI (http://oaei.ontologymatching.org/)), що передбачала міжнародну координацію в напрямі оцінки якості систем зіставлення онтологій. Основна мета OAEI полягає в підтримці порівняння систем і алгоритмів на однаковій основі й у наданні можливості будь-кому доходити висновків щодо стратегій зіставлення. Матеріали роботи OAEI розглянуто в [208, 253].

Але OAEI пропонує лише деякий попередній базис для характеристик масштабування технологій зіставлення. Тому мають розроблятися й упроваджуватися також і великі тести, що містять понад 10.000, 100.000 або 1.000.000 сутностей. Це, так само, підвищує потребу в більш автоматизованих засобах здобуття еталонного вирівнювання, тобто зменшення людських зусиль, потрібних при збільшенні обсягу даних.

Необхідні також більш точні одиниці вимірювання якості оцінки (деякі кроки в цьому напрямі запропоновані в [251]). Зокрема, потрібні специфічні для застосування одиниці вимірювання, які дають змогу з'ясувати, чи є результат зіставлення онтологій задовільним для цього застосування.

Потрібні методи оцінювання, що базуються на глибокому аналізі простору проблеми зіставлення, щоб запропонувати автоматизовані методи генерації тестів відповідної складності [279].

*2. Експлуатаційні показники технологій зіставлення онтологій* є дуже важливими для багатьох динамічних застосувань, наприклад, якщо користувач не може довго чекати на відповідь системи. Індикатор часу виконання вказує на властивості масштабування засобів зіставлення та потенціал їх використання в системах високої потужності. Крім того, важливим показником роботи таких систем є обсяг основної пам'яті, що потрібна для роботи [280].

Оптимізації у сфері зіставлення є цінними тільки тоді, коли базові технології зіставлення є стабільними. Наприклад, у випадку S-Match [278], що використовується для легких (lightweight) ontologies [275], проблема зіставлення звелася до проблеми достовірності (validity problem) для числення висловлювань. Базова версія S-Match використовує стандартну процедуру задовільності на основі DPLL – SAT4J (http://www.sat4j.org).

*3. Дослідження фонових знань, яких бракує.* Одна з причин складності задач зіставлення полягає в тому, що онтології розробляються з певними фоновими знаннями й у певному контексті, який, на жаль, не є частиною специфікації онтології й через це не може використовуватися в процесі зіставлення. Тому брак таких фонових знань ускладнює задачу зіставлення через генерацію зайвих невизначеностей. Для розв'язання цієї проблеми використовують різноманітні стратегії, такі, як: 1) декларування помилкових аксіом вручну ще до початку спроби зіставлення [337]; 2) повторне використання попередніх результатів зіставлення [245]; 3) запити до Web [294]; 4) використання специфічних правил предметної області [336]; 5) використання онтологій відповідної предметної області [474]; 6) використання онтологій, доступ до яких забезпечує

Semantic Web [405]. Крім того, у [276] розглядається автоматизований підхід до розв'язання проблеми браку фонових знань у задачах зіставлення шляхом ітеративного застосування семантичного зіставлення [277]. Наведені вище технології допомагають підвищити якість результатів роботи засобів зіставлення в різних випадках. Навіть більше, вони можуть мати різноманітні варіації залежно від того, які саме фонові знання було вибрано, як порівнюються сутності з онтологій із джерелами фонових знань і які комбінації результатів отримані з різноманітних зовнішніх джерел.

4. *Неточність у зіставленні онтологій.* Моделювання зіставлення онтологій як неточного процесу виконується за допомогою матриць подібності як міри впевненості. Потім засіб зіставлення вимірюється за допомогою оцінки його впевненості відповідно до реального світу. Приміром, у [265] прогнозується, чи буде оцінене таке зіставлення як добре чи як погане. Неточність також може бути знята ітеративно. Якщо початкові припущення будуть підсилені або відкинуті, тоді початкові межі неповноти будуть також визначені знову.

Неточність може бути визначена як порівняння К вирівнювань, кожне з яких має власну міру неточності (змодельовану як нечітке відношення над двома онтологіями), щоб покращити точність результатів зіставлення. Імовірнісні зіставлення схем – множина зіставлень з вірогідністю, що пов'язана з кожним зіставленням, – може використовуватися для знаходження відповідей на запити з неточністю для автоматизовано створених зіставлень [407].

Для успішного виконання таких робіт необхідно краще розуміти основи моделювання неточності в зіставленні онтологій з тим, щоб покращити знаходження протиріч у зіставленні, приміром, через імовірнісне виведення або ідентифікації того, коли зворотний зв'язок з користувачем є найбільш корисним. У динамічних застосуваннях часто немає точних відповідностей або ж ці відповідності недостатньо специфіковані, тому потрібно вибрати ті, що задовольняють конкретне застосування. Це, так само, потребує формалізації зв'язку між засобами зіставлення онтологій і системами інтеграції інформації, що підтримують неточність.

Досить часто наявні сьогодні засоби зіставлення в одних випадках спрацьовують добре, а в інших – ні. Це зумовлено такими проблемами:

- вибором засобу зіставлення [353];
- комбінуванням засобів зіставлення ручного або автоматизованого [329];
- зміною конфігурації засобу зіставлення для адаптації автоматизованих засобів зіставлення до потреб динамічних застосувань [247].

Складність розв'язання всіх цих проблем спричинена дуже великим простором пошуку та багатокритеріальним ухваленням рішень.

У традиційних застосуваннях автоматичне зіставлення онтологій, зазвичай, не дає високоякісних результатів, особливо для великих наборів оброблюваних даних. Більш ефективним є частково автоматизоване зіставлення. Тому потрібно знати, як найбільш результативно залучити користувача до процесу зіставлення [246]. Деякі сучасні дослідження в цій сфері спрямовані на ергономічні аспекти детального вирівнювання й на ручну перевірку та виправлення отриманих результатів. Приміром, у [256] запропонована графічна візуалізація вирівнювань на основі когнітивних досліджень.

Щоб системи зіставлення набули ширшого визнання, потрібно, щоб вони були спроможні *аргументувати* отримані результати для користувачів та інших програм, що використовують ці результати. Насправді вирівнювання, що виконується системою зіставлення, може не бути інтуїтивно очевидним для людини, а тому потребує певного пояснення. Маючи можливість зрозуміти спосіб отримання вирівнювання, користувач може свідомо редагувати його вручну, забезпечуючи тим самим, зворотний зв'язок з системою [323]. Ключовим питанням при цьому є надання пояснення в простому й зрозумілому для користувача вигляді, щоб спростити йому процес ухвалення рішень. Тому, на нашу думку, було б доцільно стандартизувати пояснення результатів зіставлення, що має полегшити взаємодію систем зіставлення з іншими програмами.

Ще один спосіб знаходження зіставлення використовує переваги мережі Інтернет: якщо одній особі надто важко досягти коректного вирівнювання між кількома парами онтологій, то та сама задача значно легше розв'язується, коли її виконує разом багато осіб. Це пов'язано з такими трьома аспектами:

- кожна особа має виконувати лише невелику частину роботи;
- кожна особа може покращувати те, що зробили інші;
- кількість помилок зменшується.

Приміром, у [475] розглядається експеримент зі створення вирівнювань онтологій спільнотою людей за допомогою використання анотацій. У різноманітних системах колаборативного створення зіставлень онтологій значна увага приділяється дружньому до користувача інтерфейсу. У [343] пропонується залучати велику кількість користувачів до спільного створення схем зіставлень на основі Web 2. 0 шляхом постановки користувачам простих запитань і вивчення їх відповідей для покращення результатів зіставлення, отриманих автоматично.

Колаборативний і соціальний підходи до зіставлення онтологій базуються на інфраструктурі, яка забезпечує спільне використання вирівнювань та їх анотувань. Ці властивості можуть використовуватися для полегшення повторного використання вирівнювань. Найбільш актуальним у колаборативному зіставленні онтологій є пошук вдалої підтримки анотування й адекватних елементів визначення, які підтримують роботу у великомасштабних прикладних програмах, а саме – суперечні й неповні вирівнювання мають набути задовільного вигляду. Крім того, слід віднайти спосіб, за допомогою якого можна було б реагувати на дії зловмисних користувачів.

Вирівнювання онтологій має підтримуватися на всіх етапах життєвого циклу відповідним інструментарієм і стандартами. Необхідні для цього функції мають бути реалізовані як сервіси. Найбільш важливими є такі сервіси:

- зіставлення двох онтологій (через вибір певного алгоритму та його параметрів, включаючи початкове вирівнювання);
- збереження вирівнювання в постійному сховищі;
- пошук вирівнювання на основі його ідентифікатора;
- пошук метаданих вирівнювання, таких, як ідентифікатор, що можуть бути використані для вибору між двома обраними вирівнюваннями;
- пошук збережених раніше вирівнювань між двома вибраними онтологіями;
- редагування вирівнювання шляхом додавання або видалення відповідностей;
- упорядковування вирівнювань на основі порогових величин;
- генерація коду для виконання трансформацій онтології, трансляції даних на основі окремого вирівнювання;
- переклад повідомлення відповідно до вирівнювання.

Така функціональна підтримка має доповнюватися багатими метаданими, що дають змогу користувачам і системам вибирати адекватні вирівнювання на основі різноманітних критеріїв. Крім того, має підтримуватися постійне сховище та ідентифікація вирівнювань для забезпечення достовірності їх використання.

Можна виділити два рівні управління вирівнюваннями:

- проміжний шар інфраструктури (редагування вирівнювань, обробка вирівнювань, розподілений доступ до вирівнювань і управління моделлю);
- оточення підтримки, що забезпечує доступ до вирівнювань, потрібних для задачі.

Ці два рівні можуть поєднуватися в одній системі [363] або функціонувати окремо [250]. Крім того, важливо забезпечити

інфраструктуру підтримки вирівнювання в масштабі Web, щоб інструментарій і застосування могли разом використовувати вирівнювання, публікувати їх і повторно ними користуватися.

Життєвий цикл вирівнювання тісно пов'язаний з життєвим циклом онтології: як тільки онтологія еволюціонує, необхідно створювати нові вирівнювання, що відповідають змінам в онтології. Такого результату можна досягти за допомогою фіксації змін в онтології і трансформування цих змін у вирівнювання (з однієї версії онтології до іншої). Це можна використовувати для обчислення нових вирівнювань, які будуть заміщувати попередні. У такому разі раніше створені вирівнювання будуть заміщені композицією оновлених вирівнювань онтологій. Коли онтологія о еволюціонує в нову версію о1, необхідно поновити екземпляри цієї онтології d та вирівнювання A з іншими онтологіями о', а для цього нове вирівнювання A' між двома версіями може бути встановлене й використовуватися для генерації необхідних для трансформації екземплярів T і для зв'язування A.A' онтологій о1 та о'.

Операція зіставлення *(*matching) позначає вирівнювання (alignment) A' для пари онтологій O1 і O2, кожна з яких складається з множини дискретних сутностей, таких, як класи, властивості та екземпляри. Є й інші параметри, які можуть розширити визначення процесу узгодження, а саме: 1) використання вхідного вирівнювання A; 2) параметри узгодження, наприклад, вага; 3) зовнішні ресурси, що використовуються в процесі узгодження, приміром, загальні знання або тезауруси предметної області [254].

Вирівнювання відображає відповідності між сутностями, що належать до різних онтологій. Для двох онтологій відповідність – це $\langle id, e_1, e_2, n, r \rangle$, де id – унікальний ідентифікатор цієї відповідності; $e_1$ та $e_2$ – сутності (наприклад, класи, властивості, таблиці), що належать до онтологій A1 та A2; n – ступінь довіри (як правило, в діапазоні [0,1]), що відображає відповідність між $e_1$ та $e_2$; r – відношення між $e_1$ та $e_2$ (наприклад, еквівалентність, узагальнення, диз'юнктивність, перекриття) між $e_1$ та $e_2$ Відповідність $\langle id, e_1, e_2, n, r \rangle$ доводить, що між $e_1$ та $e_2$ є відношення r зі ступенем довіри n. Чим вищий ступінь довіри, тим вищою є подібність між $e_1$ та $e_2$, на яку він вказує.

Зіставлення онтологій є важливою операцією для традиційних застосувань, таких, як оцінка якості онтологій, інтеграція онтологій і даних, сховища даних. Зазвичай, таким застосуванням притаманні гетерогенні структурні моделі, які аналізуються й зіставляються вручну або з частковою автоматизацією в процесі розроблення. У таких

застосуваннях зіставлення є необхідною передумовою для роботи системи.

Зіставлення незалежно створених онтологій є складною задачею, що потребує великого обсягу обчислень. Тому часто достатньо зіставити не довільні онтології, а їх окремі випадки, що являють собою моделі певних інформаційних ресурсів і об'єктів. Приміром, у процесі семантичного пошуку можна зіставляти тезаурус користувача з тезаурусом документа.

Подібно до онтологій, вирівнювання мають власний життєвий цикл (рис. 1. 2) [252]:

- створення відповідностей у процесі узгодження (вручну або автоматизовано);
- оцінка якості й удосконалення відповідностей (ітеративний процес);
- комунікація з користувачем;
- експлуатація створеної онтології.

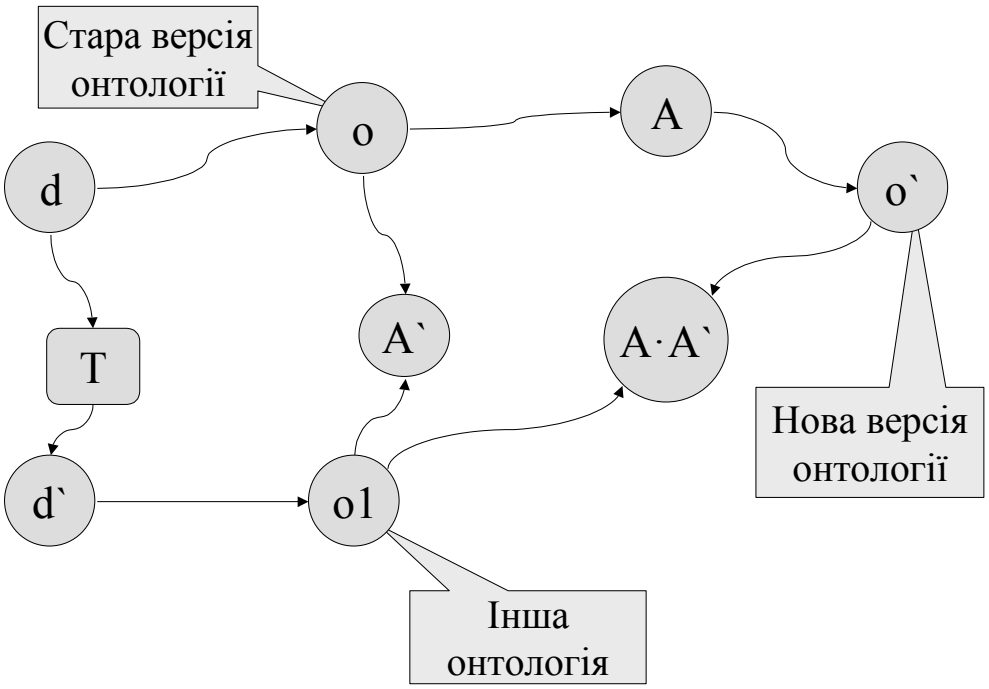



Рис. 1. 2. Еволюція вирівнювань

Оцінка якості складається з оцінювання властивостей отриманого вирівнювання. Удосконалення може відбуватися шляхом ручного внесення змін до вирівнювання або через застосування різноманітних процедур покращення, наприклад, через вибір деяких відповідностей у застосовних порогових величинах.

Коли вирівнювання видається кращим за попереднє, його зберігають і взаємодіють з іншими частинами, зацікавленими в такому вирівнюванні.

Вирівнювання трансформується в іншу форму або інтерпретується для виконання дій, наприклад, для посередництва (mediation) або об'єднання (merging).

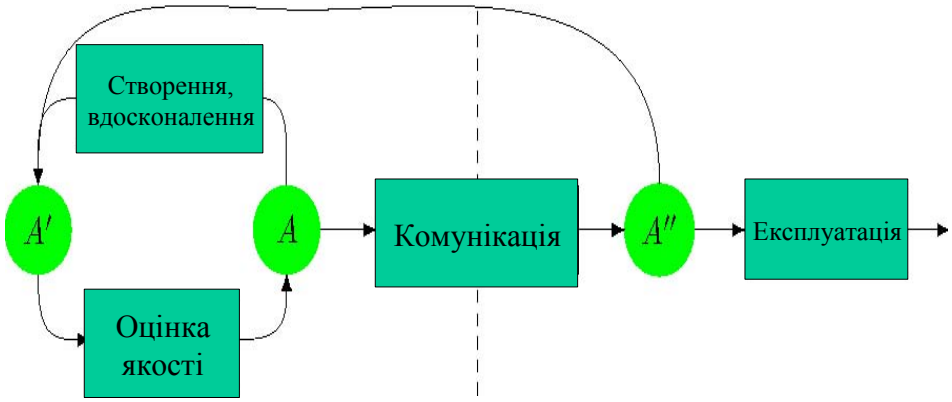



Рис. 1. 3. Життєвий цикл вирівнювання онтологій

Як видно з рис. 1. 3, створення вирівнювання є тільки першим кроком у цьому процесі. Досить часто такі вирівнювання потребують оцінювання їх якості, удосконалення й лише після цього можуть бути трансформовані в певну виконувану процедуру, яку може використовувати певна застосовна програма: перетворення онтології для її інтеграції з іншою онтологією, генерація множини аксіом переходу, що допомагають ідентифікувати відповідні поняття, переклад повідомлень, які один агент надсилає іншому, посередництво між запитами й відповідями на них в інтегрованих базах даних.

Деякі застосування, що розвиваються, можуть бути класифіковані за їх динамічністю (одноранговий обмін інформацією, інтеграція Web-сервісів, мультиагентні комунікації, технології Semantic Web). Такі застосування, на відміну від традиційних, потребують обов'язкового зіставлення в процесі виконання й набувають переваг більш явної (експліцитної) концептуальної моделі. Детальніше подібні системи проаналізовано в [324].

## 1. 8. Оцінка якості онтологій на різних етапах життєвого циклу

Міжнародна асоціація прикладних онтологій 15 червня 2013 року підготувала Комюніке з питань оцінки онтології протягом усього її життєвого циклу [374]. Цей документ носить рекомендаційний характер і відображає останні тенденції у сфері створення й підтримки прикладних онтологій як компонентів інформаційних систем. Використання семантичних моделей предметних областей і процесів, що відбуваються в них, дає змогу створити новий клас інтелектуальних систем з високим ступенем автоматизації. Аналіз цього документа [90] і пов'язаних з ним стандартів наводиться в [15].

На сьогодні немає єдиної думки стосовно методології проектування онтології й немає згоди щодо методів оцінки онтології. Відповідно, інструменти та способи оцінки онтології не дуже широко використовуються в процесі створення й розвитку онтології, що може призвести до появи онтології низької якості та стати перешкодою до успішного впровадження онтології як технології.

Тому виникла необхідність у створенні рекомендацій з оцінки онтології для розроблювачів і кінцевих користувачів на різних етапах життєвого циклу. Для цього були проаналізовані різні підходи до оцінки онтологій і оцінена застосовність кращих відомих практик з програмної та системної інженерії до онтологічних систем.

Крім того, у цьому Комюніке наведена модель життєвого циклу (ЖЦ) онтології й визначені критерії оцінки онтології в контексті різних етапів ЖЦ. Обговорюється також доступність інструментів і критерії оцінки якості онтології.

Деякі властивості онтології не пов'язані з конкретною задачею, інші потребують оцінки відношень між онтологією та її предметною областю, оточенням чи особливими варіантами її використання й можуть оцінюватися винятково в контексті сценарію використання такої онтології. Саме розмаїтість потенційних варіантів використання онтології не дає змоги створити універсальний набір критеріїв оцінки. Тому немає єдиного методу оцінки, який можна застосовувати до всіх онтологій.

Незважаючи на це, можна виділити деякі методи оцінки, необхідні для більшості онтологій. Оцінка якості онтології припускає її оцінювання як:

- моделі предметної області, придатної для розуміння людиною;
- моделі для машинної обробки;
- онтології як частини складної програмної системи.

Документ фокусується на оцінки п'яти найбільш важливих показників якості онтології:

- *зрозумілість* (intelligibility) чи придатна онтологія для розуміння її людиною;
- *точність* (fidelity) наскільки точно онтологія відображає предметну область;
- *кваліфіковане виконання* (craftsmanship) наскільки добре побудована онтологія і якою мірою дотримані оригінальні організаційні рішення й досягнуто технічної досконалості*;*
- *адекватність* (fitness) наскільки онтологічна модель придатна для задачі, що розв'язується системою за її допомогою;
- *убудованість* (deployability) чи відповідає онтологія вимогам системи, частиною якої вона є.

Для забезпечення зрозумілості онтології недостатньо, щоб онтологію могли читати спеціалісти-онтологи. Усі цільові користувачі системи повинні бути спроможними інтерпретувати її вміст (сутності, класи, відношення тощо), що є важливим для їхньої діяльності. Хоча зрозумілість не припускає прямого розпізнавання онтології користувачами, однак документація на неї повинна бути зрозумілою для всієї цільової аудиторії.

Для цього може бути потрібною наявність багатьох визначень для одного поняття (наприклад, різними мовами). Зрозумілість особливо важлива для онтологій, які використовуються як *керований словник.* Однак і для онтологій, що застосовуються як внутрішні структури програмних систем, також є бажаним забезпечення зрозумілості, оскільки підтримка онтології здійснюється людьми, які не завжди можуть бути причетними до її створення.

Точність онтології відображає коректність опису предметної області як в аксіомах, так і в документації до онтології. Технічна досконалість відповідає за акуратність виконання онтології – від наявності синтаксичних помилок до питань правильності реалізації філософського базису онтології.

Вимоги до адекватності й убудованості онтології залежать від сценарію її використання. Адекватність відображає онтологію як модель знань, убудованість – як програмний продукт

Через те, що убудованість і якість залежать від сценарію використання, оцінка онтології повинна вироблятися з урахуванням того, як саме вимоги до онтології залежать від вимог до системи, частиною якої вона є. Навіть більше, незважаючи на те, що під оцінкою онтології можна розуміти оцінку кінцевого продукту, для забезпечення успіху при розробці, упровадженні й використанні онтологій така оцінка розглядається як безперервний процес упродовж усього ЖЦ.

При цьому оцінювання онтології повинно здійснюватися за заздалегідь визначеними критеріями, що залежать від призначення

онтології та від її операційного середовища. Для досягнення цієї мети рекомендується при створенні онтологій використовувати засоби їх підтримки, що дають змогу відстежувати показники якості онтології на всіх етапах ЖЦ.

Ці рекомендації орієнтовані на всіх, хто сьогодні займається створенням або використанням онтологій: застосування наведених у документі практик дасть змогу значно підвищити ефективність створення й використання онтологій. Таким чином, первинною цільовою аудиторією таких рекомендацій є розроблювачі онтології, а вторинною – співтовариство програмних, системних інженерів і фахівців з якості. Успіх упровадження онтологій часто залежить від можливості відстежувати їх ефективність з уживанням відповідних заходів та застосуванням інженерних практик за необхідності.

Через те, що онтологія – машинно-інтерпретовані подання про деяку частину певної предметної області, які сприймаються людиною, – містить у собі поняття та їхні визначення, то вона дає змогу забезпечувати єдність термінології на підприємстві чи в організації.

Тому онтології можуть використовуватися як своєрідний глосарій. Зважаючи на те, що онтології відображають ключові концепти та їх зв’язки в машинно-інтерпретованій формі, вони близькі до моделей предметних областей у системному та програмному інжинірингу. Оскільки онтології можуть бути наповнені інформацією чи посиланнями на неї для створення баз знань, то, з операційного погляду, онтології нагадують бази даних.

Ця гнучкість є головною перевагою онтологічних технологій. Однак такого роду гнучкість одночасно й ускладнює їх оцінку. Оцінка онтології передбачає збирання інформації про її параметри, перевірку на відповідність цієї інформації деяким вимогам і оцінку придатності онтології для конкретної задачі.

### 1. 8. 1. Модель життєвого циклу онтології

ЖЦ будь-якої онтології складається з ряду процесів (паралельних чи послідовних), у ході яких онтологія зароджується, специфікується, адаптується, розгортається, використовується та підтримується. Ці процеси можуть повторюватися, окремих стадій ЖЦ може не бути в одній онтології, але вони можуть бути наявними в інших. Це не дає змоги створити єдину загальну модель ЖЦ онтології з чітко визначеною послідовністю етапів для різних задач, але для ЖЦ конкретних онтологій можна виділяти загальні стадії.

Ідентифікація фаз ЖЦ онтології дає можливість кластеризувати дії навколо цілей, входів і виходів певного типу. Навіть більше, модель ЖЦ наочно демонструє залежність одних стадій ЖЦ від інших, наприклад,

якість онтології безпосередньо залежить від того, наскільки грамотно були сформульовані вимоги до неї. Залежності між стадіями інваріантні для всіх онтологій, незважаючи на їх розходження між собою (рис. 1. 4.).

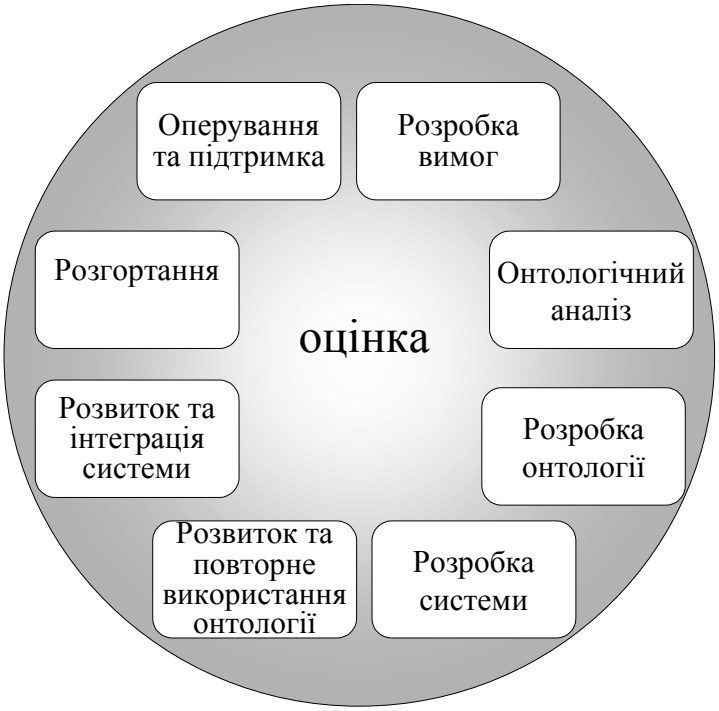



Рис. 1. 4. Оцінка онтології на різних етапах її життєвого циклу

Оцінка онтології виробляється на всіх стадіях ЖЦ, з різним фокусом, інтенсивністю та способами, що залежать від конкретної стадії ЖЦ. Оцінка на кожній стадії дає змогу зрозуміти, наскільки онтологія задовольняє вимогам наступної стадії.

Така модель може застосовуватися незалежно від того, чи використовується онтологія переважно комп'ютерною системою чи ні, оскільки такі системи є інформаційними в широкому розумінні: системами з людей, процесів, апаратного, програмного забезпечення даних, що обробляють інформацію та ухвалюють рішення.

### 1. 8. 2. Етап формулювання вимог

Завданням цього етапу є формування розуміння, контексту, мети й початкових вимог. Адекватний вибір цілей критично впливає на успіх створення й застосування онтології. Більшість оцінювальних дій залежить від результатів цієї фази.

На етапі вибору вимог розглядаються й оцінюються всі можливі сценарії застосування майбутньої онтології й на основі такої оцінки формулюються початкові вимоги. Як правило, сценарій використання

стає зрозумілим із задач онтології. На початковому етапі вимоги можуть бути представлені фрагментарно й стосуватися тільки окремих задач. Звичайно, ураховуються тільки деякі аспекти використання, і етап формулювання вимог може містити збирання інформації про інші аспекти, що є суттєвими для онтологічного аналізу та проектування.

Досить ефективним способом формулювання вимог є метод використання питань перевірки компетенції, тобто питань, на які повинна відповідати онтологія. Ці питання формулюються природною мовою, зазвичай як запити, які має підтримувати онтологія у вибраних сценаріях використання. Результатом фази формування вимог є документ, що має відповідати на такі питання:

- Навіщо потрібна ця онтологія?
- Які передбачені сценарії її використання?
- Які групи користувачів повинні розуміти певні частини онтології?
- Який масштаб має онтологія?
- Чи є онтології та стандарти, придатні для використання?
- Які питання компетенції?
- Чи відображають питання компетенції всі сценарії використання?
- Які вимоги має робоче середовище?
- Які ресурси (дані, моделі даних, глосарії, словники, схеми, таксономії, онтології, стандарти, доступ до експертів предметної області) можна використовувати при створенні онтології?

### 1. 8. 3. Етап онтологічного аналізу

Завданням фази онтологічного аналізу є виділення ключових сутностей онтології (індивідів, класів і відношень між ними), а також ототожнення їх з термінологією вибраної предметної області. Як правило, цей процес має на меті усунення двозначності, яку зумовлює різна термінологія для тих самих сутностей у різних джерелах і співтовариствах.

Результати цієї роботи, як правило, подаються в неформальному вигляді, що є придатним для сприйняття як онтологами, так і експертами ПрО. Спеціалісти використовують свої знання важливих онтологічних визначень і зв’язків для знаходження тих тверджень, які містять важливу для створення онтології інформацію. Результат онтологічного аналізу також може мати вигляд діаграм (карт концептів, діаграм UML, дерев концептів, рисунків).

Результат онтологічного аналізу має визначати таку інформацію:

- важливі *сутності* в межах передбачуваної предметної області.
- важливі *характеристики сутностей*, зокрема й відношення між ними, неоднозначність описів і властивості, важливі

для предметної області в рамках обраного сценарію використання.

- *термінологію*, що використовується для позначення цих сутностей і надання достатньої кількості контекстної інформації з метою усунення неоднозначності багатозначних термінів.

Результат аналізу забезпечує вихідну інформацію для проектування й створення онтології. Додатково ці результати надають деталі, за допомогою яких нечіткі вимоги до етапу проектування та створення онтології можуть бути сформульовані у вигляді більш точних, придатних для оцінки вимог.

Результати етапу онтологічного аналізу повинні бути оцінені відповідно до таких критеріїв:

- Чи задокументовані всі важливі терміни предметної області?
- Чи визначені всі сутності, важливі в масштабі онтології?
- Чи згодні експерти предметної області з результатами онтологічного аналізу?
- Чи є документація однозначно інтерпретованою настільки, наскільки необхідно для погодженого використання термінології?

### 1. 8. 4. Етап проектування онтології

На стадії проектування розробляється проект онтології на підставі результатів етапів формування вимог і онтологічного аналізу, зокрема обираються мови побудови онтології й мова запитів. Далі проектується структура онтології. Структура визначає поділ онтології на модулі, а також те, як ці модулі будуть взаємодіяти між собою. Наявні онтології можуть бути використані в структурі як модулі нової онтології. Опис поведінки модулів може бути здійснений на основі відповідей на питання компетенції. Ці питання, специфічні для конкретних модулів, як правило, отримують з питань для цілої онтології.

Етап проектування передбачає визначення принципів організації онтології й виділення класів верхнього рівня. Класи верхнього рівня – класи вищого рівня ієрархії (приміром, для OWL це класи – прямі нащадки owl:thing.) Ці класи визначають основні онтологічні категорії онтології. Принципи організації і класи верхнього рівня разом визначають, які базові аспекти реальності (наприклад, зміна в часі) відображає онтологія і як вона це робить. Принципи організації можуть обмежувати описові можливості онтології (наприклад, дозволяти тільки один дочірній концепт).

Одним зі способів вибору принципів організації є використання онтології верхнього рівня. Онтології верхнього рівня чи базові онтології (наприклад, DOLCE, BFO, SUMO) – це придатні для повторного

використання онтології різного ступеня складності, що визначають базові онтологічні категорії, відношення між ними та деякі методичні розв'язання щодо того, як презентувати реальність. Інші підходи, такі, як OntoClean, базуються на систематичному поданні логічних і філософських властивостей класів і відношень.

Зазначимо, що вимоги до виразності й продуктивності онтології можуть конфліктувати між собою. Таке протиріччя можна усунути шляхом створення окремих онтологій – довідкової та операційної. Довідкова онтологія відображає предметну область у всій повноті, що необхідна для розв'язання задачі. Операційна онтологія створюється на її основі з можливим уведенням деяких спрощень для збільшення продуктивності. Ці два типи онтології розглядатимуться нами в розділі розвитку й повторного використання онтології.

Результати фази проектування онтології оцінюються за такими питаннями:

- Чи достатні описові можливості мови онтології для задоволення вимог щодо побудови онтології?
- Чи досить виразна мова запитів для формалізації питань компетентності? Чи підтримує обрана мова всі необхідні можливості онтології (наприклад, якщо онтологія оперує ймовірностями, мова повинна відображати ймовірнісну інформацію)? Чи є кожен доданий до онтології клас або концепт підкласом чи екземпляром класу верхнього рівня?
- Чи визначені правила іменування концептів і чи дотримуються вони?
- Чи вимагає проект створення декількох окремих онтологічних модулів? Якщо так, то чи відображають модулі в сукупності потрібну предметну область.
- Чи відображено в проекті й чи будуть повторно використовуватися створені онтології і як?
- Чи визначено для кожного модуля, які типи сутностей у ньому представлені?
- Чи визначено для кожного модуля, як він буде оцінюватися й хто за це буде відповідати?
- Чи дає змогу спроектована онтологія уникнути додавання можливостей чи вмісту, що не відноситься до задоволення вимог до онтології?

### 1. 8. 5. Етап проектування системи на базі онтології

На стадії системного проектування ухвалюються рішення, що впливають на впровадження онтології в складну інформаційну систему. Ця взаємозалежність часто недооцінюється, що зумовлює проблеми,

пов’язані із взаємодією онтології з інформаційною системою, та збільшує ризик виникнення проблем використання онтології та системи загалом. Результат етапу системного проектування має відповідати на такі питання:

- Які операції виконуватимуться з використанням онтології? Які компоненти виконуватимуть ці операції? Як бізнес-вимоги, розроблені на стадії визначення вимог, можуть бути застосовані до цих специфічних операцій і компонентів?
- Чи будуть і, якщо будуть, то які саме зміни й додавання в онтології після розгортання системи?
- Які інтерфейси (машинні чи машинно-людські) будуть задіяні у внесенні додавань? Як будуть тестуватися ці інтерфейси щодо зміненої онтології? Яким вимогам вони повинні будуть відповідати?
- Які джерела даних використовуватимуться разом з онтологією? Через які інтерфейси буде відбуватися обмін інформацією?
- Як буде створюватися, оцінюватися й підтримуватися онтологія? Які для цього необхідні інструменти?
- Якщо онтологія буде мати модульну структуру і/чи створюватися розподіленими розроблювачами, то як це буде підтримуватися?

Важливо фактично оцінити онтологію як частину інформаційної системи й структуру системи в цілому з урахуванням того, чи може система відповісти на питання, наведені вище.

### 1. 8. 6. Етап розробки онтології

Етап розробки онтології складається з чотирьох головних дій:

- неформального моделювання;
- формалізації питань компетенції;
- формального моделювання;
- операційної адаптації.

Ці дії, зазвичай, повторюються в циклі для окремих модулів і для онтології загалом. На практиці вони часто виконуються без чітких меж між ними. Проте важливо розуміти їх концептуальні розходження, оскільки вони мають неоднакові передумови, по-різному оцінюються й ведуть до різних результатів, що, так само, оцінюються різними методами.

Етап розробки онтології стосується як створення нових онтологій, так і їх повторного використання, незважаючи на розходження між цими завданнями: успішна розробка нової онтології чи вибір вже наявної можливі тільки тією мірою, якою онтологія відповідає вимогам задачі. Отже, чи створюється онтологія з нуля, чи береться вже наявна, чи використовується комбінація з декількох попередніх, – отримані

результати залежать від коректної оцінки вимог до онтології. У такий спосіб нові та старі онтології потрапляють в один етап ЖЦ онтології.

У процесі інформаційного моделювання відбувається доопрацювання результатів онтологічного аналізу. Для кожного модуля відбувається зв'язування термінології з основними онтологічними концептами. Важливі якості сутностей можуть документуватися (наприклад, транзитивність чи відношення категоризації між двома класами). Результати, зазвичай, записуються в неформальному вигляді (схеми концептів, діаграми UML, текст природною мовою). При цьому потрібно додатково знайти відповіді на питання:

- Чи містяться в моделі винятково сутності обраної ПрО?
- Чи добре визначені всі концепти?
- Чи добре задокументована інтерпретація невизначених екземплярів, класів і відношень?
- Чи придатна документація для розуміння її експертами ПрО?
- У процесі формального моделювання вміст інформаційної моделі записується якою-небудь формальною онтологічною мовою (наприклад, OWL DL), а потім конкретизується аксіомами.

Онтологія, створена чи вибрана для повторного використання, оцінюється за трьома критеріями:

- адекватність відображення предметної області (точність);
- якість виконання онтології;
- технічна досконалість, тобто те, наскільки вона відповідає вимогам, сформульованим на етапі онтологічного проектування.

Точність відображення ПрО оцінюється за такими відповідями на питання:

- Чи коректно елементи опису (визначення, приклади, пояснення) онтології відображають онтологічні елементи (класи, властивості, аксіоми)?
- Чи істинні всі аксіоми в онтології з урахуванням рівня деталізації?
- Чи узгоджено опис онтології з аксіомами?

Оскільки оцінка точності залежить від розуміння ПрО, то вона потребує перевірки змісту онтології експертами ПрО. Є, однак, автоматизовані техніки оцінки точності. Наприклад, можна оцінювати логічну цілісність онтології, перевіряти автоматично створені моделі на відповідність вимогам до подання ПрО чи порівнювати внутрішню структуру онтології з іншими онтологіями чи з іншою версією тієї ж онтології, що мають схожу задачу.

У будь-якій інженерній дисципліні технічна досконалість пов'язана з двома окремими аспектами, які перетинаються:

- Чи продукт побудовано із застосуванням кращих практик у цій сфері?
- Чи дотримано рішення, ухвалені на стадії проектування?

Один з підходів до оцінки технічної досконалості полягає в оцінці аксіом на відповідність онтології верхнього рівня чи онтологічним метавластивостям (точність, єдність тощо). Інструменти для оцінки технічної досконалості, як правило, оцінюють внутрішню структуру онтології. Ці техніки оцінки спираються на математичні й логічні властивості онтології, зокрема такі, як логічна зв'язаність, проблеми теоретико-модельної інтерпретації тощо. Структурні метрики містять у собі коефіцієнт розгалуження, щільність, середні значення та ін.

Адекватність можна оцінити шляхом проведення тестів, що використовують онтологію як випробувальну площадку. Наприклад, якщо від онтології потрібно автоматичне індексування текстів, випробовування на адекватність може припускати апроксимацію аналізу документа й системи, що індексує. Є багато методів оцінки результату, наприклад, порівняння з деяким еталоном чи оцінка експертами.

У ході операційної адаптації довідкова онтологія адаптується до операційних задач для одержання операційної онтології. Головним питанням, на яке має відповісти нова онтологія, є питання здатності її забезпечити необхідну продуктивність. Це може потребувати спрощення онтології чи інших оптимізаційних процедур (наприклад, реструктурування) – приміром, переклад частини онтології з OWL DL на OWL EL для підвищення продуктивності.

Іноді операційна онтологія пишеться іншою мовою та з іншою семантикою, ніж онтологія посилань.

### 1. 8. 7. Етап розвитку інформаційної системи

До початку процесу створення онтології необхідно забезпечити інструментарій для її розробки та контролю. Етап розвитку охоплює інтеграцію онтології з іншими компонентами в підсистему й систему загалом за планом, який розроблено на етапі проектування системи.

Етап розвитку інформаційної системи розглядається як складник ЖЦ онтології тому, що, як правило, ефективне використання онтології можливо тільки за умов її взаємодії з іншими компонентами інформаційної системи. Таким чином, оцінка ефективності онтології може бути зроблена тільки тоді, коли інтеграція є довершеною й отримані відповідні результати.

### 1. 8. 8. Етап розгортання

На цьому етапі від розвитку та інтеграції онтології необхідно перейти до роботи з нею. Розгортанню, зазвичай, передує кілька циклів доопрацювання, тобто до цього моменту онтологія значно просувається в частині задоволення вимог до неї. Незважаючи на це, її все одно можуть додатково тестувати перед упровадженням (навіть після проходження всіх випробовувань попередніх етапів). Метою таких випробовувань є уникнення негативного впливу впровадження онтології на бізнес-процеси. Такі випробовування особливо суворі тоді, коли інформаційна система інтенсивно експлуатується й онтологія впроваджується ітераційно. Коли всі випробовування завершені, онтологія вводиться в експлуатацію і стає доступною для використання.

Питання на етапі розгортання:

- Чи відповідає онтологія всім вимогам фази створення?
- Чи виправдовують можливості, що привносяться онтологією, витрати на її створення? Чи є ризики від упровадження онтології?
- Чи використовувалися питання компетентності попередніх етапів для створення регресійних тестів? Чи були проведені регресійні тести для оцінки можливості зниження операційних показників від упровадження системи? Якщо деяке зниження прогнозується, то чи буде воно компенсовано позитивним ефектом від упровадження онтології?

### 1. 8. 9. Етап промислової експлуатації

Цей етап фокусується на підтримці наявних функцій, а не на додаванні нових. Коли онтологія перебуває на стадії експлуатації й технічного обслуговування, то відбувається збирання інформації про результати оперативного використання цієї онтології. Якщо виявляються проблеми чи факти зниження операційних показників, то можуть проводитися незначні доопрацювання для їх усунення. Одночасне виявлення нових випадків використання, бажаних покращень і нових вимог, що можуть трапитися протягом того ж періоду використання, не слід розглядати як частину технічного обслуговування діяльності; радше вони є передумовами для розробки вимог до майбутньої версії, розширення онтології чи створення нового модуля.

У процесі використання онтології ті самі засоби можуть застосовуватися для збирання інформації обох сортів – і для обслуговування, і для перспективної розробки й вимог розвитку. Технічне обслуговування передбачає виявлення й усунення помилок чи проблем функціонування.

Моніторинг онтології має бути безупинним, наприклад, збирання повідомлень про помилки й регулярне автоматичне регресійне тестування.

- Чи всі регресивні тести пройдено успішно? Якщо ні, то які заходи вживаються?
- Чи є проблеми на рівні функціонування системи? Якщо так, то чи спричинені вони онтологією чи проблеми в іншому?
- Якщо проблема в онтології, то чи може вона бути розв'язана без серйозної зміни онтології?
- Якщо проблема не може бути розв'язана без серйозного доопрацювання онтології, то чи варто продовжувати її впровадження?

### 1. 8. 10. Інструменти оцінки онтології

Є ряд ключових аспектів онтології, що не підлягають контролю чи оцінці за допомогою програмного забезпечення. Наприклад, необхідність чітких, повних і послідовних лексичних визначень термінів онтології сьогодні не можна ефективно задовольнити з використанням програмного забезпечення. Ще однією якістю онтології, яку важко оцінити за допомогою програмного забезпечення, є її точність.

На сьогодні не створено інструменти контролю онтології протягом усього її ЖЦ. Наявні лише інструменти підтримки різного ступеня ефективності на різних етапах ЖЦ. Однак нині з'являються новіші інструменти оцінки онтології, що стають доступними для користувачів.

### 1. 8. 11. Стандарти оцінки онтологій

При цьому слід зазначити, що вже сьогодні є ряд стандартів, що можуть тією чи іншою мірою застосовуватися при розробці онтологій [15]. Накопичення успішних практик і узагальнювальних методик підштовхує фахівців предметних областей, баз даних і онтологів до розробки стандартів, що фіксують корисні підходи.

*ISO 10303 (STEP).* Стандарти ISO 10303 визначають засоби опису (моделювання) промислових виробів на всіх стадіях життєвого циклу. Проект STEP розвивається з середини 80-х років минулого століття. Перша версія стандарту ISO 10303-11, присвяченого мові Express, була опублікована в 1990 році. У стандартах STEP використано ряд ідей, раніше втілених у методиках інформаційного проектування IDEF1X і функціонального проектування IDEF0. У рамках STEP зроблено спробу створення єдиних інформаційних моделей (онтологій) цілого ряду застосувань, що одержали назву прикладних протоколів.

*ISO 22745 (eOTD).* Стандарт ISO 22745 (Системи промислової автоматизації та інтеграції) містить у собі словник, що являє собою сукупність термінів, визначень і концепцій, що застосовуються для опису окремих об'єктів, організацій, адрес, товарів і послуг. У комплексі стандартів ISO 22745 відображені елементи даних, що стосуються конкретних класів і пар значень властивостей.

Відкритий технічний словник eOTD дає змогу точно характеризувати властивості відповідно до цього ISO 10303, визначати інформацію й обмінюватися даними з партнерами з інших країн без перекручування змісту даних.

*ISO 15926.* На сьогодні одним з найбільш перспективних стандартів організації онтологічних баз даних є ISO 15926. Цей стандарт визначає структуру об'єктів і специфікує модель даних, що визначає значення відомостей про життєвий цикл у єдиному контексті. Еталонна модель даних (довідкова онтологія) відображена в бібліотеці довідкових даних (RDL). Інтеграція застосування в інформаційний простір потребує узгодження класів і атрибутів прикладної моделі цього застосування з класами й атрибутами еталонної моделі (операційна онтологія) [4].

Розроблений компанією Techinvestlab (Росія) програмний продукт для роботи з даними у форматі ISO 15926 дає змогу створювати онтології ПрО в рамках базової онтології, викладеної в другій частині стандарту ISO 15926. Цей редактор здатний обробляти будь-які RDF-сумісні набори даних. Особливістю такої програми є можливість роботи з великими обсягами інформації, що недоступно більшості відкритих редакторів онтологій. Основні задачі, які розв'язує ISO 15926, – це отримання інформації з більшості наявних онтологій у максимально можливій кількості форматів, перевірка довідкових даних, створення нових довідкових даних, зокрема й шляхом автоматичного генерування їх з відомих джерел.

### 1. 8. 12. Інструментальні засоби оцінки онтологій

Поряд з розробкою засобів створення онтології з'являється все більше інструментів оцінки наявних онтологій і засобів підтримки їх життєвого циклу [14].

COLORE (Common Logic Ontology Repository) – проект, розроблений в університеті Торонто, метою якого є створення репозиторію онтологій верхнього рівня, який можна використовувати як випробувальну площадку в процесі розв'язання задач оцінки онтологій і методів їх інтеграції в рамках логіки першого порядку. Мовою опису онтології для цього проекту було обрано нещодавно прийнятий

стандарт ISO 24707, що є мовою специфікації онтологій верхнього рівня й баз знань.

Система HyQue, розроблена в університеті Карлтона, призначена для перевірки відповідності формалізованих гіпотез експериментальним даним (через ряд SPARQL-запитів) і певним формальним правилам. HyQue використовує набори правил, специфічні для конкретних ПрО. Результати роботи системи презентуються у форматах RDF і OWL.

Makleod складається з набору скриптів, покликаних надавати підтримку в розв'язанні типових задач при створенні онтології. Сьогодні розроблювачі проекту розв'язують задачу автоматизації операцій, що не торкаються семантики. До таких операцій належить, наприклад, перевірка узгодженості онтологій. У майбутньому планується об'єднати цей продукт з проектом COLORE, у рамках якого він буде використовуватися для оцінки й підтримки онтологій, що зберігаються в репозиторії.

Інтернет-портал NCBO BioPortal надає доступ до часто використовуваних онтологій у сфері біомедицини, а також до інструментів для роботи з ними. Портал підтримує пошук за кількома онтологіями одночасно, дає змогу створювати й візуалізувати зв'язки між термінами в різних онтологіях. Портал також дає можливість створювати анотації до текстів у форматі RDF.

OntoHub – це програма для керування розподіленими гетерогенними онтологіями. Вона вироблена в університеті Бремена для підтримки створення й інтеграції онтологій, написаних різними мовами. OntoHub підтримує більшість відомих мов опису онтологій (серед яких і мови формальної логіки) при побудові складних міждисциплінарних зв'язків і відношень у рамках формальної семантики [322].

Розроблений в університеті Сан-Пабло (Іспанія) набір інструментів OntologyTest призначений для перевірки відповідності онтології функціональним вимогам. Програма має вигляд набору тестів, які можна виконувати незалежно один від одного, що дає змогу спростити процедуру перевірки онтологій як у процесі створення, так і на етапі їх експлуатації.

OntoQA – це програмний засіб оцінки й аналізу онтологій, що використовує набір метрик для визначення й оцінки різних параметрів онтологій і баз знань. Метрики поділяються на дві категорії – оцінка структури організації онтології й те, як саме структуровані записи в онтології. Специфічною рисою цього продукту є його здатність до взаємодії з онтологією, яка знаходиться на стадії розробки, що значно спрощує процес аналізу.

OOPS (OntOlogy Pitfall Scanner) – це Web-застосування, призначене для виявлення потенційних проблем у логіці онтології, які можуть спричинити появу помилок у моделюванні ПрО. Ця програма дає змогу розробникам онтології перевіряти свої системи на наявність логічних помилок на етапі створення. Прикладом помилок, що виявляються за допомогою цього інструменту, є зацикленість визначень двох класів один через одного, що може призвести до проблем у роботі механізму підтримки ухвалення рішень.

OOR (Open Ontology Repository) – це онтологічний репозиторій, що надає можливості для збереження, обміну, пошуку, керування та інших додаткових сервісів для роботи з базами знань.

OpenLinkVirtuoso – універсальний сервер, що призначений для збереження й обробки мультимодельних (RDF і SQL) довідкових даних. Він надає платформонезалежну підтримку безпечного збереження, обміну й інтеграції онтологічних даних. Гібридна архітектура Virtuoso надає функціонал класичного сервера у сфері керування даними в реляційних таблицях SQL і керування даними в SPARQL-сумісних RDF.

RepOSE (Repair of Ontological Structure Environment) – система для пошуку й виправлення помилок в онтологіях. Цей продукт може розпізнавати й помилки моделювання ПрО, і семантичні дефекти онтологій. Є кілька редакцій RepOSE, одна з яких вільно поширена для некомерційного використання.

SigmaKEE (Sigma Knowledge Engineering Environment) – це система для створення, підтримки й оцінки теорій, відображених мовою логіки першого порядку. Вона працює з форматом KIF і оптимізована під онтологію SUMO.

### 1. 8. 13. Рекомендації з оцінки онтологій

На основі аналізу оцінки онтологій на різних етапах ЖЦ, наведеної в [15], можна дійти таких висновків: немає єдиного ЖЦ онтології з жорстко закріпленими етапами. Однак в онтологічному аналізі наявні повторювані ланцюжки дій з прогнозованими результатами, що замикаються один на одному. Для забезпечення якості онтології необхідна методика оцінки цих результатів. Отже, оцінка – це не разова дія, а процес, що неодноразово повторюється в ЖЦ онтології.

Результати різних етапів ЖЦ онтології відповідають різним критеріям і повинні оцінюватися відповідно до них. Зокрема неформальні моделі, довідкові онтології й операційні онтології оцінюються по-різному, навіть якщо реалізуються однією мовою.

Онтології оцінюються за критеріями, що залежать як від проектних рішень, так і від сценаріїв використання цих онтологій.

Отже, повноцінна оцінка онтології повинна вбирати в себе оцінку інформаційної системи, частиною якої є онтологія.

Сьогодні наявний дефіцит інструментів, які б давали змогу проводити постійну оцінку онтології на всіх стадіях ЖЦ. Такі інструменти необхідно розробити й упровадити в популярні середовища розробки й репозиторії.

Викладений вище матеріал підтверджує актуальність переходу до застосування семантичних технологій у процесі створення інформаційних систем, що відображається в стрімкому розвитку цієї галузі знань як у сфері стандартів, так і в появі програмних засобів створення й підтримки онтології.

### 1. 9. Методології онтологічного аналізу

Розробка онтологій відома під назвами інженерія онтологій (ontology engineering) або побудова онтологій (ontology building), а також вони можуть бути досліджені під рубрикою онтологічне навчання (ontology learning) [418].

Найпростіша побудова онтології зводиться до:

- 1) Виділення базових концептів ПрО;
- 2) Побудови зв'язків між концептами визначення відношень і взаємодій між базовими концептами.

Процес побудови онтологій може бути висхідним або низхідним. Однак у зв'язку з тим, що висхідний підхід надзвичайно трудомісткий і на сьогодні ще немає засобів, які дали б змогу створити повну систему знань («модель світу»), в основному застосовується низхідний підхід, орієнтований на конкретні, часто дуже обмежені практичні задачі. Отже, є проблема створення онтологій у вузькій предметній області, що порушує питання про створення чотирьох останніх рівнів ієрархії. Загалом застосовуються такі підходи до побудови онтологій: CyC method, Methology by Uschold і King, METHONTOLOGY, On-To-Knowledge тощо [260, 330, 429, 449]. Вони відображають процеси побудови онтології, пропонують способи виділення понять предметної області, певну модель життєвого циклу розробки онтології, способи формалізації знань.

Проведений аналіз стану розвитку методологічних досліджень свідчить, що темпи розвитку нових методологій за останні п'ять років зменшилися, хоча проблема створення онтологій і дотепер залишається нерозв'язаною, а загальноприйнятої методології, яка б відповідала більшості задач, поки що немає. Крім того, наявні методології орієнтовані на спеціалістів з онтологічного аналізу, а не на спеціалістів

прикладних галузей і потребують значних спеціальних навичок для свого ефективного застосування.

Відомі методології розробки онтологій можна поділити на прості, які надають тільки базові рекомендації для розробки онтологій (Sensus, Kaktus, Uschold and King, Gruninger and Fox, Noy and McGuinness тощо), і зрілі, що забезпечують деталізовані підходи до розробки онтологій на всіх етапах їх життєвого циклу (Methontology, OTK, UPON тощо). Крім того, іноді виділяють методології, які забезпечують розподілену розробку (приміром, HCOME і DILIGENT), але, на наш погляд, таке виділення не є суттєвим.

Онтологічний аналіз звичайно починається з укладання словника термінів, які використовуються під час обговорення й дослідження характеристик об'єктів і процесів, що становлять відповідну систему, а також зі створення системи точних визначень цих термінів. Крім того, документуються основні логічні взаємозв'язки між уведеними термінами й відповідними їм поняттями. Надалі ми не розмежовуватимемо поняття й терміни. Результатом цього аналізу є онтологія системи або сукупність словника термінів, точних їх визначень і взаємозв'язків між ними [417].

### 1. 9. 1. Основні етапи онтологічного аналізу

Отже, онтологія складається з сукупності термінів і правила, згідно з якими ці терміни можуть комбінуватися для побудови достовірних тверджень про стан розглядуваної системи в певний момент часу. Крім того, на основі цих тверджень можна дійти відповідних висновків, що дадуть змогу вносити зміни в систему для підвищення ефективності її функціонування [226].

Будуючи онтологію, насамперед необхідно укласти список або базу даних дескрипторів і за допомогою їх, якщо їх набір достатній, створити модель системи. Таким чином, на початковому етапі повинні бути виконані такі завдання:

1) створення й документування словника термінів;

2) опис правил і обмежень, відповідно до яких на базі введеної термінології формуються достовірні твердження, які передають стан системи.

3) побудова моделі, яка на основі наявних тверджень дає змогу формувати необхідні твердження для застосування.

### 1. 9. 2. Методологія IDEF5

Методологія структурного аналізу й проектування SADT (Structured Analysis and Design Technique) зумовила появу цілого ряду методів моделювання IDEFx [341] для задач інжинірингу й реінжинірингу бізнес-процесів, зокрема й IDEF5 що забезпечує

наочне подання даних, отриманих унаслідок обробки онтологічних запитів у простій природній графічній формі. Методологія SADT використовує структурний підхід, тобто роздільну побудову моделі функцій (діаграми потоків даних) і моделі даних (діаграми «сутність – зв'язок») [211]. Такий підхід передбачає п'ять основних етапів:

1) Вивчення й систематизування початкових умов. Ця дія встановлює основні цілі й контексти проекту розробки онтології, а також розподіляє ролі між членами проекту.

2) Збирання й накопичення даних. На цьому етапі відбувається збирання й накопичення необхідних початкових даних для побудови онтології.

3) Аналіз даних. Ця стадія припускає аналіз і групування зібраних даних і призначена для полегшення побудови термінології.

4) Початковий розвиток онтології. На цьому етапі формується попередня онтологія на основі відібраних даних.

5) Уточнення й затвердження онтології. Заключна стадія процесу.

Процес побудови онтології за методологією IDEF5 полегшує збирання даних про фізичні й концептуальні об'єкти разом з їх асоціаціями й забезпечує засоби для діаграмного відображення онтології.

### 1. 9. 3. Методологія METHONTOLOGY

Підхід до побудови й супроводу онтологій METHONTOLOGY базується на принципах Грубера й виділяє в «життєвому циклі» створення онтології такі етапи, як керування проектом, розробку онтології та підтримку розробки [259].

Процедури керування проектом передбачають планування, контроль і гарантії якості. Планування визначає, які завдання мають бути виконані, як вони організуються, які для цього потрібні ресурси і як багато часу необхідно на виконання завдань. Контроль гарантує, що заплановані завдання виконані й саме так, як це передбачалося. Гарантії якості потрібні для того, щоб бути впевненим у тому, що компоненти й продукт загалом перебувають на заданому рівні. Власне розробка передбачає специфікацію, концептуалізацію, формалізацію й реалізацію.

Відповідно до методології METHONTOLOGY спочатку необхідно побудувати глосарій термінів ПрО, а коли глосарій термінів досягає «істотного» обсягу – дерева класифікації концептів для ідентифікації таксономій ПрО. Потім будують діаграми бінарних відношень для фіксації відношень між концептами однієї або різних онтологій, які можуть слугувати базовим матеріалом для інтеграції різних онтологій. Для кожного дерева класифікації концептів мають бути побудовані:

1. Словник концептів, який містить усі концепти ПрО, екземпляри таких концептів, атрибути екземплярів концептів, відношення, джерелом яких є концепт, а також (опційно) синоніми й акроніми концепту.

2. Таблиця бінарних відношень для відношень, початкові концепти яких містяться в класифікаційному дереві, де фіксуються його ім'я, імена концепту-джерела й цільового концепту, інверсне відношення тощо.

3. Таблиця атрибутів екземпляра для кожного екземпляру зі словника концептів (ім'я атрибута, тип значення, одиниця виміру, точність, діапазон зміни, значення «за замовчуванням», атрибути, які можуть бути виведені з використанням цього атрибута, формула виведення атрибута).

4. Таблиця атрибутів класу для кожного класу зі словника концептів з аналогічними характеристиками.

5. Таблиця логічних аксіом, у якій наводяться визначення концептів через завжди правдиві (true) логічні вирази. Визначення кожної аксіоми охоплює її ім'я, природномовний опис, концепт, до якого аксіома належить, атрибути, що використовуються в аксіомі, логічний вираз, який містить формальний опис аксіоми.

6. Таблиця констант, де для кожної константи вказується її ім'я, природномовний опис, тип значення, саме значення, одиниця виміру, атрибути, які можуть бути виведені з використанням цієї константи тощо.

7. Таблиця формул для кожної формули, яка вміщена в таблицю атрибутів екземпляра.

8. Дерева класифікації атрибутів, які графічно відображають відповідні атрибути й константи, які використані для висновку значення кореневого атрибута й формули, які застосовані для цього. Загалом ці дерева використовуються для того, щоб перевірити наявність всіх атрибутів, що містяться у формулі, та їх описів.

9. Таблиця екземплярів для кожного входження в словник концептів.

### 1. 9. 4. Аналіз базової методології розвитку онтологій

Відповідно до аналізу наявних онтологічних методологій, розробка онтології передбачає такі фази:

- **Постановка задачі** (передпроектний аналіз). На цьому етапі потрібно встановити типи необхідних для виконання завдання онтологій, визначити інформаційні ресурси, які використовуватимуться в проекті, з'ясувати необхідні вимоги до компетенцій спеціалістів, визначити критичні й контрольні точки проекту.
- **Специфікація технічних вимог** (ТЗ). Аналіз і уточнення вимог, аналіз інформаційних ресурсів, укладання специфікації вимог для різноманітних ролей, специфікація вимог до інструментів.

- **Вибір інформаційних ресурсів**, пошук застосувань нових ІР, дослідження й оцінка ІР, вибірка та кастомізація ІР під вимоги. Необхідні інструментальні засоби: пошукові системи, обробники природної мови, системи розпізнавання текстів, системи обробки природної мови.
- **Вибір інструментальних засобів** для навчання онтології: пошук інструментарію для навчання онтології, оцінка й вибір. Необхідні інструментальні засоби: інструментальні засоби навчання онтологій (TextToOnto, Text2Onto, OntoLT, OntoLearn тощо).
- **Підготовка до навчання онтології**. Призначення інструментів СКБД доменам. Призначення (прив'язка) документів до інструментів. Специфікація виходів і точок взаємодії з користувачем. Специфікація методів для кожного інструментарію. Кастомізація обраних ІР. Специфікація порядку виконання інструментів.
- **Створення адекватного середовища навчання онтологій.** Виконання процесу навчання онтології за допомогою вибраних інструментів. Виконання відповідних ручних коригувань (ручного введення), необхідних для навчання онтологій при роботі інструментів. Оцінка й модифікація проміжних результатів. Ре-конфігурація набору інструментів. Необхідні інструментальні засоби: інструментальні засоби навчання онтологій (TextToOnto, Text2Onto, OntoLT, OntoLearn тощо).
- Оцінка й обчислення онтології.
- **Інтеграція онтологій**. Переведення в різноманітні формати подання. Інтегрування, об'єднання злиття онтологій. Оцінка результатів. Результатом цього етапу буде кінцева онтологія. Необхідні інструментальні засоби: транслятори, інструменти злиття онтологій (PROMPT, AnchorPROMPT, GLUE, FCAMerge та ін.).

На нашу думку, у цій методології бракує ще одного, останнього етапу, пов'язаного з розгортанням, тестуванням, розвитком онтологій і підтримкою версійності й походження онтологій.

Необхідно констатувати, що всі наявні на сьогодні методології побудови онтологій орієнтовані лише на досить вузьку конкретну задачу; застосовують свої, часто застарілі й закриті стандарти й підходи; а стандарти Semantic Web і мову OWL 2. 0 не використовують; орієнтовані на спеціалістів-онтологів, а не на експертів ПрО; носять занадто загальний характер і орієнтуються лише на розробку онтологій «верхнього рівня».

Отже, відкритою залишається потреба в простій узагальненій методології, придатній для використання неспеціалістами, яка підтримувала б сучасні стандарти керування знаннями й теорію відкритого світу, а також базувалася б на новітніх технологіях програмування (відкриті й гетерогенні джерела даних, Web-сервіси, API, мультиагентна парадигма).

### 1. 9. 5. Загальні етапи побудови онтології

На основі аналізу наявних методологій побудови онтологій і з урахуванням їх недоліків пропонуємо такі етапи побудови онтології:

1) Вивчення й систематизування початкових умов. Ця дія встановлює основні цілі й контексти проекту розробки онтології, а також розподіляє ролі між членами проекту.

2) Збирання й накопичення даних. На цьому етапі відбувається збирання й накопичення необхідних початкових даних для побудови онтології.

3) Аналіз даних. Ця стадія передбачає аналіз і групування зібраних даних і призначена для полегшення побудови термінології.

4) Початковий розвиток онтології. На цьому етапі на основі відібраних даних формується попередня онтологія. Він передбачає розробку прототипу структури, створення й нарощування робочої онтології, тестування й відображення, пов’язування з іншими онтологіями, а також інтеграцію.

5) Уточнення й затвердження онтології. Заключна стадія процесу. Використання, зберігання й підтримка онтологій.

6) Експлуатація й технічне обслуговування. Розширення робочої онтології та повторне використання.

Проведений аналіз інструментальних засобів підтримки процесу розробки онтологій дав змогу дійти висновку про те, що ці засоби ще досить незрілі й не пов’язані один з одним (що виключає інтероперабельність), а тому мають розвиватися в напрямі підтримки інтероперабельності, зміни інтерфейсу на API, спрощування, а також розроблятися за підтримки найбільш актуального на сьогодні напряму нового стандарту подання онтологій – OWL 2. 0 [378].

### Висновки

Розглянувши сучасні методи подання онтологічних знань, а також розроблені для них формальні моделі, стандарти й програмні засоби, можна дійти висновку, що сьогодні саме онтології є найбільш ефективним і поширеним джерелом знань для інтелектуальних інформаційних систем у Web-середовищі.

# РОЗДІЛ ІІ.
# АНАЛІТИЧНИЙ ОГЛЯД ВИКОРИСТАННЯ ОНТОЛОГІЙ У SEMANTIC WEB

## 2. 1. Технології та стандарти Semantic Web для керування знаннями

Semantic Web пропонує потужний практичний підхід до отримання засобів керування великою кількістю інформації та інформаційних послуг. Для того, щоб зробити інформацію більш корисною, необхідно врахувати семантику інформаційних ресурсів. Але це потребує нових засобів подання даних, які покращують здатність до збирання й спільного використання знань, а також нових конструкцій програмування та інструментів, що дають змогу обробляти цю інформацію в прикладних програмах [455].

### 2. 1. 1. Головні завдання Semantic Web

Головними завданнями Semantic Web є створення мов семантичної розмітки електронних документів; словників, які відображають структуру та семантику елементів семантичної розмітки, а також реалізація засобів автоматичного генерування й обробки цієї семантичної інформації [458]. У Semantic Web мають підтримуватися такі функції [31]:

- індексація та пошук інформації;
- розробка й підтримка метаданих;
- розробка й підтримка методів анотування;
- подання Web у вигляді великої, інтероперабельної бази даних;
- організація машинного здобуття даних;
- виявлення (discovery) й надання Web-орієнтованих сервісів;
- дослідження в галузі інтелектуальних програмних агентів;
- розвиток підтримки інтерфейсів різних пристроїв і всепроникні обчислення.

### 2. 1. 2. Компоненти Semantic Web

Основними компонентами Semantic Web є онтології, сервіси та програмні агенти. Для їх подання в рамках Semantic Web розроблені наступні відкриті стандарти:

- OWL – мова подання онтологій [382];
- RDF – стандарт опис метаданих [397];
- SPARQL – мова запитів до RDF [427];
- SWRL – мова визначення правил [308, 435].

Програмування в Semantic Web – це потужний новий підхід, що забезпечує найбільш ефективне використання великих обсягів інформації й робить доступними різноманітні сервіси. Модель знань Semantic Web може інтегруватися з прикладними програмами й використовуватися для окремих домено-орієнтованих бізнес-логік з самої програми. Вплив Semantic Web створює нові перспективи для розробки програмного забезпечення, що базується на моделі, орієнтованій на дані («data-centric»), яка використовує велику кількість різноманітних, розподілених даних і має суттєву виразність, полегшене спільне використання й більшу гнучкість.

Semantic Web можна уявити як Web даних, відображених і пов'язаних так, щоб створити контекст або семантику, які подаються за допомогою фактично визначених граматичних і мовних конструкцій. Розширені прикладні програми Semantic Web можуть здійснювати інтеграцію словників предметної області (та їх онтології). Урахування семантики інформаційних ресурсів дає змогу більш ефективно використовувати наявні дані для розв'язання проблем. Якщо семантика джерела інформації не формалізована, то це потребує виявлення її або користувачами, або за допомогою складних програмних інструкцій [168].

Отже, Semantic Web – це потужний напрям, що сприяє підвищенню ефективності розподіленого й спільного доступу до інформації та її використання прикладними програмами [284].

Твердження Semantic Web складаються з таких базових компонентів, як універсальний ідентифікатор ресурсу (URI), мови Semantic Web, онтології та екземпляри даних.

Для реалізації Semantic Web потрібні відповідні програмні інструменти. Інструменти бувають чотирьох типів: інструменти конструювання для створення й удосконалення прикладних програм Semantic Web, інструменти запитів для дослідження Semantic Web, різонери (засоби логічного виведення), щоб додати виведення для Semantic Web, а також машини правил для розширення Semantic Web [28]. На відміну від інших програм, прикладні програми Semantic Web зосереджені не на програмних інструкціях, а на даних. Багатство даних Semantic Web спрощує це завдання. Це дає змогу відокремлювати дані від програмних інструкцій, а також створювати більш гнучкі розв'язання. Крім того, прикладні програми Semantic Web є Web-орієнтованими: вони використовують переваги від масштабування, різноманітності й розподіленості WWW. Багато сучасних програм намагаються розв'язати ці проблеми, але вони не можуть повною мірою застосовувати WWW і працюють лише в обмеженому, ізольованому інформаційному просторі. Застосування Semantic Web здатні

використовувати розмір і різноманітність WWW шляхом створення стандартизованої виразної інформації.

Semantic Web використовує набір нових відкритих інформаційних стандартів, які можуть застосовувати й усі інші. Прикладні програми, що базуються на цих стандартах, здатні до швидкого впровадження нових джерел інформації.

На сьогодні проект Semantic Web (рис. 2. 1) активно розвивається, з'являються нові мови й стандарти роботи з розподіленими знаннями, а також удосконалюються наявні. Тому доцільно в процесі розробки моделей, методів і засобів підтримки сервіс-орієнтованих прикладних програмних систем орієнтуватися саме на ці досягнення Semantic Web і створювати семантичні Web-сервіси, що можуть ефективно використовувати всі переваги нового інформаційного середовища.

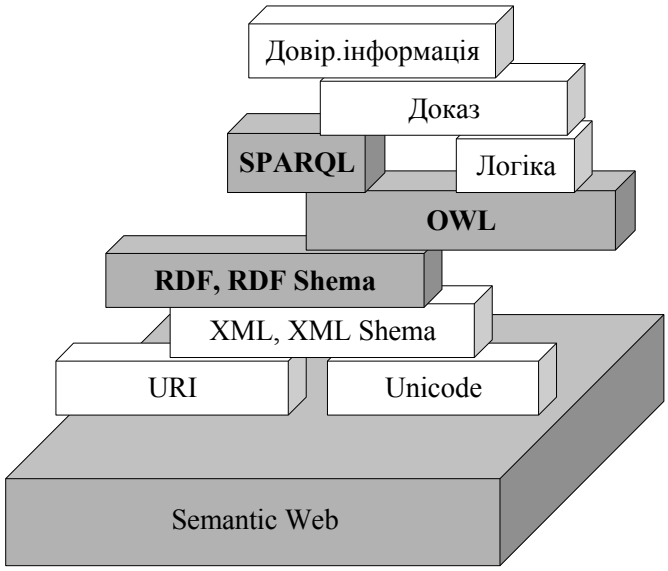



Рис. 2. 1. Стек технологій Semantic Web

Semantic Web – це злиття Web-технологій і науки про подання знань (knowledge representation, KR), що є підгалуззю штучного інтелекту (artificial intelligence, AI) і спрямовується на створення й підтримку потенційно складних моделей світу, що дає змогу розмірковувати про себе й про зв'язану з ними інформацію.

Стандартизація Resource Description Framework (RDF) і Web Ontology Language (OWL) [207] – мови, що покладена в основу Semantic Web, і визрівання нових технологій щодо вбудовування семантики в наявні Web-сторінки й у запити RDF до сховищ знань свідчать про важливість дій у цій галузі.

Підкреслимо, що в рамках проекту Semantic Web сьогодні розроблено велику кількість стандартів, методів і програмних засобів керування онтологічними знаннями [458]. Центральним компонентом концепції Semantic Web є саме застосування онтологій, які дають змогу формалізувати знання щодо ПрО [449,77]. Онтології, на відміну від XML Schema, – це безпосереднє подання знань, а не формат повідомлень [40]. Над онтологіями можна виконувати операції логічного виведення нових знань [429].

Різні інструментальні засоби надають такі можливості: створення онтологій і їх зв'язування з різними ІР; перевірка онтологій на несуперечність, удосконалення онтологій; виконання операцій логічного виведення над онтологіями [360]. Крім того, деякі питання керування знаннями у Web на сьогодні ще остаточно не розвязані: приміром, автоматизована побудова метаописів і онтологій ІР, довірча оцінка наявних метаописів, засоби зіставленення довільних онтологій, автоматизоване поповнення онтологій (рис. 2. 2).

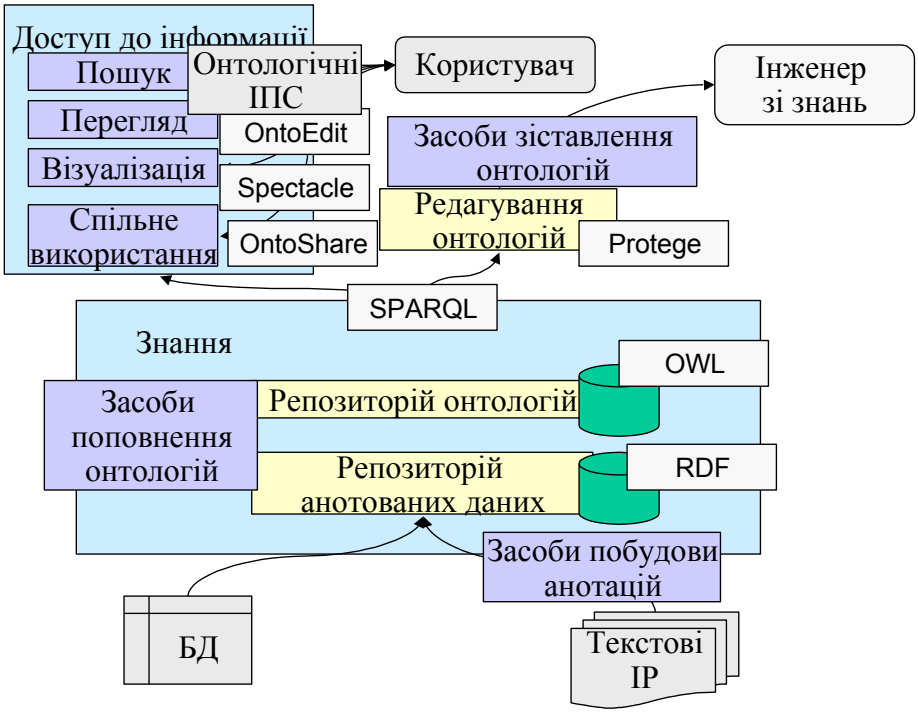



Рис. 2. 2. Використання технологій Semantic Web у керуванні знаннями

### 2. 1. 3. Мови та стандарти подання онтологій

Мови подання онтологій, які відомі сьогодні, можуть бути класифіковані як мови на основі XML або ж як не-XML-орієнтовані мови.

XML-орієнтовані мови [255] отримали найбільш широкий розвиток у Semantic Web, оскільки XML якнайкраще підходить для розв'язання проблеми інтероперабельності, що є однією з основних

вимог Semantic Web. Саме в рамках проекту Sematic Web розроблено такі мови, як RDF і OWL.

RDF(S) [396] є найпростішою і наймасштабованішою XML-мовою подання онтологій. Вона базується на обчисленні предикатів і використовує для визначення семантики трійку – суб'єкт, предикат і об'єкт. Однак вона не достатньо виразна й може використовуватися тільки для вказівки на концепти й бінарні відношення між ними.

Мова OWL [382] розширює RDF(S) і забезпечує конструкції для вираження понять, відношень, потужності, анотацій і конкретизації понять тощо. Окрім того, OWL підтримується великою кількістю редакторів онтологій і вирішувачів. Усе це дає змогу назвати OWL найбільш придатною мовою для Semantic Web.

Для формального подання онтологій раніше були розроблені мови онтологій DAML+OIL і OWL. Обидві мови базуються на RDF і RDF Schema (рис. 2. 3).

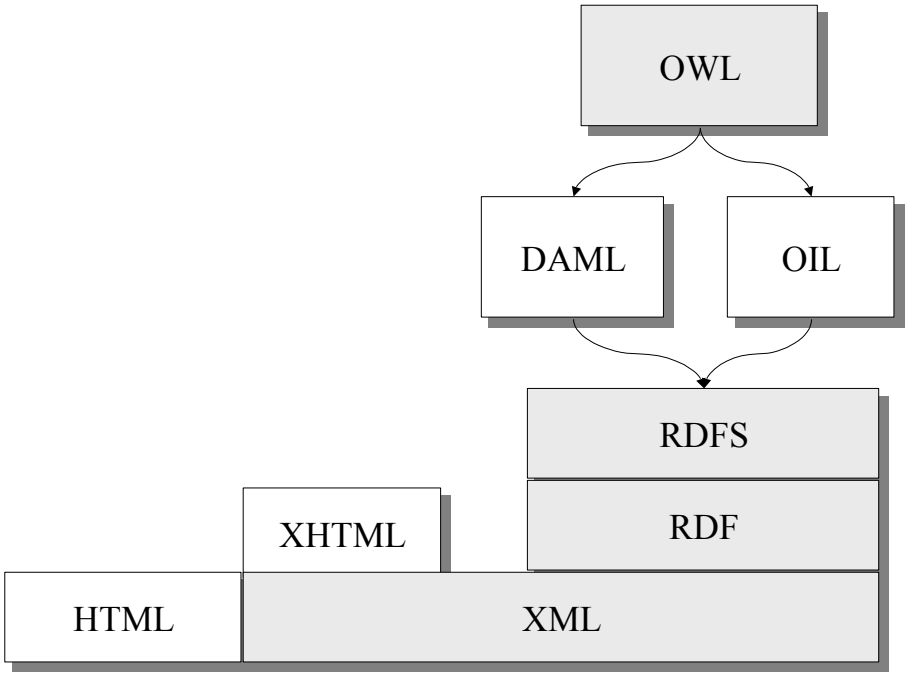



Рис. 2. 3. Мови Semantic Web

DAML+OIL – семантична мова розмітки Web-ресурсів, яка розширює стандарти RDF і RDF Schema за допомогою більш повних примітивів моделювання. Онтологія DAML+OIL – колекція RDF-трійок. Остання версія мови DAML+OIL забезпечує багатий набір конструкцій для створення онтологій і розмітки інформації так, щоб їх могла читати й розуміти машина.

Онтологія DAML+OIL містить:

- заголовки (headers);
- елементи класів (class elements);

- елементи властивостей (property elements);
- екземпляри (instances).

Мова подання онтологій OIL охоплює визначення властивостей (slot-def) і класів (class-def). Визначення Slot-def відображають відношення між двома об'єктами. Class-def зв'язує ім'я класу з його описом і містить ряд компонентів:

- тип визначення defined («зумовлений») або primitive («примітивний»);
- обмеження властивостей (slot constraint) – припустимі значення властивості для екземпляра класу;
- підклас (subclass-of) пов'язує клас зі списком, що складається з одного або кількох виразів класів (class expression): імен класів, обмежень властивостей і складених з них булевих виразів довільної складності. Це підклас класів, заданих цими виразами.

Обмеження властивостей охоплюють такі компоненти:

- ім'я (name) рядок, що позначає властивість, на яку накладається обмеження;
- значення типу (value-type) список з одного чи кількох виразів класів, що становлять діапазон обмежень для властивості щодо зумовленого класу;
- значення (has-value) список з одного або кількох виразів класів, у яких усі екземпляри класу, визначені за допомогою деякої властивості, мають бути пов'язані цією властивістю принаймні з одним екземпляром кожного виразу класу в списку.

Ontology Library – класифікація онтологій DAML, що складається з груп онтологій, об'єднаних за такими параметрами, як:

- унікальний ідентифікатор ресурсу URI;
- дата подання;
- ключові слова;
- каталог Open Directory Category;
- класи;
- властивості;
- використання просторів імен;
- джерело фінансування (організація);
- підпорядковані організації.

Мова подання онтологій OWL розширює можливості XML, RDF, RDF Schema та DAML+OIL. Ця мова базується на DAML+OIL. Проблеми, що виникли в DAML+OIL, зумовлені постійною зміною ядра специфікацій RDF, на якому ґрунтується DAML+OIL. OWL дає змогу усунути деякі обмеження порівняно з DAML+OIL, а також здатний прямо вказувати на симетричність властивостей.

Онтологія OWL (Web Ontology Language), що детальніше розглянута нижче, є послідовністю аксіом і фактів, а також посилань на інші онтології. Вони також містять компонент для запису авторства та іншої подібної інформації. Онтології OWL є документами Web, на них можна посилатися через URI.

OWL має три діалекти, що різняться за виразністю:

- OWL Lite (простота);
- OWL DL (повнота та можливість логічного виведення);
- OWL Full (висока виразність).

OWL DL є розширенням OWL Lite, а OWL Full – розширенням OWL DL. Як наслідок, будь-яка онтологія OWL Lite є онтологієюOWL DL , а будь-яка онтологія OWL DL є онтологієюOWL Full .

OWL Lite – найпростіший варіант, призначений для тих користувачів, які мають потребу класифікувати ієрархію й використовують прості обмеження. OWL Lite забезпечує швидку міграцію тезаурусів та інших таксономій.

OWL DL орієнтований на тих користувачів, які потребують максимальної виразності без втрати повноти обчислень і гарантованого завершення всіх обчислень у визначений час. OWL DL містить усі мовні конструкції OWL з обмеженнями поділу типу (клас не може бути окремою властивістю, а властивість – індивідом або класом). Назва OWL DL пов’язана з його відповідністю дескриптивній логіці.

OWL Full призначається для користувачів, яким потрібна максимальна виразність і синтаксична потужність RDF без обчислювальних гарантій. Наприклад, у OWL Full клас може одночасно розглядатися і як сукупність екземплярів, і як екземпляр. Інша суттєва відмінність від OWL DL у тому, що owl:DatatypeProperty може бути позначена як owl:InverseFunctionalProperty. OWL Full припускає такі онтології, що розширюють склад визначеного словника RDF або OWL.

OWL передбачає відкритість, тобто клас спочатку може бути визначений в одній онтології, а потім розширений в інших. Унаслідок цього судження є монотонними. Нова інформація не спростовує попередню: факти й наслідки можуть тільки додаватися до онтології й не можуть видалятися. Тому інформація може виявитися суперечливою, і розробник онтології має це враховувати.

Онтології містять інформацію про класи, властивості й окремі випадки, кожен з яких може мати ID, що є посиланням URI.

Оскільки більш детально про мову OWL йтиметься далі, то виділимо лише її головні характеристики:

- OWL використовує синтаксис XML;
- OWL має інструкції стосовно задавання дерева класів;

- OWL має інструкції стосовно задавання індивідів до належності класам;
- OWL має систему опису властивостей: область визначення, область значень;
- OWL може задавати характеристики властивостей: симетричність, транзитивність, функціональність;
- OWL має інструкції стосовно задавання еквівалентності (склеювання) класів.

Через великий інтерес до онтологічного аналізу сьогодні є також багато інших мов для подання онтологій. Хоча найбільшу популярність має мова OWL, розроблена в рамках проекту Semantic Web, та її остання версія – OWL 2. 0, проте мови з іншими виразними здатностями теж досить поширені при побудові різних типів онтологій. Більш детальний огляд мов подання онтологій наведено в [118].

Формат обміну знаннями *KIF* (Knowledge Interchange Format) – це спеціальна мова, яка призначена для обміну знаннями між різними комп'ютерними системами, базується на S-виразах синтаксису для логіки й не залежить від конкретних систем [274]. KIF може бути використаний для забезпечення перекладу з однієї мови на іншу або як загальну мову для двох агентів, що використовують різні мови подання (http://www.cs.umbc.edu/kse/kif/). Мова KIF – це мова числення предикатів для подання непроцедурних знань, доповнена засобами подання процедурних знань у вигляді умовних правил переписування виразів.

Визначення мови охоплює опис синтаксису й семантики. KIF містить логічні оператори, які допомагають подавати логічну інформацію (заперечення, диз'юнкція, правила, квантифікаційні формули тощо). KIF дає змогу кодувати знання про знання, використовуючи символи "'" і ",", операторів і пов'язаний покажчик. KIF можна також використовувати для опису процедур, тобто програм ПА. Семантика KIF (без правил і визначень) подібна до логіки першого порядку. Вона розширюється за рахунок використання нестандартних операторів і обмежується аксіомами моделювання. За винятком цих розширень і обмежень, ядро мови зберігає фундаментальні характеристики логіки першого порядку.

На сьогодні онтології системи керування знанням *CYC* – однієї з найстарших баз знань у світі – містять найбільший набір фактів, що відображають різні аспекти людської діяльності [232]. Система CYC досить об'ємна як за своєю функціональністю, так і за обсягом інформації, що міститься в ній. CYC являє собою програмну систему, бібліотека онтологій якої створена для опису знань, необхідних для міркувань на побутовому рівні, тобто таких знань, які людина

використовує у своїй щоденній діяльності. Ім'я «CYC» є фрагментом англійського слова «enCYClopedia» (енциклопедія). Розробка CYC розпочалася в 1984 р.

Знання в бібліотеці онтологій CYC поділені на так звані мікротеорії. Cyc сьогодні містить тисячі мікротеорій. Кожна мікротеорія – факти, що стосуються тієї чи іншої галузі знань. У цьому розумінні мікротеорія являє собою аналог онтології [101].

Для запису фактів у мікротеоріях CYC використовує спеціальну мову – CYC Language [340]. Мова опису онтології CYC [441] – це гібридна мова, у якій об'єднані властивості фреймів і логіки предикатів.

CYC – це формальна мова, синтаксичні конструкції якої точно відтворені й мають однозначне тлумачення. Синтаксис мови базується на двох джерелах – мові логіки предикатів першого порядку та функціональній мові LISP.

Мова CYC являє собою мову числення логіки предикатів першого порядку з аксіомами еквівалентності й можливостями розширення процедур логічного виведення. Мова також підтримує деякі властивості мови другого порядку (наприклад, квантифікацію за предикатами з деякими обмеженнями). База знань CYC складається з термів – словника мови CYC і тверджень, що пов'язують ці терми. Множина термів формується з констант, складених термів, перемінних і деяких інших типів об'єктів. Терм утворюють семантично значущі вирази мови CYC, що застосовуються для запису тверджень у базах знань проекту CYC.

Система CYC використовується не тільки як сховище для тверджень, а й надає можливості для здійснення логічного виводу за цими твердженнями.

Найважливішим компонентом бази знань CYC є лексикон, побудований на основі WordNET. CYC не є онтологією значень слів: не всі концепти мають посилання на слово, що їх позначає, і, навпаки, не всі значення слів лексикону відображені в онтології. Концепти (значення) додаються до онтології лише тоді, коли це необхідно для підтримки логічного виведення на основі здорового глузду. Хоча змістовно лексикон сильно відрізняється від онтології, однак з формального погляду він – жодним чином не виділювана частина бази знань, опис якої здійснюється за допомогою тієї ж мови Cyc*L*, що й основна онтологія.

Мова *OCML* (Operational Conceptual Modeling Language) підтримує побудову кількох типів конструкцій подання знань. Вона дає змогу задавати специфікацію функцій, зв'язків, класів, екземплярів і правил, містить механізми для опису онтологій і методів розв'язання задач – основні технології, розроблені у сфері подання знань [376].

*LOOM* – мова подання знань, розроблена за відкритою ліцензією для подання знань і міркувань у сфері штучного інтелекту [311]. Вона містить систему подання знань, що використовується для дедуктивного виведення на основі декларативних знань, які складаються з визначень, правил, фактів і правил [333].

*Ontolingua* надає розподілене середовище для спільного перегляду, створення, редагування, зміни й використання онтологій [373], що містить парсер KIF, інструменти для аналізу онтології та набір трансляторів для перетворення вихідних даних Ontolingua у форму, прийнятну для впровадження в системи подання знань [371]. Структурною одиницею знання в цій системі є онтологія, тобто набір визначень фрагмента декларативних знань формальною мовою. Ці визначення орієнтовані на спільне багаторазове використання різними користувачами у своїх застосуваннях. До такої онтології вводяться терміни, типи й співвідношення (аксіоми), що відтворюють фрагмент знання. Мова Ontolingua використовує принципи об'єктно-орієнтованого підходу і являє собою розширення комп'ютерно-орієнтованої мови Knowledge Interchange Format (KIF) для обміну знаннями та взаємодії між програмами.

Система Ontolingua реалізується за допомогою мови Common Lisp і призначена для підтримки формальної специфікації задач користувача на основі бібліотеки формальних описів фрагментів задач, моделей і понять, а також для ведення самої бібліотеки фрагментів [8]. Вона містить у собі програму синтаксичного аналізу, утиліту для перевірки правильності перехресних посилань, інструментів-перекладачів з мови системи на мови інших поширених систем подання й генератор звітів у форматі HTML.

*F-Logic* (Frame Logic) – це онтологічна мова, що базується на логіках першого порядку, однак класи й властивості в ній подані як терміни, а не як предикати. Мова створювалася для здійснення взаємодії між онтологіями, побудованими на основі предикатів, і онтологіями, побудованими на основі F-Logic [219]. Ця мова інтегрує інші мови на основі фреймів і числення предикатів першого порядку. F-Logic дає змогу відображати такі структурні аспекти об'єктно-орієнтованих і фреймових мов, як комплексні об'єкти, тотожність об'єктів, успадкування, поліморфність типів, інкапсуляцію тощо.

*SUMO* (Suggested Upper Merged Ontology, Рекомендована онтологія верхнього рівня інтеграції) – це вільно поширювана онтологія, що належить до IEEE (Institute of Electrical and Electronics Engineers). SUMO не призначена для розв'язання якої-небудь конкретної задачі, а повинна інтегрувати наявні онтології в єдину, здатну до розширення структуру, що мала б статус універсального

стандарту й могла б використовуватися в різних прикладних і дослідницьких проектах [361].

SUMO є онтологією верхнього рівня, яка містить близько 1000 понять, 5000 аксіом і близько 800 правил. Більш конкретні поняття зберігаються в окремих галузевих онтологіях і можуть додаватися за необхідності. Для зв'язку між SUMO і галузевими онтологіями розроблена онтологія середнього рівня MILO (Mid-Level Ontology). Увесь доступний на сьогодні комплекс онтологій містить близько 20000 понять і 60000 аксіом [433].

Узагальненим поняттям у SUMO є сутність. Сутності поділяються на фізичні й абстрактні. До категорії абстрактних в онтології SUMO належать такі поняття, як кількість, атрибут, Відношення, Граф, а до фізичних – об'єкти, процеси й носії контенту. В онтології SUMO припускається множинне успадкування [206].

Спочатку онтологія SUMO розроблялася на основі діалекту мови KIF, але згодом вона була перекладена мовою OWL. Розроблено повну систему відповідностей понять SUMO лексичним одиницям WordNET (для англійської мови). SUMO не має ані досить розвиненого онтологічного редактора, ані машини логічного виведення, тобто розробники SUMO надають лише інформацію, що може оброблятися програмно, увіходити як складник до різних застосувань і оброблятися засобами цих застосувань.

*KAON2* (The KArlsruhe ONtology and Semantic Web tool suite 2) – це інфраструктура для керування онтологіями, створена завдяки співробітництву двох вишів – університету Карлсруе й Манчестерського університету. KAON2 дає змогу керувати онтологіями, має програмний інтерфейс для взаємодії з іншими прикладними програмами, може використовуватися як онтологічний сервер, має машину логічного виведення й механізм здобуття знань з реляційних баз даних [317]. Система KAON2 розроблялася не під яке-небудь конкретне завдання: її розробників цікавив не стільки вміст онтології, скільки її формальна структура. З погляду структури, принциповим є поділ онтології на термінологічну частину (TBox) і частину даних (ABox). Оскільки проект орієнтований, головним чином, на розробки у сфері Semantic Web, найпоширенішими виявляються онтології з компактним Tbox і великим Abox, тобто велика кількість документів анотується на основі відносно простої онтології.

З погляду обчислень, основною функцією KAON2 є логічне виведення й відповіді на запити про склад і властивості концептуальної системи, тому модель знань KAON2 найдетальніше опрацьована й має логіко-теоретичне обґрунтування.

У KAON2 є інтерфейси для імпорту онтологій мовами OWL-DL, SWRL і F-Logic, однак для задач логічного виведення використовується власна внутрішня мова, заснована на клаузах Хорна. У процесі відповіді на запити TBox онтології переводиться в програму логічного типу, що потім наповнюється, використовуючи Abox як дані.

Зауважимо, що, крім OWL, є цілий клас формальних онтологічних моделей і пов'язаних з ними мов подання онтологій, які базуються на різних видах дескриптивних логік. Докладніший аналіз таких мов і їх використання наведено в [121]. Першою серед таких систем була KL-ONE [216], що містить ключові поняття використання DL – поняття концептів і ролей та їх взаємодію, обмеження значень, що змінили використання ролей у визначеннях концептів і найбільш важливе виведення категоризації та класифікації.

### 2. 1. 4. Мова опису онтологій OWL

Онтологія OWL (Web Ontology Language) є послідовністю аксіом і фактів, а також посилань на інші онтології. Онтології також містять компоненти для запису авторства та іншої подібної інформації. Онтології OWL є документами Web, на них можна посилатися через URI [378]. Мова подання онтологій OWL розширює можливості XML, RDF, RDF Schema та DAML+OIL [234, 258].

Онтологія визначає терміни, за допомогою яких можна відтворити ПрО. Використання онтології особливо необхідно в застосуваннях-агентах, що здійснюють пошук і об'єднують інформацію з різних джерел і різних середовищ, у яких той самий термін може означати різні речі. Незважаючи на те, що формальний опис структури XML-документів DTD (Document Type Definition) цілком достатній для обміну даними між сторонами, що заздалегідь домовилися про значення визначень і термінів, брак семантики в зазначених засобах опису структури серйозно обмежує надійність виконання завдання пошуку й об'єднання даних при використанні нових XML-словників.

Сильною стороною мови OWL можна визнати орієнтованість її на незалежну розподілену розробку онтологій. Синтаксис цієї мови такий, що будь-який клас, екземпляр або властивість в онтології можна довизначати незалежно від того, як вони були визначені спочатку. Процес довизначення не потребує жодного узгодження з автором початкового визначення й може здійснюватися в окремому документі й без зміни документа, де зафіксовано це початкове визначення. Отже, кожен, кому це потрібно, може редагувати будь-яку онтологію у форматі OWL і створювати власну версію, а тому знання про екземпляри, класи й властивості можуть накопичуватися й уточнюватися поступово, за участю великої кількості людей [188].

Слабкою стороною OWL є те, що ця мова не дає відповіді на питання щодо того, як уникнути додавання до онтології суперечливих тверджень і що потрібно зробити, якщо такі суперечності виникають. Для розв'язання проблеми несуперечності потрібно створювати спеціальні діагностичні методи й засоби [239].

Одним з основних елементів в OWL є класи, які відповідають поняттям предметної області, яку відображає відповідна онтологія. Крім того, для опису ПрО використовуються такі елементи, як властивості й екземпляри класів.

У мові OWL передбачено кілька різних способів визначення нового класу:

- використання ідентифікатора класу;
- повний перелік екземплярів, які в сукупності складають цей клас;
- за допомогою обмеження властивості;
- через теоретико-множинні операції над іншими класами.

Перший спосіб декларує наявність деякого класу, надаючи йому ім'я (ідентифікатор). Таким чином, до онтології вводиться нове поняття (і пов'язаний з цим поняттям зміст), але його властивості та зв'язок з екстенсіоналом залишаються невизначеними. Для цього використовується конструкція з таким синтаксисом:

<owl:Class rdf:ID="Ім'я_класу"/>.

Такій конструкції відповідає RDF-триплет <Ім'я_класу, rdf:type, owl:Class>.

Інші способи визначають новий клас через його екстенсіонал.

Між класами можна встановлювати ієрархічні відношення типу «клас-підклас». Для цього використовується конструкція rdfs:subClassOf. Наприклад, можна визначити клас «собака», вказавши, що він є окремим випадком класу «домашня тварина».

Такому визначенню відповідає така конструкція мови OWL:

<owl:Class rdf:ID="собака">
<rdfs:subClassOf rdf:resource="#домашня_тварина"/>.
</owl:Class>

Відповідно до семантики OWL, це відношення має теоретико-множинне трактування: екстенсіонал підкласу цілком увіходить до екстенсіоналу надкласу.

У термінології OWL властивості, що моделюють атрибутивну інформацію, називаються властивостями-значеннями (data-type properties), а відношення моделювання – об'єктними властивостями (object properties).

*Властивість* в OWL є бінарним відношенням. Мова OWL надає засоби для того, щоб вказати обмеження для області визначення й значення цих відношень. Це можуть бути як безумовні (глобальні)

обмеження, так і умовні (залежні від того, екземпляр якого класу характеризується цією властивістю). Для безумовних обмежень використовуються конструкції rdfs:domain і rdfs:range, для умовних – owl:restriction. Найпростіший спосіб визначити властивість – це оголосити про її наявність (задати ідентифікатор):

<owl:ObjectProperty rdf:ID="зроблено_у"/>.

Властивості одночасно використовуються й для декларації узагальнених тверджень щодо членів класів, і для специфікації тверджень щодо конкретних екземплярів. Прикладом узагальненого твердження може слугувати такий запис:

<owl:ObjectProperty rdf:ID="працює">
<rdfs:domain rdf:resource="#працівник"/>
<rdfs:range rdf:resource="#підприємство"/>
</owl:ObjectProperty>,

який твердить, що працівники можуть працювати на підприємствах або, більш точно, екземпляр класу «працівник» може бути пов'язаний відношенням «працює» з екземпляром класу «підприємство». Відповідним конкретним твердженням може слугувати таке:

<працівникrdf:ID="Рогушина Юлія">
<працює rdf:resource="#Інститут програмних систем"/>
</працівник>.

Крім того, мова OWL дозволяє визначати:

- ієрархію властивостей;
- логічні характеристики властивостей (симетричність, транзитивність);
- зв'зок з іншою властивістю (зворотна властивість);
- безумовні й умовні кардинальні числа, що визначають, у яких пропорціях можуть співвідноситися екземпляри, зв'язані цією властивістю.

Одним з суттєвих недоліків OWL є брак можливості для того, щоб природно визначати властивості у властивостей. Цей недолік не дає змоги моделювати атрибути предметних відношень, n-арні відношення й атрибути атрибутів.

Крім наведених вище засобів подання знань, OWL має набір конструкцій, які уможливлюють встановлення відношень еквівалентності й несумісності між класами, властивостями й екземплярами. Ці можливості певною мірою дають змогу перебороти складності, породжені принципом незалежної розподіленої розробки онтології.

- *еквівалентність класів* owl:equivalentClass визначає еквівалентність між двома класами: якщо між класами C1 і C2 є

відношення еквівалентності, тоді якщо X є екземпляром C1, то X є екземпляром і класу C2, і навпаки;
- *еквівалентність властивостей* owl:equivalentProperty визначає еквівалентність між двома властивостями: якщо між властивостями P1 і P2 є відношення еквівалентності, то, коли X пов'язано з Y відношенням P1, виходить, що X пов'язано з Y відношенням P2, і навпаки;
- *несумісність класів* owl:disjointWith визначає несумісність класів: якщо класи C1 і C2 пов'язані відношенням несумісності, то, коли X є екземпляром класу C1, виходить, що X не є екземпляром C2, і навпаки;
- *еквівалентність екземплярів* owl:sameAs визначає, що два імені екземпляра в онтології посилаються на той самий екземпляр з ПрО;
- *несумісність екземплярів* owl:differentFrom визначає, що два імені екземпляра в онтології неможуть посилатися на один екземпляр з ПрО;
- *несумісність групи екземплярів* owl:AllDifferent визначає, що жодні два екземпляри групи не можуть посилатися на один екземпляр з ПрО.

### 2. 1. 5. Особливості OWL 2.0

Практика дала змогу виявити, що, незважаючи на значне поширення OWL, обмеженість її виразних можливостей і недоліки технічного характеру (складність синтаксичного розбору, неможливість знайти помилки в іменах) [296] спричиняють багато проблем у створенні прикладних знання-орієнтованих систем. Це зумовило потребу у створенні нової версії мови – OWL 2. 0 [380].

Структура онтологій OWL 2. 0 однозначно задається з використанням Meta-Object Facility (MOF) з OMG.

MOF – це широко відома метамова, яка використовується для уточнення інших мов. Метамодель MOF, яку також називають структурною специфікацією для OWL 2. 0, може бути презентована за допомогою Unified Modeling Language (UML).

Класи метамоделі MOF відображають канонічну структуру онтологій OWL 2. 0, що не залежить від синтаксису й використовується для серіалізації онтологій. Специфікація складається з 22 діаграм класів в UML.

Наприклад, на рис. 2. 4 наведено схему UML, яка визначає онтологію, що складається з набору аксіом і набору анотацій. Крім того, це доводить, що онтологія може імпортувати множину інших онтологій, а кожна аксіома – містити набір анотацій.

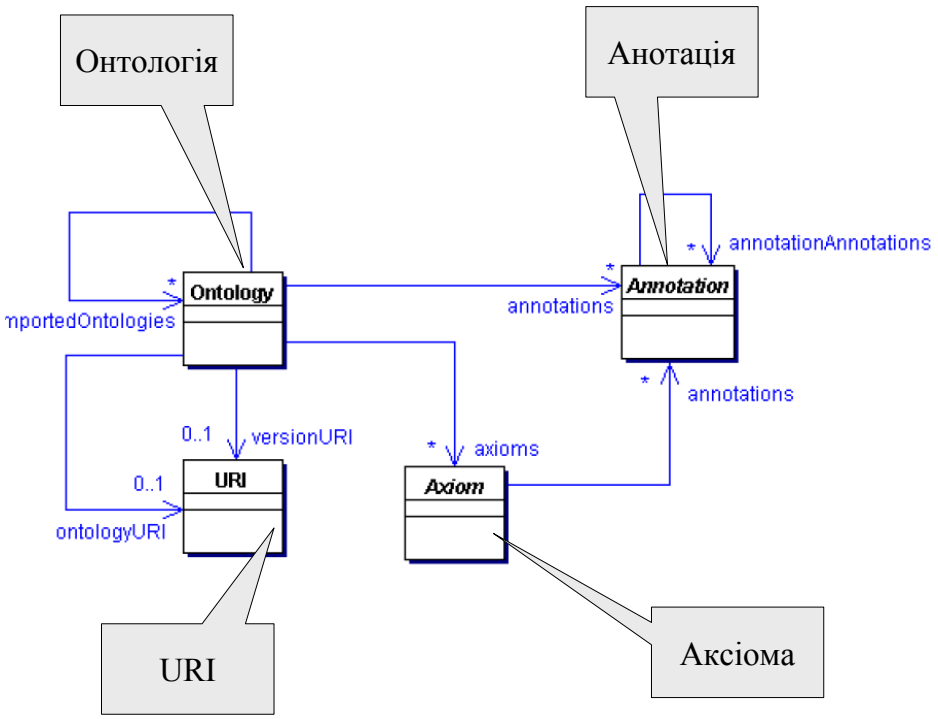



Рис. 2. 4. Об’єктна модель онтології OWL 2. 0

Метамодель MOF для OWL 2. 0 можна розглядати відповідно до специфікації об’єктної моделі документа DOM (Document Object Model) у XML. Аксіоми OWL 2. 0 визначаються як підкласи абстрактного класу Axiom.

Якщо в мові OWL можна визначати лише симетричні й транзитивні властивості, то OWL 2. 0 дає змогу розширити спектр логічних характеристик властивостей рефлексивністю, антирефлексивністю й антисиметричністю. Також додається можливість декларувати локальну рефлексивність, яка використовується тоді, коли для властивості рефлексивність не характерна, а для деяких класів об’єктів рефлексивність наявна [188].

До OWL 2. 0 додано уточнені обмеження кардинальності (qualified cardinality restrictions). Якщо OWL давав змогу лише обмежувати кількість екземплярів, пов’язаних певною властивістю (наприклад, визначити клас осіб, які мають щонайменше десять наукових публікацій), то більш точно визначені обмеження кардинальності дають можливість також обмежити клас або діапазон даних для атрибутивних властивостей тих екземплярів, що обмежуються кількісно (наприклад, визначити клас осіб, які мають щонайменше десять наукових публікацій, проіндексованих Scopus).

Якщо OWL давав змогу лише позначити класи як несумісні, то в OWL 2. 0 з’явилася можливість робити те саме й для властивостей.

Оголошення деякої множини властивостей несумісними означає, що два екземпляри не можуть бути з'єднані більш ніж однією властивістю з цієї множини (приміром, два об'єкти можуть бути пов'язані або властивістю «знаходитися над», або властивістю «знаходитися всередині», але не обома одночасно).

Ще одна нова можливість OWL 2. 0 – визначення нових властивостей через композицію інших властивостей (property chain inclusion). Це дає можливість визначати, наприклад, такі аксіоми: «Якщо стаття A опублікована в журналі B, а матеріали з B викладено у Web на сайті C, то стаття A доступна на сайті C».

OWL 2. 0 містить спеціальну конструкцію для визначення ключа – множини властивостей, яка дає змогу унікально ідентифікувати екземпляри заданого класу. Також в OWL 2. 0 розширено набір типів даних для властивостей атрибутів і введено можливість для того, щоб задавати деякі власні обмеження на діапазони значень.

Інші нововведення носять інструментальний характер і слугують для зручної розробки програм, що використовують онтології. Приміром, надається можливість використання IRI (Internationalized Resource Identifiers) замість URI (Uniform Resource Identifiers), що дає змогу присвоювати класам, властивостям та іншим елементам онтології імена довільними мовами, а не тільки англійською.

### 2. 1. 6. Формат обміну правилами RIF

Rule Interchange Format (RIF) – формат обміну правилами. Метою цього стандарту, розробленого консорціумом W3C [275] є визначення формату, який би дав змогу транслювати правила між різними мовами й завдяки цьому забезпечити обмін правилами між системами правил. Специфікація RIF може розглядатися як складник комплексу стандартів консорціуму для Semantic Web.

Найважливіша вимога до цього стандарту – забезпечення можливості його використання не тільки в поточному стані технологій, заснованих на правилах, а і його гнучкості, достатньої для забезпечення його використання в процесі їх еволюції.

RIF фокусується на обміні, а не на тому, щоб забезпечити єдину для всіх мову правил, оскільки, на відміну від інших стандартів Semantic Web , зокрема таких, як RDF, OWL і SPARQL, відразу було зрозуміло, що одна мова правил не здатна покрити всі популярні парадигми, які використовують правила для подання знань і бізнес-моделювання [399]. Навіть обмін окремими правилами швидко був визнаний надто складним завданням. Відомі системи правил поділяються на три основні категорії: першого порядку, логічного програмування й правил дій. Ці парадигми різняться за синтаксисом

і семантикою. Крім того, є значні відмінності між застосовними системами навіть у межах однієї парадигми.

Робоча група RIF зосереджувалася на двох видах діалектів: логічних діалектах і діалектах для правил з діями. Як правило, діалекти на основі логіки передбачають мови, що використовують якийсь вид логіки, зокрема такий, як логіки першого порядку (часто обмежуючись логіками Хорна) або логіки непершого порядку, що покладені в основу різних мов логічного програмування.

Розробка RIF-FLD як фреймворку для діалектів логіки виявилася можливою, оскільки, незважаючи на різноманітність логічних теорій, на яких базуються різні системи логічних правил, вони мають багато спільного синтаксично й семантично. Крім того, способи об'єднання різних частин цього механізму з метою створення цих логічних систем добре вивчені. І все-таки специфікація RIF-FLD унікальна тим, що вона переробляє багато видів знань, подає їх у когерентній формі й використовує XML навіть на рівні платформи.

RIF-FLD – узагальнена логічна мова, що охоплює велику кількість елементів широко використовуваних синтаксичним і семантичним апаратом; однак вона навмисно залишає певні параметри невизначеними, що дає змогу дизайнерам конкретних діалектів заповнити необхідні деталі. Наприклад, RIF-FLD забезпечує обладнання для налаштування правил синтаксису через поняття підписів. Вона також визначає деякі семантичні поняття, такі, як моделі логічного слідування (entailment), але залишає деякі інші варіанти відкритими (наприклад, які точні моделі будуть використовуватися для цього слідування).

### 2. 1. 7. Правила виведення нових фактів SWRL

Завдяки доповненню OWL мовою RuleML [276] (підмножина Datalog) у вигляді словника SWRL (A Semantic Web Rule Language) [277] з'явилася можливість використати диз'юнкти Хорна (Horn-like rules) для явної вказівки способу висновку нових фактів з RDF-тверджень.

Хоча роботи над цим рівнем Semantic Web ще тривають, однак у нашому розпорядженні є вже достатній набір засобів для побудови знань: твердження (тобто "і") і цитування (матеріалізація) у RDF, класи, властивості, області й документування в схемі RDF, класи що не перетинаються, властивості однозначності й унікальності, типи даних, інверсії, еквівалентності, списки, а також те, що є DAML+OIL.

### 2. 1. 8. Довіра та доказ

Наступним кроком у розробці Semantic Web є довіра та доказ. Зараз ці рівні ще недостатньо досліджені, але розбудова інформаційних систем, що базуються на технологіях Semantic Web, потребує їх розвитку й аналізу.

Основна ідея полягає в тому, що різні ресурси Semantic Web можуть містити твердження, що суперечать одне одному. Цю проблему можна розвязувати в кожному конкретному випадку, використовуючи контекст тверджень і апарат цифрового підпису, який встановлює авторство цих тверджень.

Застосування Semantic Web мають ураховувати контекст для того, щоб повідомляти користувачам, чи можуть вони довіряти наданим даним: контекст дає змогу працювати в локальних середовищах, не вдаючись до складних систем аутентифікації й перевірки.

Отже, використання контексту надає вагому допомогу в розв'язанні цієї проблеми, тому що саме контекст дає можливість працювати в локальних середовищах, спираючись на апріорні довірчі знання, не вдаючись до складних систем аутентифікації й перевірки. А для того, щоб бути впевненим, що якийсь квант даних RDF надійшов саме від певного кореспондента, пропонується використовувати цифрові підписи.

Цифрові підписи – це невеликі фрагменти коду, які можна використати для однозначної перевірки того, хто створив той або інший документ. Він базується на дослідженнях у сфері криптографії й може слугувати доказом того, що автором документа є та людина, якій належить цей підпис. Надалі було б доцільно супроводжувати цифровим підписом усі RDF-твердження.

Припустимо, наприклад, що інформація в RDF-файлі містить посилання на цифровий підпис:

this :signature <http://example.org/signature>.

:Julia :loves :Lada.

Щоб упевнитися в тому, що особа на ім'я Julia дійсно любить особу на ім'я Lada, потрібно завантажити текст RDF у «машину перевірки надійності джерела» (машину логічного виведення з вбудованою програмою перевірки цифрового підпису) і продовжувати подальшу обробку інформації, якщо ця перевірка показує, що джерелу інформації можна довіряти.

Мова перевірки істинності – це просто мова, що дає змогу проконтролювати, є чи ні твердження істинним. Реалізація мови перевірки звичайно передбачає наявність списку «елементів» логічного виведення, які використовуються для одержання інформації, яку

шукають, а також для подальшої перевірки інформації про довіру для кожного з цих елементів.

Наприклад, ми хочемо перевірити, що «Julia loves Lada». У процесі пошуку інформації виявляємо два документи на сайті, якому довіряємо. Перший з них свідчить, що ":Julia loves :Kelli Lada de Mandraka", а другий – що ":Kelli Lada de Mandraka daml:equvivalentTo :Lada". У нас також є контрольні суми персональних файлів від адміністратора сайту.

Для перевірки цієї інформації можна занести контрольні суми до локального файлу й потім актуалізувати певні правила, які підтверджують, що «якщо файл 'a' містить інформацію Joe loves Mary і в нього контрольна сума md5:0qrhf8q3hfh, то зафіксувати Success». «Якщо файл 'b' містить інформацію MJS is equivalent to Mary й у нього контрольна сума md5:0892t925h, то зафіксувати Success» і «якщо Success і Success, то Joe loves Mary».

## 2. 2. Стандарт метаописів RDF

Стандарт метаописів RDF (Resource Description Framework) – це засіб для формального опису семантики інформаційних ресурсів Web у формі, що придатна для їх автоматичного оброблення [396]. Він надає можливість формулювати твердження у вигляді, придатному для обробки комп'ютером [396, 397].

### 2. 2. 1. Призначення семантичних метаданих

Консорціум W3C у рамках проекту Semantic Web пропонує формування повної системи автоматизованого створення й збереження семантичного ядра контенту, презентованого у Web [396].

Зі збільшенням кількості інформації в документах і ускладненням їх структури простота технології Web перестала бути перевагою й почала перетворюватися на недолік. Відмінна тенденція сучасного розвитку Web – це перехід від документів, що *читаються комп'ютером* (machine readable) до документів, які *комп'ютер розуміє* (machine understandable).

У 1997 році основна увага W3C була зосереджена на моделях даних і описі контексту їх застосування. У цьому ключі виконані роботи з опису метаданих на XML, проаналізовані способи застосування гіпертекстових посилань у HTML, запропоновані потенційні його поліпшення.

Для опису предметної області ресурсів був запропонований стандарт RDF, прийнятий у 1999 році консорціумом W3C і підтриманий багатьма провідними ПЗ та постачальниками контенту. Спочатку призначенням RDF був опис XML-ресурсів з різних поглядів [473].

RDF – це модель опису метаданих. Ця мова також використовує XML-синтаксис [217, 354].

XML використовується для таких цілей [270]:

- *надання синтаксису для інших мов розмітки* (наприклад, мова Synchronized Multimedia Integration Language – SMIL).
- *семантичної розмітки Web-сторінок:* XML-подання може використовуватися на Web-сторінці разом з таблицею стилів XSL, що визначає коректний висновок різних елементів.
- *єдиного формату обміну даними:* XML-подання може передаватися між двома застосуваннями як об'єкт даних.

### 2. 2. 2. Модель Resource Description Framework

Модель Resource Description Framework має на меті стандартизувати визначення й використання метаданих, що відтворюють ресурси Web. Однак за допомогою RDF настільки ж добре й подавати дані.

Стандарт RDF (Resource Description Framework) складається з двох основних частин: власне способу опису ресурсів, а також способу задавання схем, за якими здійснюється опис ресурсу [397].

Перша частина RDF визначає просту модель для опису об'єкта, який розглядається як ресурс, і зв'язків між ресурсами в термінах пойменованих властивостей і значень. Друга (RDF Schema – RDFS) слугує для задавання структури предметної області й аналогічної діаграми класів в UML [218, 465].

За допомогою RDF можна відтворювати як структуру ресурсу, так і пов'язану з ним предметну область [233].

RDF відтворює ресурси у вигляді орієнтованого розміченого графа – кожен ресурс може мати властивості, що, так само, можуть бути ресурсами чи їх колекціями [326].

Базовий будівельний блок у RDF – це трійка «об'єкт – атрибут – значення», що часто записується у вигляді A(O,V), тобто «об'єкт O має атрибут A зі значенням V». Цей зв'язок можна також подати як ребро з міткою A, що з'єднує два вузли, O і V: [O]-A->[V]. Ця нотація дуже корисна, оскільки RDF дає змогу міняти місцями об'єкти й значення. Таким чином, будь-який об'єкт може відігравати роль значення, що в графічному поданні відповідає ланцюжку з двох ребер із мітками [209].

Крім того, RDF припускає форму подання, у якій будь-який вираз RDF у трійці може бути об'єктом чи значенням, тобто графи можуть бути як вкладеними, так і лінійними. У Web це дає змогу, наприклад, виражати сумнів чи згоду з виразами, створеними іншими людьми. Нарешті, це дає можливість зазначити, що цей об'єкт має визначений

тип, приміром, що «ISBN0012515866» – це rdf:type book, за рахунок створення дуги, що вказує на визначення book у схемі RDF.

Приклад опису ресурсу на RDF:

```
<rdf:RDF>
 <rdf:Description
 about="http://www.abc.kiev.ua/~v_sidorenko">
 <x:owner ID="Василь_Сидоренко"/>
 <x:autor ID="Василь_Сидоренко"/>
 <x:description>Домашня сторінка
 Василя Сидоренка </x:description>
 </rdf:Description>
 <rdf:Description ID=" Василь_Сидоренко ">
 <x:name> Василь </x:name>
 <x:lastname> Сидоренко </x:lastname>
 <x:email> v_sidorenko@gmail.com</x:email>
 </rdf:Description>
</rdf:RDF>
```

Мета RDF – запропонувати базову модель даних «об'єкт – атрибут – значення» для метаданих. Крім цієї передбачуваної семантики, відтвореної в стандарті лише неформально, RDF не містить яких-небудь чітких правил, орієнтованих на моделювання даних. Так само як і XML Schema використовується для визначення словника, RDF Schema дає змогу розроблювачам визначати конкретний словник для даних RDF (такий, як authorOf) і вказувати види об'єктів, до яких можуть застосовуватися ці атрибути. Іншими словами, механізм RDF Schema пропонує базову систему типів для моделей RDF. Ця система типів використовує деякі визначені терміни, такі, як Class, subPropertyOf і subClassOf, для схеми, орієнтованої на конкретне застосування. Вирази схеми RDF також є коректними виразами RDF (як і вирази схеми XML – коректні вирази XML) [388].

Об'єкти RDF можна визначити як екземпляри одного чи декількох класів за допомогою властивості type. Властивість subClassOf дає змогу розроблювачеві вказувати ієрархічну організацію таких класів, а subPropertyOf виконує те саме для властивостей. Обмеження на властивості також можуть бути зазначені за допомогою конструкцій домену й діапазону, що застосовуються як для розширення словника, так і для передбачуваної інтерпретації виразів RDF.

### 2. 2. 3. Набір елементів для створення метаданих Dublin Core

На сьогодні найбільш поширеним набором елементів для створення метаданих є набір, який формується вже протягом декількох років міжнародною групою «The Dublin Core initiative» [236].

Цей набір має назву «Dublin Core Metadata Elements» і складається з 15 базових елементів [248, 450].

Ці елементи можна умовно розподілити на три групи:

1. Content – елементи, що в основному стосуються змісту ресурсу (Title, Subject, Description, Type, Source, Relation, Coverage).

2. Intellectual Property – елементи, що в основному розглядаються з позиції інтелектуальної власності (Creator, Publisher, Contributor, Rights).

3. Instantiation – елементи, які в основному стосуються відповідного екземпляра ресурсу ( Date, Format, Identifier, Language).

Для того, щоб зберегти сумісність з найпростішим описом з 15 елементів і, водночас, збільшити деталізацію і складність описів різних організацій, самі працівники групи The Dublin Core Initiative розробляють розширення, прикладні *кваліфікатори* для базових елементів. Розширювати сам набір елементів можна з використанням уже наявних стандартів.

### 2. 2. 4. Розміщення метаданих

Як зазначається в офіційному описі RDF, метадані можуть бути вбудовані (*embedded*) у сам ресурс, наприклад у сторінку HTML чи документ MsWord (це найпростіший підхід до опису сторінок), а можуть зберігатися й оновлюватися незалежно від ресурсів. Багато виробників програмного забезпечення вже випускає ряд продуктів, приміром редактори, що автоматично формують деякий невеличкий блок RDF-опису всередині документа. Другий підхід більш універсальний, оскільки в ньому метадані можуть бути створені для будь-якого ресурсу. Сьогодні вже започатковано проект на базі Open Directory (пошукова система Google) щодо автоматичного створення репозиторію RDF-описів ресурсів Інтернет [388].

У разі розміщення метаданих окремо від ресурсу самі метадані переважно зберігаються (і передаються) у форматі XML. При цьому максимально використовуються можливості моделі RDF та забезпечується вільний обмін інформацією (*interoperability*). Обмін метаданими зводиться до пересилання RDF/XML-файлів (тобто текстових файлів формату XML чи просто посилань на ці файли), тобто може бути цілком автоматизований.

На основі RDF 23 січня 2003 р. був запропонований робочий проект RDF Vocabulary Description Language 1.0: RDF Schema. Як зазначається в документі, RDF є мовою загального застосування для подання інформації в Інтернеті. Така специфікація вказує на те, як використовувати RDF для опису RDF-словників. Вона визначає базовий словник, призначений з цією метою, й ухвалені угоди, що можуть бути

використані при створенні застосувань Semantic Web на підтримку більш складних словників RDF-описів. Мова опису словника RDF визначає класи та властивості, що можуть бути використані для опису інших класів і властивостей.

RDF Schema визначається в термінах базової інформаційної моделі RDF – структура графа, що відображає ресурси та властивості. Всі словники RDF використовують деяку базову структуру: вони відображають класи ресурсів і типи зв'язків між ресурсами. Ця спільність дає змогу використовувати різнорідні словники, створені для машинної обробки, й відповідає вимогам щодо створення метаданих, у яких твердження можуть бути отримані з множини різнорідних децентралізованих словників, створених за різними принципам, різними методам й різними співтовариствами.

Опис за допомогою RDF не обмежується тільки описом документів Інтернету. Цей стандарт досить універсальний і гнучкий для того, щоб відтворювати багато типів структурованих даних. Наприклад, у RDF природно виражаються діаграми «сутність-зв'язок», що широко застосовуються для проектування баз даних. Опис семантики ресурсу на RDF може бути як «зовнішнім», коли відтворюється ресурс загалом, так і «внутрішнім», коли відтворюється внутрішня структура ресурсу – чи то база даних, чи то XML-документ, чи то цілий сайт.

Важливою особливістю стандарту RDF, як і XML, що покладена в його основу, є розширюваність. На RDF можна задати структуру опису джерела, використовуючи й розширюючи вбудовані поняття RDF-схем, такі, як класи, властивості, типи, колекції. Модель схеми RDF передбачає успадкування: успадковуватися можуть класи та властивості.

Крім опису структури, RDF дає змогу оперувати твердженнями. Вираз «ресурс R1 як властивість P має ресурс R2» можна інтерпретувати як предикат P(R1, R2), а потім використовувати це твердження як об'єкт інших тверджень. Така інтерпретація дає можливість відтворювати за допомогою RDF концептуальну інформацію.

Отже, RDF може цілком виконувати роль універсальної мови опису семантики ресурсів і взаємозв'язків між ними.

Мову запитів до RDF-джерел даних (RDF Query) було запропоновано в 1998 р. і сьогодні вона має вже практичну реалізацію в проекті Sesame.

Розроблено вже ряд програмних продуктів, що дають змогу створювати RDF-описи для різного роду джерел (наприклад, RDFPic створений для впровадження RDF-опису в зображення), робити

валідацію RDF-описів документів (RDF-validator), редагування (RDF-editor) (рис. 2. 5) й передбачають можливості інтеграції готових сховищ інформації в загальну базу семантичного опису. У розрізі мультимедійних даних передбачається інтеграція концепції RDF-бази з форматом MPEG.

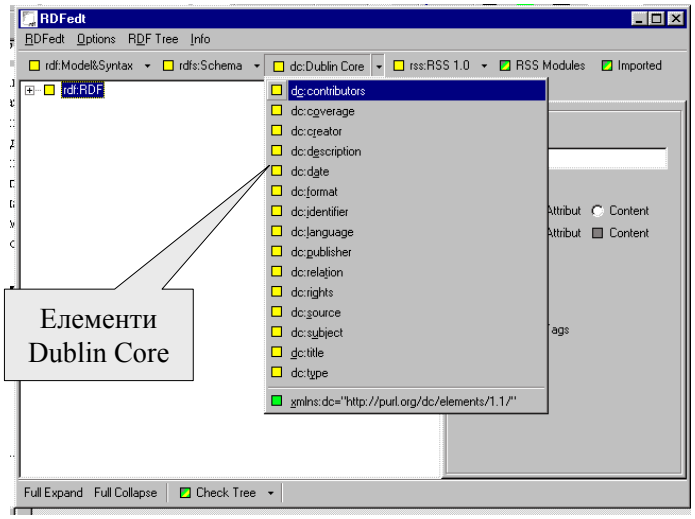



Рис. 2. 5. Елементи Dublin Core

Властивості стандарту RDF дають змогу використовувати його як засіб для інтероперабельного подання знань, отриманих засобами Data Mining унаслідок обробки різноманітних інформаційних ресурсів Web. Це забезпечує автоматизоване створення RDF-описів для природномовних і мультимедійних документів.

З іншого боку, RDF-описи можуть самі використовуватися як інформаційні ресурси, а методи Data Mining забезпечують здобування неявно поданих у них відомостей, приміром, закономірностей у результатах інформаційного пошуку або рейтингу сайтів однотипної спрямованості.

## 2. 3. Мова запитів SPARQL

З визнанням RDF назріла потреба в стандартній мові запитів для RDF, що відіграє ту саму роль, що й SQL у реляційних даних.

У 2006 році консорціум W3C започаткував розробку мови запитів до RDF і OWL-сховищ – SPARQL [428]. 15 січня 2008 року SPARQL став офіційною рекомендацією W3C.

SPARQL – мова запитів, розроблена для моделі даних RDF. Використання твердження SPARQL як стандартної мови запитів для RDF дає змогу багатьом сховищам даних стати точками доступу до SPARQL, у такий спосіб забезпечуючи гнучкий обмін даними між системами. Це відкриває шлях до Web-застосувань, у яких один компонент

використовує інший як джерело даних (наприклад, через SPARQL) і сам виступає в ролі джерела даних для третього компонента [427].

SPARQL одночасно є і мовою запитів, і протоколом доступу до даних, одним з ключових компонентів Web 2. 0: як стандарт для підтримки гнучкої моделі даних він надає спільний механізм запитів до Web 2. 0. SPARQL використовується для подання запитів до різноманітних джерел даних, незалежно від того, зберігаються ці дані безпосередньо в RDF або надаються у вигляді RDF за допомогою проміжного програмного забезпечення (middleware).

Важливий термін SPARQL – точка доступу (endpoint). Точка доступу, або SPARQL-процесор – це сервіс, що дає змогу користувачеві вводити свої запити, обробляє ці запити й видає результат, що повертається, у різних форматах.

Розрізняють два види точок доступу – загального призначення й локальні. Точки доступу загального призначення можуть робити запити щодо будь-яких зазначених RDF-документів, що розміщені у Web, тоді як локальні точки доступу здатні одержувати дані тільки з одного ресурсу.

Приклади локальних точок доступу: NASA endpoint, BBC wildlife endpoint, DBPedia endpoint.

Надання SPARQL- точок доступу (SPARQL-endpoint) є рекомендованою практикою при публікації даних у всесвітній павутині.

SPARQL є мовою запитів, що базується на патернах графів.

Синтаксис запиту в спрощеному варіанті має такий вигляд:

```
SELECT   <v_list>
FROM     <onologyURI>
WHERE    {     <template_list>.
               FILTER <filter_expr>
          }
```

де

- v_list список імен перемінних,
- onologyURI посилання на онтологію,
- template_list список шаблонів,
- filter_expr обмеження на значення перемінних.

Загальна схема SPARQL-запиту має такий вигляд:

```
PREFIX foo: <http://example.com/resources/>
# префіксні оголошення
FROM...
# джерела запиту
SELECT...
# пункт результату
```

```
WHERE {...}
# критерії запиту
ORDER BY...
# модифікатори запиту
```

Префіксні оголошення слугують для скорочення універсальних ідентифікаторів ресурсу. Джерела запиту визначають, які RDF-графи запитуються.

Наприклад, такий запит дає змогу дістати випадкову вибірку зі 100 фактів, що зберігаються у відповідному наборі даних.

```
SELECT DISTINCT *
WHERE
{
?x ?y ?z.
}
LIMIT 100
```

Виконання SPARQL-запитів до онтологій підтримується плагіном у редакторі онтологій Protégé. Наприклад, цей простий запит дає можливість знаходити всі підкласи класів поточної онтології (рис. 2. 6):

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
        WHERE { ?subject rdfs:subClassOf ?object }
```

SPARQL дає змогу користувачам писати глобально однозначні запити. Наприклад, наведений нижче запит повертає імена й адреси кожної людини у світі:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
 ?person a foaf:Person.
 ?person foaf:name ?name.
 ?person foaf:mbox ?email.
}
```

Наведені параметри використовуються для опису людини, внесеної до FOAF. Це ілюструє бачення Semantic Web як єдиної величезної бази даних. Кожен ідентифікатор у SPARQL, URI глобально однозначний, на відміну від «email» чи «e-mail», які, зазвичай, використовуються в SQL.

Цей запит може бути розподілений на кілька кінцевих точок SPAR різних комп’ютерів, і збирання результатів здійснюється за процедурою, відомою як федеративний пошук.

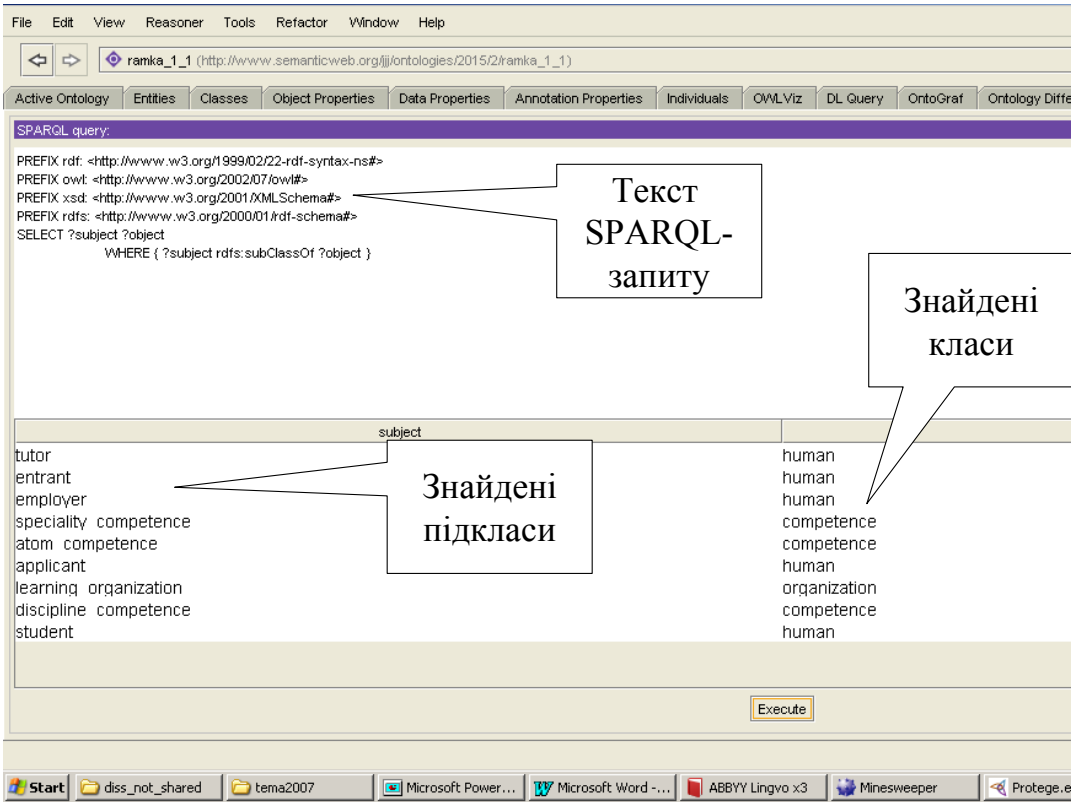



Рис. 2. 6. Приклад виконання SPARQL-запиту в Protégé

Наведемо приклад SPARQL-запиту, припустивши, що дані RDF зберігаються у вигляді таких триплетів (S, P, O):

(Foo1, category, "*Total Members*");
(Foo1, rdf:value, 199);
(Foo2, category, "*Total Members*");
(Foo2, rdf:value, 200);
(Foo2, category, "CATEGORY X");
(bar, category, "CATEGORY X");
(bar, rdf:value, 358).

Розглянемо наступний SPARQL-запит:

SELECT ?cat ?val
WHERE {?x rdf:value ?val. ?x category ?cat.
FILTER(?val>=200).}

У цьому запиті вираз SELECT здійснює опис змінних cat і val, значення яких необхідно з'ясувати.

Вираз WHERE накладає шаблони вигляду (S, P, O).

Усередині шаблону можна використовувати зв'язані перемінні – x.

Вираз FILTER накладає обмеження на значення перемінної val.

Семантика цього запиту полягає в тому, щоб видати всі об'єкти cat предиката category, суб'єкт якого (x) є також суб'єктом предиката rdf:value зі значенням val, що не менше 200. Разом зі значеннями cat видаються відповідні значення val.

Хід виконання запиту:

Замість перемінної x можуть бути підставлені Foo1, Foo2 і bar, причому Foo2 можна підставити двічі.

При підстановці Foo1 значення перемінної val не задовольняє обмеженню у виразі FILTER. У всіх інших випадках усі умови запиту виконані (див. результат).

Результат виконання запиту:

[["Total Members", 200], ["CATEGORY X", 200],

["CATEGORY X", 358]]

Мова SPARQL визначає чотири різні варіанти запитів для різних цілей:

- Запит SELECT

Використовується, щоб здобути неопрацьовані значення з точки доступу SPARQL, результати повертаються у форматі таблиці.

- Запит CONSTRUCT

Використовується, щоб здобути інформацію з точки доступу SPARQL і перетворити результати на певну форму.

- Запит ASK

Використовується для створення запитів типу Правильно/Неправильно.

- Запит DESCRIBE

Використовується, щоб здобути граф RDF з точки доступу SPARQL, зміст якого потрібно визначити кінцевій точці, ґрунтуючись на тому, що фахівець з обслуговування вважає корисною інформацією.

Кожна з цих форм запиту містить блок WHERE, щоб обмежити запит, хоча в разі запиту DESCRIBE WHERE не є обов'язковим.

У запитах SPARQL використовують такі ключові слова:

- PREFIX для скорочення URI;
- OPTIONAL позначає необов'язковий шаблон;
- GRAPH використовується для формування запиту, що застосовує шаблон до іменованих графів;
- DISTINCT вказує, що кожне рішення у відповіді на запит є унікальним;
- LIMIT задає максимальну кількість виведених результатів;
- OFFSET дає змогу не показувати в результаті перші n-рішення;
- ORDER BY дає можливість упорядковувати результати за зростанням (ASC()) чи за зменшенням (DESC()).

SPARQL 1. 0, який став стандартом у січні 2008, містить:

- SPARQL 1. 0 Мова запитів;
- SPARQL 1. 0 Протокол;
- SPARQL Формат результатів XML.
- Сьогодні актуальною версією є SPARQL 1. 1, що містить:

- SPARQL мова запитів і протокол, оновлені до 1. 1;
- SPARQL 1. 1 Відновлення;
- SPARQL 1. 1 HTTP-протокол для керування RDF-графами;
- SPARQL 1. 1 Опису служб;
- SPARQL 1. 1 Логічні наслідки(Entailments);
- SPARQL 1. 1 Основні Федеративні запити.

Jena – загальнодоступна Semantic Web платформа для Java, що забезпечує користувачам інтерфейс для створення SPARQL-запитів до онтологічних структур. Вона застосовує API, щоб здобувати знання з RDF-конструкцій і дописувати в RDF-граф нову інформацію. Графи подаються як абстрактна модель, що може надаватися разом з даними.

Jena подібна до Sesame, але, на відміну від Sesame, підтримує OWL (Web Ontology Language). Платформа має різні вбудовані різонери, а зовнішні різонери, приміром, Pellet можна також підключити для роботи в Jena.

Запит SPARQL виконується з використанням набору даних RDF, який має вигляд колекції графів. Різні частини запиту можуть відповідати різним графам, опис чого наведено нижче. Є один граф – граф за замовчуванням, у якого немає імені, і нуль або більше йменованих графів і кожен з них ідентифікований IRI.

Рівень Graph – базовий шар у Jena. Основа архітектури Jena2 – рівень Graph, що містить граф RDF. Цей рівень мінімальний за конструкцією, і можлива функціональність реалізується на інших рівнях. Такий підхід припускає широкий діапазон реалізацій цього рівня, зокрема таких, як сховища пам’яті або стійкі трійки. Рівень EnhGraph – точка розширення, на основі якої можна створити API. У Jena2 функціональність, що пропонується рівнем EnhGraph, використовується для реалізації Jena Model API та нової функціональності онтології для OWL і RDFS.

Наявність рівня Graph спростила завдання розширення основної функціональності Jena.

Для того, щоб розширити API Jena, потрібно розширити інтерфейси Graph. Наприклад, підтримка DB2-RDF у Galileo DB2 розширює рівень Graph Jena. Рівень Model – просте Jena-розширення рівня Graph.

На рівні Graph можна створювати триплети RDF, що складаються з суб’єкта, предиката й об’єкта:

```
Node s = Node.createURI("Rational Software Architect");
Node p = Node.createURI("hasAcronym");
Node o = Node.createURI("RSA");
Triple triple = new Triple(s, p, o);
graph.add(triple);
```

Створення триплетів у рівні Model:

```
Resource s = model.createResource("Rational Software Architect");
Property p = model.createProperty("hasAcronym");
Resource o = model.createResource("RSA");
model.add(subject, predicate, object);
Statement statement = model.createStatement(subject, predicate, object);
```

Model розширив базову функціональність у рівні Graph так, що дав змогу розробникам працювати з об'єктами типу Ресурс, Властивість або Оператор, замість Вузол або Трійки; крім того, більше зручних методів доступно в об'єктах рівня Model.

Третій і останній рівень в API Jena – рівень OntModel (Ontology Model). Він надає можливості логічного виведення. Так, якщо є предикат у RDF-моделі, і він визначений як проміжний з урахуванням його фактичних відношень, то:

A -> B and B -> C

це відношення передбачатиме:

A -> C

Онтологічне API Jena надає огляд онтологій і підтримку міркувань в API Jena.

Розширене подання моделі Jena, яка, як відомо, містить дані онтології, відповідно до словника онтології (такого, як OWL). Цей клас окремо не вираховував дедуктивне розширення графа за семантичними правилами мови. Замість цього використовується базова модель з її інтерфейсом онтології, який надає зручний синтаксис для доступу до елементів мови. Залежно від можливості виведення базової моделі, здаватиметься, що OntModel містить більше або менше трійок.

Приміром, якщо цей клас використовуватиметься для того, щоб охопити очевидну пам'ять або модель бази даних, то лише відношення, затверджувані документом, будуть повідомлені через цей API. Альтернативно, якщо OntModel охоплює модель логічного виведення OWL, виведені з розширення трійки, будуть наведені як позитивні.

Наприклад, розглянемо такий фрагмент онтології:

```
:A rdf:type owl:Class.
:B rdf:type owl:Class ; rdfs:subClassOf :A.
:widget rdf:type :B.
```

У моделі невиведення rdf:type віджет буде презентований тільки як клас: B. Модель, яка може обробити семантику OWL, типи віджету охоплюватимуть: B: A і owl: Thing.

OntModel – це розширення інтерфейсу InfModel. Воно необхідно для підтримки, коли модель онтології охоплює граф виведення, і ми хочемо надати спеціальну можливість InfModel, наприклад, глобальну перевірку несуперечності, доступну для клієнтських програм. Оскільки не всі OntModels застосовують різонери, то використання цих методів може призвести до відключення на етапі виконання, хоча типова поведінка характеризується повним ігноруванням таких викликів.

## 2. 4. Засоби логічного виведення на онтологіях

### 2. 4. 1. Логічне виведення

Принцип логічного виведення – це можливість виводити нові дані з тих даних, які вже є. Логічне виведення є одним з провідних принципів Semantic Web, оскільки воно дає можливість спростити створення застосувань Semantic Web і отримувати за їх допомогою потрібні знання.

Для того, щоб Semantic Web став досить виразним, необхідно побудувати потужну логічну мову, що підтримує логічне виведення. Тривають бурхливі дискусії щодо методів і навіть можливості виконання цього завдання; звертається увага на те, що в RDF недостатньо можливостей квантифікації, і що ця сфера визначена недостатньо добре. Проблеми логічного виведення на основі логіки предикатів докладно розглянуті в монографії Дж. Сови [426].

Виникає потреба в спеціалізованих засобах і мовах, орієнтованих на підтримку логічного виведення на знаннях і на формалізоване подання результатів такого виведення.

Тому на сьогодні поширені такі програми логічного виведення (reasoners), які оброблюють SWRL:

- SWRLTab плагін до редактора онтологій Protege-OWL з вільним доступом (Open Source)
- Pellet програма логічного виведення (reasoner) OWL DL з вільним доступом, написана мовою Java.
- RacerPro комерційний продукт.

Окрім цього, у межах Semantic Web запропонована мова RIF (Rule Interchange Format) – формат, який дає змогу транслювати правила між різними мовами правил і завдяки цьому забезпечує обмін правилами між системами правил.

У рамках проекту Semantic Web розроблено деякі стандарти й мови для підтримки логічного виведення.

### 2. 4. 2. Засоби логічного виведення над онтологіями

Розглянемо засоби логічного виведення над онтологіями, які в літературі часто називають різонерами, та їх можливості [386]. Загалом є два підходи до реалізації логічного виведення: на базі правил (rule-based), з використанням алгоритмів forward-chaining або backward-chaining й на основі семантичного табло (semantic tableau). На базі правил реалізовані, наприклад, Semantics.SDK і Owlim, а на базі семантичного табло – Pellet [420]. Зазвичай, доцільніше використовувати різонери на основі правил для мов з низькою виразністю, а на базі семантичного табло – для мов з високою виразністю. Pellet (OWL-DL) і Owlim (OWL-Tiny) підтверджують це спостереження, а Semantics. SDK (OWL-FULL) є винятком. Найближчими конкурентами Pellet є: FaCT++, який теж використовує алгоритм табло, і Hermi, що використовує алгоритм гіпертабло. Pellet і HermiT написані на Java, а Fact++ на C++. У Protege 4 як основна можливість цього редактора заявлена підтримка OWL 2. 0 й можливість прямого підключення до підсистем логічного виведення Pellet і FaCT++.

Усі ці різонери можна інтегрувати з OWL API. OWL API – це фреймворк для роботи з онтологіями (презентація онтології в пам'яті, можливість її модифікації на рівні об'єктів тощо), що вже де-факто став стандартом. Можна використовувати й інші фреймворки (наприклад, Pellet інтегрується ще й з Jena). З OWL API різонери працюють за визначеним інтерфейсом, тому один різонер досить просто замінити на інший.

Отже, мови подання онтологій, що базуються на DL, є на сьогодні найбільш адекватним засобом подання знань, який має достатні виразні можливості й забезпечує виконання операцій логічного виведення для отримання нових знань.

Для здобуття й оброблення знань, що зберігаються в онтологіях, використовуються блоки логічного виведення (резонери (reasoners)), які відіграють ключову роль в інтелектуальних системах, орієнтованих на знання. *Різонер* – механізм обґрунтування, механізм правил, який є частиною програмного забезпечення, призначеного для логічного виведення знань на основі ряду заявлених фактів або аксіом. Поняття семантичного різонера узагальнює поняття механізму логічного виведення, забезпечуючи багатший набір механізмів для роботи. Правила виведення, зазвичай, визначаються за допомогою мови онтології й часто мови опису. Здебільшого різонери використовують логіку предикатів першого порядку для міркування; виведення, як правило, будується прямим і зворотним ланцюжком міркувань.

### 2. 4. 3. Класифікація методів логічного виведення на онтологіях

Є багато реалізацій процесорів логічного виведення (reasoning engine) для OWL онтології, що різняться можливостями, сферами застосування та якістю виконання завдання. Узагальнений аналіз дає змогу розділити їх на три групи, залежно від методу реалізації:

1. *Табличні DL-процесори.* Традиційно були розроблені першими для розв'язку подібних задач. Мають низьку продуктивність, але здатні робити умовиводи на дуже складних онтологіях з безліччю нетривіальних конструкцій. До цього класу належать різонери Pellet, RacerPro, Fact++, а також Hermit і SHER.

2. *Диз'юнктивні Datalog-процесори.* Трансформують онтологію в диз'юнктивну Datalog-програму й використовують методику дедуктивних баз даних і правило резолюцій. Такі процесори мають задовільну швидкодію при використанні деяких оптимізацій, однак не підтримують певні OWL-конструкції, зокрема кардинальні обмеження й номінали. До цієї групи належить KAON2.

3. *Процесори правил.* Використовують системи обробки правил для умовиводів на онтологіях. Мають високу продуктивність, але можуть обробляти лише прості онтології, позбавлені багатьох важливих конструкцій. Представники цієї групи – Sesame/OWLIM, Jena, OWLjesskb.

Є два підходи до реалізації логічного виведення: на базі правил (з використанням алгоритмів forward-chaining і/або backward-chaining) і на базі семантичного табло (semantic tableau). На базі правил реалізовані Semantics, SDK і Owlim, а на базі семантичного табло – Pellet (рис. 2. 7).

Для подальшого розширення можливостей онтології необхідно використовувати мову правил SWRL. З її допомогою можна створювати свої правила логічних виведень, не передбачених мовою OWL 2 і, отже, ще більше розширювати можливості використання онтології.

У процесі виконання поставленого завдання ключову роль відіграє процесор логічного виведення. Аналіз блоків логічного виведення засвідчив, що лише невелика кількість з них задовольняють пропонованим вимогам до розробки сучасних інтелектуальних застосувань. Так, найбільш оптимальним для розроблювальних проектів є різонер Pellet. Процесори Hermit і Fact++ – не оптимізовані для роботи з екземплярами (великим ABox ), RacerPro є закритим комерційним продуктом, а SHER поки що недостатньо стабільний для промислового використання. KAON2 – не забезпечує підтримку мови OWL 2. 0 та інтерфейсу OWL API і на сьогодні припинив свій розвиток.

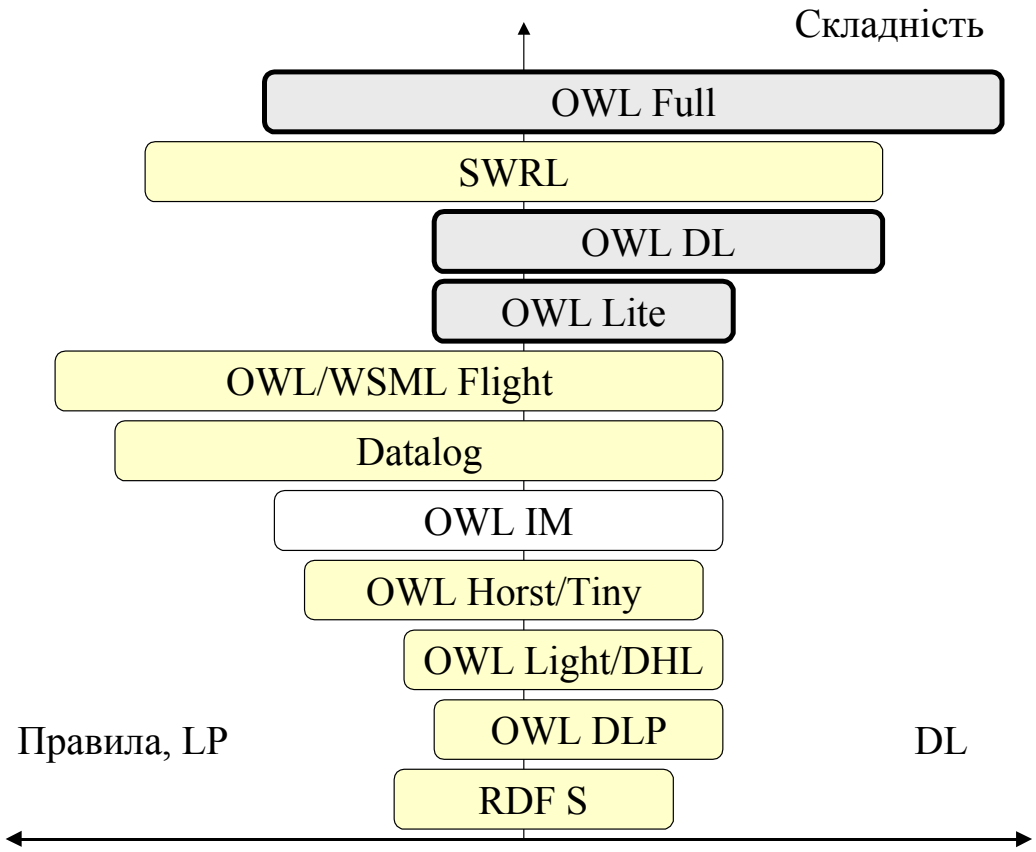



Рис. 2. 7. Розширена діаграма Owlim

До третьої групи входять найбільш перспективні процесори. Швидкість обробки даних у Jena і OWLIM достатня для практичного застосування. Однак процесори цієї групи здатні підтримувати відносно невелику експресивність онтології, обмежену в основному простими конструкціями. Наприклад, немає підтримки логічного аналізу типізованих властивостей, що унеможливлює вираження концепцій, використовуваних вище як приклад.

Різонер є ключовим компонентом у роботі з OWL-онтологіями. Це пов'язано з тим, що не всі знання в онтології є явними, і різонери потрібні для виведення неявних знань так, щоб отримати коректні результати запитів.

Можна виділити три типи атрибутів різонерів онтології:

- характеристики міркувань: ця категорія містить опис основних рис різонерів онтології, наприклад таких, як: методологія, розумність і повнота, виразність, інкрементальна класифікація, підтримка правил, обґрунтування й супровід задач міркувань ABox;
- характеристики зручності практичного використання: ця категорія визначає атрибути або реалізує різонерів в OWL API. Вони також вказують на наявність різонерів та їх ліцензії;

- показники ефективності: показники ефективності використовуються для вимірювання продуктивності різонерів онтології, наприклад: продуктивність класифікації, продуктивність перевірки несуперечності TBox тощо.

*Методологія* вказує на процедуру або алгоритм, які використовуються в різонері для розв'язання основних проблем міркування в описі логіки, наприклад: таблиці, гіпертаблиці [309] тощо.

*Розумність і повнота* – це функція різонера онтології, що оцінює, чи всі можливі висновки можна вивести.

*Виразність* припускає різні види аксіом, такі, як аксіоми транзитивності й аксіоми включення ролей у RBox.

*Інкрементальна класифікація* – коли онтологія була класифікована й після цього оновлюється (додаванням або видаленням), що дає змогу різонеру повторно використовувати попередню інформацію про класифікацію разом з оновленими аксіомами, щоб створити новий концепт ієрархії.

*Підтримка правил* дає змогу різонерові онтології об'єднати онтологію з правилами.

*Платформа* вказує на операційну систему, у якій різонер може працювати (Windows, Linux тощо).

*Обґрунтування* – здатність різонера онтології давати роз'яснення щодо невідповідностей, наявних в онтології.

*ABox міркування* – міркування з індивідами, що передбачає перевірку екземпляра, спільно з відповіддю на запит і перевіркою узгодженості ABox.

*OWL API* – API OWL-інтерфейс [23] прикладного програмування (API) для роботи з OWL-онтологією. Він забезпечує стандартний інтерфейс для OWL різонерів, так що застосування може вставляти різні міркування без зміни його реалізації.

*OWLlink* надає розширення, тобто реалізацію нейтрального протоколу для взаємодії різонерів з OWL 2. Це дає змогу включити OWL API й обізнані різонери в сервери OWLlink й отримати доступ до віддалених серверів OWLlink з OWL API-інтерфейсу (наприклад, Protégé).

*Підтримка Protege* вказує на те, чи може різонер бути використаний з інструментом Protege, чи ні.

*Підтримка NeOn* вказує на те, чи може різонер бути використаний з інструментом NeOn, чи ні.

*Ліцензія* – ліцензія з відкритим вихідним кодом чи ні.

*Підтримка Jena* вказує на те, чи може різонер бути використаний з Jena API, чи ні.

*Мова реалізації* вказує на мову, яка використовується для реалізації різонера.

*Доступність* вказує на доступність різонера.

*Рідний профіль* вказує на логічні фрагменти, які забезпечують виразну силу для ефективності аргументації.

Багато різонерів використовують логіку предикатів першого порядку, щоб виконати обґрунтування; виведення, зазвичай, відбувається через прямий ланцюжок міркувань і побудову зворотного ланцюжка [312]. Прямий ланцюжок міркувань і зворотний ланцюжок – стратегія онтологічних різонерів [320].

*Прямий ланцюжок міркувань*: згідно з цією стратегією різонер запускається з відомих фактів і отримує допустимі виведення. Цілі такого обґрунтування:

- обчислити виведене замикання;
- відповісти на певний запит;
- вивести певний вид знання (наприклад, таксономію класу).

*Зворотний ланцюжок міркувань*: згідно з цією стратегією різонер запускається з певного факту або запиту й має перевірити його або знайти всі можливі рішення.

Серед великої кількості доступних різонерів популярними й такими, що відповідають інструментарію Protégé й NeOn, є: Pellet, RACER, FACT++, Snorocket, HermiT, CEL, ELK, SWRL-IQ і TrOWL.

### 2. 4. 4. Програмні засоби логічного виведення на онтологіях

*Pellet* на основі OWL-DL міркувань з відкритим вихідним кодом Java розроблений групою Mind Swap. Він заснований на табличному алгоритмі й підтримує виразність дескриптивної логіки. Це перший різонер, який підтримується всіма OWL DL Shoin (D) і був розширений до OWL 2 (SROIQ (D)) [420]. Pellet підтримує профілі OWL 2. Це обґрунтовує онтології через Jenaas, ще один інтерфейс API OWL. Pellet також підтримує пояснення помилок. Система Pellet версії 2. 0. реалізує логічне виведення для дескриптивної логіки класу ALCQHI (D), яка розширює атрибутивну мову (AL) такими можливостями, як довільне заперечення, транзитивні відношення, інверсні відношення, ієрархія відношень, кількісні обмеження на відношення й деякі конкретні домени. Як конкретні домени підтримуються рядки й числа. Дескриптивна логіка класу ALCQHI (D) за виразністю є підкласом SHIQ; вона не підтримує типи, які лише перераховуються (nominális).

Pellet являє собою механізм міркувань для OWL-DL, що підтримує логічне виведення для екземплярів класів, визначених користувачем, написаний на Java й наданий за відкритою ліцензією [386].

Коли OWL став рекомендованим кандидатом World Wide Web Consortium, виникла думка, що звичайний блок логічного виведення для OWL-DL на основі табло буде занадто складним для людей, не обізнаних з дескриптивною логікою. Тоді й розпочалася робота над Pellet. Сьогодні Pellet – це повний і могутній механізм міркувань з високою продуктивністю. У цій системі вперше цілком реалізована процедура ухвалення рішень для OWL-DL (враховуючи й екземпляри), є підтримка міркувань для типів даних, визначених користувачем (підмова OWL-DL – синтаксичний варіант дескриптивної логіки Description Logic SHOIN(D) і онтологія OWL-DL відповідає базі даних SHOIN(D)) [420].

W3C Recommendation визначає два типи перевірки документів OWL: перевірку синтаксису OWL і перевірку несуперечності OWL. Виділяють чотири класи перевірки несуперечності – для OWL Lite/DL/Full і для повної узгодженості OWL-Lite. Перевірка узгодженості пов'язана зі специфікою семантики. Процедура розв'язання є повною, якщо вона пов'язана з відповідною семантикою. Мова OWL-Full не є мовою розв'язання, тому до неї не можна застосувати повну перевірку узгодженості.

Водночас і для OWL-DL процедура розв'язання є невідомою. Pellet повністю перевіряє узгодженість з OWL-DL і частково – з OWL-Full. При цьому Pellet покриває найбільшу кількість перевірок узгодженості. Його функції визначаються в такий спосіб: на вхід надходить документ, а на виході з'являється одне з трьох можливих розв'язань, тому узгоджено чи не узгоджено слово – не відомо.

Хоча перевірка узгодженості – важливе завдання, але вона не дає змоги робити з онтологією щось корисне з практичного погляду.

Традиційно в співтоваристві дослідників онтологій і дескриптивних логік саме набір сервісів виведення (inference services) є ключовим для більшості застосувань у галузі інженерії знань.

З огляду на те, що OWL-DL є синтаксичною варіацією високо виразної дескриптивної логіки SHOIN(D), необхідно, щоб на практиці міркування для OWL підтримували як мінімум *стандартний набір сервісів виведення* для дескриптивної логіки, а саме:

- *Перевірку узгодженості* (Consistency checking), що контролює, аби онтологія не містила жодних суперечливих (несумісних) фактів. Документ абстрактного синтаксису й семантики OWL забезпечує формальний опис узгодженості онтології, що й використовується в Pellet. У термінологіях дескриптивних логік (DL) це операція перевірки узгодженості Abox стосовно Tbox.

- *Виконуваність концептів* (Concept satisfiability), що перевіряє, чи може клас мати які-небудь екземпляри. Якщо клас нездійсненний, тоді визначення екземпляра цього класу веде до неузгодженості всієї онтології загалом.
- *Класифікацію*, що знаходить відношення «клас-підклас» між усіма поіменованими класами для створення їх повної ієрархії. Ієрархія класів може використовуватися для відповідей на запити, зокрема такі, як встановлення всіх чи тільки деяких підкласів класу.
- *Виконання* (Realization), що знаходить найбільш специфічні класи, до яких належать екземпляри (individuals). Виконання може бути здійснено тільки після класифікації.

Таблиця 2. 1.

| Скоро-чення | Використо вується для | Значення |
|---|---|---|
| ABox | Assertional Box | Компонент, що містить твердження про екземпляри, тобто факти OWL, такі, як тип, значення параметра, твердження чи рівності/нерівності |
| TBox | Terminological Box | Компонент, що містить аксіоми про класи, тобто аксіоми OWL, такі, як підклас еквівалентний чи клас аксіоми диз'юнктивності |
| KB | Knowledge Base | Комбінація ABox і TBox, тобто повна онтологія OWL |

Використовуючи класифікаційну ієрархію, можна також одержати всі класи для їх екземплярів.

Ці сервіси, за необхідності, стандартно використовуються для перевірки узгодженості.

Такі базові сервіси можуть бути доступні для виконання запитів блоком міркувань. Загалом ці запити підтримуються через API, наприклад, через інтерфейс DIG. У Pellet підтримується стандартний масив вторинних запитів і різні сервіси керування міркуваннями як через власний API, так і для підтримки зовнішніх інструментальних пакетів (Jena, WonderWeb OWL API, DIG.

Pellet також підтримує деякі менш стандартні сервіси. Наприклад, якщо класифікація вимагає ступеня проходження (наприклад, якісь відношення підкласу спираються на онтологію, і класифікація виводиться з цих відношень), але дуже обмежено. Підтримуються тільки деякі типи проходження (entailment), хоча, в принципі, довільне проходження між документами OWL може зводитися до центрального сервісу перевірки узгодженості (consistency).

Pellet має наочну підтримку для перевірки довільного проходження (arbitrary entailments). Подібно до цього можна зводити відповіді на єднальні запити для ABox до перевірки узгодженості. Запити про екземпляри, написані такими мовами, як RDQL чи SPARQL, належать до цієї категорії. Тоді, як дескриптивні логіки, в основному спрямовані на міркування, пов'язані з класами, докладно розглянуті в літературі, запити про екземпляри – набагато менше. Як наслідок, у цій сфері розроблено мало прикладних програм і технологій. У Pellet же пропонується деяка оптимізована процедура відповіді на єднальні запити.

Іншим напрямом цих сервісів є мова, якою вони виконуються. Pellet покриває всі OWL-DL, ураховуючи інверсивні й транзитивні властивості, обмеження потужності, міркування над типами даних для екстенсивної множини як убудованих, так і визначених користувачем простих типів даних схем XML, нумерованих класів і тверджень про екземпляри.

У такий спосіб Pellet є не просто суворо необхідною для повноти OWL перевіркою узгодженості, тобто він не тільки перевіряє синтаксис OWL. Pellet може автоматично застосовувати евристики до документа OWL-Full, що дають змогу визначити, чи можна трансформувати його в документ OWL-DL і потім обробляти в такому нормалізованому вигляді.

І нарешті, тільки мати повний набір сервісів без ясного й простого способу їх використання, – не ефективно. Тому в Pellet розглядаються також і способи використання запропонованих сервісів. Pellet орієнтований на користувачів з різним рівнем знань. Сьогодні є заснований на HTML сервіс, опублікований на сайті Pellet, який призначений для користувачів-початківців, що хочуть перевірити свої онтології та запити. Він дублює функції, доступні через інтерфейс командного рядка Pellet.

Pellet убудований у Swoop – редактор онтологій OWL, що теж можна запускати з Web-браузера через Java WebStart. Підтримується також інтерфейс DIG, а також інші, засновані на Java API.

Ядро Pellet – це блок виведення для DL. Але на відміну від інших, подібних засобів виведення, він з самого початку розроблявся для роботи з OWL. Такий вибір має великий вплив на загальну архітектуру, що позначається на тому, як використовується табличний механізм міркувань, наприклад, як здатність до міркувань над екземплярами класів (міркування ABox) без використання Unique Name Assumption (UNA), і який тип модулів підтримки застосовується, наприклад, блок міркувань для типів даних XML Schema й машини виводу.

Основна мета розробки Pellet полягала в одержанні невеликого ядра машини виводу, яке можна розширювати. Це дає змогу розробляти інтерфейси для різних інструментальних засобів RDF/OWL, таких, як Jena і WonderWeb OWL API, чи для підтримки прикладних програм, що взаємодіють через інтерфейс DIG.

На сьогодні система Pellet реалізує найбільш виразну дескриптивну логіку з використанням високопродуктивного алгоритму (tableau-based algorithm) логічного виведення, який використовується для обробки онтологій, опис яких здійснюється мовою OWL DL.

*RACER* (Renamed ABox es and Concept Expression Reasoner – перейменований ABox es і концепція висловлення міркувань) [305] є першим серед OWL різонерів. Він підтримує методи оптимізації FaCT, а також нові методи оптимізації для розв'язання числових обмежень і ABox es. RACER реалізує TBox і ABox міркування для логіки SHIQ.

*FaCT* (Fast Classification of Terminologies – швидка класифікація термінології) може використовуватися як класифікатор дескриптивної логіки і як засіб перевірки виконуваності модальної логіки. Система FaCT є високо розумною і має повний табличний алгоритм для виразних дескриптивних логік. Оновленою версією FaCT є FaCT ++. Цей різонер використовує той же алгоритм, що й оригінал, але з іншою внутрішньою структурою. Він реалізований на C ++. Проте перша версія FaCT++ тільки підтримувала обґрунтування в SHOIQ, OWL-DL. Однак остання версія FaCT ++ підтримує OWL і базується на дескриптивній логіці SROIQ. FaCT++ виконує засновану на таблицях процедуру розв'язання для загальних TBox es і надає неповну підтримку ABox es [228].

*Snorocket* [327] є високоефективною реалізацією поліноміально-разового алгоритму класифікації для легкої дескриптивної логіки EL+. Він реалізується на Java. Snorocket розроблялася як частина медичної інформатики й клінічної термінології дослідницької програми CSIRO.

*SWRL-IQ* (Semantic Web Rule Language Inference and Query tool – виведення мови правил Semantic Web та інструмент запиту) [436] – плагін для Protege 3. 5, який дає можливість користувачам редагувати, зберігати й подавати запити до основного механізму виведення на основі XSB Prolog. Рис. 2. 8 відображає архітектуру різонера SWRL-IQ. Цей інструмент має ряд особливостей:

- цілеспрямований зворотний ланцюжок різонерів Prolog-стилю;
- відстеження й налагодження результатів виведення;
- збереження запитів;
- експорт результату запиту у формат XML або CSV;
- не залежить від власних або закритих вихідним кодом компонентів;

- використовує XSB Prolog, що знаходиться у вільному доступі відповідно до ліцензії LGPL.

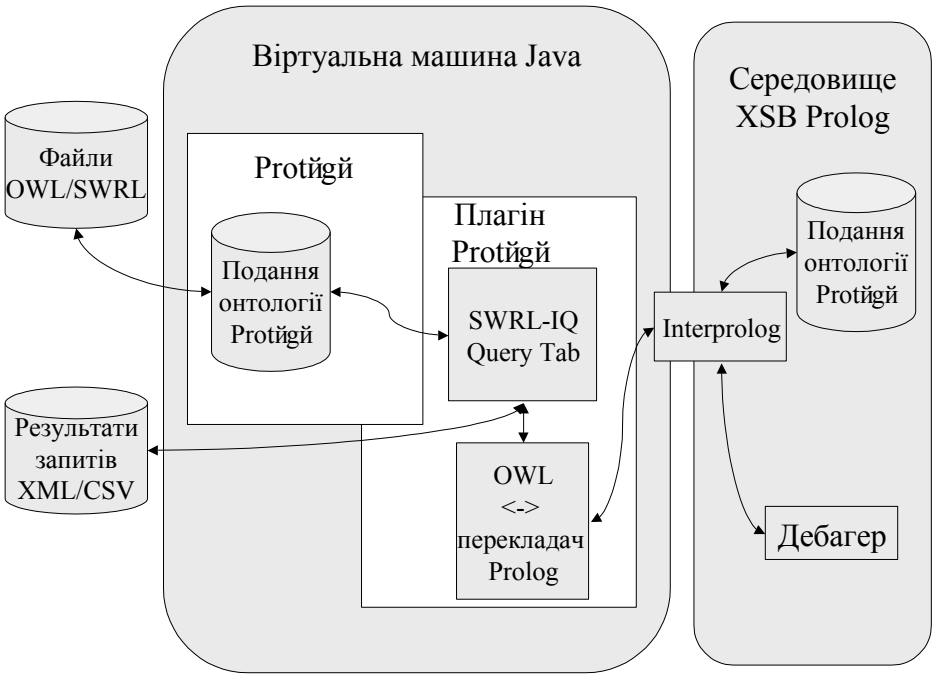



Рис. 2. 8. Системна архітектура різонера SWRL-IQ

*ELK* [249] є вільним різонером з відкритим вихідним кодом для полегшеної мови онтології OWL 2 EL. ELK-різонер написаний мовою Java й може контролюватися за допомогою OWL API. ELK доступний згідно з ліцензією Apache 2. 0. Цей різонер працює на всіх операційних системах, що підтримують Java 1.5 і вище. На рис. 2. 9 показано основні програмні модулі ELK різонера й інформаційного потоку під час класифікації.

*HermiT* – різонер для онтологій, написаних з використанням OWL. Отримавши OWL-файл, HermiT може визначити, чи є онтологія узгодженою, ідентифікувати категоризацію відношень між класами тощо. HermiT – це перший загальнодоступний OWL-різонер на основі нового «гіпертабличного» обчислення, що забезпечує більш ефективне обґрунтування, ніж будь-який раніше відомий алгоритм. Крім того, HermiT містить стратегію «блокування всюди» («anywhere blocking»), що дає змогу обмежити розміри моделей, які він створює [413].

Онтології, для класифікації яких раніше було потрібно багато часу, HermiT може класифікувати за лічені секунди. Крім того, HermiT перший різонер, що може класифікувати ряд онтологій, які раніше вважались занадто складними для будь-якої доступної системи обробки.

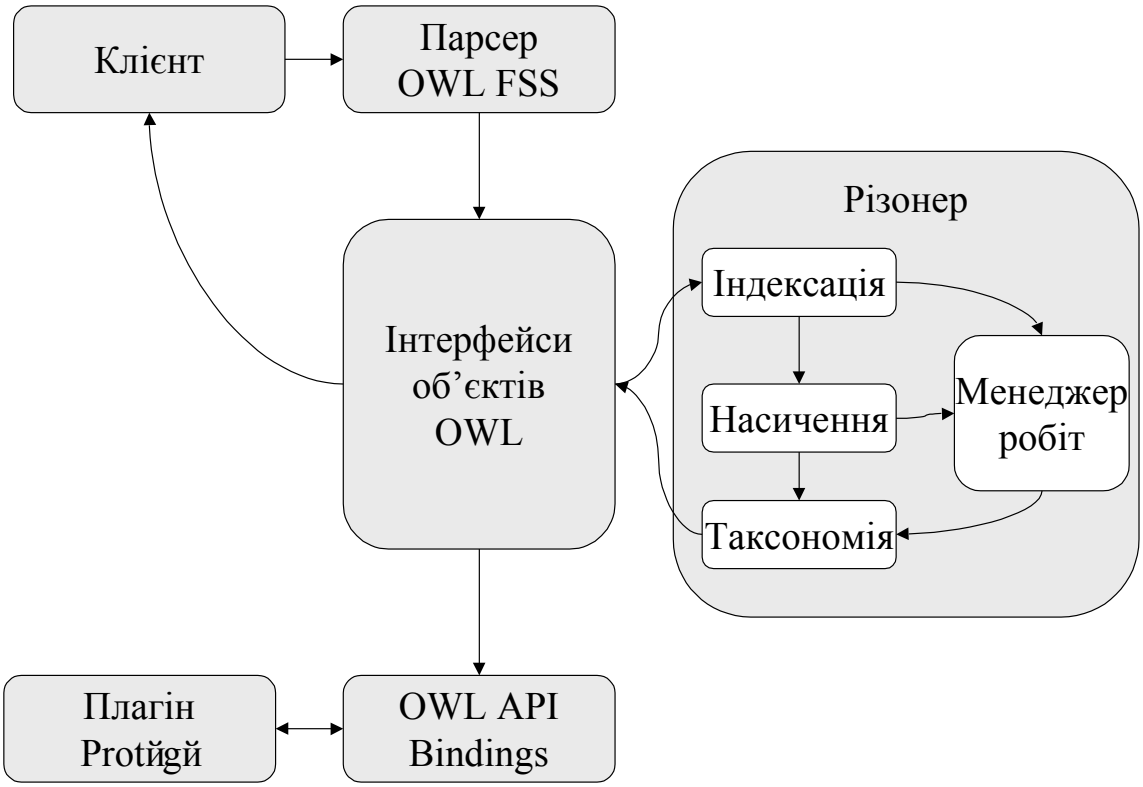



Рис. 2. 9. Основні програмні модулі різонера CEL

HermiT використовує пряму семантику й проходить усі OWL 2-тести на сумісність для прямої семантики міркувань. HermiT може обробляти правила DL-safe, і правила можуть бути безпосередньо додані до вхідної онтології у функціональному або інших стилях, що підтримує OWL.

*CEL* є різонером на основі LISP. Він має дуже просту оболонку як інтерфейс. Це надає користувачам усю важливу функціональність, зокрема й просту інтерактивну команду довідки. CEL заснований на вдосконаленій версії поліноміально-разового алгоритму класифікації. Він, головно, обґрунтовує класифікації, що містять обчислення повної ієрархії категоризації між усіма іменами поняття, що трапляються у вхідній онтології.

*TrOWL* (Tractable OWL 2 infrastructure – довірча інфраструктура OWL 2. 0) [442] є загальним інтерфейсом для ряду різонерів, розроблених в університеті Абердіна. TrOWL Quill надає обґрунтування послуг з OWL 2 QL. TrOWL REL є оптимізованою реалізацією алгоритму CEL, що забезпечує обґрунтування за OWL 2 EL. Для підтримки повного обґрунтування DL, TrOWL робить можливим використання навантажених плагінів різонерів, таких, як FaCT++, Pellet, HermiT і RacerPro.

Зазначимо, що всі, наведені вище різонери, можуть використовуватися в інструментарії Protégé й NeOn. Проте різонери сумісні не з усіма версіями Protégé [389]. У таблиці 2. 1. відображено сумісність версій різонерів з версіями інструментарію Protégé й NeOn.

## 2. 5. Інструментальні засоби для роботи з онтологіями

### 2. 5. 1. Інструментальні засоби побудови онтологій

Серед інструментальних засобів, що використовуються в онтологічному аналізі, можна виділити засоби, орієнтовані на підтримку різних етапів життєвого циклу онтології й на різні рівні обробки онтологічних структур. По-перше, це редактори онтологій, що призначені для створення й модифікації онтологій користувачем. По-друге, це програмні засоби, що призначені для виконання логічних запитів до онтологій, різноманітних операцій над онтологіями (приміром, перевірки їх на несуперечність або їх інтеграції). По-третє, це засоби, орієнтовані на візуалізацію онтологічних знань і подання їх у зручній для користувача формі. У різних програмних засобах такі функції можуть по-різному комбінуватися. На сьогодні розроблено велику кількість інструментів для роботи з RDF і OWL. До них належать редактори, CMS, середовища розробки, репозиторії RDF, генератори RDF, валідатори, сервери SPARQL [427] і механізми логічного виведення.

Системний аналіз онтологічних сервісів, необхідних для ефективного керування знаннями в Semantic Web, і аналіз послуг, які підтримують поширені сьогодні програмні засоби для роботи з онтологіями (редактори, різонери, візуалізатори тощо), свідчать, що всі онтологічні сервіси можна поділити на кілька груп відповідно до того, які функціональні можливості потрібні користувачам:

- сервіси подання й візуалізації онтологій, презентовані різними засобами та мовами, потрібні в усіх системах, які використовують онтологічне подання знань;
- сервіси редагування онтологій, які забезпечують створення нових онтологій, редагування наявних онтологій, операції над їх класами, властивостями класів і екземплярами класів, серед яких і такі, що забезпечують спільну роботу над онтологіями (потрібні в тих системах, які не тільки використовують готові онтології, а й створюють власні онтології або модифікують наявні);
- сервіси логічного виведення над онтологіями, аналогічні тим функціональним можливостям, що надають різонери, зокрема й сервіси перевірки онтології на узгодженість;

- сервіси автоматизованого поповнення онтологій (приміром, шляхом обробки природномовних текстів, індуктивного й традуктивного узагальнення відомостей з баз даних, обробки метаданих) у тому разі, якщо подібні сервіси потрібні у відповідній предметній області й реалізуються програмно;
- сервіси операцій над онтологіями, що забезпечують їх порівняння та інтеграцію, якщо для предметної області створено й програмно реалізовано відповідні алгоритми;
- сервіси підтримки виконання запитів SPARQL.

Вивчення загальних програмних застосувань дало змогу дійти такого висновку: незважаючи на те, що кількість засобів Semantic Web стрімко зростає, багато сфер семантичних задач поки що не охоплено. Наприклад, необхідно автоматизувати порівняння онтологій, щоб визначити однакові для обох онтологій класи й властивості; перетворення подання онтології з реляційної БД на OWL; розподілене збереження онтологічних даних з можливістю виконання запитів і візуалізацією онтологічних даних.

### 2. 5. 2. Редактори онтологій

*Редактор онтологій* – це програмний засіб, який підтримує створення онтологій та їх редагування. Така програма надає можливість роботи з однією чи кількома мовами подання онтологій, можливість імпорту/експорту у різні формати, доступ до бібліотек онтологій, візуалізацію, засоби логічного висновку, мови запитів, зв'язок з іншими онтологіями тощо. Сьогодні розроблено багато програмних засобів, призначених для обробки онтологічних знань, що робить актуальним завдання вибору редактора онтологій, що найбільше відповідає специфіці задачі, для якої розробляється онтологія, і визначення критеріїв, що впливають на цей вибір [121].

Стенфордському університету належить розробка трьох широко відомих редакторів онтологій – *Protégé*, *Chimaera* [225] та *Ontolingua* [372]. Найбільш поширеним серед них є Protégé [390].

Здебільшого редактори підтримують кілька форматів опису онтологій, що зумовлено браком єдиного широко використовуваного формату й залишає право вибору за користувачем. Останнім часом стає обов'язковим для редакторів підтримувати виконання логічного виведення, а саме – перевірки несуперечності побудованої моделі й забезпечення можливість виконання запитів. Позитивним фактором є використання в редакторах онтологій реляційних баз даних, що дає змогу зробити роботу редактора більш стійкою, збільшити швидкість такої роботи, значно підвищити максимальну кількість об'єктів, з якими

здатний працювати редактор, і зменшити час відновлення. Можна сформулювати три критерії вибору редактора онтологій:

1. Якщо визначена базова онтологічна модель, то варто використовувати спеціалізовані критерії.

2. Якщо необхідно забезпечити підтримку й можливість виконання імпорту/ експорту онтологій у різні формати, то як альтернативи варто розглядати універсальні редактори. Для спеціалізованих редакторів характерна підтримка обмеженої кількості форматів, як правило, формату розширюваної мови розмітки (XML) і формату мови онтології для Web (OWL, RDF).

3. Якщо необхідно забезпечити можливість виконання запитів до онтологічної бази знань, то варто використовувати спеціалізований редактор. Оскільки спеціалізовані редактори розробляються на основі базової онтологічної моделі, то з'являється можливість забезпечити ефективний механізм виконання запитів.

Сьогодні при створенні онтологій найчастіше застосовується такий програмний засіб, як Protégé, що є безкоштовним і має багато корисних властивостей. Саме тому вважаємо за доцільне розглянути детальніше можливості, які надає Protégé.

Protégé – це вільно розповсюджуваний, відкритий редактор онтологій, призначений для побудови баз знань. Платформа Protégé підтримує два основні способи моделювання онтологій: за допомогою редакторів Protégé-Frames і Protégé-OWL. Онтології, побудовані в Protégé, можуть бути презентовані в багатьох різних форматах, ураховуючи RDF і OWL [391].

Protégé має відкриту, легко розширювану архітектуру за рахунок підтримки модулів розширення функціональності. Protege підтримується великим науковим співтовариством, до якого входять розроблювачі й учені, урядові й корпоративні користувачі, що використовують його для розв'язання задач, пов'язаних зі знаннями в таких різноманітних галузях, як біомедицина, обробка знань і корпоративне моделювання [202]. Детальніше роботу з цим редактором і його плагінами розглянуто нижче.

Сьогодні редактор онтологій Protégé за кількома розглянутими вище критеріями можна визнати найкращим програмним засобом, призначеним для цих цілей. Крім того, плагіни Protégé виконують багато функцій, характерних для інших типів інструментальних засобів, зокрема таких, як різонери HermiT, Snorocket, візуалізатори, анотатори, конвертори форматів і мов подання, що пов'язані з обробкою онтологій.

## 2. 6. Створення онтологій предметних областей за допомогою Protégé

### 2. 6. 1. Онтологія в Protégé

Онтологія в Protégé – це явна специфікація концептуалізації. Цей інструментальний засіб підтримує спільне використання понять, яка містить засоби подання предметних знань і домовленості про методи міркувань. Protégé дозволяє перетворювати певний опис погляду на світ у конкретній сфері інтересів, який складається з набору термінів і правил їх використання, що обмежує їх значення в рамках конкретної ПрО, у таку форму, що вони стають придатними для машинної обробки [301], приміром, онтології, представлені мовою OWL.

Таким чином, створена за допомогою Protégé онтологія – це система, що складається з наборів понять і тверджень про ці поняття, на основі яких можна будувати класи, об'єкти, відношення, функції та теорії.

Таку онтологію можна розглядати як БЗ спеціального виду з семантичною інформацією певної ПрО. Компоненти, з яких складаються конкретні онтології, залежать від парадигми подання, але практично всі моделі онтологій містять певні *концепти* (поняття, класи), *властивості* концептів (атрибути, ролі), *відношення* між концептами (залежності, функції) та *обмеження застосування,* що визначаються аксіомами. Концептом може бути опис задачі, функції, дії, стратегії, процесу міркування тощо.

В Protégé використовується формальна модель онтології О – це впорядкована трійка O=<T, R, F>, де T – скінчена множина термінів ПрО, яку відображає онтологія О; R – скінчена множина відношень між термінами заданої ПрО; F – скінчена множина функцій інтерпретації, заданих у термінах і/або відношеннях онтології О. Відношення онтології ПрО являють собою тип взаємодії між концептами ПрО. Приклад бінарного відношення – «є частиною».

### 2. 6. 2. Властивості онтології OWL

Онтологія OWL (Web Ontology Language) [381, 382] є послідовністю аксіом і фактів, а також посилань на інші онтології. Вони також містять компонент для запису авторства та іншої подібної інформації. Онтології OWL є документами Web, на них можна посилатися через URI. Формально відобразити OWL-онтологію можна так:

<ontology> ::= Ontology ( [<authorship-etc>] {<directive>} )

<authorship-etc> ::=...

<directive> ::= <imports>

<directive> ::= <axiom>

<directive> ::= <fact>

<imports> ::= imports ( <URI> )

Фундаментальні поняття певної ПрО мають відповідати класам, що містяться в коренях різних таксономічних дерев. Кожен екземпляр в онтології OWL належить до класу owl:Thing, а кожен встановлений користувачем клас автоматично є підкласом owl:Thing. Специфічні для ПрО кореневі класи визначаються простим оголошенням іменованого класу. OWL також задає порожній клас: owl:Nothing. Визначення можуть бути розширюваними й розподіленими [381].

Фундаментальним таксономічним конструктором для класів є rdfs:subClassOf. Він зв'язує окремий клас із загальним. Якщо X – підклас Y, то кожен представник X – також представник Y. Відношення rdfs:subClassOf є транзитивним. Якщо X – підклас Y і Y – підклас Z, тоді X – підклас Z.

Визначення класу складається з двох частин: назви (або посилання на неї) й списку обмежень. Кожний вираз, який безпосередньо міститься у визначенні класу, уточнює властивості представників цього класу. Представники класу належать до перетину зазначених обмежень. Для визначення екземпляра досить оголосити його членом якогось класу.

Властивості дають змогу утверджувати загальні факти про членів класів і про екземпляри. Вони становлять бінарні відношення. Розрізняють два типи властивостей:

- властивості-значення відношення між представниками класів і RDF-літералами або типами даних, зумовлених XML Schema;
- властивості-об'єкти відношення між представниками класів.

При наданні властивості є багато способів обмежити це відношення. Можна задати домен і діапазон. Властивість може бути визначена як спеціалізація (підвластивість) наявної властивості. Можливі й більш складні обмеження. Властивості, так само як класи, можуть бути організовані ієрархічно.

OWL використовує більшість убудованих типів XML Schema. Посилання на ці типи здійснюються за допомогою URI для http://www.w3.org/ 2001/ XMLSchema.

Формальна семантика OWL містить опис того, як отримати логічні наслідки, маючи таку онтологію, тобто здобути факти, що не подаються в ній безпосередньо, проте зумовлені її семантикою. Ці наслідки можуть спиратися на один документ або множину розподілених документів, які комбінуються з використанням спеціальних механізмів OWL.

Онтологія OWL відрізняється від схеми XML тим, що вона є поданням знань, а не форматом повідомлень. Однією з її переваг є

наявність інструментального ПЗ, призначеного для аналізу знань, поданих мовою OWL. Ці інструменти забезпечують загальну домено-незалежну підтримку онтологічного аналізу.

Розглянемо приклад створення застосовної онтології, яка містить такі класи та підкласи:

- "Людина";
- "Студент";
- "Викладач";
- "Методист";
- "Предмет, що вивчається".

У процесі створення такої онтології потрібно послідовно виконати такі кроки:

- Задати атрибути цих класів, приміром, для класу «Людина» задати атрибути "Ім'я", "Прізвище", "Дата народження".
- Задати атрибути класів, що самі є класами. Наприклад, для класу "Студент" задати атрибут «вивчає», що належить до класу "Предмет, що вивчається".
- Створити екземпляри класів.
- Зберегти створену онтологію.
- Згенерувати за онтологією html-документи, що містять опис класу "Студент" і екземпляри цього класу.
- Проглянути створені документи за допомогою браузера.

### 2. 6. 3. Основні елементи онтології в Protégé

Protégé – локальна, вільно поширювана Java-програма, призначена для побудови (створення, редагування й перегляду) онтологій ПрО. Protégé містить редактор онтологій, що дає змогу проектувати онтології, розгортаючи ієрархічну структуру абстрактних і конкретних класів і слотів. На основі сформованої онтології Protégé дає можливість генерувати форми здобуття знань для введення екземплярів класів і підкласів [171, 202]. Наголосимо, що останні версії (починаючи з 4. 0) значно відрізняються від Protégé 2000 і попередніх версій як форматом подання онтологій, так і можливостями для їх обробки [444].

Основні елементи онтології, створюваної в Protégé, – це [171]:

- *класи* (Classes) множини, елементами яких є екземпляри;
- *екземпляри* класів (Individuals), що являють собою конкретні об'єкти предметної області, яка цікавить користувача;
- *властивості* (Properties) бінарні відношення між екземплярами;
- *домени,* що визначають множини екземплярів, до яких застосовується властивість (область визначення властивості);
- *діапазон* (range) множина екземплярів, що можуть бути зв'язані цією властивістю з об'єктами з домену.

Властивості можуть бути зворотними (inverse) одна щодо одної, транзитивними та симетричними. Область значень властивості може бути обмежена єдиним об'єктом, тоді властивість називається функціональною.

Властивості відповідають слотам у Protege-фреймі. Вони також називаються ролями в описі логіки й відношень в UML та інших об'єктно-орієнтованих поняттях.

Процес створення нової онтології в Protégé полягає в створенні класів, їх властивостей та екземплярів цих класів [366].

Під час запуску Protégé 4. 3 відкривається нова онтологія. Її можна зберегти з обраним ім'ям у форматі RDF/XML, вибравши в меню «File» опцію «Save as».

### 2.6.4. Створення класів в Protégé

Створення класів здійснюється в закладці «Classes». Порожня онтологія містить один клас з ім'ям THING, а всі створювані надалі класи є підкласами цього класу (рис. 2. 10).

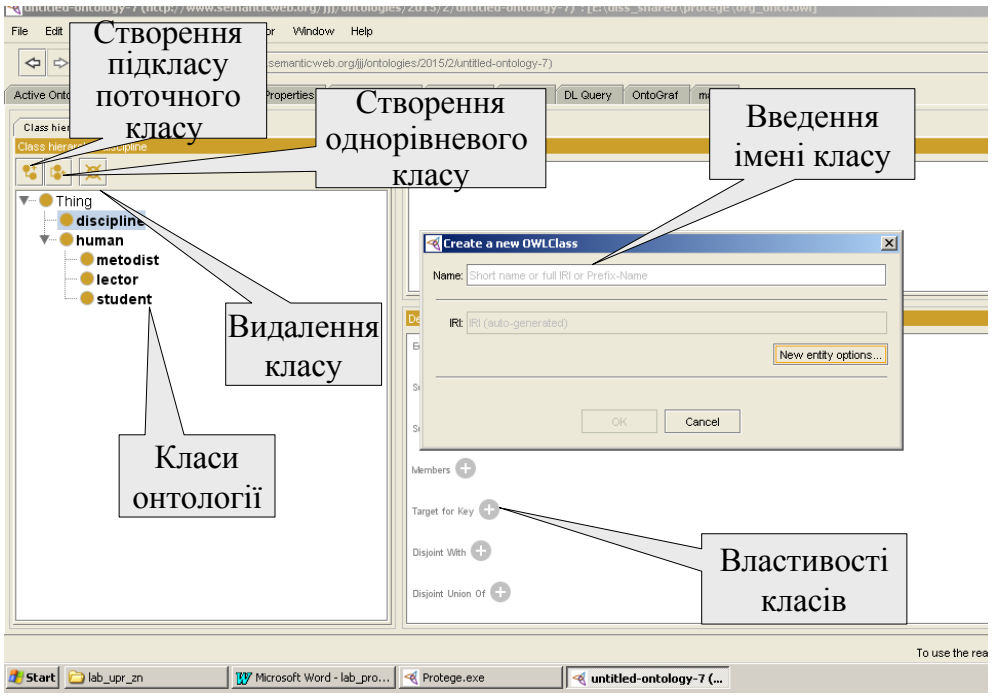



Рис. 2. 10. Створення класів у Protégé

Після того, як класи створені, можна фактично здійснити опис деяких їхніх властивостей. Наприклад, у створеній раніше таксономії класів можна вказати, що підкласи «студент» і «викладач» не перетинаються. Для цього серед властивостей класів потрібно вибрати знак «+» біля властивості «Disjoint with», а потім у вікні, що

з'явилося, з ієрархії класів вибрати мишею (при натиснутій клавіші «Ctrl») ті класи, що не перетинаються.

Крім того, у властивостях класу можна явно вказати, що класи еквівалентні («Equivalent To»). Це дає змогу використовувати різні синоніми на позначення груп об'єктів предметної області.

Вкладка «Annotation» дає змогу вводити й зберігати структуровані пояснення про клас природною мовою. Властивості анотації можуть використовуватися для додавання інформації (метадані – дані про дані) для класів, окремих індивідів і властивостей об'єктів/типів даних. Крім стандартних видів анотації, користувач може створювати нові підкласи й потім здійснювати опис їх значень.

### 2.6.5. Властивості об'єктів в Protégé

**Створення властивостей об'єктів.**

Щоб здійснити опис екземплярів класів, у Protege 4. 3 використовується два основні типи властивостей: властивості об'єктів (ObjectProperties) і властивості даних (DataProperties).

Властивості об'єктів відображають відношення між екземплярами класів.

Наприклад, властивість об'єктів «вивчає» пов'язує «Іванов» (екземпляр класу «студент») і «Матаналіз» (екземпляр класу «дисципліна»).

Властивості даних дають змогу зв'язати екземпляри класів з константами різних типів. Наприклад, властивість даних «рік_народження» зв'язує «Іванов» (екземпляр класу «студент») і константу типу Integer «1994».

Властивості об'єкта й властивості даних можуть бути позначені як властивості Анотації.

Для того, щоб створити властивості об'єктів, потрібно скористатися вкладкою «ObjectProperties». У Protégé 4. 3 за замовчуванням, автоматично створюються властивості об'єктів верхнього рівня (TopObjectProperty) за аналогією до класу THING (рис. 2. 11).

У властивостей об'єктів можуть бути підвластивості. Наприклад, властивість «Використовувати» має підвластивості «Використовувати на практиці» й «Використовувати теоретично». Зазначимо: хоча аналогічно можна створювати підвластивості властивості в DataType, однак це не означає можливість змішувати й зіставляти властивості об'єктів і властивості типів даних з їх підвластивостями. Наприклад, неможливо створити властивість об'єкта, що є властивістю підвластивості DataType й навпаки. Це безпосередньо виходить з того,

що властивості об’єктів зв’язуються з двома екземплярами класів, а властивості даних – з одним.

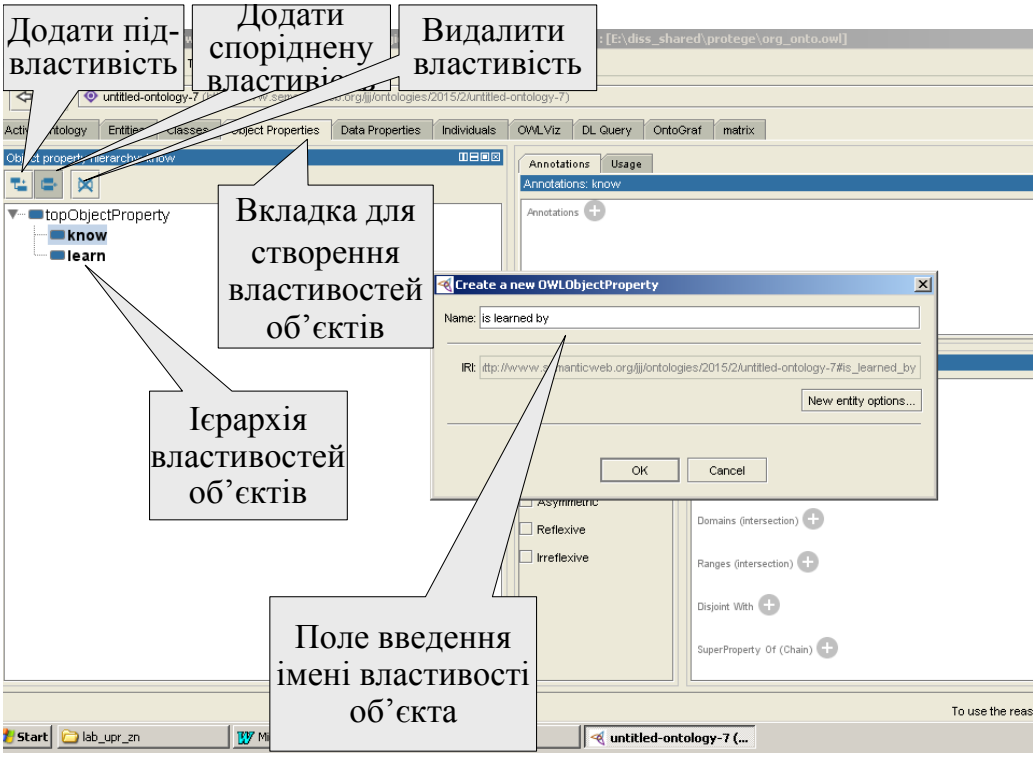



Рис. 2. 11. Створення властивостей об’єктів у Protégé

**Характеристики влествостей об’єктів**

Створюючи нову властивість об’єкта, можна явно схарактеризувати такі особливості цієї властивості, як функціональність та інверсна функціональність, транзитивність чи симетричність/асиметричність, рефлексивність чи ірефлексивність.

Кожна властивість об’єкта *x* може мати відповідну *зворотну властивість y*, таку, що, якщо властивість *x* зв’язує екземпляр *A* з екземпляром *У*, то *y* зв’язує екземпляр *B* з екземпляром *A*.

Наприклад, для властивості «learn» (вивчає) зворотною є властивість «is_learned» (вивчатися). Це означає, наприклад, якщо «learn» зв’язує «Іванов» (екземпляр класу «студент») з «Матаналіз» (екземпляр класу «дисципліна»), то зворотна йому властивість «is_learned» зв’язує «Матаналіз» (екземпляр класу «дисципліна») з «Іванов».

Щоб вказати при створенні онтології, що одна властивість є для іншої зворотною, необхідно, вибравши потрібну властивість об’єкта, у вікні «Description» обрати кнопку «+» біля напису «Inverse Of», а потім у вікні ієрархії властивостей об’єктів відзначити інверсну для нього властивість (рис. 2. 12).

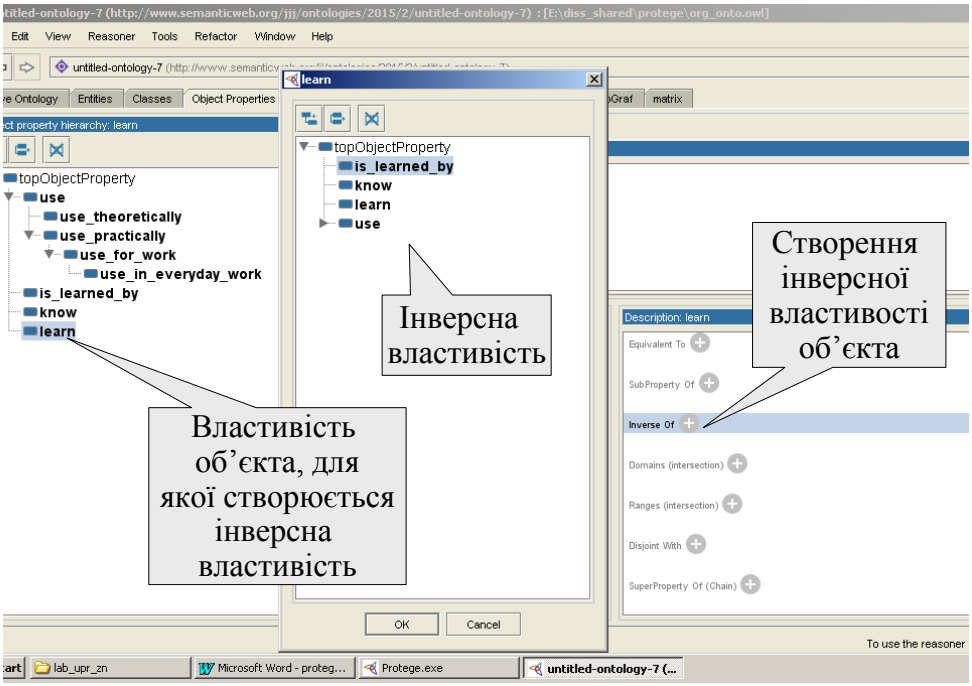



Рис. 2. 12. Створення інверсної характеристики властивостей об'єктів у Protégé

Дуже ефективним інструментом є використання функціональної характеристики властивості об'єкта: якщо властивість є функціональною, то кожен екземпляр класу може бути зв'язаний цією властивістю не більше, ніж з одним екземпляром класу. Наприклад, у людини може бути лише одна мати чи один безпосередній начальник; у конкретний момент часу людина проживає в якійсь одній країні тощо. Якщо ж один екземпляр класу зв'язаний цією властивістю з декількома екземплярами, щодо яких явно зазначено, що вони не ідентичні, то це є протиріччям.

Для того, щоб позначити властивість об'єкта як функціональну, необхідно вибрати цю властивість в ієрархії властивостей, а у вікні «Characteristics» поставити позначку коло характеристики «Functional».

Якщо властивість є зворотною щодо функціональної властивості, то це означає, що властивість є зворотно функціональною. Це значить, що для конкретного екземпляра може бути багато екземплярів, зв'язаних з ним цією властивістю, але кожний з них може зв'язуватися тільки один раз. Для того, щоб позначити властивість об'єкта як зворотно-функціональну, необхідно вибрати цю властивість в ієрархії властивостей, а у вікні «Characteristics» поставити позначку коло характеристики «Inverse functional» (рис. 2. 13).

У вікні «Description» можна виділити область визначення й область значення властивості об'єкта: «Domain (intersection)» і «Range (intersection)» дають змогу вказати, до яких класів можуть належати екземпляри, що зв'язуються відношенням. Наприклад, властивість об'єкта «проживає» зв'язує екземпляр класу «людина» з екземпляром класу «країна».

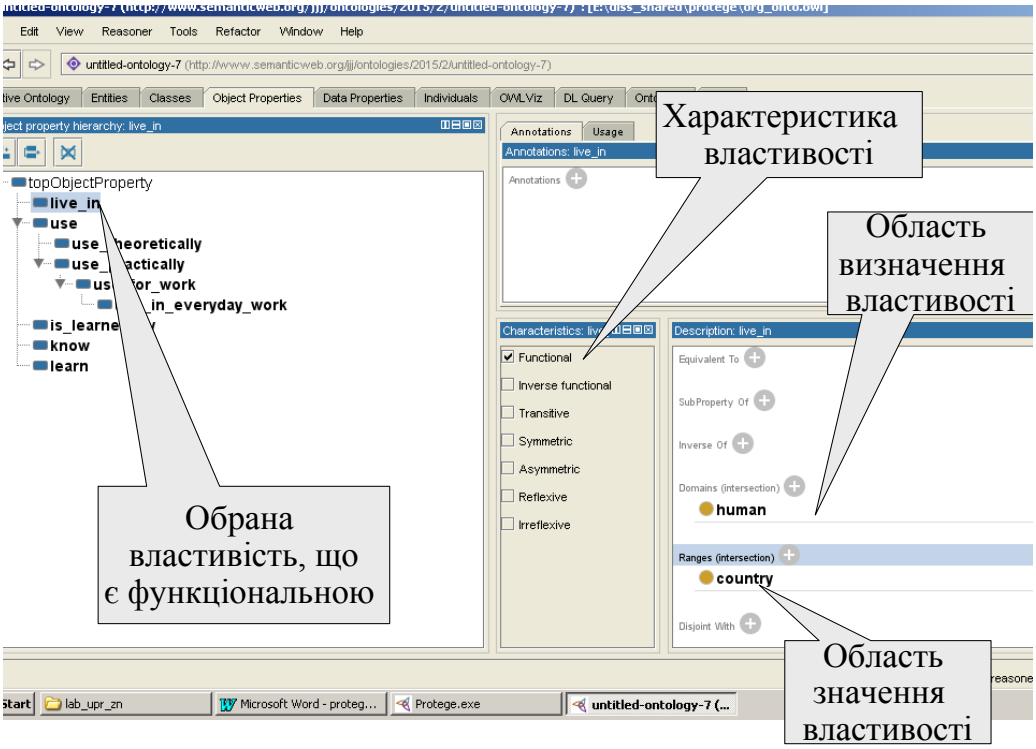



Рис. 2. 13. Створення функціональної характеристики властивостей об’єктів

Транзитивні властивості (transitive) об’єктів: якщо властивість x, що зв’язує екземпляр A з екземпляром B, а екземпляр B – з екземпляром C, є транзитивною, тоді ця ж властивість x зв’язує екземпляр A з екземпляром C. Приклад транзитивної властивості – «more_qualified» (більш кваліфікований), якою можуть бути зв’язані екземпляри класу людина: якщо Петров кваліфікованіший за Іванова, а Сидоров – кваліфікованіший від Петрова, то Петров кваліфікованіший за Іванова.

Для того, щоб позначити властивість об’єкта як функціональну, необхідно вибрати цю властивість в ієрархії властивостей, а у вікні «Characteristics» поставити позначку коло характеристики «Transitive» (рис. 2. 14).

Симетричні властивості (Simmetric) об’єктів: якщо властивість x, що зв’язує екземпляр A з екземпляром B, є симетричною, тоді ця ж властивість x зв’язує екземпляр B з екземпляром A. Приклад симетричної властивості – «more_qualified», якою можуть бути зв’язані екземпляри класу людина: якщо Петров кваліфікованіший за Іванова, то Іванов не може бути кваліфікованіший за Петрова.

Для того, щоб позначити властивість об’єкта як симетричну, необхідно вибрати цю властивість в ієрархії властивостей, а у вікні «Characteristics» поставити позначку коло характеристики «Symmetric».

Асиметричні властивості (Asimmetric) об’єктів: якщо властивість x, що зв’язує екземпляр A з екземпляром B, є асиметричною, то ця ж властивість x не зв’язує екземпляр B з екземпляром A. Приклад

асиметричної властивості – «familiar» (знайомий), яким можуть бути зв'язані екземпляри класу людина: якщо Петров знайомий з Івановим, то Іванов знайомий з Петровим.

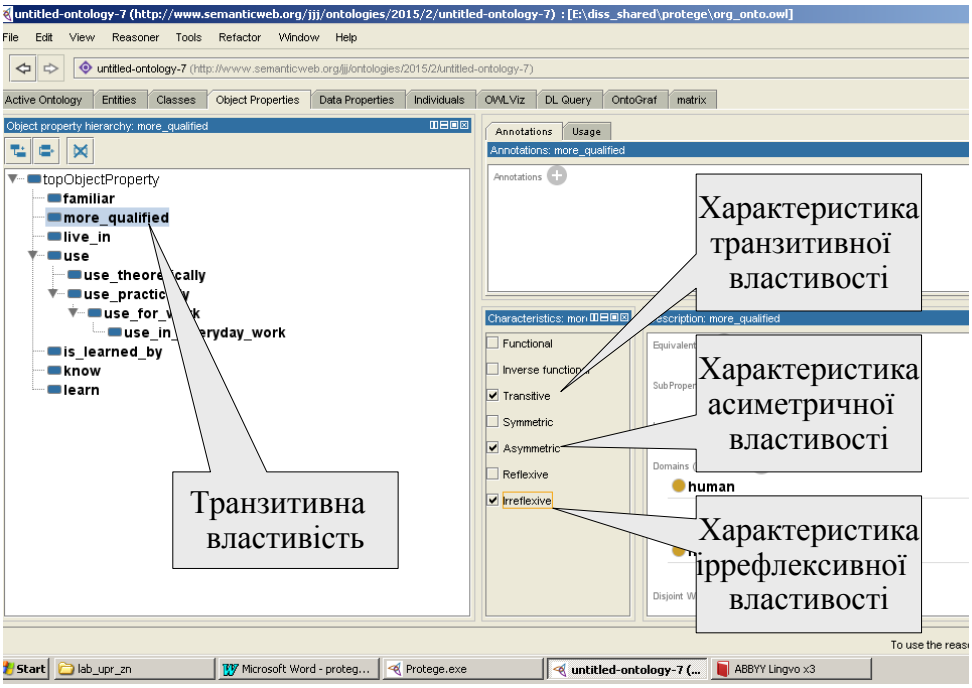



Рис. 2. 14. Створення асиметричної та іррефлексивної характеристик властивостей об'єктів у Protégé

Для того, щоб позначити властивість об'єкта як симетричну, необхідно вибрати цю властивість в ієрархії властивостей, а у вікні «Characteristics» поставити позначку коло характеристики «Asymmetric».

Рефлексивні властивості (Reflexive) об'єктів: властивість x є рефлексивною, якщо вона зв'язує кожен екземпляр A з ним самим. Приклад рефлексивної властивості – «familiar» (знайомий), якою можуть бути зв'язані екземпляри класу людина: кожна людина знайома сама з собою.

Для того, щоб позначити властивість об'єкта як рефлексивну, необхідно вибрати цю властивість в ієрархії властивостей, а у вікні «Characteristics» поставити позначку коло характеристики «Reflexive».

Іррефлексивні властивості (irreflexive) об'єктів: властивість x є іррефлексивною, якщо вона не може зв'язувати жоден екземпляр A з ним самим. Приклад іррефлексивної властивості – «more_qualified», якою можуть бути зв'язані екземпляри класу людина: жодна людина не може бути кваліфікованішою за себе саму.

Для того, щоб позначити властивість об'єкта як іррефлексивну, необхідно вибрати цю властивість в ієрархії властивостей, а у вікні «Characteristics» поставити позначку коло характеристики «Irreflexive».

### 2.6.6. Властивості даних в Protégé

**Створення властивостей даних (Data Properties).**

За допомогою вкладки Data Properties можна відображати відношення, що зв'язують екземпляри класів онтології не з іншими екземплярами класів, а з конкретними значеннями різних типів. Наприклад, для екземпляра класу «людина» це можуть бути такі властивості, як прізвище, ім'я, рік народження тощо. На відміну від Protege-фрейму, у Protege версії 4. 1 і вище екземпляри мають при створенні тільки ім'я, а згодом, при здійсненні опису в них з'являються властивості.

Усі властивості даних є підкласами класу TopDataProperty. Створювати його підкласи можна у вікні «Data property hierarchy».

Наприклад, можна створити властивість даних з ім'ям «name»: у вікні «Description» указати його область визначення (кнопка «+» біля напису «domain (intersection)») – клас «human» і область значення (кнопка «+» біля напису «ranges») – тип String (рис. 2. 15).

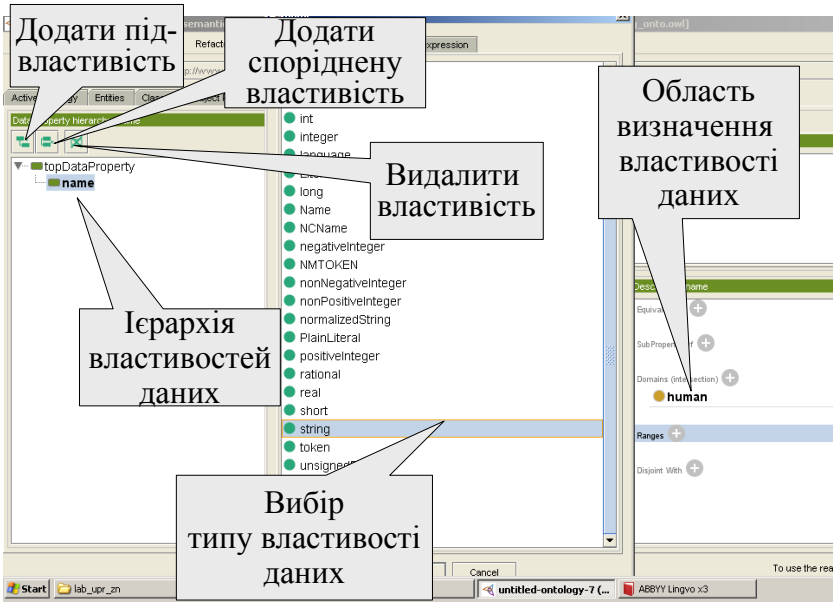



Рис. 2. 15. Створення властивостей даних у Protégé

Крім використання визначеного набору типів даних, можна також спеціалізувати використання властивостей даних шляхом вказівки на обмеження для можливих значень. Наприклад, це застосовують для вказівки на діапазон чисельних значень.

Властивість даних називається функціональною, якщо для кожного екземпляра вона може мати тільки одне значення. Наприклад, людина може мати тільки один рік народження.

Для того, щоб позначити властивість даних як функціональну, необхідно вибрати цю властивість в ієрархії властивостей даних, а у вікні «Characteristics» поставити позначку коло характеристики «Functional».

### 2.6.7. Створення екземплярів класів

Для створення екземплярів класів у Protégé 4. 3 використовується вкладка «Individuals».

Для цього потрібно у вікні «Class hierarchy» вибрати клас, до якого належить створюваний екземпляр, а потім у вікні «Members list» створити новий екземпляр, указавши його ім’я (рис. 2. 16).

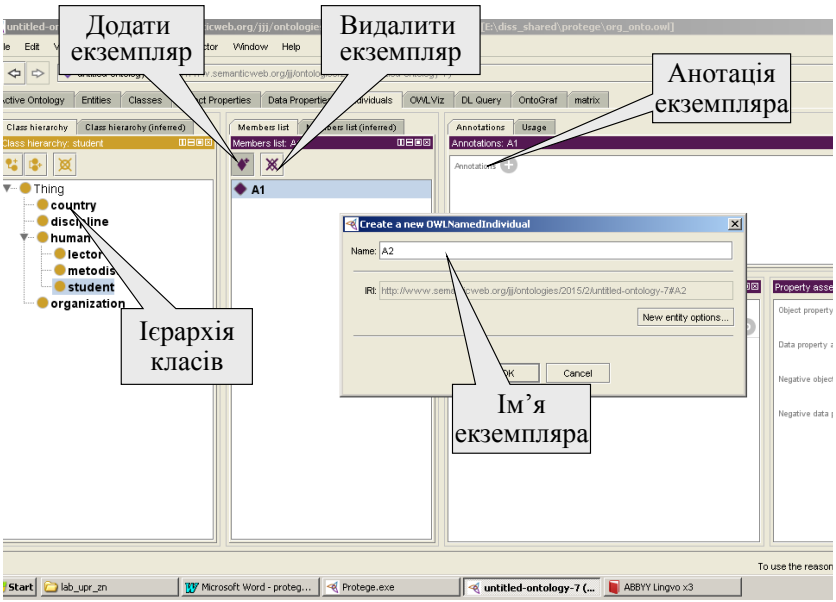



Рис. 2. 16. Створення екземплярів класів у Protégé

Після цього можна здійснити опис значення властивостей екземпляра, використовуючи зв’язані з його класом властивості об’єктів і властивості даних (рис. 2. 17).

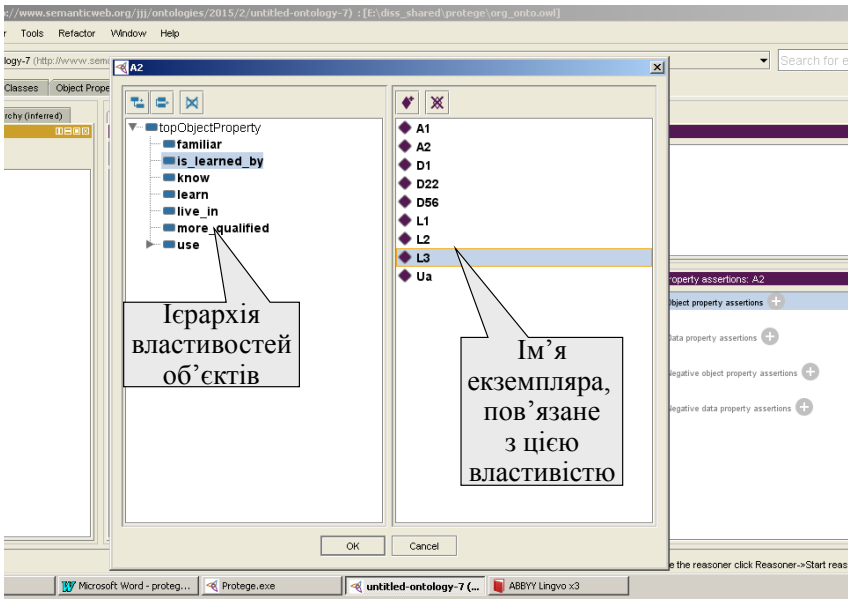



Рис. 2. 17. Створення властивостей об’єктів для екземплярів класів у Protégé

У вікні «Description» можна, вибравши знак «+» коло напису «Same individuals as», указати, що два екземпляри одного класу є ідентичними. Це дає змогу використовувати синоніми для йменування екземплярів.

Можна також явно вказати, що два екземпляри є різними. Для цього у вікні «Description» необхідно, вибравши знак «+» коло напису «Different individuals», указати, що два екземпляри одного класу не є ідентичними. Це дає можливість відстежувати протиріччя при створенні екземплярів.

### 2.6.8. Візуалізація онтологій в Protégé

За допомогою вкладок «OntoViz» і «OntoGraph» можна візуалізувати ієрархію класів створеної онтології (рис. 2. 18).

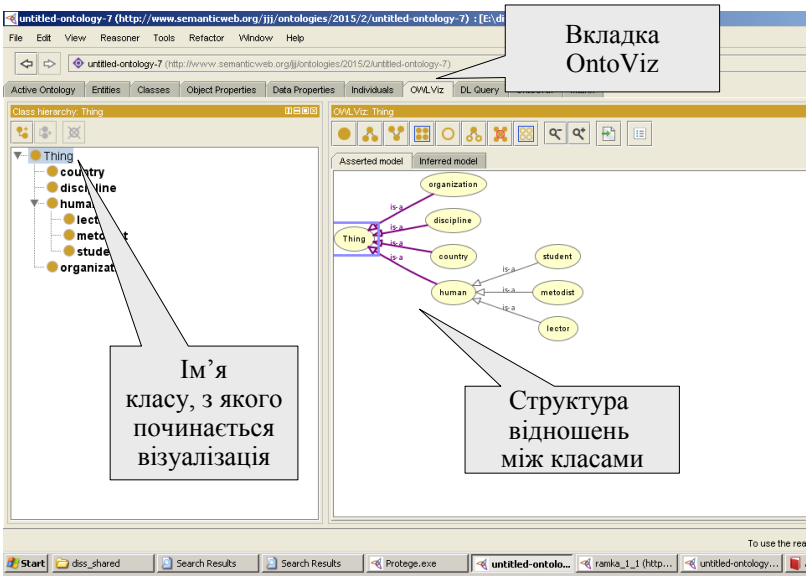



Рис. 2. 18. Візуалізація ієрархії класів у Protégé за допомогою OntoViz

Щоб побудувати візуалізацію онтології за допомогою вкладки «OntoGraph», потрібно в ієрархії класів онтології явно відзначити ті класи, зв'язки між якими потрібно візуалізувати. У цьому режимі візуалізуються не тільки ієрархічні відношення між класами, а й зв'язки, виражені через властивості об'єктів (рис. 2. 19).

Для створення онтологій у редакторі Protégé потрібно явно надати відомості, які стосуються основних понять ПрО, що моделюється, й відношень між ними. Використання Protégé дає змогу створювати онтології мовою OWL, що забезпечує інтероперабельне подання знань і можливість їх повторного використання для інтелектуального аналізу інформації.

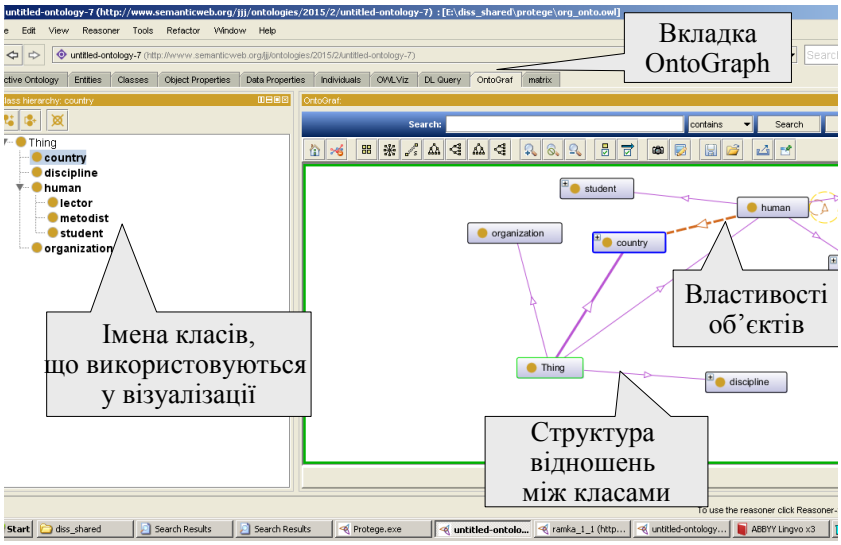



Рис. 2. 19. Візуалізація класів у Protégé за допомогою OntoGraph

## 2. 7. Fluent Editor – редактор онтологій на основі природномовних описів

### 2. 7. 1. Ontorion – система керування знанням на основі природномовного інтерфейсу

Компанія Cognitum [200], що постачає на ринок високотехнологічні IT-сервіси у сфері розробки програмного забезпечення, хмарних обчислень і рішень для Big Data, а також інструментальні програмні продукти для керування знаннями й семантичні технології, розробила розподілену систему керування знанням на основі інтерфейсу з природною мовою і машиною логічного виведення – *семантичний фреймворк Ontorion* [411]. Він сумісний з OWL 2. 0 і SWRL і являє собою хмарне, масштабоване розв'язання, призначене для зберігання великих онтологій (репозиторій) і керування ними.

Ontorion – сімейство продуктів – компонентів сервера й компонентів клієнтів для робочого стола й Web, які уможливлюють широку інтеграцію замовного програмного забезпечення та наявної корпоративної інфраструктури. Основні можливості фреймворку:

- повна підтримка OWL 2. 0 і SWRL;
- підтримка логіки OWL-DL і OWL-EL;
- сумісність з OWL API;
- підтримка контрольованої природної мови;
- убудований обчислювач логічних виразів (reasoner);

- розміщення в хмарі (Windows Azure / Cassandra; можливе розміщення на платформі Linux);
- підтримка спільної роботи над онтологіями (репозиторій) за алгоритмом систем контролю версій (update/commit);
- налаштування прав доступу, аудит безпеки;
- масштабованість, висока продуктивність і безпека;
- сумісний з Linked Data;
- сумісний з Solr/Lucene;
- убудований механізм відображення онтологій (ontologies mapping).

Ontorion найкраще відтворює свої сильні бізнес-цінності як центр перетину таких технологій, як Big Data, Linked data й керування знанням.

### 2. 7. 2. Функції редактору онтологій Fluent Editor

Найважливішою частиною фреймворку Ontorion компанії Cognitum є редактор онтологій Fluent Editor, що дає змогу створювати OWL-онтології користувачам, які не мають практичного уявлення про синтаксис OWL (хоча мати уявлення про його концепції та про моделювання інформаційних структур, звичайно, все ж потрібно).

Онтології створюються на основі природної мови – Controlled English (у системі використовується контрольована англійська мова), тобто звичайною англійською, до якої застосовуються певні правила й обмеження. Остання версія редактора дає змогу будувати онтології польською мовою, а в перспективі планується використання різних європейських мов, зокрема й української.

За допомогою такого редактора набагато простіше працювати фахівцям ПрО, які нічого не знають про онтології. Навчити таких користувачів і фахівців працювати в онтологічних редакторах Protege або TopBraid Composer [199], які повністю відповідають стандартам W3C, було б складніше.

Основними функціональними можливостяи Fluent Editor (http://www.business-semantic.ru/products/ontorion) є:

- створення онтологій шляхом запису виразів природною мовою;
- імпорт/експорт онтологій у формат OWL;
- підтримка звернень до зовнішніх онтологій;
- підтримка модальних виразів (обмеження, що накладаються на елементи моделі);
- убудований обчислювач логічних виразів Reasoner Hermit;
- можливість роботи в складі семантичного фреймворку Ontorion.

Контрольована англійська є підмножиною стандартної англійської мови з обмеженою граматикою й лексикою з метою зменшення неоднозначності й складності, властивих повній англійській.

Розглянемо основні можливості синтаксису цієї «нормалізованої» англійської мови, спостерігаючи при цьому за тим, у якому вигляді її основні вирази будуть зберігатися в OWL.

**Every** customer **is an** organization.

**Every** supplier **is an** organization.

Ключові слова редактор Fluent Editor забарвлює у синій колір (надалі в тексті вони виділятимуться грубим (жирним) шрифтом). У процесі введення тексту редактор надає підказки й попереджає про граматично некоректні вирази. Варто звернути увагу на те, що не потрібно явно повідомляти той факт, що організація, постачальник і покупець є класами: редактор сам здогадається про це з контексту. Визначивши класи, можна оголосити індивідуальні сутності:

Alpha **is a** customer.

Alpha **is a** supplier.

Beta **is a** customer.

Редактор сам визначає, що Альфа й Бета – індивідуальні сутності, які належать до класів «постачальники» й «покупці».

Якщо тепер зберегти онтологію у файлі OWL, то отримаємо таке:

```
<SubClassOf>
  <Class IRI="Customer" />
  <Class IRI="Organization" />
</SubClassOf>
<SubClassOf>
  <Class IRI="Supplier" />
  <Class IRI="Organization" />
</SubClassOf>

<ClassAssertion>
  <Class IRI="Customer" />
  <NamedIndividual IRI="Alpha" />
</ClassAssertion>
<ClassAssertion>
  <Class IRI="Supplier" />
  <NamedIndividual IRI="Alpha" />
</ClassAssertion>
<ClassAssertion>
  <Class IRI="Customer" />
  <NamedIndividual IRI="Beta" />
</ClassAssertion>
```

Постачальники постачають компанії матеріали й запчастини, а покупці закуповують продукцію. Визначимо відповідні класи:

**Every** replaceable-part **is a** product.

**Every** supplier supplies **at-least one** product.

**Every** customer buys **at-least one** product.

Це дає змогу визначити ролі «постачальника» й «покупця» залежно від того, що вони роблять з товарами й запчастинами. В OWL другий рядок з наведеного коду буде відображено так:

```
<SubClassOf>
  <Class IRI="Supplier" />
  <ObjectMinCardinality cardinality="1">
    <ObjectProperty IRI="supplies" />
    <Class IRI="Product" />
  </ObjectMinCardinality>
</SubClassOf>
```

Визначимо тепер конкретний продукт і запчастини, а також те, як будуються матеріальні відносини з контрагентами:

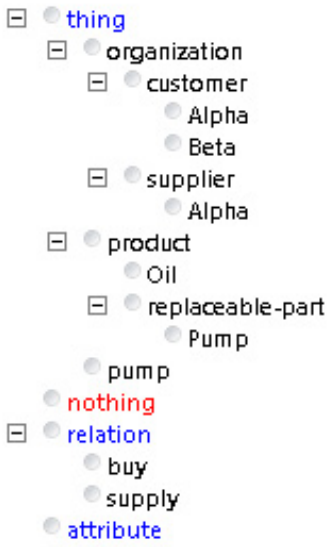
Oil **is a** product.

Beta buys oil.

Pump **is a** replaceable-part.

Alpha supplies pump.

Для того, щоб не заплутатися в написаному, Fluent Editor показує нам дерево таксономії:




Варто звернути увагу на те, що редактор сам утворює різні форми дієслова. У тексті написано «supplies», а в дереві таксономії відповідний термін називається «supply». З неправильними дієсловами редактор також здатний успішно впоратися. Синтаксис Controlled English дає можливість визначати різні обмеження й взаємозв'язки між класами:

Every-single-thing that is a replaceable-part is supplied by at-least one supplier.

**Something is a** supplier **if-and-only-if-it** supplies **nothing-but** replaceable-parts **and is an** organization.

Something is a customer if-and-only-if-it buys nothing-but products and is an organization.

**(Кожна-окрема-річ, яка є** замінюваною частиною постачається принаймні одним постачальником.

**Дехто є** постачальником, **якщо і тільки якщо він** постачає замінювані частини і **є** організацією.

**Дехто є** замовником (покупцем), **якщо і тільки якщо він** купує продукти і **є** організацією).

Далі варто додати атрибути. В організацій є назви, координати й багато інших властивостей і характеристик. Визначимо два атрибути й вкажемо їх значення для організації «Альфа». Значення, звичайно, будуть літералами.

Every organization has-name nothing-but (some-string-value).

Every organization has-tax-number nothing-but (some-integer-value).

Alpha has-name **equal-to** “Alpha, IRTC”.

Alpha has-tax-number **equal-to** +38-044-5266344.

В OWL ці фрази трансформуються приблизно так:

```
<SubClassOf>
  <Class IRI="Organization" />
  <DataAllValuesFrom>
    <DataProperty IRI="hasName" />
    <Datatype abbreviatedIRI="xsd:string" />
  </DataAllValuesFrom>
</SubClassOf>
<ClassAssertion>
  <DataSomeValuesFrom>
    <DataProperty IRI="hasName" />
    <DataOneOf>
      <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">Alpha, LLC</Literal>
    </DataOneOf>
  </DataSomeValuesFrom>
  <NamedIndividual IRI="Alpha" />
</ClassAssertion>
```

У редактор убудований різонер Hermit, якому ми можемо «ставити питання» на тему інформації, що міститься в нашій онтології. Приміром, у цьому прикладі ми можемо довідатися, чим є об’єкт Alpha:

is Alpha ?

Hermit відповість: Alpha is a customer, supplier, organization.

На рис. 2. 20 відображено зовнішній вигляд редактора Fluent Editor.

FluentEditor для OWL є комплексним інструментальним засобом для редагування й обробки складних онтологій, на основі використання контрольованої природної мови. Цей редактор підтримується також інтелектуальним модулем редактора, який забороняє вводити одному

користувачеві будь-яке речення, що не є граматично або морфологічно правильним, і активно допомагає користувачеві у формуванні онтології. На рис. 2. 21 презентовано графічне подання створеної онтології в FluentEditor.

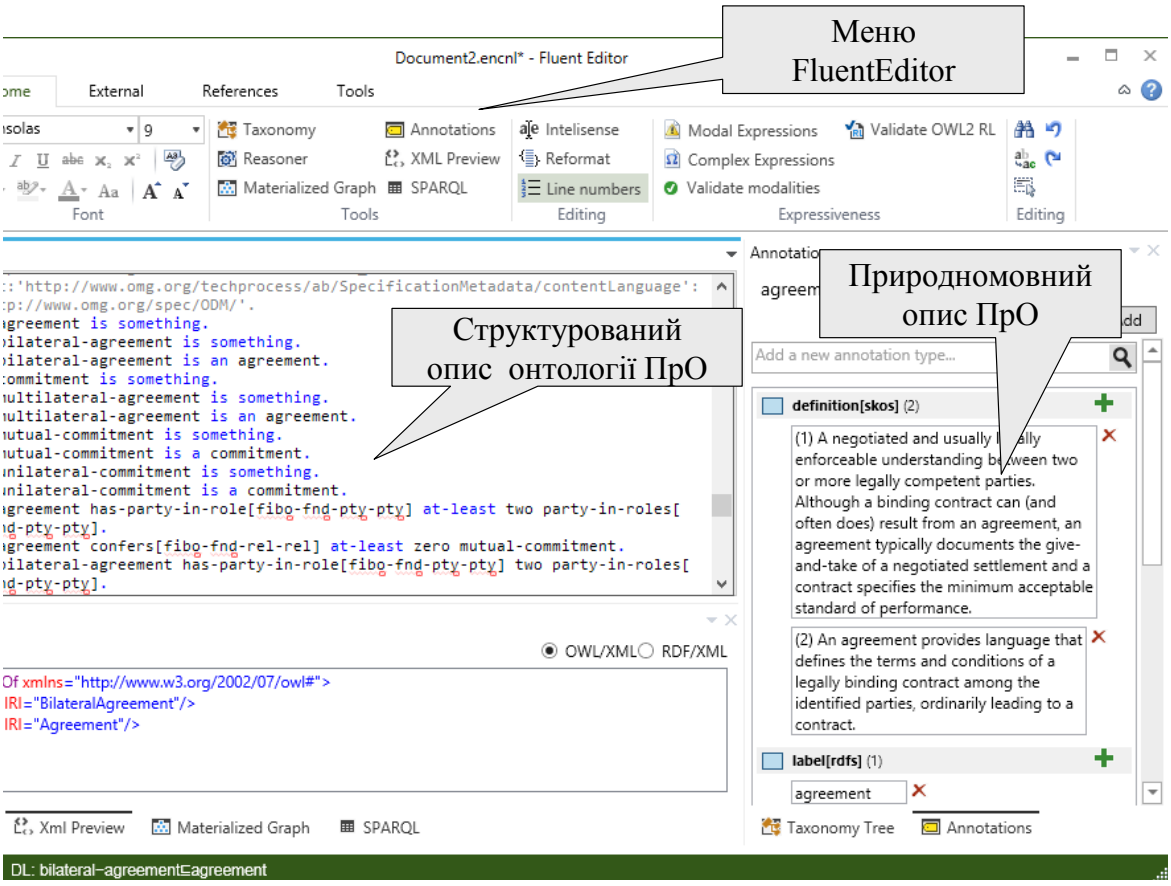



Рис. 2. 20. Обробка тексту редактором FluentEditor

Такий підхід дає змогу записувати явні, формальні концептуалізації людської діяльності. OWL може використовуватися для широкого спектра задач, які виникають при розробці застосувань Semantic Web. OWL дає можливість здійснити опис домену застосування (де формально-семантичний аспект відіграє вирішальну роль) й застосувати специфікації, бази даних, бази даних обмежень, а також бази даних контенту на основі використання природної мови. З іншого боку, OWL може бути використана як мова, що дає змогу експертам у конкретній предметній області (особа, що володіє спеціальними знаннями й навичками в певній сфері діяльності) виразити предметно-орієнтовані знання (достовірне знання використовується на позначення сфери людської діяльності).

На рис. 2. 22 відображена робота онтологічного редактора FluentEditor з природномовним текстом.

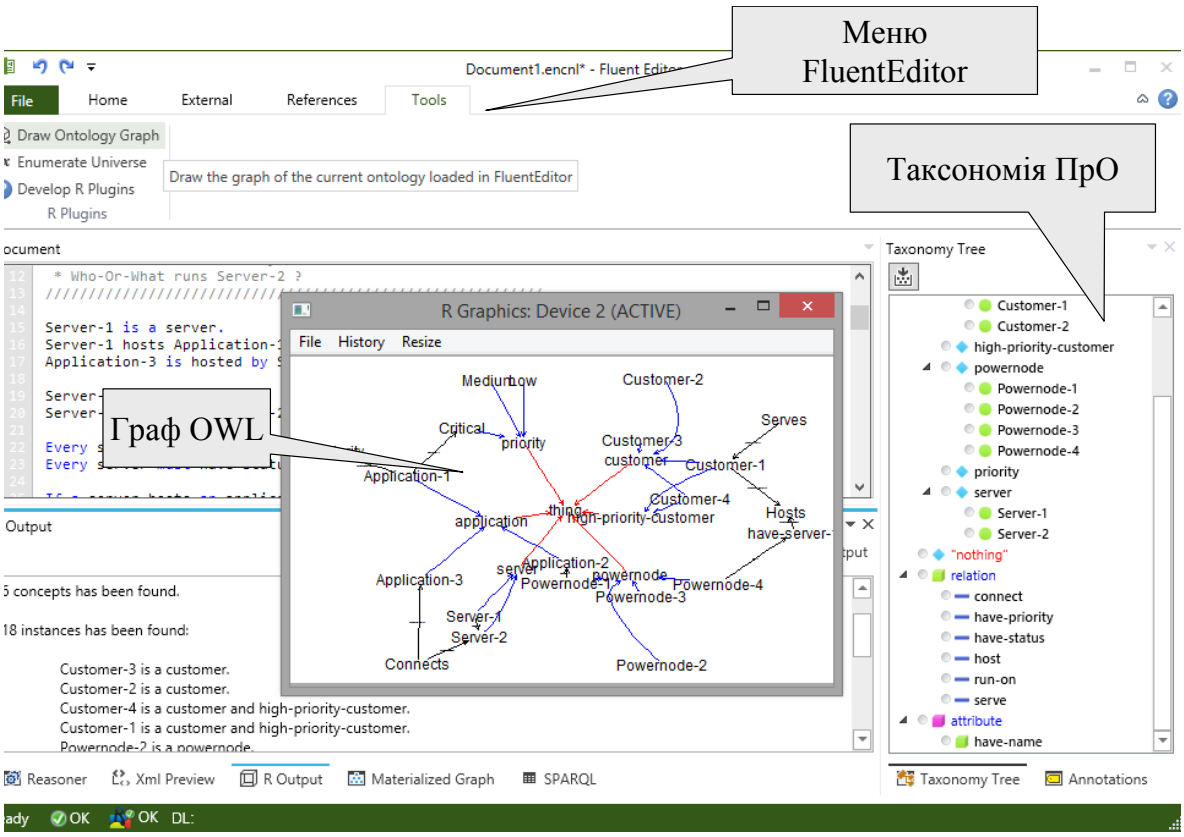



Рис. 2. 21. Побудова графічного зображення онтології в редакторі FluentEditor

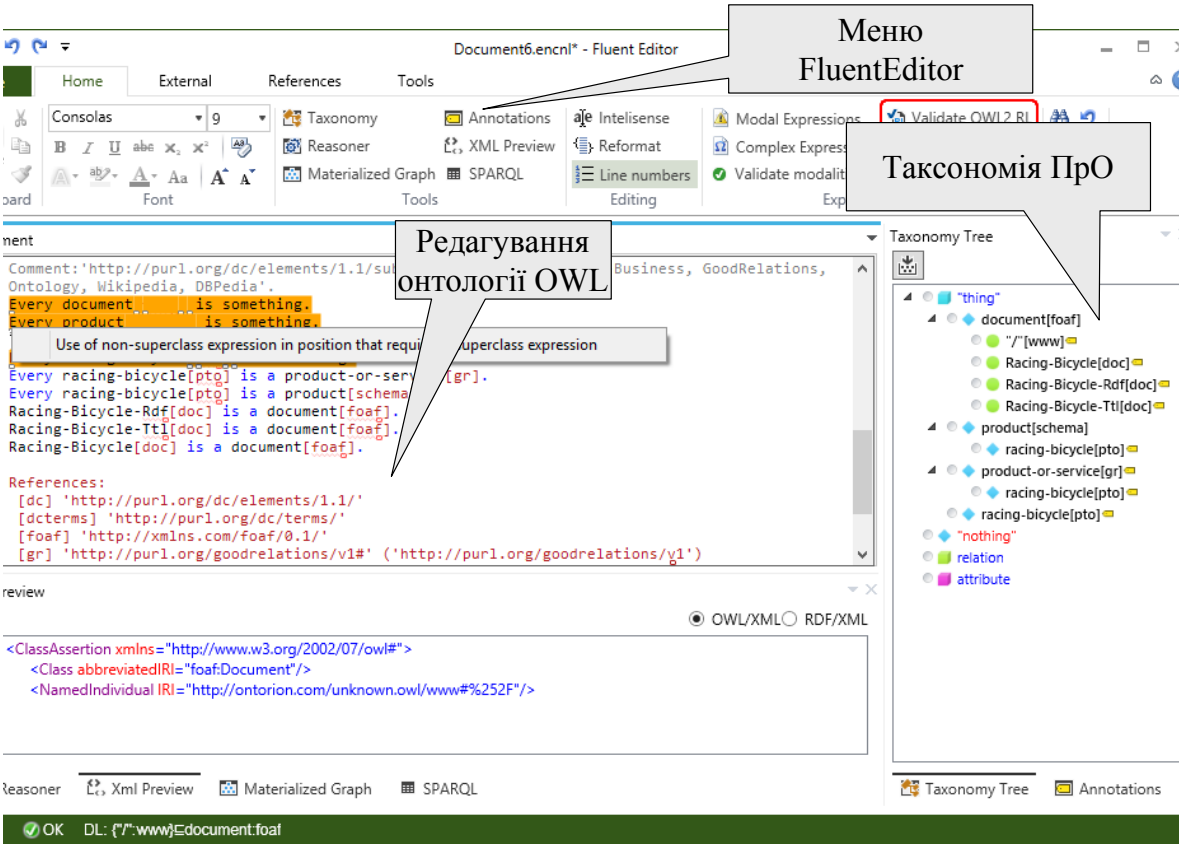



Рис. 2. 22. Лексико-морфологічний розбір речення для створення онтології у FluentEditor

**Висновки**

Технології та стандарти, розроблені в рамках проекту Semantic Web, і програмні продукти, створені на основі цих технологій і стандартів (приміром, широко вживані редактори онтологій і різонери), надають потужний базис для створення, аналітичної обробки та практичного використання онтологій у різноманітних застосовних інформаційних системах, що використовують знання.

# РОЗДІЛ ІІІ.
# РОЛЬ ОНТОЛОГІЙ В ІНТЕЛЕКТУАЛЬНИХ WEB-ЗАСТОСУВАННЯХ

## 3. 1. Використання онтологій інтелектуальними програмними агентами

### 3. 1. 1. Агентна парадигма програмування

*Програмні агенти* (ПА) – це програмні сутності, здатні діяти автономно й цілеспрямовано в динамічному середовищі для того, щоб виконати завдання користувачів [157]. Пов'язана з ПА нова парадигма програмування дозволяє перейти на новий, більш інтелектуальний рівень взаємодії користувача з програмним і апаратним забезпеченням. Вона сприяє підвищенню ефективності праці й дає можливість користувачам доручити ІС виконання досить складних завдань.

Сучасні дослідники використовують на позначення різних типів агентів такі терміни, як інтелектуальні агенти, інтелектуальний інтерфейс, персональні агенти, програмні агенти, мережні агенти тощо [318, 351, 357].

Значна частина програмних агентів – це *інтелектуальні програмні агенти* (ІПА), які спроможні здобувати, накопичувати й використовувати знання, зокрема й онтологічні [125]. Саме використання знань є визначальною ознакою інтелектуальних застосувань [109].

Термін «*агентно-орієнтоване програмування*» (АОП) був запропонований Шохамом для опису набору дій, необхідних для створення ПА – програмних сутностей, що функціонують безперервно й автономно в конкретному оточенні, часто – разом з іншими процесами й агентами [414]. Можна розглядати агентний підхід як метафору проектування та моделювання розподілених систем.

Питання про те, яку комп'ютерну програму варто кваліфікувати як ПА, і досі не розв'язано однозначно. Розмаїття підходів і застосувань свідчить про те, що ПА стали одним з базових напрямів досліджень у галузі ІТ. Однак при цьому термін «програмний агент» використовувався без будь-якої спільної угоди про його значення. Унаслідок цього, деякі програми стали називати агентами тільки тому, що вони, приміром, могли використовуватися для надання завдань віддаленим комп'ютерам або були здатні переміщуватися самостійно між ними. Тому потрібно навести більш точне визначення агентів.

Дослідження ПА має досить тривалу історію й формується на основі результатів, отриманих у рамках таких напрямів, як «розподілений штучний інтелект» (DAI – Distributed Artificial

Intelligence), «паралельний штучний інтелект» (PAI – Parallel Artificial Intelligence), «розподілені системи підтримки прийняття рішень» (DPS – Distributed Problem Solver) [131]. Теорія ПА використовує також результати, одержані в процесі досліджень таких наукових дисциплін, як теорії керування, розподіленого штучного інтелекту та когнітивної психології.

Ідея агентів була запропонована в 50-х роках XX століття Мак-Картні та Селфриджем [344]. Ці фахівці називали агентом програмну систему, яка, за наявності поставленої мети, самостійно виконує певні операції й може при цьому запитувати користувача й отримувати від нього вказівки, подані у термінах природної мови.

Нвана [368] виділяє в дослідженнях агентів два періоди: перший – з 1977 року й другий – з 1990 року. У першому періоді дослідження ґрунтувалися на досягненнях у галузі розподіленого ШІ й були присвячені взаємодії між агентами, декомпозиції задач, координації та кооперації, розв'язанню конфліктів. У другому періоді акцент досліджень змістився з моделювання міркувань (reasoning) у бік планування розподілених і віддалених дій (remote action).

### 3. 1. 2. Властивості програмних агентів

Для того, щоб ефективно використовувати ПА для розв'язання інтелектуальних задач, потрібно знати можливості ПА й теоретичні засади їх розробки.

ПА – це, насамперед, комп'ютерна програма. З цим пов'язуються такі його властивості, як коректність, повнота, ефективність, надійність. При цьому агент виконує певні функції людини, надаючи потрібні користувачеві послуги.

Вимога неперервності й автономії зумовлена потребою в тому, щоб агент був здатний гнучко реагувати на зміни середовища без постійного втручання користувача. Крім того, агент, який працює в середовищі з іншими агентами та процесами, має бути здатним до спілкування з ними.

*Мобільні агенти* – це ПА, орієнтовані на роботу в розподіленому середовищі Web. Вони, рухаючись мережею, виконуються на різних її вузлах незалежно від платформи. Саме такі агенти характерні для сучасного етапу розвитку архітектури інформаційних систем [110]. Важливий вплив на інтелектуалізацію сучасних агентних систем справив також розвиток семантичних технологій (онтологій, Web-сервісів, систем логічного виведення знань тощо) [286].

ПА – автономна фізична або віртуальна обчислювальна сутність, що базується на:

- власних ресурсах знаннях і вміннях;

- засобах сприйняття середовища (сенсорах) і впливу на це середовище (ефекторах);
- моделі середовища, заснованої на знаннях про нього.

ПА забезпечують такі функціональні можливості:

- розв'язання задач або досягнення певних цілей на основі наявних ресурсів і навичок;
- вибір розв'язань серед альтернативних і застосування цього розв'язання в певному середовищі;
- спрямована взаємодія з іншими агентами та середовищем, у якому функціонує ПА.

З погляду користувача, основна перевага використання агентів полягає в спрощенні взаємодії з програмою: користувачеві досить поставити загальне завдання перед агентом, не цікавлячись деталями того, як саме агент має його виконувати. Якщо агент має можливості для виконання цього завдання, він робить це самостійно, а інакше запитує необхідні йому послуги в інших агентів.

Агент – це програмний об'єкт, який:

- забезпечує надання однієї або кількох корисних послуг;
- надає опис семантики цих послуг іншим ПА;
- здатний функціонувати автономно без безпосередніх вказівок користувача;
- може інтерактивно взаємодіяти з іншими ПА, а також з користувачами.

Найпростіше визначення агента ґрунтується на *моделі чорного ящика*, який знаходиться в певному середовищі: агент відображається у вигляді функції f, що обробляє інформацію від сенсорів і вхідні повідомлення (рис. 3. 1), а результат роботи агента – його дії та вихідні повідомлення.

За визначенням FIPA (Federation of Intelligent Physical Agents), *агент* – це об'єкт, що знаходиться в певному середовищі, від якого він отримує дані про події в цьому середовищі, інтерпретує їх і виконує команди, що впливають на середовище. Такий агент може містити як програмні, так і апаратні компоненти. FIPA – це міжнародна організація, створена в 1996 році з метою впровадження агентної парадигми для розробки практичних застосувань [262].

Можна визначати ПА через множину його атрибутів. ПА – термін, що дає змогу об'єднати множину більш специфічних і обмежених типів агентів, які мають деякі з таких атрибутів [60]:

- *реактивність* (reactivity) зміна своєї поведінки залежно від конкретної ситуації;
- *автономність* (autonomy) самостійне виконання розпоряджень користувача без детальних інструкцій;

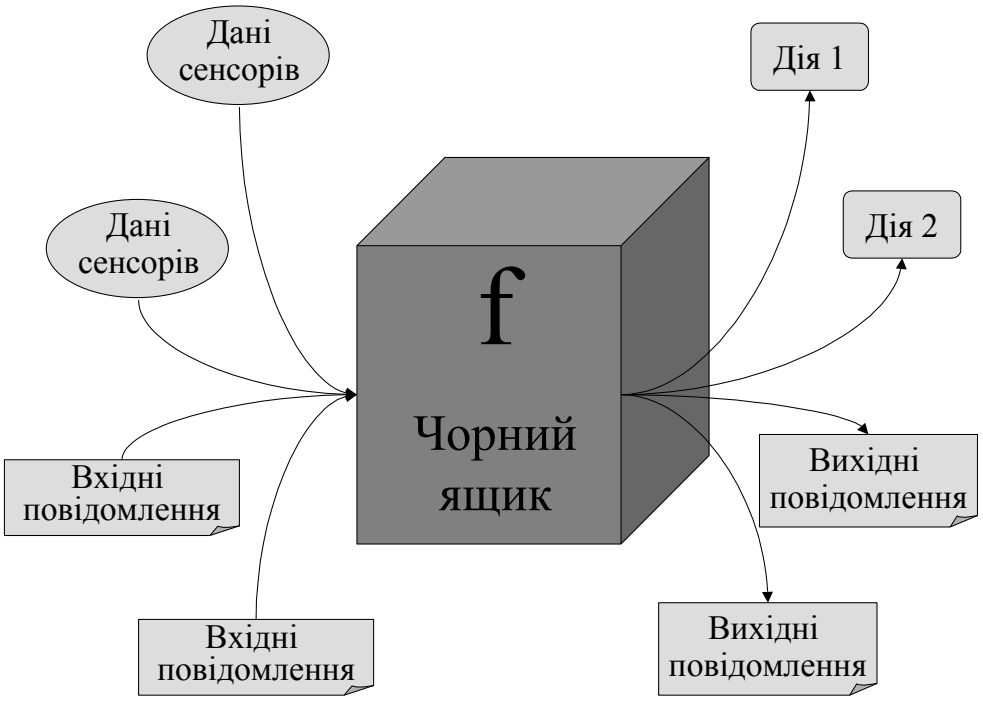



Рис. 3. 1. Визначення агента через модель чорного ящика

- *співробітництво* (collaborative behavior) здатність працювати разом з іншими агентами для досягнення спільної мети;
- *спілкування на рівні знань* («knowledge level» communication ability) спроможність спілкуватися з людьми та іншими агентами мовою, близькою до природної;
- *здатність до логічного виведення* (inferential capability) обробка абстрактного опису задачі з використанням апріорних знань щодо цілей і найбільш придатних методів їх досягнення, здатність будувати моделі власної сутності, свого користувача, ситуацій та інших агентів;
- *безперервність у часі* (temporal continuity) стійкість ідентифікації та положення протягом тривалого періоду часу;
- *персоналізація* (personality) наявність персоналізованих значень атрибутів власної поведінки;
- *адаптивність* (adaptivity) навчання й удосконалення на основі власного досвіду;
- *мобільність* (mobility) здатність самостійно переходити з однієї платформи на іншу.
- *правдивість* припущення про те, що агент не буде свідомо поширювати помилкову інформацію;
- *лояльність* намагання робити те, що потрібно іншим агентам;
- *раціональність* виконання тільки тих дій, які ведуть до досягнення цілей.

Однією з основних характеристик агента є комунікабельність – здатність до гнучкого спілкування як з агентами, так і з іншими програмними компонентами [62].

Крім того, агент повинен мати такі властивості, як: автономність, реактивність, проактивність, раціональність, наявність соціальної поведінки тощо.

Підкласом ПА є інтелектуальні агенти (ІА), які мають засоби для збереження знань (БЗ) і механізми для їх обробки (рис. 3. 2).

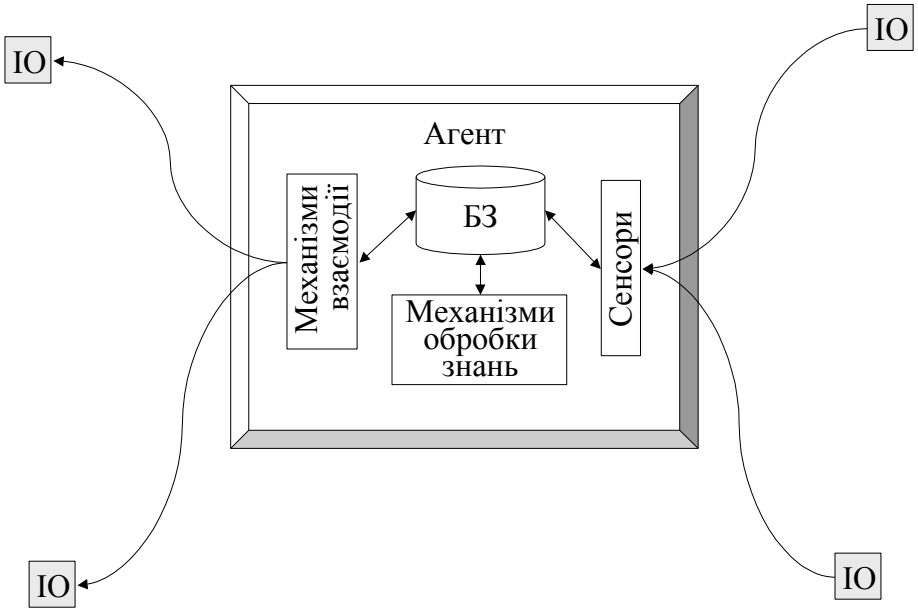



Рис. 3. 2. Загальна концептуальна схема інтелектуального ПА

ІА в широкому розумінні має такі ключові ознаки:

- *автономність* (autonomy) функціонування значною мірою незалежно від втручання людини й контроль власних дій і внутрішнього стану;
- *соціальність* (social ability) інтелектуальна й конструктивна взаємодія з іншими агентами й людьми завдяки обміну з ними повідомленнями за допомогою деякої загальнозрозумілої мови комунікацій;
- *реактивність* (reactivity) сприйняття змін середовища й вчасне реагування на них;
- *проактивність* (pro-activity) здатність агента генерувати цілі й діяти раціонально для їх досягнення, а не тільки реагувати на зовнішні події.

Деннет використовує термін «інтенсіональні системи» для того, щоб відображати сутності, прогнозування поведінки яких здійснюється шляхом приписування їм атрибутів переконання, бажання

й раціональності [240], а Маккарті розглядає сферу застосування таких систем [344]. Чим менше відомо про систему та її структуру, тим корисніші інтенсіональні пояснення її поведінки. Крім того, для досить складної системи (навіть за наявності повної інформації про неї) інтенсіональні пояснення її поведінки часто практично більш корисні, ніж механістичні.

Інтелектуальність ПА визначається його здатністю до міркування й навчання, що, зазвичай, базується саме на Data Mining; наявністю моделі користувача, його потреб і механізму пошуку засобів їх задоволення.

Виділяють такі ознаки інтелектуальності агентів:

- автономне виконання своїх функцій;
- взаємодія з іншими агентами та користувачами;
- здатність стежити за оточенням;
- використання абстракції;
- застосування знання ПрО;
- адаптивність поведінки;
- навчання на власному досвіді;
- толерантність до помилок у вхідних сигналах;
- здатність працювати в реальному часі;
- спілкування природною мовою.

Використання агентів сприяє розвитку принципово нових інформаційних технологій.

Модель ПА містить модель ПрО, модель користувача, засоби сприйняття (сенсори), засоби виконання дій (ефектори), цілі й планувальник дій на підставі цілей, моделі інших агентів і засоби взаємодії з ними.

Модель ПрО, у якій функціонує ПА, відображає структуру та ієрархію об’єктів (наприклад, у вигляді онтології), на які спрямовані дії агентів і які впливають на способи досягнення його цілей. ПА має явно задану символьну модель світу, у якій рішення (наприклад, вибір дії) ухвалюються через логічні (або щонайменше псевдологічні) міркування, що базуються на відповідності між зразками та символьними маніпуляціями. Це спричиняє дві проблеми: як за час, упродовж якого інформація ще буде актуальною, адекватно відобразити реальний світ за допомогою символів і як агентам обробити цю інформацію.

Модель користувача призначена для того, щоб ПА правильно інтерпретував завдання користувача та сповіщав про отримані результати в зручній і зрозумілій формі. У процесі роботи агент може поповнювати модель користувача, накопичуючи досвід взаємодії

з конкретним користувачем (або класом користувачів) для підвищення ефективності своєї роботи.

Засоби сприйняття й засоби виконання дій ПА залежать від функцій, які агент має виконувати. Цілі агента визначають його дії відповідно до принципу раціональності: агент виконує тільки ті дії, які за наявною в нього інформацією та правилами її обробки забезпечать виконання хоча б однієї з його цілей. Планувальник дій агента визначає його дії та їх послідовність.

Моделі інших агентів потрібні ПА для того, щоб успішно взаємодіяти з цими агентами й обмінюватися з ними інформацією в процесі досягнення спільних цілей.

### 3. 1. 3. Класифікація програмних агентів

Для опису та класифікації агентів більшість дослідників використовує різноманітні таксономії, що, залежно від мети дослідження, містять різні набори атрибутів. Наприклад, Гілберт у [273] здійснив опис ПА за допомогою термінів тривимірного простору через дії, інтелект і мобільність (рис. 3. 3).

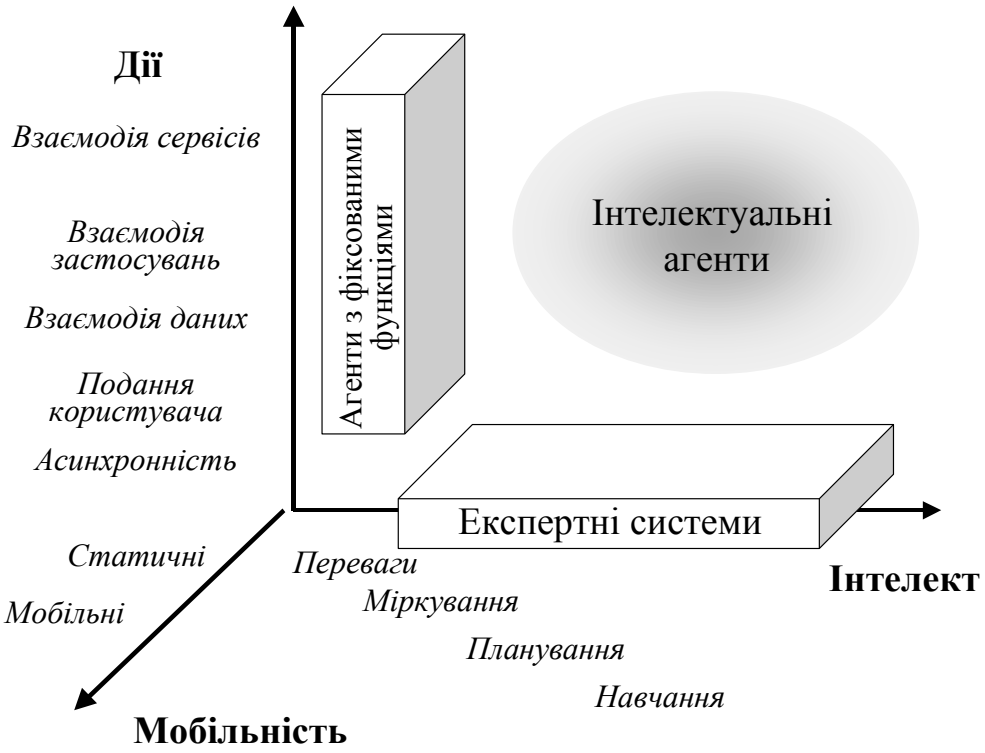



Рис. 3. 3. Класифікація агентів за діями, інтелектом і мобільністю

*Дія* – рівень автономності й повноважень агента, що може оцінюватися через природу взаємодії між агентом і іншими сутностями в системі. Агент має бути асинхронним. Ступінь його дії підвищується,

якщо агент має певне уявлення про користувача. Більш розвинений агент може взаємодіяти з даними, застосуваннями та іншими агентами.

*Інтелект* – рівень поведінки, пов’язаний з міркуваннями й навчанням. Здатність агента сприймати твердження користувача про цілі й виконувати його завдання пов'язана саме з цією якістю. Як мінімум, агент має вміти визначати пріоритет тверджень. Вищий рівень інтелекту потребує моделі користувача й здатності до міркувань. Ще більш інтелектуальними є системи, що спроможні навчатися на власному досвіді для того, щоб адаптуватися до свого середовища.

*Мобільність* – рівень самостійності агента в процесі його переміщень у мережі. Мобільні сценарії можуть формуватися на одній машині й надсилати запит для виконання на інший вузол мережі. Мобільні об'єкти переміщуються між вузлами під час виконання, переносячи накопичені ними дані.

Нвана [368] пропонує типологію агентів, вибравши такі параметри класифікації, як: мобільність, реактивність, первинні атрибути (автономність, співробітництво, навчання).

На основі цих характеристик розрізняють чотири типи ПА:

- співробітництва (collaborative);
- навчання й співробітництва (collaborative learning);
- інтерфейсу (interface);
- розумні (smart).

Додавши до цього опису такі характеристики, як роль, гібридні підходи, що поєднують кілька підходів в одному ПА, і вторинні атрибути (часова безперервність, вірогідність тощо), Нвана виділяє ПА семи категорій:

- співробітництва;
- інтерфейсу;
- мобільні;
- інформаційні;
- реагувальні;
- гібридні;
- розумні.

### 3. 1. 4. Архітектури агентів

Основні принципи побудови та функціонування ПА називаються їх *архітектурою*. Залежно від того, які принципи визначають дії агентів, архітектури поділяються на деліберативні (агенти вибирають план дії на основі логічного виведення з наявних у них знань) і реактивні (дії агентів визначаються як реакція на події в зовнішньому середовищі).

На практиці, як правило, використовують різноманітні комбінації цих архітектур, які називають *гібридними*. Крім того, деякі дослідники виділяють в окремі класи архітектури з певними спільними рисами (приміром, інтерактивні архітектури, архітектури з планувальником дій, архітектури інтелектуальних агентів, інтенсіональні архітектури). Архітектура ПА відображає внутрішню організацію та взаємодію між основними компонентами (рис. 3. 4).

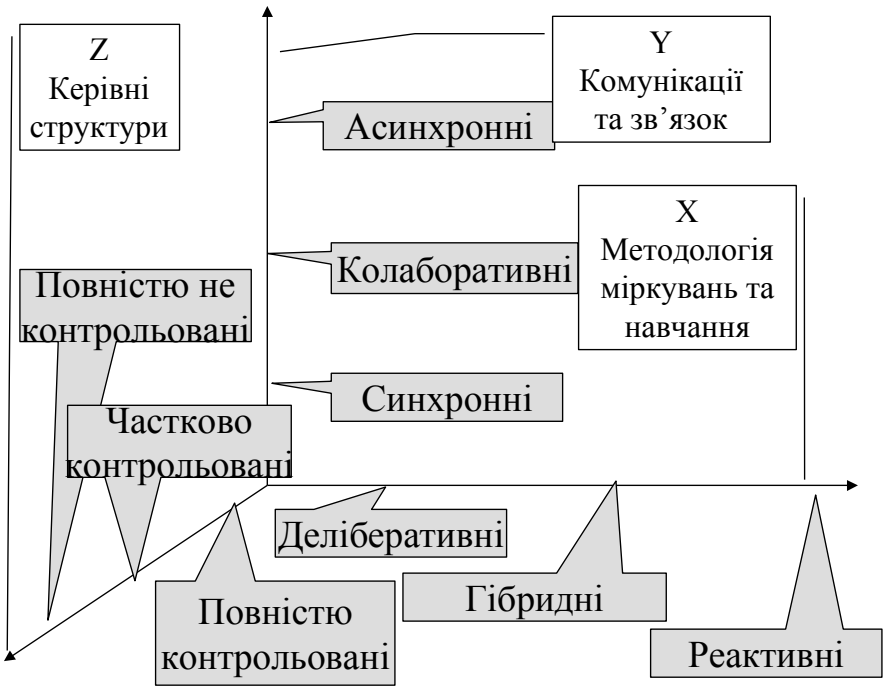



Рис. 3. 4. Таксономія архітектур мобільних агентів

*Деліберативна архітектура* ПА містить символьну модель світу, подану в явній формі, за допомогою якої ПА на основі міркувань логічного чи псевдологічного типу ухвалює рішення про дії, які він здійснює. Такий агент може розглядатися як спеціальний випадок системи, заснованої на знаннях. Використання деліберативної архітектури ПА спричиняє кілька принципових проблем:

- обсяг символьної інформації, яку зберігає ПА, пропорційний розміру внутрішнього стану, тому його збільшення призводить до зниження мобільності ПА;
- символьне подання інформації про середовище, на основі якої міркують ПА, має формуватися в режимі реального часу, щоб знайдені агентами рішення були корисними;
- перетворення сценаріїв реального світу на точне й адекватне символьне визначення є нетривіальною задачею.

*Реактивні архітектури* не використовують централізовану символьну модель світу й не застосовують складні символьні

міркування. Така архітектура оперує на низькому рівні абстракції. Невеликий час очікування відповіді забезпечує ефективну взаємодію між агентами з такою архітектурою, одного з одним і з середовищем. Недолік архітектури – неможливість проведення глибокого аналізу даних від сенсорів.

Реактивна архітектура забезпечує ухвалення рішень за значно меншим обсягом інформації про навколишнє середовище й за значно менший час, використовуючи прості емпіричні правила, специфічні для певної ПрО.

ПА, який здатний міркувати, має явно подану символьну модель світу, у якій рішення (наприклад, про те, яку дію виконувати) продукуються через логічні (або щонайменше псевдологічні) міркування, засновані на відповідності зразків і символьних маніпуляцій.

З іншого боку, архітектури агентів класифікують залежно від типу структури функціональних компонентів ПА й методів організації взаємодії між ними. Як правило, архітектура агента організується у вигляді кількох рівнів.

Тільки найпростіші ПА можуть бути реалізовані за однорівневою схемою. Рівні відображають функції ПА, зокрема такі, як сприйняття зовнішніх подій і прості реакції на них; координування взаємодії з іншими ПА; відновлення переконань про зовнішній світ; визначення своїх дій на черговому кроці тощо. Найчастіше в архітектурі ПА наявні рівні, що відповідають за такі функції:

- сприйняття й виконання дій;
- реактивна поведінка;
- локальне планування;
- кооперативна поведінка;
- моделювання ПрО;
- формування намірів;
- навчання.

*Гібридні архітектури.* Як доводять наведені вище приклади, і реактивний, і деліберативний підходи мають певні переваги й недоліки. Тому доцільно поєднати ці підходи й розробити ПА, що містить дві підсистеми: 1) здатну міркувати, обробляти символьну модель світу, розробляти план і ухвалювати рішення; 2) здатну реагувати на події, що відбуваються в навколишньому середовищі без проведення складних міркувань.

### 3. 1. 5. Мультиагентні системи

Завдання, які постають перед програмними агентами, досить часто є занадто складними, щоб виконувати їх за допомогою одного агента, тому це спонукало до створення нової концепції – *мультиагентних систем* (МАС) [402]. Дослідження в галузі ШІ засвідчили, що взаємодія та декомпозиція загальної задачі ефективно позначаються на результатах розв'язання цієї задачі. Тому краще створювати окремі автономні блоки такої системи, а потім організовувати їх спільне функціонування [24]. Саме для таких систем онтології постають не тільки джерелом знань, а й основою для інтеграції та успішної спільної роботи.

Термін «*мультиагентні системи*» використовується на позначення ІС, які складаються з множини автономних модулів ПА і мають такі властивості:

- кожен ПА є автономним, мобільним та інтероперабельним;
- ПА, що входять до складу МАС, здатні обмінюватися інформацією для досягнення спільних цілей;
- керування ПА може бути децентралізованим;
- джерела даних і доступ до них децентралізовані;
- робота агентів є асинхронною.

Теорія МАС базується на теорії відкритих систем, розподіленого ШІ й загальній теорії складних систем [185]. Розподілений штучний інтелект (РШІ) пов'язаний з аналізом систем, що складаються з окремих незалежних об'єктів, які взаємодіють один з одним, і механізмів їх координації. МАС теж є предметом розгляду РШІ. У цьому разі незалежними об'єктами є ПА [264]. Для вивчення поведінки МАС використовують методи таких наукових дисциплін:

- *розподілений ШІ* (теорія розподілених систем, теорії ухвалення рішень), що займається найбільш загальними аспектами колективної поведінки агентів;
- *теорія ігор*, яка використовується для дослідження ситуацій, аналогічних до кооперативних ігор, стратегій ведення переговорів;
- *теорія колективної поведінки автоматів*, яка досліджує колективну поведінку великих груп автоматів з примітивними функціями, спроможних навчатися за допомогою системи штрафів і заохочень;
- біологічні, економічні й соціальні моделі.

У розробці МАС використовуються також результати з інших галузей теоретичних досліджень (рис. 3. 5).

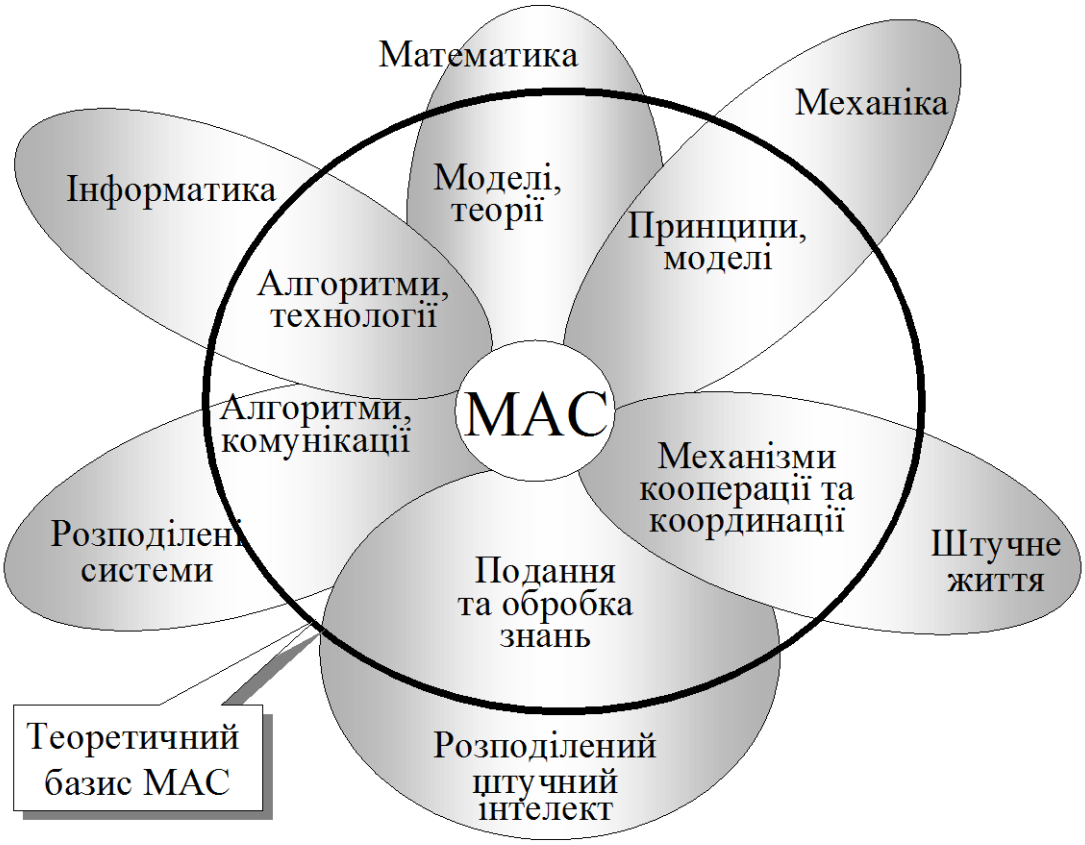



Рис. 3. 5. Основні ПрО, що використовуються в розробці МАС

МАС – співтовариство ПА, які пов'язані один з одним цілями й ресурсами. ПА, що входять до складу МАС, здатні взаємодіяти з метою обміну послугами, які потрібні їм для досягнення цілей, поставлених перед ними користувачами. Агент, що не здатний самостійно розв'язати задачу, поставлену перед ним користувачем, звертається до інших МАС поширені в багатьох ПрО: при керуванні виробничими процесами й промисловими підприємствами; плануванні руху транспорту (повітряного, залізничного, автомобільного); аналізі й пошуку інформації; у навчанні; бізнесі тощо [38, 63, 64].

МАС складається з множини ПА, які [438]:

- взаємодіють шляхом комунікацій;
- здатні діяти в певному середовищі;
- мають певні сфери впливу, які можуть перетинатися або збігатися.

Функціонування МАС пов'язано з кооперацією та конкуренцією агентів у процесі колективного розв'язання задач. Агент, що не може розв'язати власну задачу самостійно, має взаємодіяти з іншими ПА. При цьому агенти можуть будувати плани спільних дій, спираючись не тільки на власні можливості, а й на аналіз планів і намірів інших ПА

(використовуючи різні комбінації інтенсіональних відношень). Кооперативні дії ПА – важлива перевага МАС. За такої організації роботи група ПА виявляє новий, ефективніший тип поведінки.

Мотивація використання МАС базується на їх властивостях:

- здатність розв'язувати проблеми, складні для одного централізованого ПА через обмеженість ресурсів;
- можливість взаємодії та інтероперабельності з різноманітними застосунками (ЕС, СППР тощо);
- розв'язання дійсно розподілених проблем (приміром, керування рухом транспорту);
- знаходження рішень на основі розподілених ІР;
- розв'язання проблем з розподіленою експертизою (приміром, медичний догляд);
- підвищення швидкості та надійності;
- здатність збільшувати кількість обчислювальних пристроїв (процесорів), що використовуються для розв'язання задачі;
- можливість опрацювання нечітких і неповних даних і знань.

Знання й уміння МАС здобуваються від великої кількості відносно простих ПА, що поєднуються разом деякою архітектурою. У процесі колективної роботи агенти мають будувати плани дій, спираючись не тільки на свої можливості, а й ураховуючи плани та наміри інших агентів (на основі своїх знань і переконань щодо цих планів і намірів) [27].

Причини взаємодії ПА:

- наявність спільних цілей;
- брак індивідуальних ресурсів ПА для досягнення цілей;
- нездатність ПА самостійно розв'язати задачу;
- наявність взаємних зобов'язань.

Перехід до відкритих МАС уможливлює перехід до нової якості функціонування системи завдяки тому, що система (група агентів) більше, ніж сума властивостей її членів (агентів) [45].

Найпростіший метод координації МАС – *організаційне структурування* – полягає в чітко визначених і довгострокових відношеннях між ПА. При цьому використовують ієрархічні структури *master-slave* або *client-server*. Організаційне структурування передбачає, що принаймні один ПА має глобальне уявлення про все співтовариство, однак для багатьох ПрО це не реально.

Цей метод використовується в двох варіантах:

- головний ПА планує й розподіляє завдання між підлеглими ПА, які, на відміну від головного, мають часткову автономність;

- для координації використовується дошка оголошень з метою обміну інформацією між ПА, операції з якою (зчитування й запис) визначає головний ПА.

Інший варіант координації – *укладення контракту* – припускає децентралізовану структуру. ПА мають дві ролі:

- менеджер поділяє задачу на підзадачі й шукає виконавця для їх розв'язання;
- виконавець реалізує свою підзадачу.

Виконавець може рекурсивно стати менеджером і виділити у своїй підзадачі ще дрібніші задачі й доручити їх розв'язання іншим ПА.

Пошук виконавця передбачає такі етапи:

- менеджер повідомляє задачу;
- виконавці оцінюють цю задачу щодо можливості її розв'язання;
- менеджер отримує таблицю виконавців, оцінює надані пропозиції, вибирає виконавця й доручає йому розв'язання певної задачі;
- виконавець розв'язує запропоновану задачу й повідомляє менеджера про одержані результати.

Така модель координації забезпечує розподіл задачі й засоби для самоорганізації співтовариства агентів. Її доцільно застосовувати в таких ситуаціях:

- задача має чіткий ієрархічний характер;
- у задачі можна виділити великі підзадачі;
- взаємозв'язок між під задачами має бути відносно невеликим.

Переваги цього підходу координації полягають у динамічному розподілі задачі завдяки пошуку оптимального виконавця; у збалансованому завантаженні ПА й надійному механізмі для розподіленого керування.

Є два типи планування діяльності МАС: централізоване й розподілене.

У *централізованому плануванні* діяльності МАС є координаційний ПА, що отримує індивідуальні плани від інших, аналізує їх, щоб знайти потенційні протиріччя й конфлікти у взаємодії ПА. Потім координаційний ПА намагається змінити ці індивідуальні плани й поєднує їх у план МАС, у якому усунені суперечливі взаємодії та додані команди зв'язку, що синхронізують взаємодію ПА в потенційно можливих конфліктах.

У *розподіленому плануванні* діяльності МАС кожен ПА отримує моделі планів інших ПА, що входять до складу МАС. ПА взаємодіють, щоб побудувати модифікації своїх індивідуальних планів, які не конфліктують з планами інших ПА. Координація в розподіленому плануванні діяльності МАС набагато складніша, ніж

у централізованому, оскільки жоден ПА не має глобального уявлення про всю розподілену систему.

*Переговори* – один з найскладніших методів координації, який використовує методи ШІ, логіку тощо. У динамічному співтоваристві агенти самостійно розподіляють роботу між собою. Часто одну й ту ж роботу в співтоваристві можуть виконати кілька ПА. Агенти на момент розподілу робіт перебувають у різних станах (приміром, мають різний ступінь завантаженості). Для ефективного розподілу робіт між ПА вони мають взаємодіяти один з одним – вести переговори, у процесі яких можна виявити оптимального виконавця для кожної роботи.

Процес переговорів складається з таких компонентів:

- протоколу переговорів – набору правил, за якими відбувається взаємодія агентів (учасники переговорів, їх типи, стани переговорів, правила, за якими змінюються стани переговорів, можливі дії учасників тощо);
- об'єкта переговорів – діапазону проблем, стосовно яких потрібно досягти згоди;
- моделі ухвалення рішення ПА – апарат ухвалення рішення, який використовують учасники відповідно до протоколу переговорів.

Отже, у процесі побудови МАС, що розв'язує реальну задачу, як правило, необхідно використовувати комбінацію наведених вище підходів. При моделюванні діяльності віртуальних і реальних підприємств переважно застосовують організаційне структурування, яке задовільно відображає ієрархічну структуру підприємства. Крім того, це один з найпростіших підходів до координації. Інші методи координації виявляють серйозні недоліки в процесі розв'язання подібних задач: переговори – через складність реалізації; планування МАС придатне лише для специфічних задач, таких, як планування авіарейсів. Укладення контракту при моделюванні віртуальних підприємств не дає переваг, порівняно з організаційним структуруванням, оскільки мережа контрактів фактично визначена до початку роботи, а тому немає необхідності в пошуку виконавця.

Для МАС характерні децентралізованість даних, асинхронність обчислень і наявність засобів комунікації. Упровадження МАС дає змогу розв'язувати проблеми, що є надто складними для окремого ПА або пов'язані з обмеженими ресурсами, і забезпечує збільшення ефективності системи через паралельні обчислення й повторне використання ПА в різних співтовариствах агентів. Для успішного функціонування вони потребують великих обсягів знань, які мають постійно оновлюватися й перевірятися. Це досягається шляхом обміну знаннями між ПА [185].

МАС можуть бути як централізованими, так і децентралізованими. У централізованих МАС, створених для розв'язання агентами спільних задач, конфлікти між цілями агентів не виникають (або ж наявні централізовані механізми усунення таких протиріч). Приміром, в інформаційно-пошуковій МАС, що обслуговує групу користувачів, пріоритети користувачів визначають порядок їх обслуговування. Проте такі МАС значно важче адаптувати до нових потреб користувачів або нових завдань.

Значна частина сучасних МАС складається з ПА, які не є повністю автономними, тобто здатними вибирати для себе цілі й вирішувати, як саме досягати їх і які саме дії виконувати для цього. Набір ПА розробляється цілісно, і проблема взаємодій між агентами розв'язується в процесі їх розробки. Такі проблеми, як конфлікти або недостатність ресурсів, узагалі не розглядаються. Агенти, що входять до складу такої МАС, мають обмежену автономію: їх роль у процесі розв'язання загальної проблеми звичайно заздалегідь визначається розробником системи, але агенти вільні у виборі способів досягнення своїх цілей.

Є багато різноманітних типів МАС, але можна виділити характеристики, яким задовольняють деякі з цих систем:

- агенти, що входять до складу МАС, автономні;
- агенти поєднують свої індивідуальні потреби з потребами всієї системи.

МАС, що відповідає цим умовам, називають *відповідальним співтовариством.* Його компонентами є:

- *елементи*, призначені для розв'язання певних проблем;
- *середовище*, у якому знаходяться ці елементи;
- *способи взаємодії* між елементами;
- *цілі* мотиваційна сила, під впливом якої елементи розв'язують проблеми.

Разом ці компоненти характеризують систему.

### 3. 1. 6. Мови взаємодії програмних агентів

ПА можна розглядати як застосунки, для яких здатність до комунікації з іншими застосунками й до пошуку знань має першорядну вагу. Компоненти ПА можна поділити на такі, як подання комунікацій, і такі, що не стосуються безпосередньо розподіленої взаємодії.

Для розв'язання проблеми комунікацій між інтелектуальними ПА необхідна спільна мова, тобто спільні синтаксис, семантика й прагматика. На сьогодні немає загальноприйнятої мови подання інформації та запитів. Такі мови, як KIF і SQL, досить поширені, але цього недостатньо, щоб визнати їх як стандарт.

Агенти мають взаємодіяти на рівні знань, а тому не можуть послуговуватися звичайними мовами й протоколами, що призначені для розподілених обчислень і звичайно фокусуються на процесах, а не на програмах або коаліціях ПА. Мова комунікації агентів має бути досить потужною для підтримки взаємодії між програмами високого рівня. Мова комунікації – не тільки протокол.

Протокол взаємодії – це високорівнева стратегія, що здійснюється агентом і взаємодіє з іншими агентами. Цей протокол може мати такий діапазон: від схеми переговорів і протоколів теорії ігор до простих протоколів типу «*щораз, коли я чогось не знаю, я знаходжу кого-небудь, хто це знає, і я запитую його»*. Вибір транспортного протоколу залежить від специфіки його застосування.

Можна сформулювати такі вимоги до мови взаємодії агентів:

- *форма*: декларативна, синтаксично проста, зрозуміла;
- *зміст*: відрізняє мови опису комунікаційних актів від мов передачі змісту повідомлень;
- *семантика*: надає бажані властивості;
- *реалізація*: проста, сумісна з наявними ПЗ;
- *робота в мережі*: ураховує основні аспекти сучасних ІТ і має незалежний транспортний механізм;
- *середовище*: підтримує гетерогенність і динамізм;
- *надійність*: забезпечує надійність і безпеку взаємодії агентів.

Ці вимоги можуть конфліктувати одна з одною. Приміром мова, що є простою для розуміння її людиною, може бути не настільки стислою, як належить. Вибір вдалого співвідношення всіх вимог є для розробника мови нетривіальним завданням.

Крім того, мова комунікацій має забезпечувати надійність, конфіденційність і безпеку взаємодії агентів на основі механізму ідентифікації агентів і виявлення помилок.

Для конструктивної та інтелектуальної взаємодії між ПА необхідна не тільки спільна мова, а й спільне розуміння термінів, що використовуються в конкретному контексті.

Отже, ПА взаємодіють у трьох напрямах:

- спільної мови;
- спільного розуміння знань, якими вони обмінюються;
- здатності обмінюватися знаннями за допомогою спільної мови.

Вимоги до мов комунікації агентів можна поділити на такі групи: вимоги до форми, змісту, виконання, роботи в мережі, середовища, надійності. Для розв’язання проблем спілкування між агентами в рамках проекту KSE з метою створення мови для підтримки різноманітних архітектур агентів розроблено протокол KQML (Knowledge Query Manipulation Language) [261]. При цьому особливий

інтерес становить спеціальний клас ПА – *посередники* (facilitators) комунікацій. Посередники – це агенти, що реалізують різноманітні комунікаційні сервіси, приміром, підтримують реєстрацію імен сервісів, переправляють повідомлення для них, виконують маршрутизацію повідомлень на підставі вмісту й забезпечують проміжний сервіс.

Подання знань ПА може бути поділено на дві підзадачі: 1) переклад з однієї мови подання на іншу; 2) пошук семантичного контексту подання знань для різних застосувань.

Комунікації зумовлюють проблеми, які пов'язані з протоколом взаємодії, мовою комунікацій, транспортним протоколом.

Крім протоколів, ПА можуть містити інші компоненти, що допомагають виконувати свої завдання. Здатність міркувати про свої дії, подавати метазнання, планувати дії або моделювати інших агентів дають змогу підвищувати можливості застосувань. Такі компоненти є периферійними для бази знань. Вони також можуть використовувати базу знань і комунікаційну мову.

*KSE* – це ініціативна група з розробки й розвитку технічної інфраструктури для підтримки пошуку знань між системами. KSE організувала три робочі групи, кожна з яких розв'язувала одну з проблем технології подання знань: Interlingua, Shared Reusable Knowledge Bases, External Interfaces. Група *Intrelingua* розробила спільну мову подання вмісту БЗ. Цією групою був опублікований документ, що відтворює формат обміну знаннями KIF (Knowledge Intrchange Format). KIF може бути використано для забезпечення перекладу з однієї мови на іншу або як спільну мову для двох агентів, що використовують різні мови подання (http://www.cs.umbc.edu/kse/kif/). Група *SRKB* (Shared Reusable Knowledge Bases) займалася розробкою розподілених БЗ і пошуком знань за визначеними темами, а також розробкою предметно-незалежних інструментів і методологій. Група *External Interfaces* розглядала питання взаємодії систем, заснованих на базах знань.

*KIF* є версією числення предикатів першого порядку з додаванням підтримки немонотонних міркувань і визначень. Визначення мови передбачає опис синтаксису й семантики. KIF містить логічні оператори, які допомагають подавати логічну інформацію (заперечення, диз'юнкція, правила, квантифікаційні формули тощо). KIF дає змогу кодувати знання про знання, використовуючи символи «'» і ",", операторів і пов'язаний покажчик. KIF можна також застосовувати до опису процедур, тобто програм ПА. Семантика KIF (без правил і визначень) подібна до логіки першого порядку. Вона розширюється за рахунок використання нестандартних операторів і обмежується

аксіомами моделювання. За винятком цих розширень і обмежень, ядро мови зберігає фундаментальні характеристики логіки першого порядку.

*KQML* – мова й набір протоколів, що забезпечують ПА ідентифікацію, зв'язок і обмін інформацією з іншими агентами.

Основні риси KQML:

- повідомлення KQML непрозорі щодо вмісту;
- перформативи (performatives) примітиви мови визначають припустимі операції, які може здійснити ПА для комунікацій з іншими агентами;
- середовище агентів, що використовують KQML, може бути поповнене спеціальними агентами для виконання сервісних функцій: зв'язку фізичних адрес з символьними іменами, реєстрації БД, комунікаційних послуг тощо.

KQML має три рівні – вмісту; комунікацій; повідомлень.

Рівень *вмісту* – це фактичний зміст повідомлення власною мовою подання (KQML може переносити висловлювання будь-якою мовою подання, зокрема й у вигляді рядків ASCII). Деякі ПА, що використовують KQML (приміром, програми маршрутизації, узагальнені посередники), можуть ігнорувати вміст повідомлення й тільки визначати, коли воно закінчується.

Рівень *комунікацій* містить набір властивостей, що відображають низькорівневі параметри комунікації (такі, як ідентифікація відправника й одержувача, ідентифікатор комунікації тощо).

Рівень *повідомлень* використовується для кодування того, що одна програма намагається передати іншій. Цей рівень формує ядро мови KQML і визначає вид взаємодії з агентом, що використовує KQML. Основною функцією рівня повідомлень є ідентифікація протоколу, що використовується для передачі повідомлення й забезпечення виконання певної дії, яку відправник пов'язує з повідомленням (запиту, команди тощо). Крім того, на цьому рівні можуть бути відображені властивості мови вмісту, а також визначено його тематику. Це дає змогу KQML здійснювати аналіз , маршрутизацію й доставку повідомлень навіть тоді, коли їх вміст недоступний.

Синтаксис KQML ґрунтується на збалансованому списку з дужками. Початковий елемент списку – перформатив, а інші елементи – його аргументи у вигляді пар *(ключове слово, значення)*.

### 3. 1. 7. Засоби інтелектуалізації програмних агентів

Для досягнення успіху в складному динамічному середовищі інтелектуальний ПА потребує засобів адекватного реагування на зміни у своєму оточенні або ж на невідповідність апріорних припущень про це оточення. Поведінку агента відображає *принцип раціональності*: він

вибирає дію, якщо має знання про те, що ця дія допоможе йому досягти одну з його цілей. Агенти, що мають однакові знання й цілі, на рівні знань не розрізняються, навіть якщо у них різна фізична структура.

Рівень інтелектуальності ПА визначається його здатністю до міркування й навчання, тобто можливостями здобуття необхідних знань за доступною інформацією. Критерії оцінювання самостійно здобутих знань збільшують автономність агента, а вміння ПА враховувати свій досвід спілкування з користувачами підвищує його ефективність. Тому засоби здобуття знань мають бути невіддільним складником моделей інтелектуальних ПА.

Система автономна (у розумінні «не контролюється безпосередньо людиною») настільки, наскільки її поведінка визначається її власним досвідом.

Агент, що планує свої дії на основі вбудованих припущень, діє успішно тільки тоді, якщо ці припущення виконуються, але цим припущенням може бракувати гнучкості. Унаслідок динамічності оточення ПА використання вбудованих припущень часто зумовлює неефективні дії. Автономний ПА успішно діє навіть тоді, коли оточення є дуже різноманітним, оскільки він має достатньо часу для адаптації. Для цього ПА використовує елемент навчання, який за допомогою зворотного зв’язку порівнює переконання агента про доцільність виконання доступних дій з реальними результатами виконання цих дій. Якщо знаходяться розбіжності, то елемент навчання намагається змінити переконання ПА. У процесі розробки елемента навчання треба враховувати, які переконання ПА потрібно вдосконалити, як вони подані, як здійснюється зворотний зв’язок і яка апріорна інформація доступна ПА.

Можливість навчання є важливою властивістю будь-якої системи, яка функціонує в середовищі, що динамічно змінюється. У МАС проблеми навчання дуже актуальні й мають ряд особливостей. При цьому інтелектуальність розглядається вже не як властивість окремого ПА, а як властивість групи агентів, що взаємодіють між собою. Тому необхідно враховувати такі питання:

- у чиїх інтересах має здійснюватися навчання (окремого ПА, групи агентів, всієї МАС);
- як здійснюється процес навчання (самостійно, через взаємодію з іншими ПА, усією групою);
- що є об’єктом навчання (поведінка ПА, взаємодія тощо);
- який метод навчання використовується;
- як перевіряти ефективність навчання.

Інтелектуальна поведінка системи пов’язана зі здатністю класифікувати об’єкти (а саме задача класифікації є найпоширенішою

в Data Mining). Проблема, яку має розв'язати ПА, полягає в тому, щоб за набором прикладів (із власного досвіду або досвіду інших ПА, з якими він здатний взаємодіяти) побудувати гіпотезу, яка узагальнює ці приклади або дає змогу знаходити рішення в аналогічних ситуаціях. Задача прогнозування властивостей полягає у визначенні невідомих характеристик об'єкта дослідження.

Використання механізмів навчання на базі прецедентів у МАС дає можливість забезпечити адаптивність їх поведінки. Механізми навчання можуть бути застосовані для вибору моделі поведінки агента, стратегії переговорів, протоколів взаємодії, ухваленні рішень тощо. Це пов'язано з використанням різноманітних методів ШІ: логіки, виведення на основі фактів, цілеспрямованого пошуку тощо. Розрізняють такі напрями:

- розподілене виведення на основі фактів;
- колективне виведення на основі фактів;
- проактивне навчання;
- здобуття знань на основі аукціонів координоване здобуття прецедентів у МАС.

Якщо навчається один ПА з МАС, то він самостійно вирішує, чи є певний прецедент корисним для його навчання. У МАС приклад, який не потрібний одному ПА, може бути корисним для іншого. Тому такий процес навчання складається з двох етапів: акумуляції прецедентів і обміну прецедентами.

Для етапу акумуляції прецедентів характерні три стратегії:

- агент ніколи не додає прецедент до своєї бази фактів;
- агент завжди додає прецедент до своєї бази фактів;
- агент додає прецедент до своєї бази фактів, якщо він є цікавим (приміром, якщо ПА неправильно класифікує ситуацію, відтворену в прецеденті).

На етапі обміну прецедентами визначають стратегію поведінки ПА. Наприклад, агент ніколи не пропонує прецедент для навчання іншим агентам або завжди пропонує його іншим ПА.

Агент $A_i$ може отримати від агента $A_j$ опис прецедентів, які можуть допомогти йому розв'язати певну задачу за таким протоколом:

- $A_i$ надсилає опис проблеми P іншим агентам $A_j, j=\overline{1,n}$, що входять до складу МАС;
- $A_j$ здійснює спробу розв'язати проблему, використовуючи свою базу прецедентів $C_j$;
- $A_j$ надсилає $A_i$ такі повідомлення:

– відмову, якщо він не може розв'язати проблему;

– розв'язання у вигляді трійки $\langle S_k, P, A_j \rangle$, яке означає, що агент $A_j$ вважає $S_k$ найбільш імовірним розв'язанням проблеми P.

Схема голосування визначає механізм, за допомогою якого агент $A_i$ будує узагальнене розв'язання на основі розв'язань, що надійшли від інших агентів $A_j, j = \overline{1,n}$. Приміром, узагальнене розв'язання може визначатися як клас, що отримав найбільшу кількість голосів (комітетна політика).

### 3. 1. 8. Використання агентів в електронній комерції

Програмні агенти використовуються як потужний програмний засіб для розробки й реалізації складних інформаційних систем [24].

Агенти реєструються на власній платформі для взаємодії з іншими агентами на цій же платформі або агентами на інших платформах. Агент може реалізуватися різними способами: як Java-компонент, COM-об'єкт, самовбудована LISP-програма, TCL-скрипт.

Розробка програм-агентів потребує застосування стандартизованих профілів агентів і методології розробки програм агентів для кожного з конкретних застосувань [24].

Покращення ефективності виконання завдань е-бізнесу потребують подальшого розвитку методів автоматизації ділових процесів. Головною перевагою внаслідок упровадження агентно-орієнтованих систем е-бізнесу є скорочення вартості й швидкість реалізації бізнес-процесів, а також надання більш зручних послуг клієнтам.

Розглянемо це на прикладі мультиагентної системи для завдань е-комерції [39]. Щоб бути застосовними (корисними), мобільні агенти повинні зв'язуватися з різними комп'ютерами, агентами й рухатися всередині різнорідних мереж. Це потребує реалізації стандартизованого каркаса й методології для дій агента через телекомунікаційні мережі.

У системі е-комерції виділяються дві взаємозалежні підсистеми: торгівля й керування діалогом. У торговельній підсистемі можуть бути агенти товарів і замовлень, а також агенти продавців і покупців, складу, постачальників тощо. Агенти товарів і замовлень ведуть переговори щодо стратегій знижок для постійних покупців, знижок за оптову покупку, знижок, зумовлених становищем конкурентів, знижок, що залежать від затоварювання складу, та ін. При цьому декілька агентів-покупців (потенційних конкурентів) можуть об'єднати свої замовлення для одержання більшої знижки, тобто перейти від конкуренції до кооперації. У підсистемі ж керування діалогом потрібно створити систему видачі результатів переговорів агентів.

Технологія Java надає відкриту стандартну універсальну платформу для мережних обчислень. Під час розробки технології особливий акцент було зроблено на незалежності застосувань Java від конкретної апаратно-програмної платформи (що й уможливило успішний обмін у гетерогенному обчислювальному середовищі застосуваннями й навіть їх фрагментами). Ця мета досягається за допомогою мови Java й віртуальної Java-машини, у коди якої (так звані байти-коди) транслюються Java-застосування.

Модель МАС містить такі модулі: Offer – визначення ціни товару на торгах; Basic Negotiation – ведення базових переговорів (правила); AboutDialog – ведення діалогу (інтерфейс з користувачем); Marketplace Frame – блок координації й керування МАС; BaySel Message – інтерфейс взаємодії з користувачем; FacilitatorAgent – агент-посередник; Better BuyerAgents – найкращий агент-покупець; Best BayerAgent – кращий агент-покупець; BayerAgent – агент-покупець; Better SellerAgents – найкращий агент-продавець; Best SellerAgents – кращий агент-продавець; SellerAgents – агент-продавець.

На основі мови програмування Java розроблений FacilitatorAgent (агент-посередник), що керує ринком, а також агенти BuyerAgents (агент-покупець) і SellerAgents (агент-продавець), які використовуються для взаємодії всередині цього ринку. Агенти клієнта й продавця (торговця) розрізняються насамперед за складністю їх переговорних стратегій. Переговори починаються з простої логіки (у термінах if-then-else), а потім переходять до методів формування правил, які базуються на набутих фактах.

FacilitatorAgent є посередником між клієнтами й торговцями. Всі агенти зобов’язані зареєструватися в посередника перед тим, як почати взаємодію з якими-небудь іншими агентами на ринку (marketplace). Торговці рекламують бажання продати товари або послуги за допомогою посередника, водночас клієнти також просять посередника рекомендувати їм імовірного продавця. Як тільки агенти клієнта й торговця будуть представлені посередником, вони продовжать спілкування тільки через нього.

На рис. 3. 6 відображена головна панель програми Marketplace, реалізована на основі моделі МАС із застосуванням мови Java. У ній використовуються дві текстові зони, які показують повідомлення від посередника, покупця й продавця на ринку. Пункти меню «Файл» містять команди «Початок» і «Зупинка». Меню «Вид» забезпечує альтернативне виведення деталізованої інформації або повідомлення сумарного змісту [36].

Усі агенти запускають свої власні потоки через конкретні фіксовані проміжки. Всі зв’язки між клієнтами, продавцями

й посередником використовують інтерфейс CiAgent-Events і CiAgentEventListener. Об'єкт аргументу, що передається з CiAgentEvents, є новим об'єктом за назвою BuySellMeswisard і формується за зразком стандартного повідомлення KQML [62]. Для встановлення взаємодії між продавцями й покупцями агенти-посередники використовують онтологію відповідної ПрО для узгодження термінології в їх запитах.

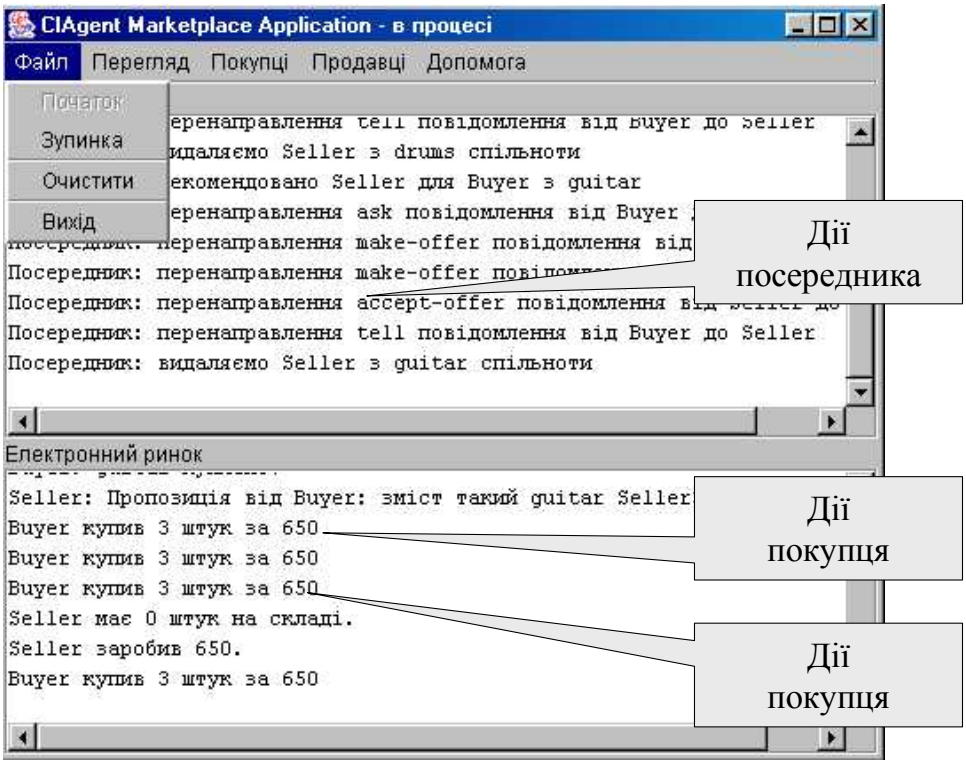



Рис. 3. 6. Вікно переговорів між продавцем і покупцем через посередника

Прикладом використання МАС у е-комерції є інтелектуальна мультиагентна MIMCO для електронної комерції, що забезпечує автоматизацію закупівель на основі агентів покупця й продавця, яким допомагає агент-посередник [36]. Система враховує індивідуальні переваги й персональні властивості покупця та рейтингує надійність та інші важливі для покупця якості продавця.

Агентна парадигма привносить ряд принципово нових властивостей і можливостей в інформаційні технології і являє собою якісно новий рівень її розвитку, який забезпечується розподіленими обчисленнями в гетерогенному інформаційному середовищі Web. Упровадження агентних технологій має підвищити ефективність інтелектуальної діяльності людини, позбавивши її від рутинних операцій. Одним з факторів інтересу до МАС став розвиток інформаційного простору Web і мережі Інтернет, у якій функціонують

розподілені автономні програмні системи, що обробляють гетерогенні інформаційні ресурси.

### 3. 2. Системи доступу до даних на основі онтологій

За останнє десятиліття діяльність консорціуму W3C у сфері Semantic Web і створення стандартів мов опису онтології спричинила нову хвилю активності в розробках інструментарію для систем семантичного доступу до баз даних і нового класу систем баз даних, які називають *системами доступу до даних, що базуються на онтологіях,* – OBDA-системами (Ontology-Based Data Access Systems). У таких системах онтології використовуються як концептуальна схема предметної області й основа користувальницького інтерфейсу для SQL-систем баз даних. Детальний аналіз використання онтологій і DL у таких системах наведено в [89].

На сьогодні вже вдалося створити різноманітні мови опису онтології, які дають змогу досягти прийнятного компромісу між їх виразністю, що залишається достатньою для багатьох застосовних програм, обчислювальною складністю логічного виведення на онтологіях і обробкою запитів даних, збережених у великих базах даних. Наявність таких мов створює передумови для появи індустріальних технологій розробки систем обробки даних на основі онтологій.

#### 3. 2. 1. Дескриптивні логіки в OBDA-системах

OBDA-система розглядається як *надбудова над множиною наявних джерел структурованих даних,* що уможливлює надання користувачам системи концептуального подання даних, логічного виведення й доступу до даних, що містяться в них, у термінах цього подання. При цьому як концептуальну схему інтегрованого джерела даних пропонується використовувати *формальну онтологію предметної області,* відображену засобами дескриптивної логіки сімейства DL-Lite.

Більшість запропонованих у цій області підходів використовує мову опису онтологій, що презентовані дескриптивними логіками, як мову концептуального моделювання предметної області.

Сьогодні для опису таких онтологій широко використовується OWL 2.0 [378] разом з його діалектами (EL, QL і RL) [379], у яких виразні можливості OWL 2.0 обмежуються для досягнення ефективності міркувань у деяких важливих сферах застосування.

У [223] запропоновано сімейство дескриптивних логік DL-Lite, одна з яких – DL-LiteR – є основою профілю OWL 2.0.

Пропонуються також експериментально перевірені на дослідницьких прототипах підходи до створення систем інтеграції даних і OBDA-систем на основі логік зазначеного сімейства.

Термін «*доступ до даних, що базуються на онтологіях»,* був уведений авторами сімейства логік DL-Lite, що запропонували їх не тільки для опису онтології, для чого вони спочатку розроблялися, а й для створення систем інтеграції даних, а також для OBDA-систем.

Важливе припущення полягає в тому, що СКБД може існувати й розроблятися незалежно від онтології, яку потрібно створити й пов'язати з цією системою. Процес створення OBDA-системи починається з розробки онтології ПрО, дані якої зберігаються в SQL-системі бази даних.

### 3. 2. 2. Основні властивості OBDA-систем

Найважливіші можливості OBDA-систем:

- надання розвинених виразних засобів для формального подання БД і специфікації запитів;
- забезпечення декларативності запитів у термінах такого подання;
- наявність механізмів для логічного виведення над онтологіями, а також для обробки сформульованих у термінах онтології запитів даних у СКБД, ураховуючи перетворення запиту, сформульованого в термінах онтології, на запит, специфікований засобами мови SQL;
- здатність здійснювати логічне виведення й обробку запитів даних з прийнятною продуктивністю.

Крім того, використання онтології як концептуальної схеми забезпечує ряд інших корисних можливостей. Зокрема, забезпечується більш абстрактне подання БД, ніж у разі використання традиційних моделей даних, яке не пов'язане з логічною структурою БД. З'являється можливість використовувати в запитах явно не визначені в БД, приховані відношення. Можна перевіряти якість даних, виявляючи їх неповноту чи суперечливість.

Одна з головних функцій OBDA-системи – забезпечення логічного виведення на онтології. Слід ураховувати, що онтологія створюється відповідно до *гіпотези відкритого світу* на відміну від БД, яка будується відповідно до *гіпотези замкнутого світу.* Тому, обробляючи запити в термінах онтології, можна виявити *неповноту чи суперечливість даних* у БД.

Інша важлива вимога – забезпечення *прийнятної складності* виконання логічного виведення, а також обробки сформульованих у термінах онтології запитів до БД великого обсягу, з якими доводиться мати справу в реальних системах.

Через те, що в OBDA-системах передбачається використовувати формальні онтології як концептуальні схеми, потрібно, щоб логіка, яка застосовується до їх подання, давала змогу виражати основні модельні елементи, властиві традиційно використовуваним мовам концептуального моделювання.

Останніми роками для концептуального моделювання (у реалізаціях експериментальних проектів для наукових досліджень) використовують RDF [233] разом з мовою запитів SPARQL [427] для доступу до RDF-специфікації ПрО. Сукупність цих мов визначає повнофункціональну семантичну модель даних, що містить як дескриптивні, так і операційні засоби. Більш широкому використанню мови RDF для концептуального моделювання сприятиме здійснювана в консорціумі W3C розробка нового стандарту опису відображення реляційних даних у RDF [1]. У деяких інших дослідницьких проектах як мову концептуального моделювання використовують також OWL і OWL 2.0.

Дескриптивні логіки дають змогу здійснювати опис поняття *(концепти)* ПрО за допомогою виразів, що будуються з атомарних понять *(унарних предикатів)* і атомарних ролей *(бінарних предикатів)* з використанням конструкторів понять і ролей, які надаються цією логікою.

Важливо, що DL мають чітку формальну семантику й вагомі для практичних застосувань властивості, сформовані на основі компромісу між ступенем виразності й можливістю розв'язання. Завдяки цьому, DL доцільно використовувати як мови подання термінологічних знань, що дають змогу здійснювати логічне виведення для відображеної їх засобами понять ПрО.

БЗ, що використовує DL, складається з двох частин:

- *термінологічної частини* TBox (від англ. Terminological), що аналогічна за своїми функціями схемі БД і відображає поняття (класи) ПрО та їх ролі (бінарні відношення між ними);
- *частини тверджень* ABox (від англ. Assertional), що містить твердження, які відображають конкретну ситуацію в ПрО, встановлюючи властивості її індивідів. Ця частина аналогічна за своїми функціями наповненню бази даних.

Найбільш привабливою сферою застосування дескриптивних логік стало їх використання як формальних мов опису онтологій для семантичних інформаційних систем. На початку 90-х рр. XX ст. для цього був розроблений і реалізований цілий ряд дескриптивних логік: KL-ONE (1985), KRYPTON (1983), LOOM (1987), BACK (1988), K-REP (1991), CLASSIC (1991), KRIS (1991). Пізніше створювалися логіки, які забезпечували більшу виразність і при цьому мали обчислювальну складність, що не перевищує поліноміальну. Це дало

змогу реалізувати механізми логічного виведення (різонери) з прийнятною продуктивністю навіть для складних онтологій: FaCT (1998), RACER (2001), CEL (2005), KAON 2 (2005).

Останніми роками дескриптивні логіки розглядаються як формальні не тільки для опису онтології, а й для створення систем семантичної інтеграції даних, систем баз даних з користувацьким інтерфейсом, що забезпечує доступ до даних на основі онтологій і логічне виведення на онтологіях – OBDA-системах.

Основні результати у сфері OBDA-систем базуються на DL сімейства DL-Lite. Для OWL 2.0 розроблено його діалекти – OWL 2 EL, OWL 2 QL і OWL 2 RL, які також схвалені консорціумом. Ці підмови OWL 2.0 за рахунок обмежень його виразної сили забезпечують певні переваги в аспекті реалізації в деяких конкретних сферах застосування, порівняно з повною мовою.

Найбільший інтерес становить діалект OWL 2 QL, орієнтований на прикладні програми, що мають справу з великим обсягом екземплярів даних (з великим ABox), для яких важливим завданням є ефективне логічне виведення на онтологіях. OBDA-системи належать саме до цієї категорії систем.

Прототипом профілю OWL 2 QL послужила дескриптивна логіка $DL\text{-}Lite_R$ сімейства DL-Lite. У специфікації профілю OWL 2 QL стверджується, що інші, більш виразні логіки сімейства DL-Lite можуть підтримуватися над OWL 2 QL, але можуть знадобитися додаткові обмеження на структуру онтологій, що подаються їхніми засобами.

Метою розробки сімейства логік DL-Lite було забезпечення прийнятної для реальних практичних систем обчислювальної складності логічного виведення на TBox і обробки запитів на ABox великого обсягу, які зводяться до кон'юнктивних запитів чи їх об'єднання, що могли б відображатися в середовищі традиційної SQL-системи бази даних і оброблятися у СКБД. Такі можливості були досягнуті завдяки компромісу між виразною силою логік цього сімейства, з одного боку, і обчислювальною складністю логічного виведення на онтологіях і обробки запитів даних – з іншого.

Логіки сімейства DL-Lite забезпечують поліноміальну складність логічного виведення щодо розміру TBox і складність LogSpace щодо розміру ABox (складність за даними). Важливою рисою цих логік є можливість здійснювати логічне виведення на TBox, незалежно від ABox, і виконувати обробку запитів на ABox, незалежно від TBox. При цьому обробка запитів на ABox зводиться до виконання запитів засобами реляційної SQL-СКБД. Завдяки цьому може здійснюватися оптимізація запитів за допомогою механізмів, які мають для цього традиційні SQL-СКБД.

Авторам сімейства DL-Lite вдалося довести [222], що логіка DL-Lite цього сімейства є максимальним фрагментом мови OWL DL, який має зазначені вище властивості. У процесі використання її як прототипу в профілі OWL 2 QL для узгодження зі стандартом OWL 2.0 було внесено деякі зміни. Вилучено підтримку припущення про унікальність імен (Unique Name Assumption), оскільки воно не підтримується в OWL 2.0. У профілі не виражається твердження ідентифікації (Identification Assertion), що характеризує функціональність ролей і атрибутів, хоча воно може бути представлено в логіці-прототипі. Таке рішення аргументується необхідністю забезпечити для профілю прийнятні характеристики складності логічного виведення. Разом з тим, ці зміни не впливають на обчислювальну складність твердження щодо властивостей ролей і підтримують типізацію даних мови OWL 2.0, якої також немає в логіці-прототипі.

Підкреслимо, що для логік сімейства DL-Lite за умов деяких розширень їх виразних можливостей зберігаються зазначені оцінки складності. Прикладом може слугувати логіка $DL\text{-}Lite_{A,id}$, яка детально розглянута в [222].

Відображення запитів, сформульованих у термінах онтології, у середовищі системи бази даних формується на основі відображення між онтологією і SQL-схемою БД, попередньо визначеного проектувальником конкретної OBDA-системи. Опис цього відображення складається з сукупності тверджень відображень двох видів: типізованих тверджень відображення й тверджень відображення «дані-об'єкти». Типізовані твердження відображення визначають відповідність типів онтології елементам схеми БД. Твердження відображення «дані-об'єкти» відтворюють відображення даних БД в екземплярах концептів, ролей і атрибутів онтології.

Складність зв'язування онтології з системою БД полягає в тому, що онтології базуються на *гіпотезі відкритого світу,* а БД – на *гіпотезі замкнутого світу.* У базі даних може бракувати даних, які запитує користувач у термінах онтології. При цьому в традиційних технологіях БД факт, якого немає в базі даних, вважається помилковим, на відміну від інтерпретації такої ситуації в середовищі онтології, де для такого факту розглядаються три можливості – він може бути істинним, помилковим чи невідомим.

### 3. 2. 3. Інструментальні засоби створення OBDA-систем

Для формування OBDA-систем є ряд інструментальних засобів, створених авторами сімейства дескриптивних логік DL-Lite, а також іншими дослідницькими групами, аналіз яких наведено в [89]. Деякі

з них мають більш широку сферу застосування. Коротко розглянемо деякі з цих інструментів.

*Система QuOnto (QUerying ONTOlogies)* [203] – різонер для дескриптивних логік сімейства DL-Lite. Вона володіє також засобами обробки запитів даних у разі великого обсягу ABox і забезпечує сумісність з OWL 2.0 і SPARQL. Механізм логічного виведення в системі добре оптимізований. Запити даних можуть бути адресовані до «внутрішніх» даних, що підтримуються у власному репозиторії QuOnto, чи до даних, що містяться в SQL-системі бази даних. Система QuOnto має у своєму інтерфейсі ряд поширених СКБД: Oracle, DB2, SQL Server, MySQL, Post-gres тощо, надаючи тим самим змогу презентувати ABox у вигляді звичайної SQL-бази даних. QuOnto реалізується мовою Java, має інтерфейси прикладного програмування, зокрема й розширені DIG-інтерфейси.

DIG-інтерфейс [244] – широко вживана специфікація стандарту інтерфейсу для взаємодії в мережному середовищі різних систем з різонерами на дескриптивних логіках, створена колективом дослідників DL Implementation Group (DIG). Специфікація DIG визначає мову опису онтологій, що базується на дескриптивній логіці SHOIQD-n. Також DIG визначає мінімальний набір повідомлень, якими обмінюються різонери, що мають DIG-інтерфейси (цей набір припускає розширення), та їхні клієнти. DIG можна розглядати як протокол доступу клієнтів до DIG-різонерів. Більшість відомих редакторів онтологій і різонерів підтримують цей інтерфейс. Остання його версія – DIG 2.0 [445].

*OBDA-розширення для DIG* [243] доповнює функціональність DIG-інтерфейсу можливостями специфікації параметрів джерел даних і відображення запитів на онтології в запитах до джерел даних для різонерів OBDA-систем.

*DIG-QuOnto сервер* [393] – реалізація розширення інтерфейсу DIG 1. 1 для системи QuOnto, що дає змогу виконувати логічне виведення за допомогою її різонера на онтологіях і здійснює переписування запитів даних в об’єднання кон’юнктивних SQL-запитів.

*OBDA Plugin for Protégé* [401] – плагін з відкритим кодом для редактора онтології Protégé, який перетворює Protege на інструмент для OBDA-систем. При цьому забезпечується не тільки проектування онтології з відображенням їх у джерела SQL-даних, а й можливості специфікації та виконання логічного виведення на онтологіях, а також об’єднань кон’юнктивних SQL-запитів даних шляхом взаємодії з різонерами, розробленими для архітектури OBDA-систем. Розширений у такий спосіб Protégé може використовуватися як клієнт, наприклад, для DIG-QuOnto.

## 3. 3. Використання онтологій у корпоративних системах

### 3. 3. 1. Аналіз даних в інструментальних бізнес-застосуваннях

Термін «Business Intelligence» використовується порівняно давно, хоча згодом дещо змінився зміст цього поняття. Уперше термін «Business Intelligence» було вжито в публікаціях у IBM JOURNAL в 1958 році, де під ним розумілися автоматична обробка й анотування документів для ефективного пошуку необхідної користувачеві інформації, що розподілено зберігається на різних типах носіїв. Сьогодні створення й упровадження технологій Business Intelligence виокремилося в самостійний напрям індустрії інформаційних технологій (IT) [459], що динамічно розвивається.

Аналітики Gartner почали використовувати термін «Business Intelligence» наприкінці 80-х років XX ст., позначаючи ним орієнтований на користувача процес, що забезпечує доступ до інформації, її дослідження й аналізу, формує інтуїтивне розуміння, сприяючи тим самим поліпшенню неформального ухвалення рішень. У 1996 році з'явилося уточнення: BI – це інструменти для аналізу даних, побудови звітів і запитів, що можуть допомогти бізнес-користувачам опрацювати море даних з тим, щоб синтезувати з них значущу інформацію.

Мета Business Intelligence – перетворення даних на знання, а знань – на бізнес-дії для отримання певної користі [355]. BI – це процес аналізу інформації, вироблення інтуїтивного розуміння для більш ефективного й неформального ухвалення рішень бізнес-користувачами, а також інструменти для здобуття з даних значущої для бізнесу інформації.

Через те, що сучасні застосування, призначені для аналізу й обробки знань у Web, здебільшого використовують онтологічний підхід, виникає потреба в інтеграції засобів Business Intelligence з онтологіями.

Business Intelligence є процесом збору багатоаспектної інформації про досліджуваний предмет. Розроблено програмні застосунки, що забезпечують користувачів можливістю проводити такий процес для відповідей на питання бізнесу й для виявлення важливих тенденцій чи правил у досліджуваній інформації. На жаль, адекватного перекладу Business Intelligence (BI) на сьогодні в українській і російській мовах немає, а ті терміни, що використовуються, тільки сильніше заплутують читача або є не зовсім вдалою калькою з англійської («бізнес-інтелект», «бізнес-аналіз») [40].

Слово «інтелект» однозначно розуміється як здатність людини до мислення. Однак «інтелект» англійською звучить як «intellect», а

слово «intelligence» в англійській мові є багатозначним і в різних контекстах має такі значення[127]:

- здатність дізнаватися й розуміти;
- готовність до розуміння;
- знання, передані чи отримані шляхом навчання, дослідження або досвіду;
- дію або стан у процесі пізнання;
- розвідку, розвідувальні дані.

Можливо, варто звернутися до перекладу слова «Intelligence» у словосполученні «Artificial Intelligence», де під Intelligence розуміється зовсім не «інтелект» і навіть не «інтелігентність», а просто здатність до логічного виведення [96, 126]. Отже, словосполучення «Business Intelligence» можна перекласти як «*засоби логічного виведення, орієнтовані на бізнес-застосунки»*.

Іноді як переклад BI використовують словосполучення «Інтелектуальний аналіз даних», але це не зовсім точно, оскільки в літературі, зазвичай, «Інтелектуальний аналіз даних» використовують як переклад терміна «Data Mining», що так само складно перекласти правильно: Data Mining – це напрям в інформаційних технологіях, що пов’язаний з автоматизованим отриманням знань, неявно наявних в оброблюваній інформації, і базується на інтелектуальному аналізі даних (у буквальному перекладі з англійської – «здобуття даних») [33].

Один з відомих експертів у галузі Data Mining Г. Піатецький-Шапіро, визначив його як процес виявлення в сирих даних раніше невідомої, нетривіальної, практично корисної та доступної інтерпретації знань, необхідних для ухвалення рішень у різних сферах людської діяльності. Тому Business Intelligence можна розглядати як одну з найбільш перспективних сфер застосування технології Data Mining. Але важливо розуміти, що методи Data Mining можуть використовуватися не тільки в бізнес-застосунках (тобто у Business Intelligence), а й для різноманітних наукових досліджень (генетика, хімія, медицина тощо).

Метою Data Mining є виявлення прихованих правил і закономірностей у наборах даних. На відміну від оперативної аналітичної обробки даних (OLAP), у Data Mining задачу формулювання гіпотез розв’язує не людина, а комп’ютер. Найбільш важливі й часто використовувані методи Data Mining – це класифікація та кластеризація. Платформа Business Intelligence на основі таких методів дає змогу користувачам створювати застосунки, що надають можливість організаціям вивчати й розуміти власний бізнес.

Основний напрям сучасних BI-проектів – доставка інформації, але варто також приділити увагу аналізові нових тенденцій і проблемі інтеграції.

### 3. 3. 2. Технології Business Intelligence

Класифікація компанією Gartner програмних засобів BI базується на методі функціональних задач, де програмні продукти кожного класу виконують певний набір функцій або операцій з використанням спеціальних технологій. При цьому Gartner виділяє набір програмних продуктів BI таких класів:

- засоби побудови сховищ даних (*data warehousing*);
- системи оперативної аналітичної обробки (*OLAP*);
- інформаційно-аналітичні системи (*Enterprise Information Systems, EIS*);
- засоби інтелектуального аналізу даних (*data mining*);
- інструменти для виконання запитів і побудови звітів (*query and reporting tools*).
- Аналітична компанія IDC програмні засоби BI поділяє на такі дві категорії:
- засоби кінцевого користувача для звітності, запитів і аналізу (*end-user query, reporting and analysis QRA*);
- розширена аналітика (advanced analytics software).

До першої категорії входять засоби для формування й виконання запитів користувача та звітності, а також для виконання простого аналізу. До них належать: нерегламентовані запити, звіти, засоби багатомірного аналізу, інструментальні панелі. Інструменти багатомірного аналізу охоплюють як OLAP-сервери, так і клієнтські аналітичні інструменти, що забезпечують середовище керування даними для моделювання бізнес-проблем і аналізу бізнес-даних. Основними вендорами цієї категорії є компанії SAP, IBM, Oracle, Microsoft, SAS.

Розширена аналітика передбачає засоби здобуття даних (*data mining*) і статистичне програмне забезпечення (*statistical software*). Інструментальні засоби цієї категорії використовують такі технології, як *нейронні мережі*, *вивід правил методом індукції, кластеризація, виявлення зв'язків у даних*, *прогнозування прихованих тенденцій* (які неможливо розпізнати тільки за рахунок OLAP-засобів). До цієї категорії входить специфічне математичне забезпечення, яке використовується в бібліотеках статистичних алгоритмів і тестах для аналізу даних. Основними вендорами цієї категорії є компанії SAP, IBM, Microsoft, KXEN, TIBCO, FICO, Oracle.

Аналітики з Gartner зазначають, що в системах типу BI обов'язково повинні бути реалізовані три основні категорії функцій:

- можливість інтеграції;
- подання інформації;
- аналіз даних.

Визначаючи платформу BI, Гартнер виділяє 12 її базових можливостей, що можуть бути розділені на три категорії:

1) інтеграція:

- Інфраструктура BI;
- Керування метаданими;
- Розвиток;
- Потоки робіт і колективна робота.

2) доставка інформації:

- Звітність;
- Інформаційні панелі;
- Незапланований запит;
- Інтеграція з Microsoft Office.

3) аналіз:

- OLAP;
- Розширена візуалізація;
- Прогнозоване моделювання й data mining;
- Протоколи результатів.

З погляду онтологічного аналізу, серед цих функцій можна виділити такі важливі особливості.

Для реалізації BI-інфраструктури всі інструменти BI-платформи повинні бути реалізовані «в одному ключі», зокрема й використовувати спільні метадані, єдину об'єктну модель, наскрізну модель безпеки, адміністрування, портальної інтеграції, спільний інструмент виконання запитів, тому саме підхід Semantic Web може стати основою для спільного подання метаданих.

З погляду керування метаданими, усі інструменти в рамках єдиної платформи повинні поєднувати не лише модель метаданих, а й однакові засоби для пошуку, отримання, зберігання, повторного використання та публікації об'єктів метаданих, наприклад таких, як масиви, ієрархії, множини, метрики та елементи оформлення звітів, що теж можна інтегровано реалізовувати на основі RDF, OWL і засобів їх обробки.

BI-платформа повинна надавати як користувачам, так і розробникам набір специфічних інструментів для створення BI-застосувань, які можуть інтегруватися між собою й будувати наскрізні бізнес-процеси, зокрема й для використання в інших (зовнішніх) бізнес-застосуваннях. Але саме онтології забезпечують повторне використання

зовнішніх баз знань у Web без додаткової обробки, гарантуючи уніфіковане й однозначне розуміння їх семантики.

BI-застосування повинні мати можливість на підставі певних бізнес-правил призначати завдання конкретним користувачам і відстежувати їх виконання. Для опису таких правил можна використовувати мови задавання правил на онтологіях.

Більшість BI-інструментів використовує шар метаданих або репозиторій. Бізнес-метадані передбачають визначення даних, які зберігаються в джерелах даних, у термінах предметної області. Вони також можуть містити правила й обчислення, які повинні бути визначені для цього бізнесу. Крім того, є технічні метадані для доступу до фізичних даних. CASE-засоби, реляційні СКБД, засоби добування, перетворення й завантаження даних використовують метадані. При створенні сховища й вітрин даних часто можна автоматично видобувати метадані з джерел даних, але іноді користувачам самим доводиться створювати метадані. Так, можлива складна ситуація з декількома репозиторіями, які є в одній організації. Нестача спільних метаданих для інструментів (через брак стандартів для метаданих) – серйозна проблема для підрозділів IT.

Крім того, у застосуваннях Business Intelligence висувається багато вимог до подання інформації та її аналізу.

*Звітність* – можливість створення форматованих та інтерактивних звітів, з розвиненими механізмами для їх поширення й оновлення. *Контрольні (інформаційні) панелі (dashboards)* – особливий вид звітності, який дає змогу подавати дані в наочному, інтуїтивно зрозумілому вигляді, за допомогою різних шкал, показників, індикаторів тощо. За допомогою таких контрольних панелей користувачі можуть стежити за поточним станом ключових показників і процесів і порівнювати їх з наміченими, цільовими значеннями. *Оперативні (ad hoc) запити* – можливість для відповідних користувачів самостійно (без залучення IT-спеціалістів) створювати й виконувати унікальні, нетипові запити. Для реалізації таких можливостей у BI-платформі повинен бути розвинений семантичний шар, який дає змогу знаходити й отримувати потрібну інформацію з наявних джерел. Крім того, у системі повинні бути відповідні засоби для аудиту цих запитів для перевірки правильності їх виконання.

*Оперативна аналітична обробка даних (OLAP)* – підтримка OLAP-клубів значно прискорює процеси обробки запитів і виконання розрахунків, забезпечуючи аналіз даних у різних зрізах. *Розвинена візуалізація* – максимально наочне подання даних з використанням різних інтерактивних зображень, схем і графіків (замість традиційних таблиць).

Сучасна BI-система містить кілька десятків компонентів. При цьому архітектура BI-системи повинна охоплювати не лише такі параметри, як звітність, аналітика, надання інформації, а й виявлення залежностей у даних, прогнозування, інтеграція, керування якістю даних та інші складні аналітичні функції, серед яких:

- операційна звітність для масового поширення;
- інструментальні засоби для створення нерегламентованих запитів;
- OLAP-інструменти;
- інструментальні панелі й інтерактивний візуальний інтерфейс користувача;
- моніторинг бізнес-операцій (BAM) для звітності за даними в реальному режимі часу й для обробки інформаційних потоків;
- прогнозоване моделювання;
- робочий простір BI, який забезпечує самостійну роботу користувача;
- інструменти пошуку.

Серед зазначених функцій все більшої ваги в сучасних BI-системах набувають функції статистичного аналізу й прогнозної аналітики.

Отже, Business Intelligence і керування знаннями – це дуже близькі напрями досліджень, у взаємодії яких Business Intelligence є реалізацією і практичною перевіркою ефективності різних підходів до подання й аналізу знань.

Business Intelligence і системи керування знаннями (СКЗ) (Knowledge Management System, KMS) сьогодні являють собою синергетичну систему концепцій, технологій і засобів, кожна з яких може грати головну або підпорядковану роль в об’єднаній системі. Такі системи вже функціонують у передових організаціях і забезпечують їх власникам важливі конкурентні переваги [32].

Організаціям у сфері бізнесу, діяльність яких пов’язана з ухваленням рішень на основі обробки значних обсягів інформації, доцільно реалізувати свої стратегії Business Intelligence разом з KMS (Business Intelligence 2. 0 (3.0), Big Data) на основі кращих світових практик, які необхідно адаптувати до особливостей свого бізнесу. Це важлива необхідна умова їх виживання й підвищення економічної ефективності у високо конкурентних і високо прибуткових середовищах.

### 3. 3. 3. Системи керування знаннями й Business Intelligence

Завдання СКЗ – накопичувати не розрізнену інформацію, а структуровані, формалізовані знання – закономірності й принципи, які

дають змогу виконувати реальні виробничі завдання. Можна розглядати керування знаннями як комплексну організаційно-технічну діяльність, спрямовану на підвищення ефективності використання знання в бізнес-процесах організації (підприємства) [40].

При цьому знання класифікуються й розподіляються за категоріями відповідно до визначеної онтології, яка розвивається, структурованих і слабко структурованих баз даних і баз знань. Основна мета СКЗ – зробити знання доступними й повторно використовуваними на рівні всієї корпорації.

Керування знаннями – це сукупність процесів і технологій для виявлення, створення, поширення, обробки, зберігання й надання знань для використання.

У середині 90-х років минулого століття у великих корпораціях, де проблеми обробки інформації набули особливої гостроти й критичності, стало очевидним, що основним вузьким місцем є обробка знань, накопичених фахівцями компанії, оскільки саме знання забезпечують перевагу перед конкурентами. Так з'явився термін Knowledge Management – сукупність процесів, які керують створенням, поширенням, обробкою й використанням інформації всередині підприємства.

Під керуванням знаннями загалом розуміється дисципліна, яка забезпечує інтегрований підхід до створення, збирання, організації, доступу й використання інформаційних ресурсів організації. Ці ресурси містять у собі корпоративні бази даних, текстову інформацію, зокрема таку, як документи, що відображують правила й процедури, і, що найважливіше, неявні знання й досвід співробітників організації.

Керування знаннями – це формальний процес, який полягає в оцінці організаційних процедур, людей і технологій і в створенні системи, яка використовує взаємозв'язки між цими компонентами для надання потрібної інформації потрібним людям у потрібний час, що сприяє підвищенню продуктивності.

Є три основні компоненти керування знаннями:

А) *люди*, які здобувають, генерують і передають знання;

Б) *процеси*, які використовуються для поширення знань;

В) *технології*, які забезпечують ефективну роботу людей і процесів.

Функції систем керування знаннями:

1. Збирання знань:
- Доступ до різнорідних джерел інформації;
- Здобуття знань;
- Виділення структурованої інформації (контент-аналіз тощо);
- Виділення зв'язків між документами;

- Попередній аналіз (анотування, виділення імен, дат тощо);
- Кластеризація й рубрикація;
- Створення рубрик за запитом або набором еталонних документів;
- Автоматичне створення рубрикатора;
- Накопичення знань користувачів;
- Зворотний зв'язок з користувачами.

2. Зберігання й обробка знань:
- Зберігання знань;
- Структурування знань у різних розрізах;
- Модифікація знань.

3. Доставка знань:
- Перегляд інформації без пошуку;
- Пошук інформації в текстах (повнотекстовий, атрибутивний, за зразком) і базах даних;
- Оповіщення користувачів про зміни;
- Зв'язування документів і експертів.

Підкреслимо, що Knowledge Management більше орієнтована на аналіз неструктурованої або слабкоструктурованої інформації (наприклад, HTML), яка не є предметом аналізу BI-інструментів. KM забезпечує категоризацію, аналіз і семантичну обробку текстів, розширений пошук інформації тощо, а технологія BI призначена для аналізу фактографічної структурованої (бази даних, пласкі файли та інші ODBC або OLE DB-джерела даних) і слабкоструктурованої інформації (наприклад, XML).

На жаль, у наявних системах слабко використовуються сучасні стандарти подання метаданих, зокрема такі, як RDF.

### 3. 3. 4. Тенденції розвитку засобів Business Intelligence

Сьогодні системи типу Business Intelligence розвиваються за такими основними напрямами:

*«Хмарний» (cloud) BI або SaaS (Software as a Service)* BI – BI-вендори надають свої власні майданчики для побудови BI-застосувань, здійснюють адміністрування системи й, за необхідності, її масштабування. Ця схема дає змогу зменшити операційні витрати на керування BI-інфраструктурою, передати ці функції кваліфікованішому персоналу.

*Open-source BI* – на думку аналітиків Forrester, BI-системи з відкритим вихідним кодом мають ті ж переваги, що й інші системи з відкритим кодом, починаючи від нижчих початкових витрат до гнучкішого супроводу й можливостей інтеграції. Прикладами таких систем є, зокрема, продукти Actuate, Jaspersoft, SpagoBI і Pentaho, які мають функціонал бізнес-аналітики. Разом з тим, такі системи ще

не мають такого функціоналу як традиційні BI-продукти, а також відповідних можливостей щодо масштабування й безпеки.

*Нереляційні in-memory BI-інструменти* – це BI-продукти, яким взагалі не потрібна СКБД: вони використовують власний механізм обробки інформації, що працює за принципом in-memory. Він передбачає завантаження всіх даних в оперативну пам'ять і подальше виконання аналітичних запитів на них без потреби звернення до дискової підсистеми. Представниками цієї групи є такі продукти, як QlikView, Tibco Spotfire, Cognos TM1.

*Операційний BI* – використання BI-інструментів не лише для стратегічного або тактичного менеджменту, а й для цілей операційної діяльності підприємства.

*Використання BI в SOA архітектурі*. BI-система організується як набір сервісів, які можуть викликатися ззовні як безпосередньо користувачами, так і зовнішніми застосуваннями, наприклад, обліковими системами, системами електронного документообігу або ж системами workflow management. За сервісною архітектурою можуть бути побудовані не лише засоби візуалізації (генерування звітів), а й робота ETL-процедур. Така організація відкриває гарні можливості для створення BI-застосувань, що функціонують у складній, географічно розподіленій IT-інфраструктурі, у режимі реального часу або близькому до нього. У цьому напрямі найактивніше працюють компанії IBM, SAP, Oracle.

Розглянемо останнє питання детальніше.

### 3. 3. 5. Використання Web-сервісів у Business Intelligence

Останнім часом багато компаній почали презентувати свої сховища даних і BI-системи як Web-сервіси для використання іншими застосуваннями й процесами, пов'язаними сервісно-орієнтованою архітектурою (SOA) або ПЗ проміжного рівня, таким, як корпоративна сервісна шина (enterprise service bus – ESB).

Технологія Web-сервісів спрямована на інтеграцію застосунків підприємства. Базуючись на використанні точної інформації в реальному масштабі часу, що надходить з усіх куточків підприємства, ця технологія покликана допомогти Business Intelligence досягти бажаних результатів, підсилити його потенціал для правильного ухвалення рішень. Саме тому її використання в продуктах BI є дуже ефективним.

Багато хто ототожнює SOA з технологією Web-сервісів або певними інформаційними системами, наприклад, системами для керування бізнес-процесами (Business Process Management, BPM). Але SOA – це не програмний продукт або технологія, а ідеологія

інформатизації бізнесу, що базується на процесному підході й методології керування бізнес-процесами BPM [414].

Є багато визначень поняття «сервіс». Однак для SOA найбільш точним є таке: сервіс – це закінчений функціональний компонент, що багаторазово може використовуватися в різних бізнес-процесах [461].

Web-сервіси розгортаються в межах підприємства, щоб активувати більш ефективну інтеграцію операційних процесів. Інтеграція застосувань (Business-To-Business, B2B) усередині Extranet на основі використання Web-сервісів сьогодні активно досліджується. Більшість застосувань BI нині є Web-базованими й технологія Web-сервісів розширює сферу дій BI поза мережами Extranet. Web-сервіси сьогодні дають змогу численним корпораціям обмінюватися інформацією BI через стандартні протоколи зв'язку між програмами й централізувати сервіси довідників без попереднього встановлення з'єднань «точка-точка», необхідних BI від мереж Extranet. Web-сервіси дають можливість компонентам BI-застосувань багаторазово використовувати їхні функціональні можливості й презентувати їх як сервіси для інших застосувань, використовуючи стандартний Web-базований протокол. Це дало б змогу компонентам, написаним різними мовами й на різних платформах, динамічно зв'язуватися один з одним. Отже, використовуючи методи Web-сервісів, не складно, наприклад, програмі на Java, що виконується на платформі UNIX, зв'язатися з програмою, написаною в коді Microsoft, що виконується на платформі Windows.

Загалом SOA пропонує компаніям численні переваги як з погляду бізнесу, так і інформаційних технологій. SOA дає змогу зв'язати різні системи, наявні на підприємстві, та формалізувати бізнес-процеси їх взаємодії. У центрі уваги SOA перебувають не дані, а сервіси, які є бізнес-функціями, призначеними для забезпечення узгодженої роботи більшості частин, що складаються з множини застосувань.

З іншого боку, у фокусі BI знаходяться дані, які потрібно обробляти й відображати. Технологія SOA має дуже гарний потенціал щодо BI-систем. Вона дає змогу забезпечити прозорий доступ до інформації, зібраної у «віртуальне» сховище даних з різних операційних і аналітичних джерел у реальному часі. Крім того, використання сервісів як основи для побудови BI-системи уможливлює подолання багатьох труднощів, пов'язаних з клієнт-серверною архітектурою. Так, наприклад, стає можливим управляти подіями, виконувати багато завдань у режимі реального часу, автоматизувати аналіз і обробку інформації, робити легкомасштабовані й «інтегровані» системи.

Ці функціональні можливості Web-сервісів дають змогу підприємствам з'єднати розриви між різними застосуваннями

й системами. Вони можуть використовуватися для того, щоб забезпечити доступ до бізнес-процесів застосуванням, BI-кубам, звітам, запитам і функціям інтеграції даних, базам даних тощо.

Як відомо, Web-сервіси базуються на дуже поширених і відкритих протоколах: HTTP, XML, UDDI, WSDL і SOAP. Саме ці стандарти реалізують основні вимоги SOA: сервіс, по-перше, повинен піддаватися динамічному виявленню й виклику (UDDI, WSDL та SOAP) і, по-друге, повинен використовуватися незалежно від платформи інтерфейсу (XML). Нарешті, HTTP забезпечує функціональну сумісність.

### 3. 3. 6. Business Intelligence 2.0 і Semantic Web

Сучасний етап розвитку інформатики сприяв виникненню нового покоління BI, який отримав назву BI 2.0 (за аналогією з Web 2.0). BI 2.0 – це радше загальна парадигма, абстрактна концепція, а не конкретні застосування. Якщо в традиційному BI аналізована інформація залучалася до рішень, орієнтованих на бізнес (оперативно обробляється інформація й подається в зручній для користувача формі), то у BI 2.0 інформація допомагає ухвалювати рішення ще до того, як відбулася деяка подія. Можна розглядати BI 2.0 як нове покоління програмного забезпечення для BI, яке швидше за все проактивне, ніж реактивне, тобто певні кроки виконуються ще до того, як надійде відповідна інформація [32]. BI 2.0 значно більше орієнтується на інтероперабельне подання знань, використання онтологій і семантичних Web-сервісів.

Ця зміна підтримується сервіс-орієнтованою архітектурою (СОА), яка забезпечує гнучке, компоноване й адаптивне ПЗ проміжного шару. BI 2.0 забезпечує зворотний зв’язок ухвалених рішень і інформації, яка надходить у режимі реального часу.

Основними особливостями технологій BI 2.0 є засоби контролю за продуктивністю, засоби інтегрованого планування й прогнозування, які вбудовуються в операції, аналітичні портали, робочі простори для колективної роботи, а також інтеграція з Microsoft Office, розширені технології пошуку й візуалізації даних, реалізація функцій BI як сервісів, стандартизація й розширення застосування відкритих рішень, розвиток семантичних методів, засоби роботи зі слабкоструктурованими даними й спеціалізовані високопродуктивні програмно-апаратні рішення не тільки для сховищ даних і задач BI.

BI 2.0 забезпечує більш високий рівень абстракції та працює з семантичною моделлю даних. Це дає змогу здійснювати пошук потрібної користувачеві інформації на семантичному рівні, використовуючи її метаопис [33]. Для підтримки цього використовуються такі технології Semantic Web:

- Resource Description Framework (RDF);
- Web Ontology Language (OWL);
- SPARQL (SQL like query language for RDF).

Це забезпечує новий рівень взаємодії.

Можна виділити наступні ключові фактори, пов’язані з переходом BI на новий рівень:

- поширення BI за межі окремого підприємства шляхом об'єднання оперативних і аналітичних застосувань;
- підтримка аналітичної культури й автоматизація ухвалення рішень;
- готовність до взаємодії з новим поколінням працівників, яке, на відміну від своїх попередників, залучає технологію BI до свого персонального життя;
- застосування семантичних технологій, які використовують метадані та онтології для інтеграції взаємодії між людьми та процесами;
- адаптація стандартів і технологій Web 2.0 для спільної роботи, масштабування й прискорення створення BI-продуктів і застосувань;
- досягнення високої продуктивності BI.

Найбільш характерними явищами, які очікуються у BI, найближчим часом стануть:

- всеосяжна BI-технологія (Pervasive BI) – розширення використання Business Intelligence у всьому бізнес-співтоваристві за рахунок застосування цієї технології серед співробітників різних рівнів;
- доступність BI для мас – розширення діапазону BI-можливостей, доступних малому й середньому бізнесу й невеликим IT-відділам;
- рольова BI-технологія – аналітичні задачі, націлені на завдання та інтереси конкретної аудиторії. Показники, метрики, тенденції, інструментальні й оціночні панелі, призначені для конкретних функцій (дослідження, маркетинг, продажі, фінанси тощо) на бізнес-рівні (стратегічному, тактичному, операційному);
- дослідницька аналітика – інтерактивні, дослідницькі процеси виявлення нових зв’язків і явищ, коли кожне нове питання породжує прикладні питання;
- динамічна аналітика – забезпечення швидкого відгуку на різні ситуації, де є необхідним аналіз. Йдеться про можливість швидкого й ефективного адаптування до мінливих обставин як у бізнес-питаннях, так і в технічних.

Основні риси Business Intelligence 2.0:

- оперативність і наскрізна взаємодія;
- вбудовування в процеси;
- якість обслуговування;
- доступність.

BI 2.0 містить у собі декілька важливих нових концепцій, які торкаються використання інформації в бізнесі, організаціях і урядових структурах. Цей термін по суті своїй пов'язаний з BI в реальному часі, з технологією, керованою зовнішніми подіями, але основна ідея криється в застосуванні цих методів до бізнес-процесів.

Засоби BI нового покоління будуть керуватися подіями й зможуть виявляти аномальні ситуації й проблеми, що виникають, для чого в них будуть вбудовані відповідні інтелектуальні й адаптивні можливості. Джерелом даних для аналізу слугуватимуть не тільки сховища, а й дані найрізноманітніших внутрішніх і зовнішніх для компанії джерел і програмних агентів. У технологіях нового покоління переважатимуть колективні методи роботи в мережі з використанням віртуальних форумів, блогів, Wiki, соціальних мереж.

BI 2.0 має ряд властивостей:

- *Керується подіями.* Автоматизовані процеси керуються подіями, тому очевидно, що для розробки більш гнучких процесів потрібно аналізувати й інтерпретувати події.
- *Виконується в реальному часі.* Інакше не можна реалізувати можливості BI як етап процесу й навіть більше – не вдасться автоматизувати операції. Для порівняння: процеси пакетної обробки містять звіти про ефективність процесу, але не можуть бути частиною самого процесу, за винятком тих випадків, коли час не відіграє вирішальної ролі. Будь-яке застосування, що стосується торгівлі, динамічного ціноутворення, оцінки попиту, безпеки, оцінки ризиків, виявленню шахрайства, поповнення складу й будь-якої взаємодії з клієнтом, це процес, який залежить від часу, а отже, потребує обробки в реальному часі.
- *Автоматизований аналіз.* Щоб автоматизувати щоденний процес прийняття рішень, організаціям необхідно не просто подати дані у вигляді інструментальної панелі або звіту. Завдання перетворити дані в реальному часі на щось дієве, тобто автоматично й динамічно їх інтерпретувати. Продукти BI 2.0 повинні як основу використовувати деяку норму як на детальному, так і на агрегованому рівнях і порівнювати автоматично конкретні події з цією нормою.
- *Далекоглядність.* Щоб зрозуміти, як певна подія вплине на потреби організації, необхідно мати деяку далекоглядність. Щоб відповісти на запитання: «чи надійде замовлення вчасно?»,

«чи відмовить система сьогодні?», потрібно вміти робити прогнози. Ця можливість надає особливого значення операційним відділам, які повинні уявляти собі перспективу зміни ефективності своєї роботи протягом дня, тижня або місяця.

- *Орієнтація на процеси.* Продукти BI 2.0 повинні бути орієнтовані на процеси. Це не означає, що процеси моделюються за допомогою інструмента керування. Дії можна оптимізувати, виходячи з результатів конкретного процесу, хоча він може й не мати точного визначення.
- *Масштабованість.* Масштабованість є наріжним каменем BI 2.0. Потоки подій можуть бути непередбачуваними й дуже інтенсивними. Наприклад, якщо в роздрібній торгівлі розробляється застосування для оцінки попиту, для відстеження продажу найпопулярніших товарів, то виникає така ситуація: у компанії може бути, приміром, 30 тис. товарів, які продаються в 1 тис. магазинів. Унаслідок цього утворюється 30 млн. комбінацій «магазин-товар», які потрібно відстежувати. За день може бути продано до 10 млн. товарів. Такий масштаб пересічна ситуація для BI 2.0. Фактично, навіть така масштабованість потребує застосувань нового класу, які в традиційній технології BI були недоступними.

Сьогодні спостерігається перехід до більш комплексної, глибокої аналітики з використанням інтелектуального аналізу даних, статистичних методів і технологій, що підтримують аналіз у стилі ad hoc.

Ці технології підвищили рівень знань аналітиків-прогнозистів і спроможність до ухвалення рішень, що дає змогу застосовувати їх в оперативних інформаційних потоках. Зараз компанії здатні до оперативної чи вчасної бізнес-аналітики, що уможливлює доступ передових працівників до результатів такої аналітики й використання цих результатів, у поєднанні з оперативними даними, у повсякденній діяльності.

Засоби BI нового покоління дають змогу керуватися подіями й здатні виявляти аномальні ситуації та проблеми, що виникають, завдяки вбудованим у них відповідних інтелектуальних і адаптивних можливостей. Як джерела даних для аналізу можуть слугувати не тільки сховища, а й дані найрізноманітніших внутрішніх і зовнішніх для компанії джерел і програмних агентів (що безпосередньо корелюється з таким напрямом Semantic Web, як Linked Data).

Linked Data Project вибудовує загальновживані ієрархії класів, укладає словники загальних і власних імен, а також допомагає

власникам масивів даних поєднувати їх бази знань в одну, зв'язну систему знань. Часто учасники проекту поєднують уже наявні великі бази, допомагаючи один одному встановити відповідність між ідентифікаторами однієї й тієї ж речі в різних базах. Якщо якась база знань заповнюється деякою автоматичною процедурою, то ця процедура може почати використовувати імена, що вже використовуються іншими учасниками, а якщо вона пишеться вручну, тоді автори можуть заглядати в Dbpedia, GeoNames, WordNet [176] чи Yago як у словник, одночасно з цим перетворюючи свої дані на «замітки на полях» великої енциклопедії. Це корисно як авторам невеликих баз, так і укладачам великих словників. Наявність перехресних зв'язків між різними базами знань не тільки робить ці бази більш корисними – часто самі знання очищаються від помилок. Дуже швидко зростає не лише сумарний обсяг баз проекту – одночасно збільшується й кількість SPARQL-запитів до бази, що створює багато проблем.

У технологіях нового покоління Business Intelligence переважатимуть колективні методи роботи в мережі з використанням онтологій, віртуальних форумів, блогів, Wiki, соціальних мереж. Реалізація застосунків спиратиметься на SOA з застосуванням загальнодоступних рішень і активних елементів Web 2.0 (наприклад, AJAX), а також функціонально багатого зовнішнього інтерфейсу.

BI 2.0 – це *нове покоління програмного забезпечення,* що допомагає ухвалювати рішення ще до того, як відбулася певна подія, використовуючи сервіс-орієнтовану архітектуру й технології Semantic Web для обробки й аналізу розподілених даних і знань. BI є таким напрямком IT, що швидко розвивається й широко застосовується, тому його інтеграція з Semantic Web впливає як на розвиток BI-платформ, так і на інтерес до практичного застосування таких компонентів Semantic Web, як метадані, онтології, семантичні Web-сервіси та інтелектуальні програмні агенти.

Сервіси Semantic Web, для опису яких, крім стандартних засобів COA, використовується OWL-S, є основою для побудови систем Business Intelligence 2.0. Такі сервіси можуть інтероперабельно використовуватися різними інтелектуальними агентами, що підтримують функції сучасних систем Business Intelligence.

### 3. 4. Використання онтологічного аналізу в застосуваннях Semantic Grid

Сучасний етап розвитку IT безпосередньо пов'язаний з розвитком інтелектуальних мереж [34], які забезпечують пошук інформації

в розподіленому середовищі на семантичному рівні й надають користувачам відповідні моделі та сервіси [37].

Останнім часом виникла потреба в Grid наступного покоління (NGG – Next Generation Grids) і орієнтованих на служби утиліт знань (SOKU) [315], у яких технології проміжного програмного забезпечення стають більш інтелектуальними й автономними. Класифікація новітніх Grid-систем дає змогу більш чітко визначити їх важливі розходження й додаткові можливості, надані користувачам. Для категоризації традиційних Grid використовуються такі характеристики, як забезпечувані Grid-системами типи рішень і розмір організацій, що вони обслуговують. Для новітніх Grid-систем доцільно враховувати також доступність, інтерактивність, здатність до налаштування й керованість. Залежно від того, як розставляються пріоритети в Grid-дослідженнях між проникністю (властивість, що залежить від таких атрибутів, як доступність, спрямованість на користувача й динамічна взаємодія) і здатністю до самоуправління, можна виділити такі категорії новітніх Grid, як Grid доступу, Grid, що конфігуруються, інтерактивні Grid, Grid знань і керовані Grid [359].

### 3. 4. 1. Напрями інтелектуалізації сучасних Grid

Grid знань є розширенням стандартного Grid, де дані, ресурси та служби супроводжуються чітким описом змісту, що анотований семантичними метаданими настільки, що як машина, так і людина можуть ними оперувати. Метою подібного розширення є створення такої інфраструктури, яка б призначалася не тільки для обчислень і керування даними, а й проникала б до керованої знаннями інфраструктури. Прикладами проектів Grid знань можуть слугувати OntoGrid (www.ontogrid.net/ontogrid/index.jsp), InteliGrid(www.inteligrid.com)і K-Wf Grid (www.kwfgrid.eu). Над реалізацією Grid знань працює кілька груп, зокрема й група Semantic Grid Group (www.ogf.org/gf/group_info/view.php?group=sem-rg) від Відкритого Grid-форуму (Open Grid Forum, www.ogf.org).

Сьогодні є кілька принципово відмінних визначень такого терміна, як Knowledge Grid, що реалізуються в рамках досить успішних проектів.

У [478] Knowledge Grid – це інтелектуальне людино-машинне середовище, що дає змогу людям і програмним агентам ефективно здійснювати керування такими ресурсами, як знання (збирати їх, зберігати, забезпечувати доступ за допомогою відповідних сервісів). При цьому Knowledge Grid не зводиться до інтеграції Grid з такими засобами ШІ, як методи здобуття знань з розподілених даних чи керування базами знань. У [439] Knowledge Grid – це середовище для підтримки співробітництва вчених, яким необхідно робити

інтелектуальний аналіз даних, що зберігаються в різних дослідницьких центрах, а також використовувати системи керування знаннями, що працюють з різними сховищами даних, які базуються на різноманітних сервісах здобуття знань.

Напрями інтелектуалізації сучасних Grid відрізняються саме тим, для чого в них використовуються знання та їх обробка – для більш ефективного функціонування інфраструктури самого Grid (наприклад, метадані про ресурси Grid). До того ж самі знання є основним об'єктом і продуктом функціонування Grid. Крім того, значні розходження спостерігаємо у формалізмах, що використовуються для подання знань, і методах їх здобуття та обробки.

Можна узагальнити всі ці підходи до побудови інтелектуалізованої надбудови над наявною інфраструктурою Grid:

- застосовуються знання як з певної ПрО, так і з різних ресурсів Grid, презентовані у вигляді онтологій для забезпечення їх інтероперабельності й повторного використання;
- в обробці й пошуку знання використовуються методи та засоби штучного інтелекту (data mining);
- знання виділяються як окремий клас ресурсів Grid;
- використовуються стандарти й технології, розроблені для трансформації Web у єдину базу знань (проект Semantic Web).

Процес створення семантично-орієнтованих Grid-застосувань охоплює такі етапи:

- Створення онтології, що відображає ПрО, з якою буде працювати застосування.
- Розробка Web-сервісів, призначених для доступу до онтології (це базові сервіси й сервіси даних, які повинні забезпечувати додавання, зміну, видалення й пошук метаданих і знань).
- Розробка Web-сервісу, призначеного для здійснення логічного виведення на онтології, тобто одержання знань, поданих в онтології неявно.
- Розробка прикладних Web-сервісів, що використовують знання в реалізації своєї функціональності (сервіси знань).

Унаслідок цього одержуємо більш гнучке застосування, простіше у конфігуруванні й більш пристосоване для повторного використання й композиції сервісів, ніж звичайні Grid-застосування. Це дасть змогу мінімізувати витрати часу на розробку й технічну підтримку Grid-застосувань.

Використання онтологій дає змогу ефективно й автоматизовано створювати різноманітні віртуальні спільноти – віртуальні співтовариства, віртуальні дослідницькі середовища – для виконання

найрізноманітніших дослідницьких завдань. Навіть більше, використання онтологічного аналізу дає змогу підвищити ефективність використання вже наявних віртуальних дослідницьких проектів. Прикладом цього є Semantic Grid – розвиток третього етапу Grid-технології, пов'язаний з переходом до взаємодії на семантичному рівні й упровадженням стандартів і методик проекту Semantic Web.

Semantic Grid співвідноситься з Semantic Web так само, як Grid – з Web. Semantic Grid характеризується як відкрита система з високим рівнем автоматизації, яка підтримує гнучку співпрацю й обчислення на глобальному рівні, у якому використовується аналіз ресурсів і сервісів на рівні їх семантики.

Метадані, тобто дані про дані (у Grid це дані про користувачів, ресурси та сервіси тощо), є критичним елементом у дослідницькій інфраструктурі. Ефективне використання метаданих потребує спільної презентації знань як базового словника, з якого можна отримувати твердження метаданих. Онтологія, як спільне й загальне розуміння домену, може це забезпечувати. Тому саме онтології можуть використовуватися для подання семантики метаданих.

### 3. 4. 2. Віртуальні організації та віртуальні дослідницькі середовища Grid

У термінології Grid сукупність осіб і організацій, що виконують спільно ту чи іншу загальну задачу й надають одна одній свої ресурси, називається *віртуальною організацією* (ВО) [306]. Наприклад, віртуальною організацією може бути сукупність усіх людей, що беруть участь у якому-небудь науковому проекті. Віртуальні організації можуть розрізнятися за складом, масштабом, часом існування, родом діяльності, цілями, відносинами між учасниками (довірчі, недовірчі) тощо. Склад віртуальних організацій здатний динамічно змінюватися.

Інфраструктура Grid заснована, з одного боку, на наданні ресурсів у загальне користування, а з іншого – на використанні привселюдно доступних ресурсів. З огляду на це, ключовим поняттям інфраструктури Grid є віртуальна організація, у якій кооперуються як споживачі, так і власники ресурсів. Кожна ВО має у своєму розпорядженні визначену кількість ресурсів, що надані зареєстрованими в ній власниками (деякі ресурси можуть одночасно належати декільком ВО). Кожна ВО самостійно встановлює правила роботи для своїх учасників, виходячи з дотримання балансу між потребами користувачів і наявним обсягом ресурсів, тому користувач повинний обґрунтувати своє бажання працювати з Grid-системою й дістати згоду керівних органів ВО.

### 3. 4. 3. Semantic Grid

Ще одним напрямом інтелектуалізації Grid є Semantic Grid. Для створення нових застосувань Grid бажано мати можливість багаторазово використовувати наявні компоненти та інформаційні ресурси й збирати ці компоненти деяким гнучким способом. Таким чином, підсилилася увага до *сервіс-орієнтованої моделі обчислень* і *використання метаданих,* що стали двома ключовими характеристиками систем Grid третього покоління. Вони тісно пов'язані між собою, тому що сервісно-орієнтований підхід має безпосереднє значення для структури інформації. Гнучке збирання ресурсів Grid у застосування потребує семантичної інформації про функціональні можливості, доступність та інтерфейси різних компонентів, і ця інформація повинна мати узгоджене тлумачення, яке може бути оброблено машиною, що пов'язано з використанням онтологічного підходу [210].

Web усе більше стає інфраструктурою для розподілених застосувань, де радше відбувається обмін інформацією між програмами, ніж читання її людиною. Такий інформаційний обмін забезпечується сімейством рекомендацій XML від W3C. Поняття Grid і Semantic Grid мають багато спільного, але можуть розрізнятися в акцентах: Grid традиційно зосереджена на обчисленнях, тоді як Semantic Grid – на виведенні, доказі й перевірці. Третє покоління Grid сьогодні називають Semantic Grid, тобто Grid, заснована на використанні метаданих і онтологій, де інформація подається, запам'ятовується, стає доступною та спільно використовуваною й підтримується. Така інформація розуміється як дані, що мають семантику. Передбачається, що наступне покоління буде мати справу вже зі *знаннями*, що здобуваються, використовуються, надаються, публікуються й підтримуються з тим, щоб допомогти Е-вченим досягти їх специфічних цілей. Знання розуміється як інформація, що застосовується для досягнення мети, розв'язання проблеми чи ухвалення рішення. Semantic Grid охоплює всі три концептуальні шари Grid: знання, інформація й обчислення/дані. Ці додаткові шари, зрештою, забезпечать доступ до глобально розподілених гетерогенних ресурсів.

Метафора Grid інтуїтивно підвищує погляд на структуру е-науки як на набір послуг, які забезпечуються окремими особами або інституціями для споживання іншими. Пов'язуючи це з тим фактом, що багато показників діяльності в дослідженнях і стандартизації пов'язуються подібним чином, можна адаптувати сервіс-орієнтований підхід до Grid. Цей підхід базується на записі різних сутностей (*entities),*

які забезпечують послуги *(services)* одна одної через різноманітні форми контракту *(contract)* або домовленості на рівні послуг (рис. 3. 7).

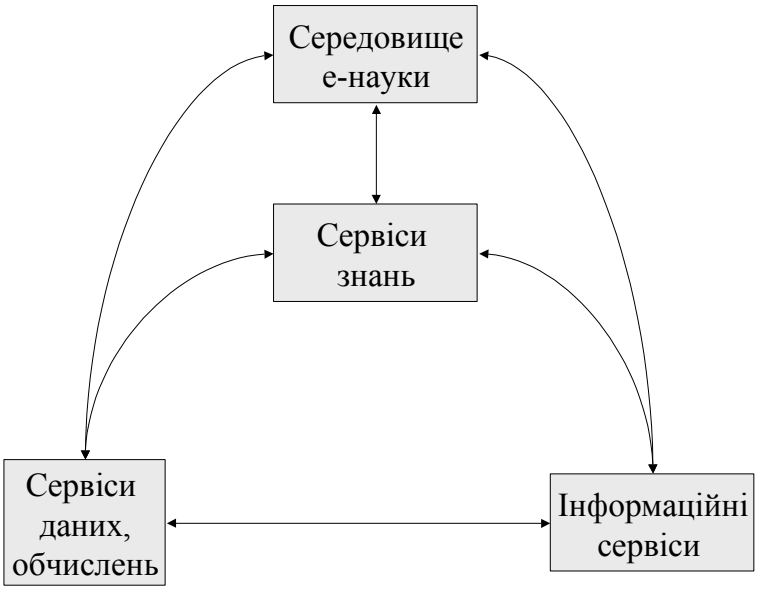



Рис. 3. 7. Трирівнева архітектура сервісів у Semantic Grid

Призначення Semantic Grid – допомогти в інтеграції та обміні гетерогенними даними. Більш формально, Semantic Grid – це середовище, у якому семантично відображені функції його ресурсів і сервісів з використанням метаданих. Якщо інформація та сервіси мають чітко визначені значення, то комп'ютери й користувачі можуть взаємодіяти значно ефективніше. Але для цього потрібно визначити семантику ресурсів Grid так, щоб комп'ютери могли інтерпретувати ці визначення, розуміти їх функції та мати можливість взаємодіяти з ними. Для цього потрібно створювати й використовувати онтології відповідних доменів і пов'язувати семантику ресурсів і сервісів Grid саме з їх поняттями.

Крім того, такі онтології дають змогу створювати спільну термінологічну базу в різноманітних віртуальних організаціях і віртуальних дослідницьких середовищах, платформою для функціювання яких і є Grid.

Як формальна структура, що використовується в Semantic Web, онтологія може забезпечити метаінформацію, яка відображає дані семантично. Це надає спільне розуміння домену інтересів для підтримки комунікацій між людьми та програмними агентами. Онтологія може використовуватися для пошуку потрібних ресурсів і даних у Grid на рівні семантики.

Третє покоління Grid акцентується на розподіленому співробітництві. Нові форми Grid для наукових досліджень – це живі інформаційні системи. Один з аспектів спільної роботи використовує

ідею «колабораторії», або центру без стін, де національні дослідники можуть виконувати дослідження безвідносно до географічного місця розташування шляхом взаємодії з колегами, спільно використовуючи інструментарій, дані й обчислювальний ресурс і звертаючись за інформацією до цифрових бібліотек. Таке подання фактично перетворює інформаційні прилади, якими є комп'ютери й мережна інфраструктура, на лабораторне устаткування, що може, наприклад, містити електронні журнали й інші портативні пристрої.

У Web нині дуже поширена подача інформації, що є могутнім стимулом для створення кіл спілкування. Однак така парадигма взаємодії, власне кажучи, реалізує принцип «один в одного», підкріплений службами електронної пошти й груп новин, що також підтримують *асинхронне співробітництво*. Незважаючи на це, основна інфраструктура Інтернету, однак, цілком здатна до підтримки *живих* (у реальному масштабі часу) інформаційних послуг і *синхронного співробітництва:* живі дані від експериментального устаткування; живе відео за допомогою Web-камер через однобічну широкомовну мережу чи передачі; відеоконференції; чати; системи моментального інформування; багатокористувацькі діалоги та ігри; колаборативні віртуальні середовища. Всі вони відіграють визначну роль у підтримці Е-науки, *безпосередньо зв'язуючи людей* поза процесами інфраструктури. Зокрема, вони підтримують розширення Е-науки в напрям до нових співтовариств, переборюючи встановлені організаційні та географічні кордони. Акцентується на забезпеченні розподіленого співробітництва, що все більше й більше охоплює інтелектуальні сфери роботи Е-вчених. Ця сфера досліджень підпадає під напрям «Перспективні колаборативні середовища» Робочої групи Глобального форуму Grid (ACI Grid), у якому розглядаються середовища колективної роботи й наявні всюди (ubiquitous) обчислення.

### 3. 4. 4. Метадані в Semantic Grid

Метадані різного рівня деталізації потрібні для опису ресурсів Grid, учасників, проектів та осіб тощо. У кожному проекті метадані в Grid управляються незалежно, тобто кожен окремий проект може впроваджувати власні специфічні метадані. Такий підхід придатний для управління доменно-залежними метаданими, наприклад, у біомедицині використовуються онтології «life science» (біологічних наук) для анотування даних. З іншого боку, є типи метаданих, спільні для різних проектів Grid. Інформацією про проекти, ресурси й організації Grid можна управляти в інтегрованій формі; вона може бути доступною для безпосереднього редагування всім авторизованим особам і проектам. Навіть більше, інформація про ресурси має

надаватися застосуванням і сервісам Grid, тобто інтерфейс до них має формуватися саме за допомогою метаданих. Забезпечення інтегрованого доступу до метаданих Grid дає змогу проектам більш ефективно обмінюватися інформацією про їх поточну роботу, забезпечуючи кооперацію між різними проектами й запобігаючи повторному виконанню робіт [454].

Система менеджменту метаданих повинна мати простий користувацький інтерфейс, щоб користувачі-члени віртуальних організацій, які не мають спеціальних навичок з управління знаннями, були здатні користуватися такою системою. Крім того, менеджмент метаданих, який триває досить довго, має надавати можливості щодо внесення змін до моделі метаданих (еволюційний аспект), коли застосування Grid генерують нові вимоги до метаданих [307]. Використовують модель трирівневого подання менеджменту метаданих, у якій виділяють такі рівні (рис. 3. 8):

- Метамодель, яка складається з двох основних частин: типів контенту, що визначають структуру метаінформації для екземплярів даних, і категорій, що використовуються для семантичної анотації екземплярів контенту;
- модель даних (схема), специфічна для кожної віртуальної організації Grid;
- екземпляри даних.

Тип контенту має ім’я та набір атрибутів, що відображають властивості екземплярів контенту – обов’язкових і додаткових. Тип даних атрибута має певне значення (приміром, дані, URL або рядок символів). Типи контенту можуть бути пов’язані бінарними взаємовідношеннями. Відношення можуть бути двобічними. Наприклад, Особа «є членом» Організації, а Організація «включає» Особу.

Крім таких специфічних для домену відношень, підтримуються два загальні типи відношень з попередньо визначеною семантикою: ієрархічні (генералізації) та «бути частиною». Похідні типи контенту використовують повторно метадані своїх попередників в ієрархії генералізації та можуть визначати додаткові атрибути або відношення. Кореневі вузли відношення генералізації називають базовими типами контенту. Наприклад, базовий тип контенту ‘GridResource’ може передати свої атрибути й властивості для таких більш специфічних типів контенту, як ‘GridHardwareResource’ або ‘GridSoftwareResource’. Відношення «бути частиною» взаємодіє з типами контенту в створенні ієрархії агрегації. Наприклад, рекурсивне відношення «бути частиною» може встановлюватися між організаціями. Такі ієрархії використовуються для підтримки навігації та опису контексту екземплярів контенту. Наприклад, може бути кілька елементів з назвою

«Департамент інформаційних технологій», які пов'язані з різними організаціями відношенням «є частиною».

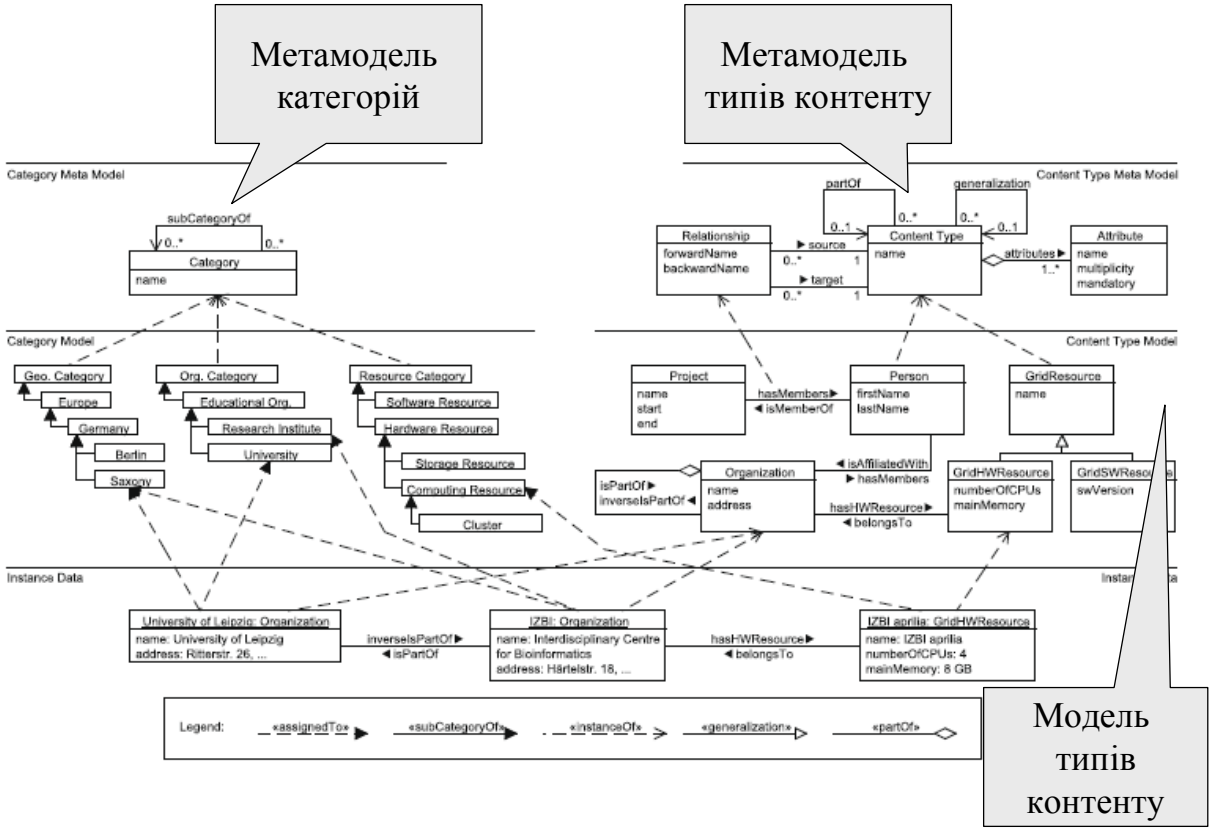



Рис. 3. 8. Трирівнева модель подання метаданих у Grid

Цей підхід базується на технології Wiki (а саме – Semantic MediaWiki – SMW) і комбінації семантичних методів. У редагуванні онтологій, на відміну від класичного редактора онтологій Protégé, підтримуються додаткові можливості (такі, як визначення логічних класів через об'єднання, доповнення, перетин тощо).

### 3. 4. 5. GCube – програмна платформа віртуального дослідницького середовища

GCube як програмна платформа для віртуального дослідницького середовища дає змогу дослідникам динамічно, на вимогу (on-demand) створювати інформаційно-обчислювальні середовища (Virtual Research Environments, VREs), агрегуючи й формуючи контент-ресурси, прикладні сервіси та комп'ютерні ресурси як за рахунок власних у прикладних проектах, так і за рахунок наявних у EGEE. У складі gCube достатньо засобів для моніторингу використання розподілюваних ресурсів з гарантією їх оптимального розподілу й експлуатації, а також легкого створення Web-порталів для VREs, через які користувачі можуть мати доступ до контенту й сервісів; надається також набір

типових для DL функцій (пошук, анотування, формування, візуалізація документів тощо) [266].

Система gCube не тільки надає повний набір сервісів для підтримки реалізацій VRE, а й забезпечує засоби адаптування й розширення своїх сервісів і можливостей, щоб відповідати потребам предметної області конкретної наукової галузі.

Спільноти науковців і дослідників використовують ПЗ gCube і ресурси Grid для обміну контентом, а також для співробітництва у VRE (рис. 3. 9).

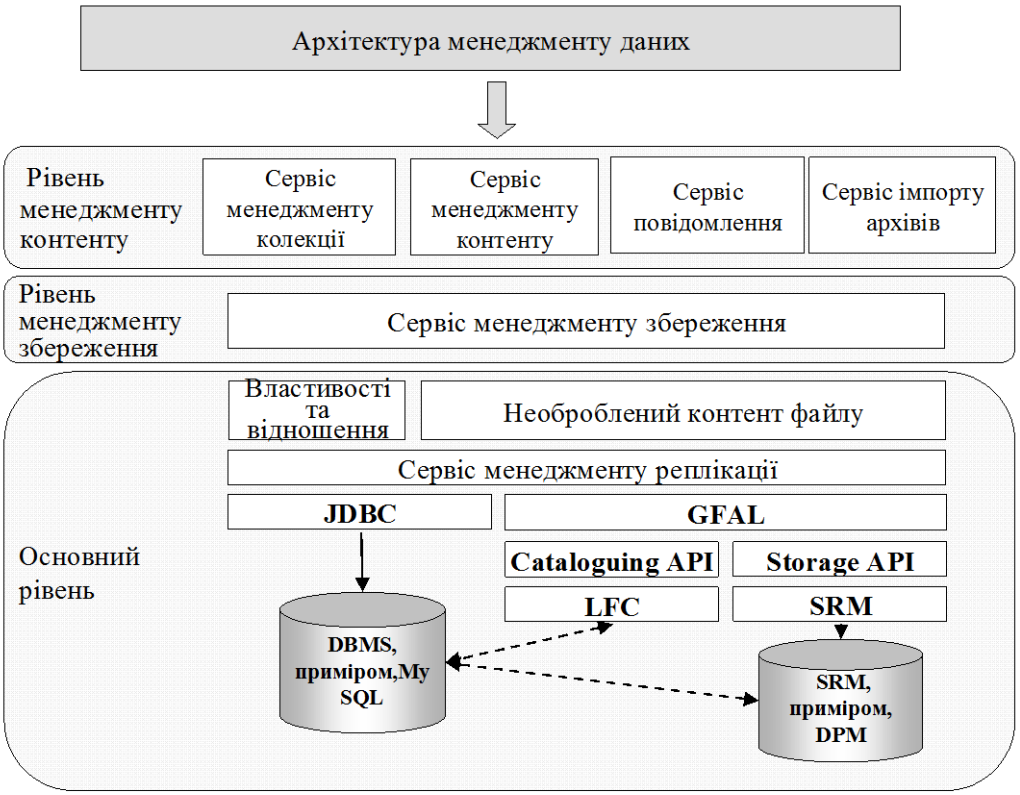



Рис. 3. 9. Архітектура менеджменту даних у gCube

Менеджмент метаданих у gCube здійснюється засобами збереження на рівні управління збереженням і призначений для управління базовими й додатковими відношеннями між інформаційними об’єктами. Ця підсистема gCube призначена для впорядкування метаданих і колекцій метаданих, тобто підтримки їх створення, доступу, збереження й видалення у форматі, придатному для ефективного пошуку [267].

Функціональні можливості, що пропонуються цією підсистемою, виконуються двома сервісами, що взаємодіють між собою: один забезпечує створення й управління метаданими та колекціями метаданих (Metadata Manager), а інший підтримує пошук за метаданими (XML Indexer), а саме:

- Metadata Manager – сервіс, який відповідає за збереження й доступ до метаданих, їх обробку відповідно до моделі метаданих, що використовується в gCube;
- XML Indexer – сервіс, що виконує запити до метаданих за допомогою розв'язання довільних виразів Xquery і XPath у їх контенті. Він індексує однорідні колекції даних XML.

Об'єкт метаданих (Metadata Object – MO) – джерело інформаційних об'єктів для IDB-відношень. MO може бути пов'язаним з одним і лише одним цільовим інформаційним об'єктом (навіть у контексті різних колекцій метаданих), що відображається цим MO.

MO зазвичай використовується як XML документ у чітко визначеній обгортці. Відповідно до контенту, сервіс менеджменту метаданих адаптується до моделі обміну, щоб додати властивості системного рівня, необхідні для відповідного управління MO.

gCube пропонує дві базові функції, пов'язані з метаданими, – дивитися й редагувати метадані.

Портлет для перегляду метаданих – це графічний інтерфейс користувача, що показує метадані об'єкту залежно від його схеми. Цей портлет забезпечує також зміну способу візуалізації метаданих відповідно до профілю користувача.

Портлет редагування метаданих – це графічний інтерфейс користувача, що може «викликатися» для редагування метаданих об'єктів. Залежно від схеми метаданих, можуть використовуватися різні редактори (приміром, візуальні редактори для схем dc та eiDB).

### 3. 4. 6. Семантичні зв'язки між онтологіями в Semantic Grid

З різноманітних причин різні люди й організації використовують різні онтології навіть для опису близьких задач і доменів. Така ж ситуація характерна й для використання онтологій доменів різними віртуальними організаціями й віртуальними дослідницькими середовищами, що функціонують на основі технологій Semantic Grid. Тому часто виникає ситуація, коли різні локальні онтології, що розвивалися незалежно, потрібно інтегрувати для отримання інформації. Семантичні зв'язки забезпечують рівень, на якому кілька онтологій мають доступ одна до одної і можуть обмінюватися інформацією з урахуванням її семантики. Семантичні зв'язки встановлюють зв'язок між сутностями двох онтологій, що стосуються одного домену так, що зберігаються логічна структура й призначені інтерпретації онтологій. Семантичні зв'язки між двома онтологіями можуть належати до одного з таких типів [479]:

- семантична еквівалентність;
- семантична подібність;

- семантичне посилання;
- семантична імплікація (властивості: транзитивність);
- підклас (властивості: транзитивність);
- послідовність (продовження контенту).

Сьогодні задача автоматизованого знаходження таких зв'язків ще не розв'язана і є актуальною науковою проблемою, однією з важливих сфер упровадження якої є менеджмент метаданих у Semantic Grid. Тому потрібно розробляти відповідні методи й алгоритми онтологічного аналізу, що придатні для різних мов подання онтологій.

**Висновки**

Аналіз публікацій і дослідницьких проектів, наведений вище, свідчить про сьогоднішній великий інтерес до використання онтологічного аналізу й технологій Semantic Web у різноманітних сферах: Business Intelligence, Grid, керування базами даних тощо. Використання онтологій дає змогу ефективно й автоматизовано створювати різноманітні віртуальні спільноти – віртуальні співтовариства, віртуальні дослідницькі середовища – для виконання різноманітних дослідницьких завдань [51], повторно використовувати знання й інтегрувати їх у різні програмні комплекси. Навіть більше, використання онтологічного аналізу уможливлює підвищення ефективності використання вже наявних віртуальних дослідницьких проектів. Прикладом цього є Semantic Grid – розвиток третього етапу Grid-технології, пов'язаний з переходом до взаємодії на семантичному рівні й упровадженням стандартів і методик проекту Semantic Web. У Semantic Grid онтології використовуються для укладання спільного словника й встановлення домовленостей щодо понять домену дослідження та опису специфічної для домену інформації всередині віртуальних дослідницьких середовищ, а також для інтегрованого менеджменту метаданими для встановлення співпраці й взаємодії між різними віртуальними спільнотами. Шляхом до цього є, зокрема, встановлення семантичних зв'язків між поняттями різних онтологій. Це зумовлює необхідність інтеграції та порівняння онтологій, створених незалежно для опису структури знань одного або кількох близьких доменів.

Іншим важливим підтвердженням корисності онтологічного аналізу є впровадження онтологій у Business Intelligence 2.0, що забезпечує більш високий рівень абстракції і працює з семантичною моделлю даних. Використання онтологічного подання знань у МАС забезпечує інтелектуальність поведінки окремих ПА й надає їм можливість до навчання на основі власного досвіду.

# РОЗДІЛ IV.
# РОЗРОБКА ОНТОЛОГІЧНИХ МОДЕЛЕЙ І МЕТОДІВ СЕМАНТИЧНОГО ПОШУКУ У WEB

## 4. 1. Сучасні засоби інформаційного пошуку

Пошук інформації з різних джерел – у Web, локальній мережі або на окремому комп'ютері – є єдним з найпоширеніших завдань у галузі ІТ [138].

Результатом такого пошку має стати задоволення *інформаційної потреби*, що виникає в користувача як наслідок нестачі наявної інформації для розв'язання певної проблеми [112]. Щоб знайти необхідну інформацію, користувач зазвичай звертається до *інформаційно-пошукової системи* (ІПС). У процесі звернення до ІПС користувач повинний сформулювати інформаційну потребу у вигляді *запиту* [149, 160] – з ключових слів, природномовного речення або прикладів.

Задачею сучасних ІПС є забезпечення доступу користувачів до гетерогенніих ресурсів – як до природномовних, так і до мультимедійних. Крім того, ІПС мають виконувати пошук серед різних типів структурованої інформації, у репозиторіях інформаційних об'єктів, в базах знань та в метаданих.

Запит користувача являє собою опис інформації, доступ до якої він хоче отримати. Такий запит може, наприклад, містити ключові слова, зв'язані логічними операторами; документ-зразок; тип документа та його тему за класифікатором; списки рекомендованих чи заборонених користувачем інформаційних джерел; обмеження на час чи обсяг пошуку; обсяг, час створення, мова шуканого документа.

*Релевантність* – формальна відповідність інформації, яку знаходить ІПС, запиту, який надіслав користувач. Релевантність документа не обов'язково повинна оцінюватися в термінах бінарної логіки («так – ні»). У деяких розвинених системах використовуються більш тонкі оцінки, що обчислюються як значення спеціально підібраної числової функції (функції релевантності), що приймає значення в інтервалі від 0 до 1. Тоді доречно говорити про ступінь релевантності документа, розуміючи її як значення цієї функції. Деякі системи текстового пошуку видають користувачу множини документів, отриманих унаслідок обробки запитів, упорядковуючи документи за зменшенням ступеня релевантності. Таке впорядкування знайдених документів називають їх ранжируванням. Користувач більш ефективно може аналізувати таку множину результатів запиту. З більшим ступенем

імовірності найцікавіші документи серед знайдених розміщуються на початку виведеного списку документів.

Однак для користувача більш важливим є інший параметр оцінки якості функціонування ІПС – *пертинентність*, тобто співвідношення між обсягом корисної для нього інформації й загальним обсягом отриманої інформації (приміром, якщо користувач уводить запит «прогноз погоди», то він здебільшого прагне отримати певні документи, що містять відомості про погоду в тому місті, де він мешкає, на найближчі кілька днів, а не про погоду в інших країнах у минулому році).

При цьому необхідно враховувати, що формальний запит до системи є спробою користувача формалізувати свою інформаційну потребу і, на жаль, не завжди точно відображає останню (унаслідок низької виразної потужності мови створення запитів до ІПС або через низьку кваліфікацію користувача) [105, 115, 140, 194].

Сьогодні основна проблема, що виникає в процесі пошуку інформації в Інтернеті, пов’язана з фільтрацією отриманих результатів і добором тих інформаційних ресурсів, що відповідають реальним інформаційним потребам користувача. Для цього потрібно формалізувати уявлення користувача про предметну область, що його цікавить, і розробити засоби автоматизованого формування відповідної бази знань. Доцільно використовувати для таких задач технології Semantic Web, а саме – онтологічне подання знань і тезауруси [117,138]. Це зумовлює необхідність розробки методів автоматизованого формування онтологій і тезаурусів предметних областей, а також способів їх використання в інформаційному пошуку.

Основні проблеми технологій текстового пошуку пов’язані зі складністю однозначної автоматичної інтерпретації змісту текстів документів і формулювань інформаційних потреб користувачів природною мовою. Висловлювання природною мовою часто бувають двозначними й надмірними. Необхідно враховувати синонімію й омонімію термінів, різноманіття граматичних форм елементів мови. Змістовні зв’язки між словами в реченні часто виражаються в прихованій формі. Лексика природних мов динамічна, тому в багатьох ПрО досить часто з’являються нові поняття й терміни.

### 4. 2. Специфіка пошуку інформації в Web

Пошук інформації у Web має сьогодні значне теоретичне забезпечення: створено когнітивні моделі для пошуку саме у Web-середовищі, розроблено ментальну модель користувачів пошукових механізмів. Більшість цих дослідження базуються на тому, що дії

користувачів у процесі пошуку зумовлені певною інформаційною потребою.

Відповідно до намірів користувачів, запити поділяють на три класи [314]:

1) *навігаційні* запити, метою яких є отримання доступу до конкретного сайту;

2) *інформаційні* запити, що відображають намір отримати деяку інформацію, яка подається на кількох різних Web-сторінках;

3) *транзакційні* запити, метою яких є виконання певної діяльності у Web-середовищі (приміром, здійснення купівлі в Інтернет-магазині чи завантаження файлів).

Сьогодні в науковій літературі є багато різних визначень пошуку у Web, у яких пошук пов'язують з певним програмним забезпеченням, що дає змогу отримувати доступ до Web-сайтів, файлів і документів, презентованих у Web, у відповідь на пошуковий запит. Це можуть бути як системи типу «запитання – відповідь», так і більш складні системи, що використовують процедури логічного виведення. Залежно від шляхів створення індексної БД, ІПС поділяють на каталоги (БД створюється людьми) і пошукові машини (БД створюються автоматизовано програмами-кроулерами).

Мета-пошукові механізми використовують ресурси та індексні БД інших ІПС, але самостійно фільтрують і ранжують отримані від них результати.

У процесі еволюції пошукових механізмів для Web можна виділити кілька поколінь. У пошукових системах першого покоління ранжування знайдених посилань на сайти враховувало тільки контент Web-сторінок (щільність ключових слів на Web-сторінці, місцезнаходження ключових слів у документі, а також метатеги, використання ключових слів в імені домену й в url-адресі), приміром, AltaVista, Excite, Webcrawler тощо [477].

У другому поколінні ІПС для визначення релевантності знайдених IP використовувалася також інформація щодо структури самої мережі Web: аналіз посилань на цю сторінку та даних, що передаються за http-запитом, індекс популярності й репутація сторінки, кількість часу, що проводять відвідувачі на сторінці тощо.

Першою ІПС, яка почала використовувати аналіз посилань між сторінками як один з основних факторів ранжування, стала Google (PageRank), а DirectHit – перша ІПС, у якій ранжування базується на аналізі даних, що передаються під час http-запиту. Сьогодні ці системи продовжують розвиватися в напрямі їх семантизації, залучаючи до пошуку зовнішні бази знань і онтології [205].

ІПС третього покоління намагаються інтегрувати інформацію, знайдену в різних ІР, на основі її семантичного аналізу, з урахуванням персональних інформаційних потреб користувачів. Крім того, такі ІПС здатні враховувати в процесі пошуку знання про предметну область пошуку, приміром, використовувати тезаурус ПрО.

Прикладом ІПС третього покоління є Wolfram Alpha [469] – це Web-сервіс, який розробники позиціонують як засіб для обчислення знань, що дає змогу переводити одиниці виміру з однієї системи в іншу, за хімічною формулою знаходити інформацію про певну речовину / хімічний елемент, обчислювати кількість калорій у продуктах, надавати інформацію про географічні об'єкти й Web-сайти, розв'язувати рівняння й будувати графіки функцій, обробляти фінансову інформацію тощо й наводити посилання на використані для цього джерела інформації.

Ще одна ІПС третього покоління – Google Squared [370] – пропонує користувачеві замість списку посилань на ІР таблицю зі структурованою інформацією за запитом, отриману з різних джерел. SenseBot теж позиціонується як семантична пошукова система, що у відповідь на пошуковий запит генерує текстові анотації знайдених ІР і «семантичну хмару» понять, що містяться в цих текстах, використовуючи методи text mining для аналізу семантики знайдених ІР.

В ІПС Hakia семантична технологія забезпечує пошук, орієнтований не на популярність, а на якість, що визначається за трьома критеріями: інформація має надходити з Web-сайтів, що заслуговують на довіру, відображати найбільш свіжу наявну інформацію й бути релевантною запиту. Інтелектуалізувати процес пошуку можна за допомогою структурування даних, отриманих з Інтернету; семантичної фільтрації за якістю; організації пошуку серед структурованих даних в Інтернеті; пошуку в режимі реального часу в Інтернеті; пошуку в «глибинному» Web [257, 328, 362, 423].

Є ІПС, що спеціалізуються на пошукові серед структурованої інформації, поданої у форматі OWL і RDF, тобто базуються на технологіях і стандартах Semantic Web [412, 419, 434]. Наприклад, сервіс SWSE [437] індексує RDF або OWL, знайдені у Web, і надає послугу щодо пошуку серед цих даних. Swoogle – ІПС, що здійснює пошук за ключовими словами серед даних у форматі RDF і видає посилання на джерела, які їх містять. Аналогічні функції виконують такі системи, як Semanticwebsearch [412], WatsOn [460], Falcons [257] і Sindice.

Подальша інтелектуалізація пошуку пов'язана з застосуванням онтологічного подання знань, яке забезпечує їх багаторазове використання [170].

## 4. 3. Особливості семантичного пошуку

Сучасний рівень розвитку ІТ та Інтернет зумовив трансформацію завдання пошуку інформації – з виявлення документів, що містять певні ключові слова, у пошук знань, необхідних для розв'язання поставленої перед користувачем задачі. Семантичний пошук – це процес пошуку інформації, що задовольняє інформаційну потребу користувача, яка виникає в нього в процесі розв'язання певної проблеми [42, 54, 328].

Для семантичного збагачення ІР і запитів користувача можна використовувати:

- персональний профіль цього користувача ІПС, що містить історію його попередніх запитів, модель області його інтересів тощо [158, 285];
- прикладну семантичну інформацію, фактично надану користувачем (наприклад, синоніми термінів або онтології відповідної ПрО) [42, 143, 291];
- досвід роботи ІПС та історії запитів інших користувачів цієї ІПС (або підгрупи користувачів, у чомусь подібних до нього) [138, 147];
- зовнішні джерела знань, доступні через Web (приміром, Вікіпедію) [135, 162].

З розвитком інфраструктури Semantic Web семантичні метадані стали більш доступними. Розробка стандартів семантичної розмітки, мов опису ІР (RDF), онтологічних мов, сервісів, онтологічних баз, систем пошуку в семантичних даних (Swoogle, SWSE, WatsOn), точок SPARQL-доступу, систем логічного виведення, обробки правил тощо сприяє подальшому розвитку області інформаційного пошуку в напрямі використання семантики [290, 293].

Виконання семантичного пошуку потребує створення формалізованих моделей як доступних інформаційних ресурсів, серед яких здійснюється пошук, так і інформаційної потреби та тієї задачі, яку намагається розв'язати користувач за допомогою шуканої інформації. У процесі пошуку потрібно зіставляти ці моделі й знаходити ІР, пертинентні задачі.

*Семантичний пошук* – це надбудова над традиційним інформаційним пошуком, у якому для підвищення пертинентності пошуку (тобто для більш ефективного задоволення інформаційних потреб користувача) використовується обробка знань, що стосуються як самого користувача та його інформаційних потреб (персоніфікація пошуку), так і інформаційних ресурсів, серед яких здійснюється пошукова процедура [166].

Результатом семантичного пошуку може бути як здобуття інформації, неявно наявної в певному ІР (як текстовому, так і мультимедійному), так і надання користувачеві відомостей про наявні ІР у певному порядку й певній формі, що відповідають персональним потребам саме цього користувача.

Семантичний пошук може здійснюватися в розподіленому інформаційному середовищі з використанням додаткових знань щодо цього середовища, приміром, пошук в Internet of Things [292] або у Web of Things, у середовищі GRID [29, 47].

Користувач може явно вказувати тип інформаційного об'єкта, який він прагне знайти, використовуючи для цього відповідні стандарти й таксономії, приміром, для здійснення пошуку в репозиторіях онтологій [51] або Web-сервісів [53], серед RDF-описів і XML-структур [237].

Результатом семантичного пошуку може бути інтегрування відомостей з різних доступних ІР.

Під час семантичного пошуку, на відміну від звичайного, предметом пошуку може бути не конкретний інформаційний ресурс (ІР) – документ чи його фрагмент, а інформаційний об'єкт певного класу, тобто користувач може (явно чи неявно) вказати клас шуканого об'єкта. Це може бути досить простий і поширений клас, наприклад, «людина» чи «мультимедійний об'єкт» або клас, специфічний для певної предметної області (ПрО), наприклад, «наукова публікація» [49, 140].

Для розв'язання найрізноманітніших прикладних задач потрібно використовувати не стільки знання про всю предметну область загалом, скільки саме ту їх частину, що дає змогу встановити зв'язок між початковими умовами задачі та її результатом. При цьому можна виділити деякі структурні елементи, що наявні в постановці інтелектуальної задачі, і певним чином їх класифікувати. Це зумовлює необхідність створення певної формальної моделі задачі, що використовує онтологічне подання знань і встановлення зв'язків цієї онтології задачі з відповідними онтологіями ПрО.

В інформаційному пошуку з використанням семантики користувачеві можуть бути запропоновані:

- запити інших користувачів, схожі з введеним ним запитом, і їх результати;
- посилання на довідкову інформацію, пов'язану з пошуковим запитом (довідники, словники, статті Вікіпедії тощо);
- семантично анотовані результати пошуку (результати пошуку посилання на сторінки /документи мають виділені

фрази, імена, назви, терміни, семантично пов'язані з пошуковим запитом);
- фрагменти природного тексту, подібні до введеного користувачем;
- уточнення запиту через семантичні та синтаксичні властивості ключових слів (приміром, уточнюється частина мови терміна або його зміст, наприклад, прізвище людини чи назва міста);
- терміни та відношення онтології й таксономії ПрО;
- пошук семантичних структурованих даних;
- фасетний і кластерний пошуки.

Аналіз багатьох наявних семантичних ІПС засвідчив різноманітність розуміння поняття семантичного пошуку, а саме – використання семантичної розмітки під час індексації ІР і виконанні пошуку; використання персональної інформації користувачів; зворотний зв'язок з користувачем (рейтингування й ранжування результатів пошуку); узагальнення статистичних даних для врахування спільного досвіду користувачів (часу, проведеного на сторінці, типових запитів); удосконалення фільтрації результатів (вилучення спаму, реклами); виправлення помилок у запиті.

Наприклад, у ІПС Hakia семантичний пошук передбачає обробку морфологічних змін, синонімів і узагальнювальних термінів для слів запиту; визначення предметної області запиту; обробку природномовних запитів; пошук на рівні не документу, а його контенту; здатність до навчання на основі власного досвіду й спроможність змінювати власну поведінку.

### 4. 4. Використання онтологій в семантичному пошуці

Сьогодні завдання пошуку інформації трансформувалося з виявлення документів, що містять певні ключові слова, у пошук знань, необхідних для розв'язання певної задачі. Семантичний пошук – це процес пошуку документів за змістом. Він складається з формування інформаційних моделей користувача, предметної області, що його цікавить, задачі, яку розв'язує користувач, та інформаційних моделей доступних інформаційних ресурсів, що характеризують їх семантику й подальше зіставлення [53]. Отже, завдання семантичного пошуку передбачає формування й зіставлення онтологій.

Для використання онтологічних знань у семантичному пошуку потрібні механізми автоматизованого створення онтологічних моделей предметних областей та інформаційних потреб, а також методи їх зіставлення. Іноді такі моделі значно легше зіставляти, ніж довільні онтології [26, 260, 289, 345].

### 4. 4. 1. Тезаурус як засіб моделювання природномовних інформаційних ресурсів

Тезаурус можна розглядати як окремий випадок онтології. Це словник основних понять мови, що позначаються окремими словами чи словосполученнями, з визначеними семантичними зв'язками між ними. Можна говорити про те, що ТС є відображенням (моделлю) певної ПрО [73, 108, 117].

*Формальна модель тезауруса* – це пара $Ts = \langle T, R \rangle$, де T – множина термінів, а R – множина відношень між цими термінами [46].

**Методи створення тезаурусів.**

Сьогодні застосовуються два способи створення тезаурусів – ручний і автоматичний. Ручна побудова тезауруса є дуже дорогою, кропіткою та трудомісткою справою, що вимагає значного часу. Тому на практиці часто використовують автоматичне або автоматизоване створення тезаурусів. Автоматичне створення тезаурусів здійснюється, зазвичай, на основі заданих колекцій текстових документів, тому такі тезауруси призначені для роботи саме з цими колекціями.

*Тезаурус ПрО* – це сукупність термінів ПрО, знайомих користувачеві, що містяться в ІР, які були знайдені раніше за запитами користувача й визнані ним як такі, що стосуються цієї ПрО.

Щоб формалізувати інформаційні потреби користувача, необхідно (рис. 4. 1):

- сформувати тезаурус ПрО, що відповідає інформаційним потребам цього користувача (шляхом аналізу ІР, які користувач вважає релевантними цій ПрО);
- побудувати тезаурус для кожного ІР, знайденого ІПС (простий словник без стоп-слів);

– порівняти тезауруси ІР, знайдені ІПС, з тезаурусами ПрО й віднайти ті, що містять найбільше слів у перетині.

Формування тезауруса інформаційного ресурсу.

Через велику кількість ІР пропонується використовувати спрощений алгоритм побудови їх тезауруса: з повного переліку слів, що використовуються в ІР, відкидаються стоп-слова, що містяться в спеціально розробленому користувачем списку (або в кількох списках) [54, 285]. Цей алгоритм застосовується тільки для тих ІР, що не супроводжуються метаописами. В іншому випадку з метаописів (у форматі RDF чи OWL) «витягаються» терміни тезауруса й зв'язки між ними, що доповнюють побудований за контентом ІР словник.

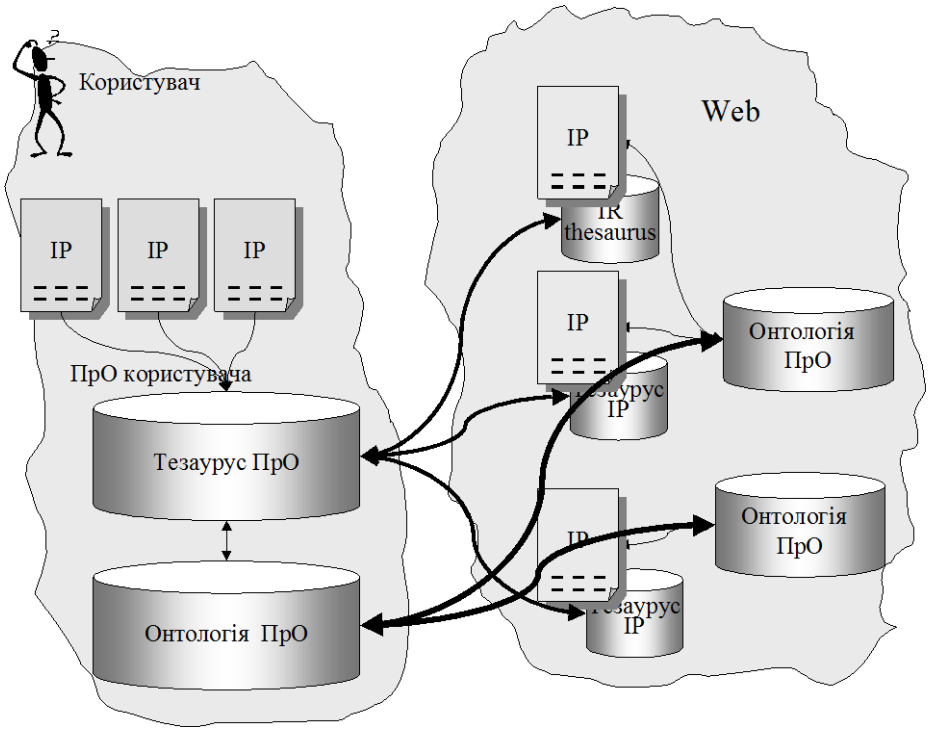



Рис. 4. 1. Інформаційний пошук на базі тезаурусів

**Побудова тезауруса ПрО.**

Тезаурус – це об'єднання тезаурусів IP, релевантних цій ПрО [54], який удосконалюється, якщо потрібно, за допомогою онтології відповідної ПрО (рис. 4. 2).

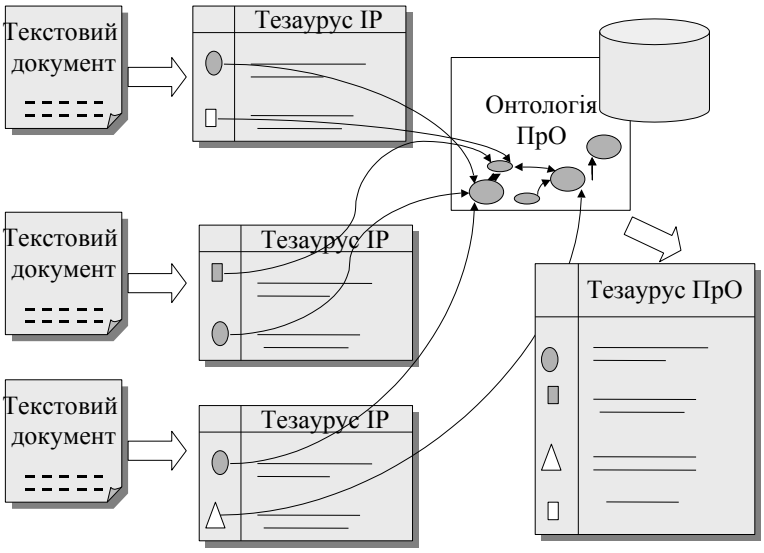



Рис. 4. 2. Побудова тезауруса ПрО

Тезаурус ПрО досить просто створити за її онтологією як множину її термінів, але побудова онтології ПрО є надто складним завданням для більшості користувачів (навіть якщо вони мають відповідні знання й застосовують їх у своїй діяльності), оскільки потрібно вказати основні поняття ПрО й зв'язки між ними.

На першому етапі формування тезауруса користувач може обрати одне або кілька із запропонованих рішень:

- самостійно побудувати чи відредагувати онтологію ПрО за допомогою одного з редакторів онтологій і здобути з неї множину термінів;
- знайти (наприклад, в Інтернеті) яку-небудь онтологію, подану мовою OWL, що відображає ПрО, близьку до сфери його інформаційних інтересів;
- сформувати множину понять ПрО, що містить найбільш характерні слова та словосполучення, що трапляються в ІР, які його цікавлять;
- знайти набір документів, релевантних ПрО, автоматично згенерувати їх тезауруси й поєднати їх.

### 4. 4. 2. Фільтрація ІР на основі тезаурусів

Алгоритм фільтрації результатів запиту користувача щодо зовнішнього ІПС [54]:

- Користувач вводить запит, у якому ідентифікує свою інформаційну потребу через набір ключових слів.
- ССП передає запит до зовнішньої ІПС, від якої одержує результати його виконання n посилань на ІР та їх короткі описи $I = \{Ref_j, D_j\}, j = \overline{1,n}$, де $Ref_j$ http-адреса відповідного ІР, знайденого ІПС, а $d_j$ інформація про цей ІР, що зовнішня ІПС надає користувачеві у відповідь на запит.
- Якщо множина I не порожня ($n \geq 1$), то потрібно встановити порядок, відповідно до якого й пропонувати користувачеві відомості про знайдений ІР. З цією метою для всіх ІР з множини $I = \{Ref_j, D_j\}, j = \overline{1,n}$ формують їх спрощені тезауруси $Ts(IP_j) = \langle T_{j_j}, \varnothing \rangle, j = \overline{1,n}$ і відповідні їм словники термінів $T_j = \{t_{j_w}\}, j = \overline{1,n}, w = \overline{1,q_j}$. $t_{j_w}$ слова, що використовуються в інформації про j-м ІР, знайдений ІПС. $D_j, j = \overline{1,n}$ $q_j, j = \overline{1,n}$ кількість слів, що використовуються в описі $D_j, j = \overline{1,n}$. Якщо слова в описі повторюються, то в словнику термінів вони фіксуються тільки один раз.
- Користувач формує тезаурус ПрО, яка його цікавить (чи вказує на раніше сформований тезаурус) $Ts_{ПрО}$ і відповідний йому словник термінів цієї ПрО $T_{ПрО} = \{t_m\}, m = \overline{1,q}$. $T_{ПрО}$ це множина, що складається з m термінів, які належать до ПрО користувача, що будується аналогічно до словника термінів ІР і, зазвичай, формується як об'єднання словників термінів, що містяться в документах, які користувач знайшов раніше й вважає

релевантними ПрО, яка його цікавить (як у їх контенті, так і в метаописах).

- Виконується порівняння $T_{ПрО}$ і $T_j, j=\overline{1,n}$, підраховується коефіцієнт їх близькості $K_j=\sum_{m=1}^{q}\sum_{w=1}^{w_j} f(t_{j_w}, t_m), m=\overline{1,q}, w=\overline{1,w_j}$, де $f(t_1,t_2)=\begin{cases}0, \text{если } t_1 \neq t_2\\ 1, \text{если } t_1 = t_2\end{cases}$ (1).

Коефіцієнт (1) являє собою кількість термінів, що трапилися як у тезаурусі ІР, так і в тезаурусі ПрО.

- Знайдені ІР впорядковуються залежно від значень $K_j$, користувачеві подаються насамперед ті ІР, що мають найвищий коефіцієнт близькості до ПрО.

У коефіцієнті (1) слова, що відповідають одному терміну, але є різними словоформами, синонімами чи перекладами різними мовами, обробляються як різні терміни. Більш доцільно використовувати онтологію ПрО й виділяти групи слів, що відповідають одному терміну [46]. Для цього користувач повинний зв'язати елементи словника термінів тезауруса ПрО з одним із термінів онтології ПрО O=<X,R,F>, тобто $\forall t_m \in T_{ПрО}, m=\overline{1,q}=$ задати функцію $g(t_m)\in X$. Потім, для обчислення коефіцієнта близькості ДО, ця функція використовується в такий спосіб:

$$K^O{}_j=\sum_{m,l} f(t_{j_w}, t_m), m=\overline{1,q}, w=\overline{1,w_j}, \text{ де } f(t_1,t_2)=\begin{cases}0, \text{если } g(t_1) \neq g(t_2)\\ 1, \text{если } g(t_1) = g(t_2)\end{cases} \quad (2).$$

Коефіцієнт (2) – кількість термінів у тезаурусах ІР і ПрО, що посилаються на той самий термін онтології ПрО. Порівняно з коефіцієнтом (1) коефіцієнт (2) дає змогу використовувати менший обсяг документів для побудови тезауруса ПрО, але потребує більше часу для обчислень.

Можна використовувати в тезаурусі одночасно як позитивні (відомі, бажані, релевантні терміни), так і негативні (незнайомі, незрозумілі, нерелевантні проблемі терміни) [42, 167]. Терміни з *позитивного* тезауруса (тезауруса, підключеного до пошукового модуля для того, щоб вказати, що користувач цікавиться цією ПрО) підвищують рейтинг ІР, терміни ж з негативного тезауруса (тезауруса, підключеного до пошукового модуля для того, щоб вказати, що користувач не цікавиться цією ПрО) – знижують [169].

## 4. 5. Пошук у Web як розпізнавання інформаційних об'єктів

У найзагальнішому розумінні *пошук інформації* – це зіставлення подання користувача про потрібні йому знання з контентом доступних ІР і побудова на основі цього зіставлення інформаційного об'єкта (ІО) [65], значення властивостей якого здобуваються з цих ІР. Найбільш перспективні напрями розвитку інформаційного пошуку пов'язані нині з його персоніфікацією та інтелектуалізацією [68].

Під час *семантичного пошуку* в цьому зіставленні використовуються *знання* щодо різних суб'єктів пошуку – користувачів, ресурсів, результатів раніше виконаних пошукових процедур, а також знання про ПрО пошуку [112, 257, 285, 328, 362]. Якщо для формалізації знань про суб'єктів цього процесу використовуються онтології, то можна говорити про *онтологічну модель* пошуку [161, 287, 402].

Семантичний пошук можна розглядати як окремий випадок проблеми розпізнавання, якщо результатом пошуку має стати інформаційний об'єкт зі складною структурою, знання про який використовуються в пошукових процедурах [52].

### 4. 5. 1. Семантичне розпізнавання інформаційних об'єктів

*Інформаційний об'єкт* (ІО) – модель об'єкта предметної області (ПрО) в інформаційному просторі, що визначає структуру, атрибути, обмеження цілісності й, можливо, поводження цього об'єкта.

Сам об'єкт предметної області може бути як матеріальним (наприклад, людина, організація, місце), так і віртуальним, що не має конкретного матеріального еквівалента (наприклад, книга, програма), хоча і зв'язаного з тим чи іншим матеріальним носієм інформації. Крім того, об'єкт ПрО може являти собою абстрактне поняття, яке в принципі не зв'язане з конкретним носієм інформації й відображається лише у свідомості людини (наприклад, «час», «мораль»).

Класична постановка задачі розпізнавання образів може бути сформульована так:

- задано скінчену множину класів P, $P = \{p_j\}, j = \overline{1,m}$;
- задано скінчену множину об'єктів ПрО, $O = \{o_i\}, j = \overline{1,n}$, про кожен з яких відомо, що він може бути однозначно зарахований до одного з класів з P: $\forall o_i \in O \exists p_j \in P : o_i \in p_j$;
- кожен об'єкт $o_i = \langle a_{i_1},...,a_{i_k} \rangle$ з ПрО має набір із k властивостей $A_q, q = \overline{1,k}$ таких, що $a_{i_q} \in A_q$;

- задано класифіковану підмножину об’єктів (навчальну вибірку) $O_{cl} \subseteq O$ таку, що $\forall o_i \in O_{cl}$ відомо, до якого саме класу $p_j \in P : o_i \in p_j$ належить цей об’єкт.

Необхідно визначити, до яких класів належать усі об’єкти з O.

Для об’єктів реального світу типова постановка задачі класифікації припускає виділення визначеного конкретного об’єкта серед подібних йому. У найзагальнішому випадку в задачі розпізнавання може бути виділено дві підзадачі, що виконуються ітеративно до одержання прийнятного результату:

Крок 1. Побудова гіпотези про те, до яких класів можуть належати об’єкти і як саме характеризуються об’єкти всередині класу;

Крок 2. Зарахування конкретного об’єкта до одного з цих класів.

Якщо на 2-му етапі розпізнавання виявляється неуспішним, то необхідно повернутися до кроку 1 та сформувати іншу множину класів (розширити початкову множину чи змінити критерії належності до класу). Наприклад, почувши звук кроків у квартирі, людина може припустити, що їх робить інша людина, оскільки їй відомо про наявність у приміщенні інших людей. Однак якщо потім вона чує, наприклад, нявчання, то може припустити, що джерелом звуків є або кішка, або ввімкнений телевізор.

### 4. 5. 2. Пошук інформаційних об’єктів у Web

В інформаційному просторі Web задача розпізнавання об’єктів безпосередньо пов’язана з інформаційним пошуком і його метою. Онтологію ПрО можна використовувати як основу для презентації структури ІО (класи), а ІР – для створення екземплярів ІО. Це дає змогу інтегрувати інформацію з різних ІР.

При цьому виникає ряд підзадач:

- пошук онтології, що відображає структуру ІО, знання про які необхідні користувачу;
- пошук ІР, що містить відомості про ці ІО;
- здобуття знань про ІО з ІР;
- надання здобутих знань у зрозумілій і зручній формі.

Дуже часто цей процес є ітеративним, а повторний пошук потребує оновлення інформації (наприклад, через Web чи корпоративну мережу).

Під час семантичного пошуку ІО може мати заздалегідь задану складну структуру, формалізовану у вигляді класу відповідної онтології. Приклади ІО – організація, навчальний заклад, людина-експерт, Web-сервіс, аналітична модель бізнес-процесу [25].

## 4. 6. Онтологічна модель взаємодії користувачів та ІР

Взаємодію користувачів та ІР у процесі семантичного пошуку доцільно відобразити за допомогою онтологічної моделі (рис. 4. 3), у якій використовуються такі основні класи [159]:

- *користувач***,** що здійснює опис властивостей, і інтереси особи, що шукає інформацію за допомогою ССП;
- інформаційна потреба користувача;
- *онтологія ПрО*, що відображає область, до якої належать інформаційні потреби користувача [53];
- інформаційний ресурс;
- інформаційний об'єкт;
- задача користувача;
- *тезаурус задачі* терміни онтології, сукупність яких характеризує ту конкретну задачу з ПрО, що в цей момент розв'язує користувач, та їх вага [46];
- *запит* множина ключових слів, що характеризують одну з інформаційних потреб користувача, пов'язану з конкретною задачею за допомогою тезауруса;
- *тема* множина запитів, пов'язаних з однією інформаційною потребою різних користувачів, що дає змогу поєднувати семантично зв'язані запити;
- *результат запиту* посилання на ІР та їх оцінки;
- група користувачів.

Крім цих основних класів, доцільно ввести до моделі для опису знань та інтересів користувача два службові класи – «компетенція» і «атомарна компетенція» [159]. Екземпляри цих класів можуть імпортуватися з різних організаційних онтологій і з онтологій ПрО, пов'язаних з навчальними закладами, працевлаштуванням, науково-дослідними організаціями тощо (докладніше засоби аналізу компетенцій на основі онтологій розглянуто нижче). Запропонований підхід орієнтований на відносно вузьку підмножину ІПС, призначених для підтримки науково-дослідної та освітньої діяльності й розрахований на користувачів зі стабільними, складно структурованими знаннями й інформаційними потребами в досить вузькій ПрО [152]. Такі обмеження зумовлені тим, що для ефективної роботи подібних систем користувачам потрібно докласти певних зусиль для формалізованого опису сфери власних інформаційних потреб.

Набір компетенцій дає змогу формалізувати не всі накопичені в процесі навчання й роботи знання, а лише ті, що активно використовуються в практичній діяльності.

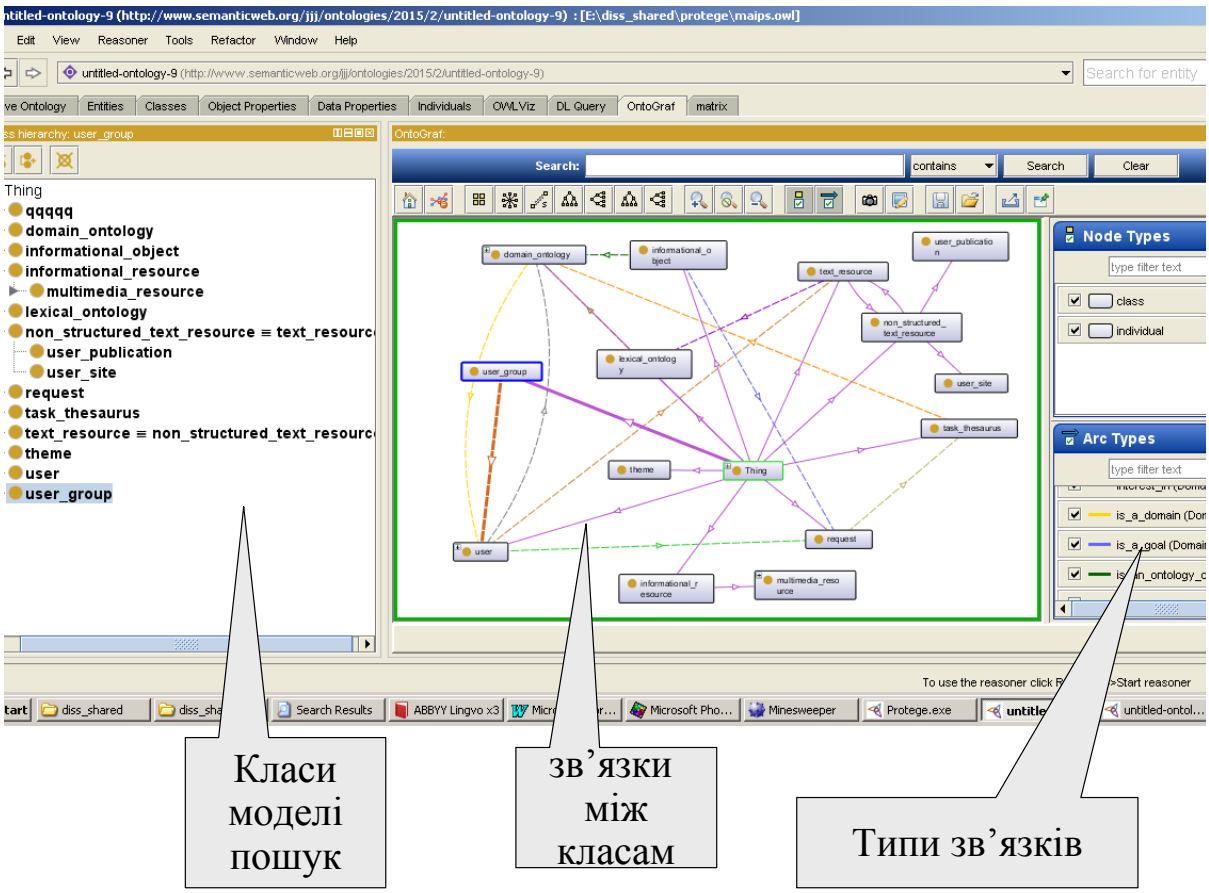



Рис. 4. 3. Онтологічна модель семантичного пошуку

На такій онтологічній моделі базується робота інтелектуальної інформаційно-пошукової системи МАІПС, яка створена для пошуку інформації у відображених користувачем відносно вузьких ПрО, пов'язаних з професійними чи науковими інтересами [67].

МАІПС можна розглядати як рекомендувальну систему, орієнтовану на формування колаборативних рекомендацій щодо ІР [147]. Рекомендувальні системи (РС) здатні проактивно створювати рекомендації на основі досвіду взаємодії з іншими користувачами, тобто пропонувати ті інформаційні ресурси, що зацікавили інших користувачів з подібними інформаційними потребами [398].

На відміну від ІПС, користувачу РС не потрібно явно формулювати пошуковий запит – система сама, на підставі наявних відомостей про користувача, пропонує йому *рекомендовані елементи* (РЕ) [204]. Персоналізовані рекомендації, що виробляються такими системами, – це впорядковані списки РЕ, тобто робота РС зводиться саме до ранжування доступних РЕ [205].

МАІПС дає змогу зберігати й повторно виконувати запити, ураховуючи реакцію користувача на раніше запропоновані йому ІР (персональна фільтрація), відстежувати появу аналогічних запитів в інших користувачів (колаборативна фільтрація), зберігати формальний

опис сфери інтересів користувача у вигляді онтології (семантична фільтрація) тощо (рис. 4. 4).

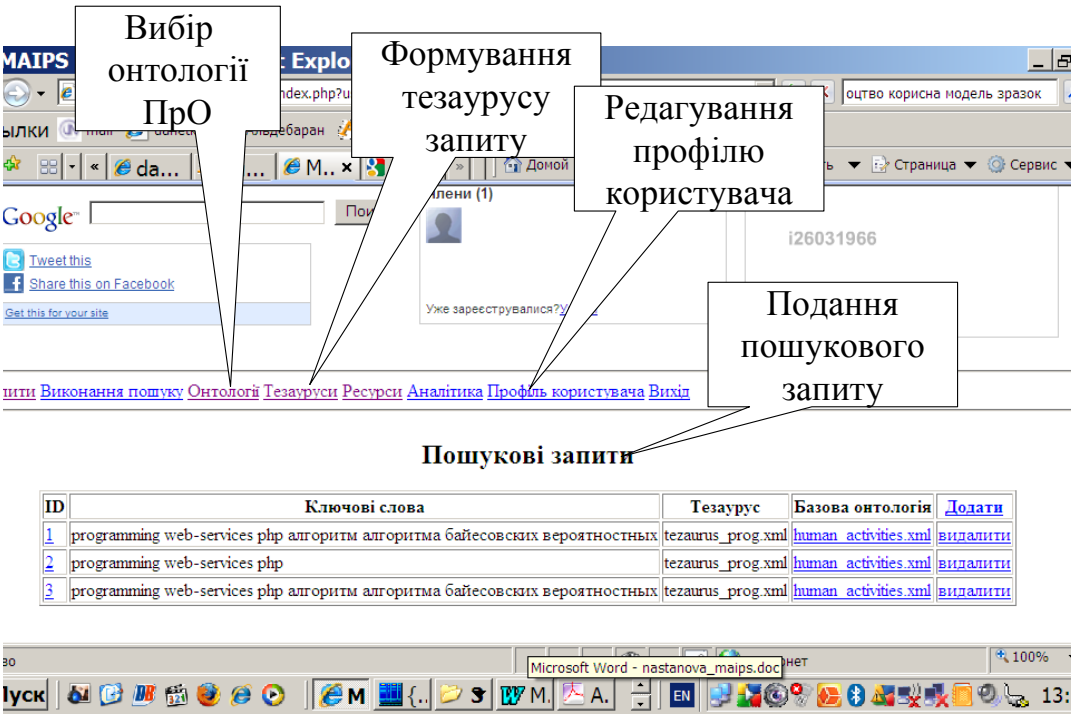



Рис. 4. 4. Пошукова система МАІПС

Під час виконання запиту він переадресується до зовнішньої ІПС, а отримані результати перевпорядковуються з урахуванням знань про користувача; ПрО, що цікавить його (онтологія), й розв'язувану задачу (тезаурус задачі) [151, 154].

Для впорядкування результатів пошуку в МАІПС ураховуються також відомості про індекс легкості читання природномовних ІР [141], які застосовують як до всього ІР, так і до його анотації, що пропонується користувачеві у відповідь на запит пошукової системи (рис. 4. 5).

## 4. 7. Джерела відомостей про екземпляри класів моделі

Відомості про значення елементів інформаційної моделі рекомендувально-пошукової системи МАІПС зберігаються як опис екземплярів цих класів.

Відомості щодо ІР отримуються шляхом аналізу контенту ІР та їх оцінок користувачами, яким ці ВР були запропоновані. Крім традиційних параметрів ІР, отриманих від зовнішніх ІПС, МАІПС оцінює, наскільки отримані ресурси будуть зрозумілі конкретному користувачеві, застосовуючи для цього критерії легкості читання

їх [141]. Основні параметри цих критеріїв – загальна *кількість слів* у тексті, середня *довжина* речення й *довжина* використовуваних *слів* (середня довжина слова може вимірюватися в складах і в символах, підраховується кількість багатоскладових і односкладових слів). Найпоширеніші методи визначення легкості читання тексту – це індекс туманності Ганнінга [350], формула Флеща [263], індекс Фрая [332], формула SMOG [346]. Вони дають змогу визначити, чи буде зрозумілим певний текст читачеві відповідного віку й з певним рівнем освіти. Крім характеристик самого тексту, доцільно враховувати й характеристики читача – його досвід у цій ПрО, навички та мотивації.

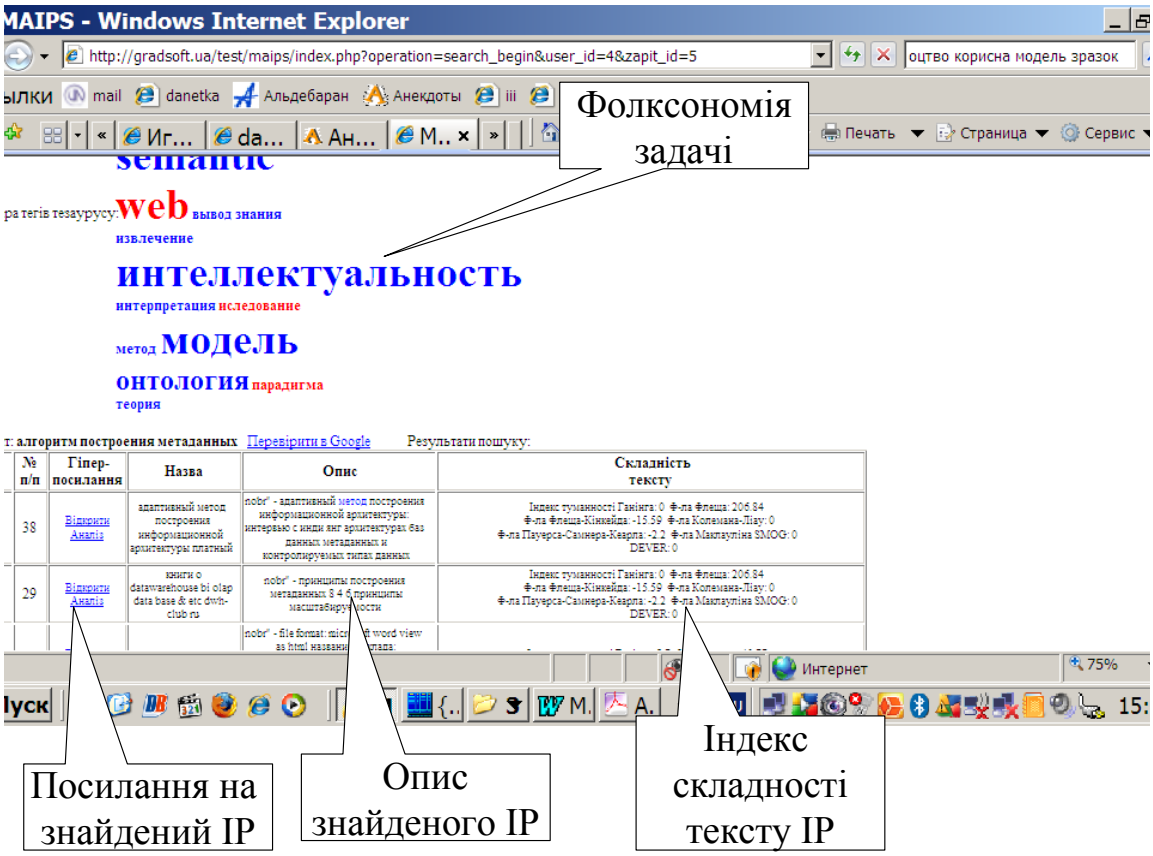



Рис. 4. 5. Приклад результатів виконання пошуку

Відомості про користувачів ССП, такі, як персональні переваги користувача, здатність до сприйняття інформації, його інформаційні потреби, задачі, що стоять перед ним та ІО, з'ясування відомостей про які дасть змогу користувачеві розв'язати цю задачу, а також досвід взаємодії його з ССП, є основою для персоніфікації семантичного пошуку й створення рекомендацій. Частину цієї інформації користувач уводить під час реєстрації. Відомості про сфери інтересів і компетентності користувача можуть імпортуватися із зовнішніх джерел – соціальних мереж, Wiki-ресурсів, зовнішніх онтологій тощо.

У процесі взаємодії з ССП знання про користувача поповнюються: у його профіль автоматично вносяться дані про те, чи користувач обирає якусь онтологію, чи створює, модифікує тезаурус, формує запит і виконує раніше створений запит. Користувач може вносити зміни у свій профіль, щоб він більш коректно відображав його інформаційні потреби. У профілі користувач за допомогою індексу легкості читання може зазначити, які ІР є зрозумілими для нього, явно вказавши індекс або запропонувавши ІР із заданими характеристиками.

Крім того, для персоніфікації пошуку враховуються психофізіологічні властивості користувача, що впливають на його здатність до сприйняття інформації та створюють для користувача комфортний індивідуальний адаптаційно-інформаційний простір [97]. Ці властивості базуються на таких типах класифікації користувачів за швидкістю реагування й гальмування, як холерик, сангвінік, флегматик і меланхолік або на таких типах особистості, як екстраверт та інтроверт.

Для подання знань у МАІПС використовуються *внутрішні* й *зовнішні* онтології: внутрішні онтології створюються розроблювачами МАІПС, а зовнішні імпортуються з Web. Лексичні онтології ПрО розробляються автоматизовано для кожної внутрішньої онтології ПрО, що входить до складу МАІПС.

Тезаурус задачі користувач будує для того, щоб відобразити специфіку своєї задачі за обраною онтологією ПрО, вказуючи на потрібні терміни й визначаючи їхню вагу. Він відображається у формі фолксономії [339], де розмір і колір шрифту позначає вагу термінів. Реалізовані в системі теоретико-множинні операції над тезаурусами – об’єднання, перетинання й доповнення – дають можливість користувачам легко створювати нові тезауруси як комбінації вже наявних.

Групи користувачів формуються користувачами відповідно до семантики їх колаборативних інформаційних потреб за певними умовами, приміром, поєднання користувачів, що використовують одну онтологію чи тезауруси, що перетинаються, або через фактичний перелік.

Значення службових класів – «компетенція» і «атомарна компетенція», які відображають обізнаність користувачів у різних ПрО, можуть імпортуватися як з компетенцій спеціальності (чи спеціальностей) користувача, що підтверджують його дипломи та сертифікати або з опису посад(и), яку(і) він обіймає ним, так і з онтологій ПрО, пов'язаних з навчальними закладами, працевлаштуванням і науково-дослідними організаціями [403]. Це

забезпечує найбільш пертинентне зіставлення користувачів з онтологіями ПрО і властивостями ІР [66, 164].

Запропоновану вище модель семантичного пошуку досить легко адаптувати до розв'язання різних прикладних задач. Приміром, її можна застосувати до розв'язання проблеми зіставлення компетенцій, що входить до складу таких задач, як порівняння кваліфікації фахівців, що здобули освіту в різних країнах; пошук роботодавцем виконавців робіт; порівняння навчальних закладів з погляду абітурієнта з певними інтересами та здібностями; зміна місця навчання тощо.

Модель ПрО відображає базові поняття – «дисципліна», «спеціальність», «змістовний модуль», «компетенція», «кандидат» тощо й зв'язки між ними, а також структуру ІО – результат пошуку (наприклад, клас «студент», «фахівець») (рис. 4. 6).

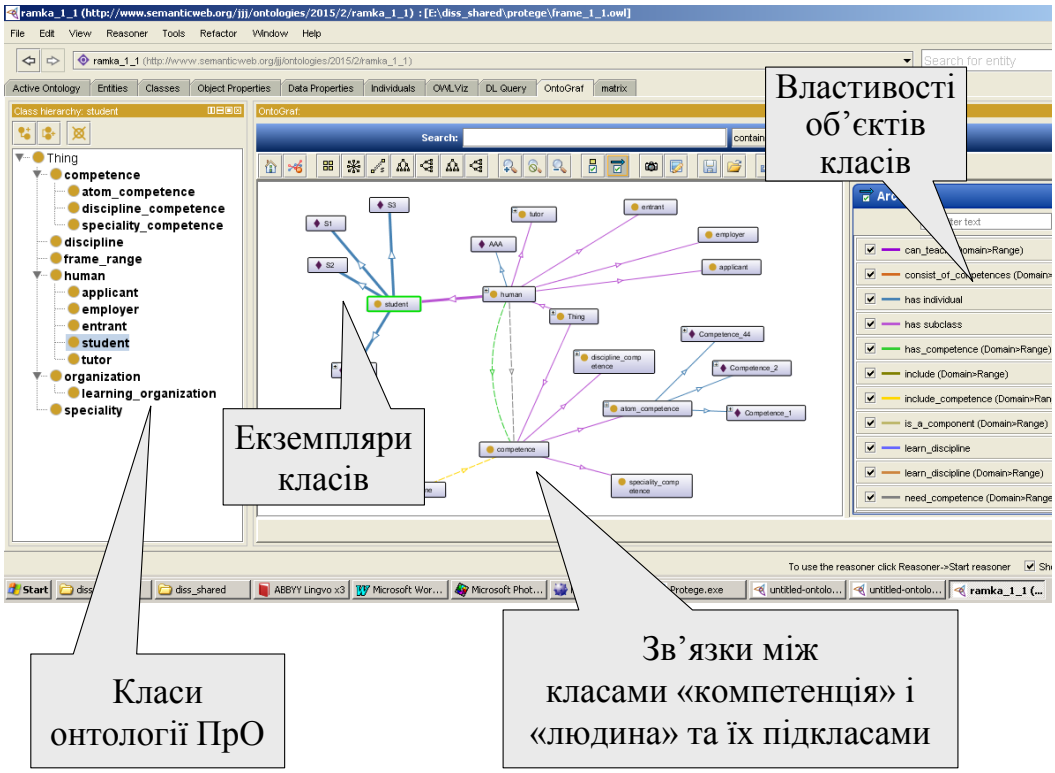



Рис. 4. 6. Онтологічна модель ПрО аналізу компетенцій

Це уможливлює здійснення опису екземплярів потрібного класу в термінах онтології ПрО. Для зіставлення спеціальностей, умінь і компетенцій людей і організацій різних країн доцільно використовувати набір еталонних атомарних екземплярів кожного класу. Екземпляр вважається атомарним, якщо жоден інший екземпляр цього класу не є його підмножиною.

Використання онтологічної моделі семантичного пошуку дає змогу ефективно забезпечити прикладні системи актуальними знаннями, презентованими у Web. Цей підхід дає можливість

ураховувати персональні особливості користувачів зі складними й стабільними інформаційними потребами, характерними для науково-дослідної діяльності й освітньої сфери.

**Висновки**

Використання онтологій для підтримки семантичного пошуку дає змогу значно підвищити його пертинентність, більш ефективно враховувати персональні відомості щодо користувачів та інформаційних ресурсів. Крім того, онтології уможливлюють структурування результатів пошуку, дослідження складних інформаційних об’єктів і здійснення семантичної розмітки знайденої інформації. За допомогою онтологій до пошуку можна залучати нові, динамічні відомості, доступні через Web.

# РОЗДІЛ V.
# ІНТЕЛЕКТУАЛІЗАЦІЯ WEB-СЕРВІСІВ НА ОСНОВІ ОНТОЛОГІЙ

## 5. 1. Сервіс-орієнтовані обчислення

*Сервіс-орієнтована архітектура* (СОА) – це концепція проектування, розробки й керування функціональних модулів (сервісів), кожен з яких доступний через мережу й здатний виконувати певні дії [321]. Одним з сучасних напрямів розвитку розподілених ІТ є сервіс-орієнтовані обчислення. Це обчислювальна парадигма, яка використовує сервіси як фундаментальні елементи для розробки застосувань. Вони базуються на СОА і забезпечують виконання операцій керування сервісами. Розробка таких систем – це процес пошуку, дослідження й композиції сервісів, що задовольняють вимогам користувача.

СОА створює комунікаційне середовище для модулів, що реалізовують прикладну бізнес-логіку. Інформація про модулі публікується в такій формі, що їх використання не потребує знань про використані в них рішення й технології. Розробник не повинен знати, як працює програма, необхідно лише розуміти, які вхідні й вихідні дані потрібні і як викликаються ці програми для виконання.

Можливість композиції сервісів часто розглядають як одну з основних переваг їхнього використання. Вона полягає в знаходженні набору елементарних сервісів, необхідних для реалізації функцій, використовуваних у запиті користувача, і визначенні порядку їх виконання.

SOA (Service Oriented Architecture) – це архітектурний шаблон програмного забезпечення, модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабко пов’язаних замінних компонентів, оснащених стандартизованими інтерфейсами для взаємодії за стандартизованими протоколами.

Технологія SOA для керування бізнес-процесами є великим кроком уперед з погляду підвищення ефективності розробки систем; за значущістю її можна порівняти зі створенням наприкінці 50-х років минулого століття компіляторів мови високого рівня. Цей підхід дає змогу спростити використання Web-сервісів з будь-якого місця розташування та їх виконання на основі бізнес-правил.

SOA змушує використовувати альтернативні технології та підходи (такі, як обмін повідомленнями) для побудови застосувань за допомогою зв’язування сервісів, а не за допомогою написання нового програмного коду. У такому разі, за належного проектування,

застосування повідомлень дасть змогу компаніям вчасно реагувати на зміну ринкових умов – налаштовувати процес обміну повідомленнями, а не розробляти нові програми.

SOA – це термін, який з'явився для опису виконуваних компонентів, зокрема таких, як Web-сервіси, що можуть викликатися іншими програмами, які виступають як клієнти або споживачі цих сервісів.

У найзагальнішому вигляді SOA припускає наявність трьох основних учасників:

- постачальника сервісу,
- споживача сервісу,
- реєстру сервісів.

Взаємодія учасників має досить простий вигляд: постачальник сервісу реєструє свої сервіси в реєстрі, а споживач звертається до реєстру із запитом.

Для використання сервісу необхідно дотримуватися угоди про інтерфейс для звернення до сервісу – інтерфейс не повинен залежати від платформи. SOA реалізує масштабованість сервісів – можливість додавання сервісів, а також їх модернізацію. Постачальник сервісу та його споживач виявляються непов'язаними – вони спілкуються за допомогою повідомлень. Оскільки інтерфейс не повинен залежати від платформи, то й технологія, що використовується для визначення повідомлень, також не повинна залежати від платформи. Тому, як правило, повідомлення є XML-документами, які відповідають XML-схемі.

## 5. 2. Концепція Web-сервісів

Концепція Web-сервісів виникла наприкінці 90-х років XX ст. і стала галузевим стандартом у сфері ІКТ. Стандарти Web-сервісів розроблені такими компаніями, як IBM, Microsoft, Ariba, Sun Microsystems, SAP за підтримки Консорціуму W3C. У межах W3C було створено робочу групу Web Services Architecture Working Group, яка опублікувала глосарій термінів у сфері Web-сервісів [462].

Web-сервіси використовують XML для обміну даними між застосуваннями, незалежно від використання операційної системи, апаратної платформи й розробника. Web-сервіс – це набір логічно пов'язаних функцій, які можуть бути програмно викликані через мережу Internet. Web-сервіс – це програма, що ідентифікується через URI, інтерфейс якої може бути подано у вигляді мови XML.

*Web-сервіси* – це реалізована програмними засобами система для підтримки міжкомп'ютерної взаємодії телекомунікаційних мереж,

що підтримується такими стандартами: SOAP (Simple Object Access Protocol) – протокол обміну повідомленнями; WSDL – мова опису програмних інтерфейсів Web-сервісів; UDDI (Universal Description, Discovery and Integration) – класифікатор Web-сервісів.

Інформація про те, які функції пропонує конкретний Web-сервіс, міститься в його описі – WSDL-документі. Інші системи взаємодіють з Web-послугами, використовуючи повідомлення в стандарті за протоколом SOAP, передані з використанням HTTP і XML, у поєднанні з іншими Web-стандартами.

Для пошуку Web-сервісів використовують спеціальні реєстри, що підтримують UDDI. Є два основні методи публікації Web-сервісів для користувачів – UDDI і DISCO. UDDI – це централізований структурований сервіс реєстру, а DISCO пропонує вільну форму механізму пошуку через браузер. Реєстр UDDI – центральне сховище для специфікацій та інформації про підприємства, з урахуванням послуг, які компанії надають через Internet.

### 5. 2. 1. SOAP

Web-сервіси стають доступними через протоколи HTTP GET, HTTP POST, HTTP SOAP.

SOAP (Simple Object Access Protocol) – стандарт передачі повідомлень через Internet, розроблений фірмою Microsoft для віддаленого виклику процедур (RPC, Remote Procedure Call) через протокол HTTP [421]. Він дає змогу передавати інформацію мережею у форматі XML. Можуть використовуватися: будь-яка мережа, будь-який протокол передачі даних, довільна інформація, різні обчислювальні пристрої (зокрема мобільні). Специфікація SOAP визначає XML-«конверт» для передачі повідомлень, метод для кодування програмних структур даних у форматі XML, а також засоби зв’язку через протокол HTTP [422].

### 5. 2. 2. WSDL

WSDL – заснований на XML стандарт опису того, як користуватися сервісом, запропонований Консорціумом W3C [453]. Опис Web-сервісу мовою WSDL містить технічні деталі, необхідні для інтеграції Web-сервісу із застосуванням (формат повідомлень, операції). На сьогодні WSDL підтримують продукт від Microsoft – SOAP Toolkit 2.0 (WSDL Generator) і продукт від IBM – WSDL Toolkit. Мова опису Web-сервісів (Web Services Description Language (WSDL)) визначає синтаксис того, як Web-сервіс може бути викликаний [230].

### 5. 2. 3. UDDI

Стандарт UDDI надає механізм виявлення Web-сервісів. UDDI формує бізнес-реєстр (UDDI Business Registry), у якому провайдери Web-послуг можуть реєструвати свої послуги, а розробники – відшукувати необхідні їм сервіси [446]. Компанії реєструють себе у Business Registry, який є базою даних загального користування. UDDI дає можливість відображати, інтегрувати й публікувати сервіси. UDDI сам є спеціалізованим Web-сервісом, що дає змогу користувачам і застосуванням знаходити необхідні їм сервіси [447].

Інтелектуальний пошук і автоматичне компонування Web-сервісів можуть бути здійснені за допомогою можливостей семантичного опису Web-сервісів, запропонованих у спеціалізованій онтології опису Web-сервісів OWL-S [383].

## 5. 3. Онтологічний опис Web-сервісів

### 5. 3. 1. Онтологічна модель Web-сервісів

*Інтелектуальні,* або *семантичні* Web-сервіси – елементи програмної логіки з однозначно відтворюваною семантикою, доступні через Інтернет і придатні для автоматичного пошуку, композиції й виконання з урахуванням їхньої семантики. Вони розширюють поняття звичайних Web-сервісів.

Для опису семантики Web-сервісів недостатньо звичайних засобів подання відомостей про Web-сервіс (приміром, стандарт WSDL, UDDI), хоча WSDL, наприклад, допускає наявність у цих описах довільного XML-вмісту, але необхідно розробити для опису Web-сервісів семантичних мов розмітки, подібні до тих, що розробляє Консорціум W3C для статичної частини Web: RDF, RDF Schema, OWL тощо (рис. 5. 1).

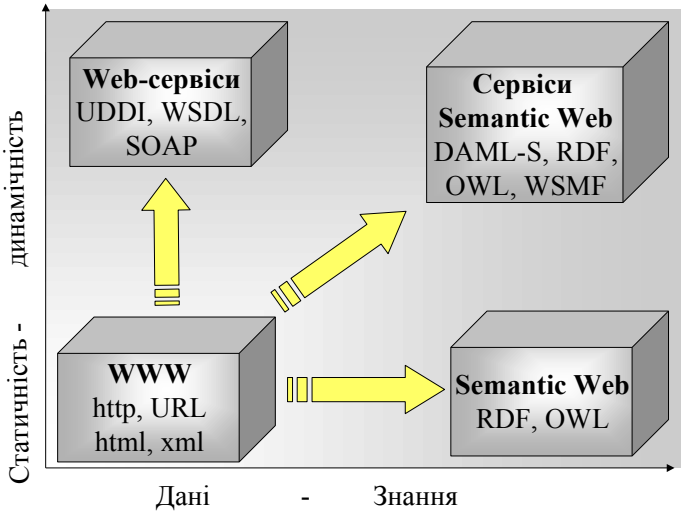



Рис. 5. 1. Процес інтелектуалізації Web-сервісів

### 5. 3. 2. OWL-S – мова семантичного опису Web-сервісів

OWL-S забезпечує онтологічний опис Web-сервісів. Мета розробки OWL-S полягає в тому, щоб зробити можливим використання логічного виведення для Web-сервісів, планування автоматичного компонування Web-сервісів, автоматичного використання сервісів програмними агентами [383].

OWL-S забезпечує декларативні описи властивостей Web-послуг і можливості, які можуть використовуватися для автоматичного виявлення сервісу.

Використовуючи OWL-S, Web-сервіс може повідомляти потенційним користувачам про свої функціональні можливості. Запит на обслуговування може бути узгоджений з оголошенням Web-сервісів за допомогою процесу підбору (matchmaking).

OWL-S забезпечує механізм для моделювання бізнес-процесів, проте відрізняється від нього виразністю термінів, уявлень, семантики, підтримки пошуку й виконання, обробки помилок. Опис OWL-S для сервісу складається з профілю сервісу, моделі сервісу й обґрунтування сервісу, тобто пояснення того, що виконує цей сервіс, як він працює, як можна отримати до нього доступ [153].

### 5. 3. 3. Профіль сервісу

OWL-S дає змогу здійснювати опис Web-сервісів у термінах *профілю*, що визначають, що саме робить сервіс; *моделі процесу*, що відображає, як сервіс працює, і *бази (Grounding)*, що допомагає з'ясувати, як одержати доступ до сервісу. Профіль і модель процесу є абстрактними специфікаціями в тому розумінні, що вони не здійснюють детальний опис окремих форматів, протоколів і мережних адрес, з якими зв'язаний Web-сервіс. Для забезпечення цих конкретних деталей призначена підстава. Мова опису Web-сервісів WSDL, розроблена безпосередньо від OWL-S, надає для цього цілком задовільні засоби. Тому автори OWL-S ухвалили рішення – використовувати його як основу для OWL-S. Передбачається, що поняття бази в OWL-S є розвитком поняття зв'язування (*binding*) у WSDL. Наслідком цього стає створення бази для атомарних процесів OWL-S як основного завдання.

*Профіль сервісу* – абстрактна характеристика функцій сервісу. Профіль сервісу будується на основі контенту UDDI, що відображає властивості сервісу, необхідні для його автоматичного виявлення, такі, наприклад, як входи й виходи, попередні умови та дії. На основі профілю, що надає інформацію про провайдера, з урахуванням функціональних можливостей і функціональних атрибутів сервісу

можуть бути створені описи й запити сервісу. Профіль Web-сервісу містить опис його властивостей: категорію сервісу (наприклад, за класифікацією UNSPSC) і його якісну оцінку (швидкість, надійність тощо).

З метою семантичного обґрунтування параметрів Web-сервісів застосовують онтології різного рівня. Використовують різну архітектуру для опису семантики джерел інформації: підходи, засновані на єдиній онтології, єдина глобальна онтологія, що забезпечує спільний словник для специфікації семантики; підходи, засновані на множинних онтологія х (кожне джерело інформації відображається за допомогою власної онтології); гібридні підходи, подібні до підходів, заснованих на множинних онтологіях у тому, що семантика кожного початкового тексту відображається за допомогою її власної онтології, але для того, щоб зробити локальні онтології порівняними, формується глобальний словник для загального використання.

Розмітка Web-сервісів за допомогою OWL-S полегшує використання Web-сервісу: його автоматизоване виявлення Web-сервісу, його виконання, взаємодію, компонування й контроль за виконанням.

Щоб використовувати Web-сервіс, потрібно інтерпретувати комп'ютером визначення сервісу й засоби доступу до нього.

OWL-S – онтологічна мова, призначена для опису Web-сервісів. Ця специфікація розроблена W3C на основі DAML-S і пропонує структуру мови, що базується на іншій розробці концерну W3C – мові подання онтологій OWL. OWL-S має допомогти користувачам і пошуковим агентам виявляти, викликати, компонувати й контролювати Web-сервіси. OWL-S – це онтологія сервісів, призначена для того, щоб допомогти користувачам і пошуковим агентам виявляти, викликати, компонувати й контролювати Web-сервіси.

OWL-S (Web Ontology Language for Services) – важливий крок у напрямі до більш інтелектуальних Web-сервісів, що забезпечує онтологічний опис Web-сервісу. OWL-S дає змогу використовувати логічне виведення для Web-сервісів, планувати компонування Web-сервісів і автоматично викликати сервіси програмними агентами.

Для інтероперабельного подання онтологічного опису сервісів розроблено модифікацію OWL – OWL-S, що забезпечує онтологічний опис Web-сервісу. Метою розробки OWL-S є надання можливостей для використання логічного виведення для Web-сервісів, планування компонування Web-сервісів, автоматичне використання сервісів програмними агентами. OWL-S забезпечує декларативні описи

властивостей Web-сервісу й можливості, що можна використовувати для автоматичного виявлення сервісу.

Для використання Web-сервісу програмні агенти мають визначити сервіс, що інтерпретується комп'ютером, і засоби доступу до цього визначення.

Щоб використовувати Web-сервіс, потрібно автоматизовано інтерпретувати визначення сервісу й засоби доступу до нього. OWL-S – це онтологія сервісів, що реалізує цю функціональність. Поштовхом до її розробки став проект Semantic Web, що намагається забезпечити доступ до інформаційних ресурсів Web не за ключовими словами, а за контентом. Одна з цілей Semantic Web – створення засобів, що дадуть змогу користувачам автоматично вибирати, використовувати, компонувати й відстежувати Web-сервіси. Важливим результатом у цьому напрямі стала розробка мов розмітки Web, таких, як OWL і його попередник DAML+OIL. Ці мови уможливлюють створення онтології для кожної предметної області (ПрО) і встановлення зв'язків з цими онтологіями Web-сайтів для їх опису.

Сервіси можуть бути *простими* в тому розумінні, що вони активізують тільки якусь одну доступну через Інтернет програму, сенсор чи пристрій, не використовуючи інші Web-сервіси, а тому взаємодії між користувачем і сервісом, крім простого виклику, немає. Сервіси можуть бути й *комплексними*, що складаються з кількох простих. Вони, звичайно, потребують взаємодії чи діалогу між користувачем і сервісами, тобто користувач може здійснювати вибір або накладати певні обмеження.

OWL-S призначений для підтримки обох категорій сервісів, але комплексні (складені) сервіси висувають більше вимог до властивостей мови.

Кожна доступна через Web програма, сенсор чи пристрій, що оголошені сервісом, можуть розглядатися як сервіс. OWL-S не виключає оголошення сервісом простих, статичних Web-сторінок. Однак основною метою OWL-S є підтримка більш складних задач.

### 5. 3. 4. Структура онтології OWL-S

Онтологія OWL-S структурує відомості про сервіси за трьома категоріями (рис. 5. 2):

- Профіль сервісу: клас SERVICE презентує SERVICEPROFILE;
- Модель сервісу: клас SERVICE відображається в SERVICEMODEL;
- База сервісу: клас SERVICE підтримується в SERVICEGROUNDING.

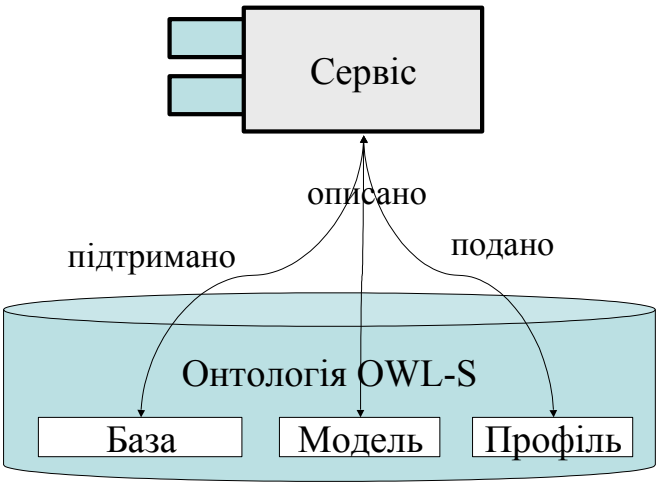



Рис. 5. 2. Подання онтології Web-сервісу в OWL-S

Для кожного опублікованого сервісу має бути один екземпляр класу SERVICE. Використовуючи OWL-S, Web-сервіс може повідомляти про свої функціональні можливості потенційним користувачам. Запит на обслуговування може бути узгоджений з оголошеннями Web-сервісів за допомогою процесу підбору (matchmaking). OWL-S забезпечує механізм для моделювання бізнес-процесів, але відрізняється від нього виразністю термінів, подань, семантики, підтримки пошуку й виконання, обробки помилок.

### 5. 3. 5. Задачі OWL-S

OWL-S дає змогу розв’язувати задачі чотирьох типів:

1. Автоматичне виявлення Web-сервісів передбачає автоматичне визначення місцезнаходження Web-сервісів, що надають певну послугу й задовольняють накладеним обмеженням. Наприклад, користувач хоче знайти дистанційний курс «Системи штучного інтелекту», у якому навчальні матеріали подані російською мовою, кількість годин не перевищує 60, а сам курс надається безкоштовно. Якщо сервіс розмічений за допомогою OWL-S, тоді семантична інформація, необхідна для знаходження сервісу, подана на сайті Web-сервісу у формі, що інтерпретується комп’ютером, отже, пошукова система, що здатна обробляти онтологічну інформацію, може знайти такий сервіс автоматично.

2. Автоматичний виклик Web-сервісу полягає в автоматичному виконанні ідентифікованого раніше Web-сервісу комп’ютерною програмою чи агентом. Наприклад, сервіс реєструє користувача як слухача певного дистанційного курсу й надає йому доступ до інтерактивних лекцій, а потім викликає модуль тестування. Виконання Web-сервісу можна розглядати як послідовність викликів функцій. Розмітка OWL-S Web-сервісу має забезпечувати декларативний, інтерпретований комп’ютером API для виконання цих

функціональних викликів. Програмний агент повинен бути здатним інтерпретувати розмітку, щоб зрозуміти, які вхідні дані потрібні для виклику сервісу, яка інформація буде повернута і як виконати сервіс автоматично.

3. Автоматична композиція і взаємодія Web-сервісів припускає автоматичний пошук, композицію та взаємодію Web-сервісів для розв'язання визначеної задачі, зумовленої високорівневим описом завдання. Наприклад, під час дистанційного навчання потрібно правильно визначити порядок досліджуваних дисциплін (від простіших до більш складних), а тестування повинно викликатися після вивчення курсу [36].

4. Автоматичний моніторинг виконання Web-сервісів дає змогу визначити, на якому етапі знаходиться процес виконання запиту й чи не виникли які-небудь непередбачувані перешкоди. Наприклад, користувач системи дистанційного навчання може одержувати інформацію про свої поточні оцінки, про те, скільки годин потрібно для завершення навчання, які нові курси з'явилися в ході його навчання тощо.

Для семантичного обґрунтування параметрів Web-сервісів використовують онтології різного рівня: онтології застосування, онтології ПрО й онтології верхнього рівня. Доцільно користуватися загальним словником (тезаурусом), що містить базові терміни ПрО, які використовуються в онтологіях застосувань (рис. 5. 3). Кожне поняття в такому тезаурусі може розкриватися через набір інших понять, що веде до появи семантичного поля.

Створення профілю сервісу – відношення з моделлю процесу. Профіль сервісу забезпечує коротке визначення зареєстрованого сервісу. Але після того, як користувач звертається до сервісу, він використовує модель процесу для керування взаємодією з сервісом. Профіль і модель – це два різні подання одного й того ж сервісу.

OWL-S не нав'язує які-небудь обмеження щодо моделей профілів і процесів, тобто обидва визначення можуть бути несумісними без впливу значущості виразу OWL.

### 5. 4. Пошук Web-сервісів на основі онтологій

*Web-сервіс* – це складний інформаційний об'єкт, структура якого визначається його моделлю та профілем. Для семантичних Web-сервісів така модель містить також опис семантики сервісу. Тому пошук і дослідження (discovery) Web-сервісів можна розглядати як окремий випадок розпізнавання ІО.

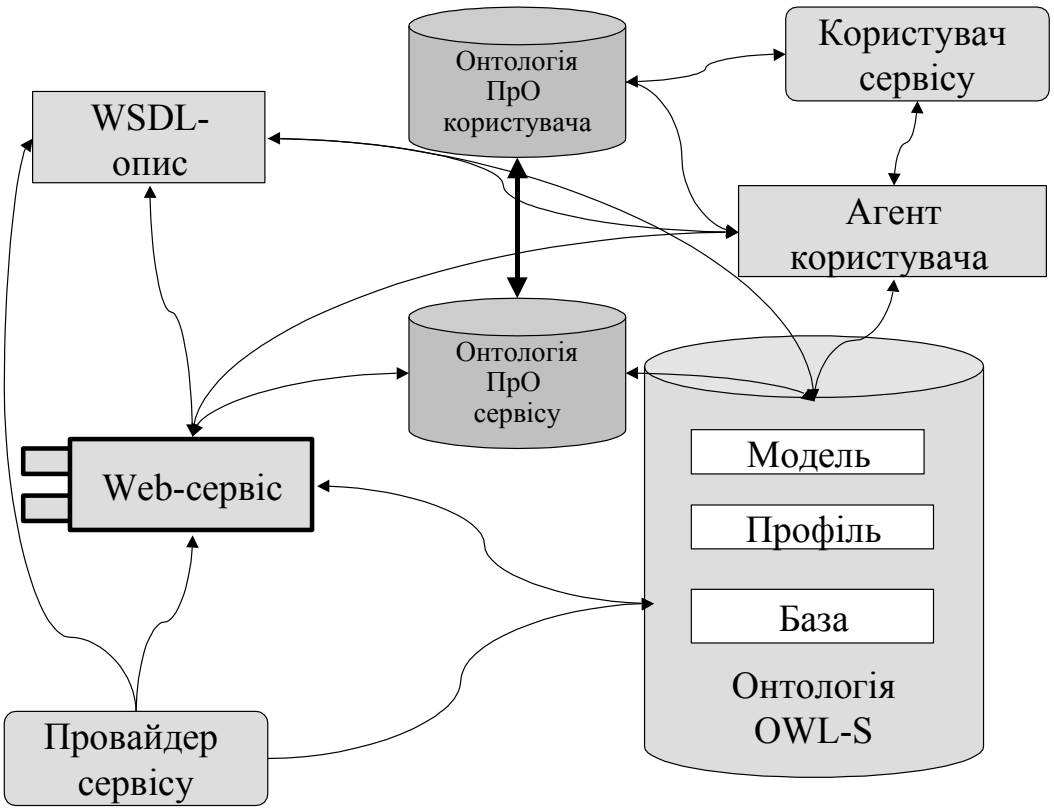



Рис. 5. 3. Засоби подання семантики Web-сервісу

Основою для цього розпізнавання є семантична розмітка елементів Web-сервісу, яка явно встановлює їх семантичні зв'язки (наприклад, ієрархічні). Семантична розмітка дає змогу дає можливість значно знизити обчислювальну складність розпізнавання ІО, оскільки замість задачі кластеризації розв'язується задача класифікації.У тому разі, коли для завдання семантики зв'язків між тегами використовується онтологія, то говорять про онтологічну розмітку.

Необхідно забезпечити Web-сервіси такими описами, щоб можна було автоматично розпізнавати їх зміст, тобто пов'язувати семантику Web-сервісу з певною предметною областю (ПрО) й знаннями цієї ПрО.

Отже, доцільно пов'язати засоби презентації Web-сервісів, що входять до стеку технологій СОА, з розробками Semantic Web, що базуються на онтологічному поданні знань (рис. 5. 4).

### 5. 4. 1. Семантична розмітка Web-сервісів

Web-сервіси є найсучаснішою спробою реконструкції великомасштабних розподілених обчислень. Вони базуються на стандартах, що є чинними на синтаксичному рівні й не мають можливостей для подання семантики. Семантика забезпечує більш якісні й масштабовані розв'язання для таких проблем, як інтероперабельність сервісів, виявлення й композиція сервісів і оркестровка процесів.

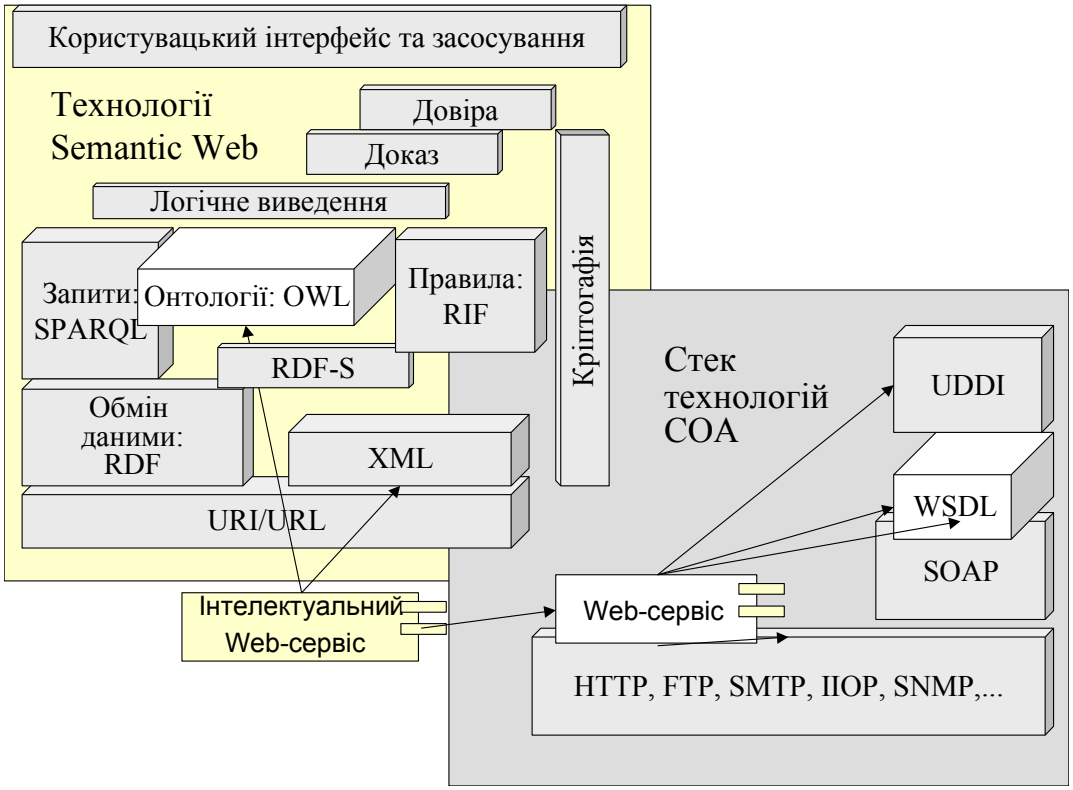



Рис. 5. 4. Інтелектуальні Web-сервіси як інтеграція технологій

Базова архітектура Web-сервісів реалізує сервіс-орієнтовану архітектуру (SOA), що забезпечує їх інтеграцію з внутрішніми чи зовнішніми застосовними програмами, містить у собі три ролі компонентів (клієнти, постачальники й реєстри), де постачальники презентують свої сервіси в реєстрах, а клієнти запитують реєстри для виявлення сервісів. В архітектурі Web-сервісів транспортом є Web. Web-сервіс – програмне забезпечення, призначене для взаємодії між інтероперабельними машинами через Інтернет.

Web-сервіси базуються на розширюваній мові розмітки XML, що являє собою їх базову технологію. Проте описи Web-сервісів (як правило, синтаксичні) здійснюються за стандартами UDDI, SOAP, WSDL.

Universal Description, Discovery and Integration (UDDI) – віртуальний реєстр, що надає інформацію про Web-сервіси. Simple Object Access Protocol (SOAP) – це протокол для обміну структурованою інформацією в децентралізованому й розподіленому середовищі.

Він використовує XML, щоб визначити розширювану інфраструктуру повідомлень, що забезпечує обмін складеними (constructed) повідомленнями за допомогою різних базових протоколів. Протокол SOAP не залежить від якої-небудь конкретної моделі програмування й від якої-небудь конкретної семантики реалізації. Мова

Web Service Description Language (WSDL) забезпечує модель і формат XML для опису Web-сервісів. Він відокремлює опис абстрактної функціональності, пропонованої сервісом, від конкретних деталей опису сервісу. WSDL здійснює опис тільки синтаксичного інтерфейсу Web-сервісу. Таким чином, чистий WSDL не може бути використаний для автоматичної композиції Web-сервісів: для того, щоб зробити інформацію доступною для автономних агентів, потрібна семантична анотація.

Різні підходи стосовно додавання семантики до Web-сервісів запропоновано в [210, 214, 215, 406]. Вони по-різному базуються на анотуванні Web-сервісів за допомогою онтологій, з використанням методів зіставлення онтологій.

***Таблиця 5. 1.***

Підходи до анотації Web-сервісів

| Посилання | Елементи | Ресурс анотації | Метод | Інстру-мент |
|---|---|---|---|---|
| Belhajjame K., Embury S-M., Paton N-W., Stevens R. and Goble C-A. «Automatic annotation of web services based on workflow definitions» | Параметри операцій | Потоки робіт | Правила сумісності параметрів | Редактор анотацій |
| Bouchiha D., Malki M., Alghamdi A. and Alnafjan K. «An Empirical Approach for Annotating Web Services» | Імена складних типів і операцій | Онтологія домену | Зіставлення онтологій | SAWSDL Builder |
| Hess A., Johnston E. and Kushmerick N. «ASSAM: A tool for semi-automatically annotating semantic Web services» | Операції, частини повідом-лень і дані | Онтологія домену | Методи класифіка-ції тексту | ASSAM |

| | | | | |
|---|---|---|---|---|
| Patil S., Oundhakar A. Sheth and V. Kunal. «METEOR-S Web service annotation framework». | Дані (входи і виходи сервісів) | Онтологія домену | Методи зіставлення схем | MWSAF tool |
| Grcar M. and Mladenic D. «Visual OntoBridge: Semi-automatic Semantic Annotation Software» | Запит природною мовою | Онтологія домену | Методи Text mining | Visual OntoBridge (VOB) |
| Lerman K., Plangprasopchok A. and Knoblock C-A. «Automatically labeling the inputs and outputs of web services» | Дані (входи і виходи сервісів) | Метадані (WSDL) | Методи машинного навчання | Інстру-мент семантичних міток (labelling) |
| Bowers S. and Ludascher B. «A calculus for propagating semantic annotations through scientific workflow queries» | Анотація і запит | Workflow | Метод поширення (Propaga-tion) | Інстру-мент семантичної розмітки Prolog |
| Carman M-J. and Knoblock C-A. «Learning Semantic Definitions of Online Information Sources» | Визначення Datalog | Визначен-ня джерел | Індуктив-ний логічний пошук | EIDOS |

У таблиці 5.1 наводяться такі характеристики підходів до анотації Web-сервісів:

- стовпчик «Посилання» вказує на дослідників, що запропонували цей підхід;
- стовпчик «Метод» відповідає підходу до питання;
- стовпчик «Елементи» містить опис елементів, що враховуються в процесі анотацій;

- стовпчик «Ресурс анотації» вказує на модель, з якої «витягаються» семантичні анотації;
- стовпчик «Інструмент» містить вказівку на інструмент, що підтримує підхід.

### 5. 4. 2. Анотування Web-сервісу термінами онтології ПрО

Різні дослідження пропонують для семантичного опису Web-сервісів такі засоби, як: OWL-S, Web Service Modelling Language (WSML), Web Service Modelling Ontology (WSMO), Web Services Description Language Semantic (WSDL-S) і Semantic Annotations for Web Services Description Language (SAWSDL).

Вони враховують наступні критерії опису:

- *ресурс*: семантичний опис (XML-схеми, WSDL, UDDI );
- *властивість* (Property): що саме семантично відображено в документі функціональні властивості, такі, як входи, виходи, або ж не функціональні властивості, такі, як виконання (Implementation);
- *мова:* мова подання семантичної моделі (OWL, WSML);
- *анотація*: чи зберігаються анотації всередині документа (внутрішні) чи є незалежними (зовнішніми);
- *модель*: є модель семантичної області внутрішньої чи зовнішньої;
- *співставлення*: тип алгоритму зіставлення та його властивостей.

Процес анотування Web-сервісів складається з двох етапів:

*1 етап* – етап категоризації, що дає змогу класифікувати WSDL-документи у відповідний їм домен;

*2 етап* – етап зіставлення, що дає можливість зв’язати кожну сутність із документів WSDL з відповідною сутністю в онтології домену.

Процес *категоризації* передбачає класифікацію WSDL-описів сервісу у відповідний домен. Для цього опис сервісу розбивається на основні елементи WSDL (типи даних XSD, інтерфейс, операції та повідомлення). Перелік понять також витягується з кожної онтології. Подібності між двома наборами на основі ступеня подібності між двома сутностями обчислюється для того, щоб визначити, які поняття онтології будуть зберігатися для подальшого процесу. Обрана онтологія вказує на домен WSDL чи категорію.

Процес *співставлення* передбачає проектування елементів WSDL на поняття онтології. Подібності між елементом WSDL і поняттям обраної онтології обчислюються для того, щоб визначити, яке поняття буде зв’язано з початковим елементом WSDL. Ця операція повторюється для всіх елементів WSDL.

Як категоризація, так і співставлення використовують методи зіставлення онтологій.

Мета такого зіставлення онтологій – знайти відношення між сутностями, виражені в різних онтологіях. Дуже часто ці відношення є відношеннями еквівалентності, що виявляються через ступінь подібності між сутностями онтологій (рис. 5. 5).

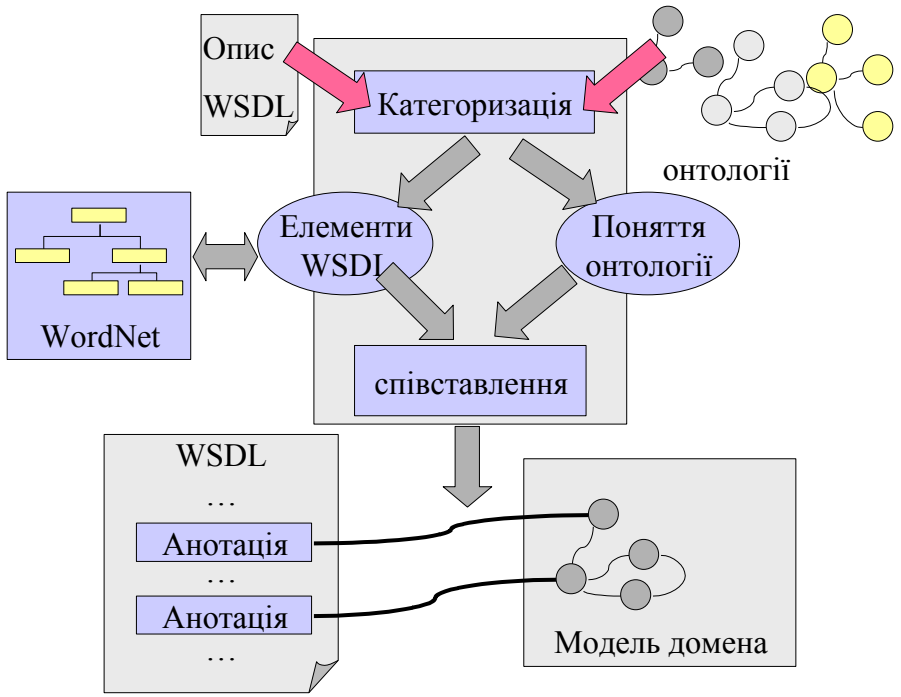



Рис. 5. 5. Процес анотування Web-сервісу термінами онтології ПрО

Для завершення процесу співставлення онтологій та їх відповідності використовуються ступені подібності (similarity measure) між об’єктами. Ступінь подібності спрямований на кількісну оцінку того, наскільки дві сутності подібні.

Як ступінь подібності можна використовувати оцінку на основі WordNet [387]. WordNet являє собою лексичну базу даних.

Ступені подібності, що залежать від WordNet, можна поділити на три категорії:

- ступені подібності на основі довжини шляху між поняттями;
- ступені подібності на основі інформаційного змісту;
- ступені зв’язаності на основі типу відношень між поняттями.

Коли доступний набір онтологій, подібність між двома наборами обчислюється шляхом порівняння набору сутностей файлу WSDL і набору сутностей кожної онтології. На основі таких ступенів системи ухвалюють рішення про те, між якими онтологіями запустити алгоритм зіставлення. Обрана онтологія предметної області визначає категорію файлу WSDL. Цей процес називається процес категоризації.

Онтологія розглядається як набір сутностей (понять), і файл WSDL також як набір сутностей (типів даних XSD, інтерфейсу, операцій, повідомлень). Деякі стратегії можуть бути адаптовані до обчислення подібності між двома наборами.

У процесі використання для іменування параметрів Web-сервісів багатозначних термінів, що застосовуються в різних ПрО (наприклад, «ім'я», «оцінка», «адреса»), автоматична класифікація Web-сервісів або вкрай неточна, або взагалі не можлива. Тому доцільно, щоб розроблювачі Web-сервісів явно вказували простір імен, що використовуються на позначення вхідних і вихідних параметрів їх сервісів. Наприклад, як джерела таких імен можуть використовуватися схеми RDF-S чи OWL.

У такому разі: 1) Web-сервіс однозначно може бути класифікований за допомогою відповідної онтології; 2) між термінами (і позначеними цими термінами параметрами Web-сервісів) уже встановлені семантичні зв'язки, що можуть оброблятися автоматизовано не тільки для пошуку необхідних користувачеві сервісів, а й для композиції з них складених сервісів.

### 5. 4. 3. Дослідження семантичних Web-сервісів на основі логічного виведення в дескриптивних логіках

У [242] для пошуку Web-сервісів пропонується підхід, що базується на дескриптивних логіках. Окремі сервіси подаються в термінах DL, а потім над ними здійснюється логічне виведення.

Як правило, підходи на основі DL використовують стандартні сервіси для логічного виведення для DL систем – підпорядкування (subsumption) й здійсненності (satisfiability) для пошуку відповідності потенційних партнерів в електронних транзакціях. Якщо постачання відображає концепція Sup, а попит – концепт Dem, нездійсненність кон'юнкції Sup і Dem визначає несумісність пропозицій, а здійсненність ідентифікує потенційних партнерів, що можуть домовитися за певних обмежень, і категоризація між Sup і Dem означає, що вимоги з Dem повністю виконуються в Sup.

Класифікація в сумісних і несумісних зіставленнях може бути марною за наявності кількох сумісних джерел; потрібно визначити, як ранжувати найбільш перспективні з них. Також слід з'ясувати, яке пояснення щодо мотивації такого рангу може бути оцінене. З іншого боку, коли бракує сумісних зіставлень, то можна прийняти для включення несумісні зіставлення, які ще можуть бути цікавими, якщо переглянути деякі з оригінальних вимог, що висуваються в запиті, якщо можна досить легко ідентифікувати такі вимоги.

Іншими словами, потрібний якийсь метод, щоб забезпечити оцінку на основі логіки як для сумісних, так і для несумісних зіставлень і, зрештою, забезпечити часткову або повну впорядкованість, даючи можливість користувачеві чи автоматизованому агентові вибрати найбільш перспективні з цих зіставлень. Крім того, потрібно надати оцінку, щоб забезпечити логічні пояснення отриманого бала, що дає змогу зрозуміти результат оцінювання й полегшити подальшу взаємодію з уточнення або перегляду запиту.

Хоча цей процес досить простий для людини, але не є таким для повністю автоматизованого логічного виведення. Тому необхідно визначити немонотонні сервіси міркування в термінах DL, щоб мати справу з наближенням і ранжуванням. Для цього можна використовувати абдукцію понять (Concept Abduction) і скорочення понять (Concept Contraction) як сервіси, які допомагають позитивно розв'язати окреслені вище питання. Для цього використовуються:

- логічна структура для висловлювання запитів і пропозицій у термінах опису поняття й властивостей, які мають задовольняти посередників із зіставлення (matchmaking facilitator);
- абдукція понять як логічна основа для ранжування сумісних пропозицій і створення логічних пояснень для результатів ранжування;
- скорочення понять як логічна основа для ранжування несумісних зіставлень, спрямоване на виявлення найбільш перспективних «близьких до мети промахів», і створення логічних пояснень для результатів ранжування;
- алгоритми, що реалізують формалізоване виведення для цілей зіставлення й результатів складності для класу задач зіставлення;
- опис системи, що реалізує семантичні сервіси зіставлення та її оцінки.

Такий підхід має глибоке теоретичне підґрунтя з дескриптивних логік і може надавати корисні й зважені результати. Проте він має надто складний як для використання, так і для розуміння користувачами вигляд, а тому потребує подальшого розвитку й реалізації.

### 5. 5. Дескриптивні логіки та Web-сервіси

Зазначимо, що, незважаючи на величезну кількість доступних сервісів, автоматично знайти серед них коректний Web-сервіс, який задовольнятиме вимоги конкретного користувача, не завжди можливо. Через це виникає потреба в засобах, які дали б змогу сформувати набір

з наявних сервісів, що частково задовольняють цим вимогам, та інтегрувати їх так, щоб задовольнити функціональність користувача. Така інтеграція називається *композицією* Web-сервісів. Потрібно, щоб така операція композиції виконувалася автоматично за інтерактивної взаємодії з користувачем на основі набору наявних і доступних сервісів і цільового запиту.

Ураховуючи велику кількість сервісів, які треба інтегрувати для реалізації функціональності сучасних прикладних систем, питання автоматизації процесу побудови композиції стає критичним для забезпечення масштабованості сервіс-орієнтованих систем. Для цього, насамперед, потрібно доповнити моделі сервісів семантикою, яка може бути потім використана механізмами міркувань під час розв'язання різних задач життєвого циклу сервісів. Необхідність реалізації автоматизованого логічного виведення зумовлює доцільність використання в алгоритмах розв'язання задач Web-сервісів дескриптивних логік, що надають готові, уже досить розвинені механізми міркувань [79].

У [80] поданням Web-сервісу S розглядається як шістка, $S = \langle CI, I.A, AO, O, CO \rangle$, де CI – передумови сервісу, I – список вхідних параметрів, A – ефекти сервісу, AO – об'єкти, що виробляються як ефекти сервісу, O – список вихідних параметрів і CO – післяумови сервісу.

Одним із підходів до ефективного розв'язання задачі виявлення Web-сервісів на базі їх можливостей є завдання знаходження найкращого покриття, що використовує апарат дескриптивних логік.

Якщо задано репозиторій Web-сервісів і отримано запит щодо потреби в сервісі з певними властивостями, тоді автоматичне знаходження в цьому репозиторії Web-сервісу, що відповідає запиту, є завданням виявлення Web-сервісу. Валідні запиту Web-сервіси мають задовольняти таким умовам:

- генерувати хоча б один вихідний параметр, указаний у запиті, й задовольняти всім передумовам запиту;
- мати вхідні параметри тільки з наданого в запиті списку вхідних параметрів і задовольняти передумовам запиту;
- виробляти ефекти запиту.

Деякі рішення можуть бути занадто обмеженими, але вони все одно розглядаються як чинні, якщо задовольняють вимогам вхідних і вихідних параметрів, перед і постумів та ефектів.

Дескриптивні логіки дають змогу презентувати домен, що цікавить користувача, у термінах концептів або описів (унарні предикати), що характеризують підмножини об'єктів (екземплярів) у домені, й ролей (бінарні предикати) у такому домені. Таким чином,

найменше загальне включення множини концептів відповідає найбільш специфічному опису, що містить усі концепти з множини.

Внутрішні описи, що містяться в БЗ і побудовані з використанням DL, називаються термінологією.

Якщо A – ім'я концепту, а C – опис концепту, тоді A=C – це визначення концепту. Термінологія T – це скінчена множина визначень концептів, таких, що кожне ім'я концепту трапляється в лівій частині визначення не більше одного разу.

Процес виявлення сервісів можна розглядати знайти Web-сервіси для виконання конкретних як процес перезапису (rewriting), у якому запит Q перезаписується найближчим до нього описом E, який виражається кон'юнкцією Web-сервісів заданої онтології T (онтології, на базі якої визначаються сервіси й запит). Такий опис і є найкращим покриттям запиту.

Щоб знайти Web-сервіси для виконання конкретних задач у бізнес-процесі, запитувач повинен, насамперед, визначити умови, яким повинен задовольняти цей Web-сервіс. Такий підхід називається зіставленням на основі цілі (goal based matchmaking) [206, 207] і пов'язаний з визначенням умов на оголошення Web-сервісу (вхідні й вихідні параметри, передумови та ефекти) і перевіркою того, чи здатний Web-сервіс задовольняти цим умовам.

Через низьку ймовірність знаходження сервісу, повністю еквівалентного запиту, для відбору множини сервісів, які становитимуть покриття, потрібний гнучкий механізм зіставлення. Таку гнучкість може забезпечити використання операції різниці на описах сервісів. Використання операції різниці дає можливість витягти з підмножини описів Web-сервісів частину, що є семантично спільною з сервісом, що запитується, і частину, що семантично відрізняється від запиту.

Зауважимо, що на практиці застосовують два рівні презентації Web-сервісів – функціональну й процесну модель сервісів [215]. На *функціональному* рівні сервіси розглядаються як окремі, презентовані в мережі прикладні програмні компоненти, які можна викликати шляхом відправлення повідомлень. Отримавши таке повідомлення, сервіс має виконати своє завдання і, якщо потрібно, згенерувати відповідь для того, хто його викликав. Таким чином, між запитувачем сервісу й самим сервісом безперервної взаємодії немає. Опис таких Web-сервісів фокусується на їх функціональності в термінах імені сервісу, імен операцій, імен повідомлень (що відомі також як вхідні й вихідні повідомлення /параметри), імені інтерфейсу .

Сервіси *процесного* рівня складаються з набору операцій, які відповідають загальній поведінці сервісу. Тому вони потребують розширеної взаємодії між запитувачем сервісу й множиною операцій,

забезпечуючи конкретну функціональність. Це, як правило, композитні сервіси, і взаємодія з ними не може бути зведена лише до запиту й відповіді на запит – для досягнення потрібного результату такі сервіси повинні діяти за значно складнішим протоколом.

Функціональна модель сервісу відображає його функціональність і визначає способи взаємодії клієнта з цим сервісом для використання його функціональних можливостей. Сервіс розглядається як атомний елемент, а композитний сервіс – як пов'язана сукупність атомних елементів. Процес, що відбувається всередині сервісу, не аналізується.

Можна визначити такі базові завдання життєвого циклу сервісу на функціональному рівні, для виконання яких доцільне використання апарату DL [80]:

- формалізація Web-сервісів на функціональному рівні (передумови й ефекти сервісу задаються через твердження DL);
- виявлення Web-сервісів;
- визначення цілі виявлення як кон'юнктивного запиту, що є кон'юнкцією тверджень DL;
- зіставлення пошукового запиту й Web-сервісів на функціональному рівні, перевірка відповідності їх передумов і ефектів для дослідження цілі й перевірка істинності передумов;
- верифікація Web-сервісів перевірка відповідності сервісу його специфікації за допомогою різонерів DL;
- перевірка нефункціональних характеристик сервісів, таких, як вартість, ступінь довіри до сервісу, репутація сервісу тощо.

Опис Web-сервісу відповідає запиту, якщо запит є достатньо близьким до сервісу, щодо якого надано запит. Якщо висунути вимогу, щоб опис сервісу й запит повністю збігалися, то така ситуація призведе до того, що відповідності виявлятимуться дуже рідко. У [79] наводиться алгоритм, який дає змогу зіставляти оголошення Web-сервісів із запитами.

Через це виникає потреба в більш гнучкому визначенні достатньої близькості запиту й сервісу. Для цього необхідні механізми зіставлення, які розпізнають ступінь подібності між оголошеннями сервісів і запитами, і можливість визначати ступінь близькості, яка задовольнила б запитувача. Чим меншою є гнучкість, тим менша вірогідність знаходження сервісів, що задовольняють вимогам. Для цього механізм зіставлення має підтримувати гнучке семантичне зіставлення між оголошеннями сервісів і запитами на основі онтологій, що доступні сервісам і механізму зіставлення; запитувач повинен мати можливість контролювати ступінь гнучкості зіставлення; механізм зіставлення має контролювати чесність як запитувача, так і провайдера сервісу щодо питань вартості сервісу; а сам процес зіставлення повинен

бути ефективним і не навантажувати запитувача надмірними затримками.

**Висновки**

Онтологія OWL-S забезпечує механізм для моделювання бізнес-процесів, але відрізняється від нього виразністю термінів, уявлень, семантики, підтримки пошуку й виконання, обробки помилок. Опис OWL-S для сервісу складається з профілю сервісу, моделі сервісу й обґрунтування сервісу, тобто пояснення того, що виконує цей сервіс, як він працює, як можна отримати до нього доступ. Саме ця структура може використовуватися як основа для інтелектуального аналізу відомостей щодо сервісів і застосування методів Data Mining для автоматизованого компонування сервісів відповідно до завдань, сформульованих користувачами.

# РОЗДІЛ VI.
# ВИКОРИСТАННЯ ОНТОЛОГІЙ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ У WEB

Одним з основних факторів розвитку сучасного інформаційного суспільства є застосування знань, що зумовлює потребу в засобах їх здобуття, накопичення, пошуку й аналізу [351, 408] з наявних інформаційних ресурсів.

Значна частина знань, що людство здобуло в процесі свого розвитку, нині доступна за допомогою відкритого інформаційного простору Web [339, 424]. Але саме великий обсяг цієї інформації стає проблемою – користувачам усе важче знайти потрібну інформацію й проаналізувати її з погляду своїх потреб [74, 172].

Для того, щоб Web перетворився на глобальну базу знань, придатну для автоматизованої обробки, необхідно розробляти й використовувати методи інтелектуального аналізу інформаційних ресурсів Web, що враховують динамічну, гетерогенну й розподілену природу цього середовища й здатні генерувати знання в інтероперабельній формі, приміром, поповнювати й удосконалювати онтології. Це зумовлює необхідність створення автоматизованих засобів інтелектуального аналізу даних для поповнення й удосконалення онтологій знаннями, здобутими з IP Web [360].

*Інтелектуальний аналіз інформації* – процес виявлення придатних до використання закономірностей у великих наборах даних. Зазвичай, такі закономірності неможливо знайти безпосередньо, без використання спеціальних методів через те, що такі закономірності досить неочевидні, а для їх виявлення потрібно аналізувати надмірний для людини обсяг даних.

Саме це зумовило розвиток технологій і програмних засобів Data Mining та їх різновидів, що призначені для аналізу специфічних видів даних (природномовних текстів, Web-контенту, структурованої інформації тощо) та генерації певних видів знань. Отже, Data Mining можна розглядати як джерело знань для онтологічного аналізу.

Виявлення нових знань є складною проблемою, що потребує багато часу й обчислювальних ресурсів. Тому доцільно, щоб здобуті знання були придатними для повторного застосування. Це зумовлює необхідність пов’язувати методи інтелектуального аналізу інформації з онтологічним аналізом, тобто зберігати отримані закономірності в онтологіях.

## 6. 1. Технології Data Mining як засіб здобуття знань

*Data Mining* [76] – це процес пошуку закономірностей (кореляцій, тенденцій, взаємозв'язків), що використовує різноманітні математичні й статистичні алгоритми – кластеризації, класифікації, регресійного, кореляційного аналізу тощо [6] і на їх основі здобуває з даних відношення між різними групами інформаційних об'єктів. Саме такі відомості містяться в онтологіях ПрО і дають змогу більш ефективно розв'язувати застосовні задачі.

Технологія Data Mining тісно пов'язана з виявленням знань у базах даних (knowledge discovery in databases – KDD). KDD – це процес отримання з інформації, що зберігається в базі даних, нової, потенційно корисної інформації про предметну область, що містить накопичення сирих даних. Це також відбір, підготовка, перетворення даних, пошук закономірностей у даних, оцінка, узагальнення й структурування знайдених закономірностей.

Розвиток KDD багато в чому зумовлений швидким розвитком різноманітних сховищ даних (data warehouse) – збирання даних, які відрізняються предметною орієнтованістю, інтегрованістю, підтримкою хронології, незмінюваністю, а також ускладненням структури інформаційних ресурсів Web і призначені для подальшої аналітичної обробки.

Data Mining відрізняються від засобів OLAP (On-Line Analytical Processing) тим, що замість перевірки передбачуваних користувачем залежностей вони на основі наявних даних самі здатні розробити моделі, які дають змогу кількісно оцінити ступінь впливу різних досліджуваних факторів на задану властивість. Крім того, засоби Data Mining дають змогу створювати нові гіпотези про характер невідомих, але реально наявних відношень у даних.

Найбільш поширеною задачею, що розв'язується за допомогою Data Mining, є задача *класифікації*: розв'язання задачі класифікації дає змогу виявити ознаки, що характеризують групи об'єктів досліджуваного набору даних – класи, за якими новий об'єкт можна зарахувати до того чи іншого класу.

Ця задача безпосередньо пов'язана з онтологічним аналізом і дає змогу зарахувати екземпляри до відповідних класів.

Для розв'язання задачі класифікації можуть використовуватися такі методи: найближчого сусіда (Nearest Neighbor); k-найближчого сусіда (k-Nearest Neighbor); Байєсівські мережі (Bayesian Networks); індукція дерев рішень; нейронні мережі (neural networks).

Задачу *кластеризації* можна розглядати як логічне продовження ідеї класифікації, що полягає в розподілі множини об'єктів на групи

(кластери), при цьому в кожному кластері зібрані об'єкти, які схожі за параметрами. Варто зауважити, що на відміну від класифікації, кількість кластерів та їхніх характеристик визначають у процесі побудови кластерів, виходячи зі ступеня близькості поєднуваних об'єктів за сукупністю параметрів.

В онтологічному аналізі ця задача постає на попередньому етапі й дає змогу побудувати набір базових класів онтології й встановити між ними ієрархічні відношення.

Задача *асоціації* – задача пошуку асоціативних правил (визначення взаємозв'язків), що полягає у визначенні наборів об'єктів, які часто трапляються серед множини подібних наборів. Відмінність асоціації від двох попередніх задач Data Mining: пошук закономірностей здійснюється не на основі властивостей аналізованого об'єкта, а між декількома подіями, що відбуваються одночасно.

Інші поширені задачі Data Mining – задачі прогнозування, асоціації, визначення відхилень тощо – також можуть застосовуватися для вдосконалення онтологій шляхом обробки даних відповідних ПрО, доступних через Web.

Головна цінність Data Mining – це практична спрямованість цієї технології, що забезпечує шлях від сирих даних до конкретних знань (зокрема й онтологічних), від постановки задачі до готового застосування, за підтримки якого можна ухвалювати рішення.

Більшість аналітичних методів, що використовується в технології Data Mining – це відомі математичні алгоритми й методи. Новим у їх застосуванні є можливість їхнього використання в процесі розв'язання тих чи інших конкретних проблем, зумовлена новими можливостями сучасних технічних і програмних засобів. Зауважимо, що більшість методів Data Mining була розроблена в рамках теорії штучного інтелекту. Основною особливістю Data Mining є поєднання широкого математичного інструментарію (від класичного статистичного аналізу до нових кібернетичних методів) і останніх досягнень у сфері інформаційних технологій. Технології Data Mining інтегрують суворо формалізовані й неформальні методи аналізу, кількісний і якісний аналіз даних.

## 6. 2. Особливості інтелектуального аналізу інформації у Web

Зростання обсягу Web-ресурсів і ускладнення його структури зумовило широке застосування методів Data Mining для здобуття знань з ресурсів Web, які містяться в їх контенті й структурі неявно. Методи Data Mining, що пристосовані до специфіки аналізу неструктурованої, неоднорідної, розподіленої і значної за обсягом інформації, яка

міститься у Web-вузлах, називаються *Web Mining*. Різні дослідники працювали над застосуванням методів Data Mining у Web уже наприкінці 90-х років XX століття, однак особливої актуальності такі дослідження набули останніми роками. Це пов'язано зі швидким поширенням Інтернету завдяки поширенню мобільних засобів доступу й бездротових технологій [6, 98].

На основі Web Mining можна будувати й поповнювати онтології, що відображають взаємодію між користувачами й різноманітними Web-застосуваннями, інтероперабельно переносячи персональні дані до інших інтелектуальних систем.

Web Mining є основою для персоніфікації Web-застосувань [1, 229]. Проаналізувавши поведінку відвідувача сайту, його попередні відвідування та інші дані, за допомогою Web Mining можна класифікувати користувача як потенційного покупця або кредитного шахрая й запустити певний сценарій роботи з цим клієнтом [111, 124].

Для аналізу інформації про користувача слід якнайменше використовувати декларовану ним інформацію, а краще спиратися на стійкі шаблони його поведінки в мережі – вибір ресурсів, послідовність дій усередині ресурсу, переходи на інші ресурси, періоди мережної активності, здійснювані покупки тощо [149].

Це, наприклад, дає змогу визначити характеристики клієнтів Web-магазину, які приносять найбільший прибуток, а потім застосувати ці знання для формування персональних рекламних оповіщень.

Розроблено багато підходів до виконання завдань щодо виявлення знань з шаблонів навігації користувачів.

Під час застосування алгоритмів інтелектуального аналізу даних, у процесі пошуку шаблонів поведінки користувача у Web найчастіше використовують такі методики:

- *кластеризація* – пошук груп схожих відвідувачів, сайтів, сторінок тощо;
- *асоціація* – пошук спільно запитуваних сторінок або замовлених товарів;
- *аналіз послідовностей* – пошук послідовностей дій (найчастіше застосовується варіант алгоритму apriori, розробленого для аналізу частих наборів, але модифікованого для виявлення частих фрагментів послідовностей і переходів).

Особливо цікавим є підхід кластеризації послідовностей – пошук груп користувачів зі схожими послідовностями дій. На першому етапі в цьому підході виділяються послідовності класифікованих дій користувача, наприклад, у межах однієї сесії. Потім підраховуються частоти переходів між різними діями для складання Марківського ланцюга заданого порядку. На завершальному етапі отримані

Марківські ланцюги кластеризуються для виявлення груп зі схожими частотами переходів.

Для прогнозування наступної дії користувача спочатку на підставі історії його дій у межах сесії визначається група, до якої він належить з найбільшою ймовірністю. Потім визначається дія, яка виконується з найбільшою вірогідністю в цій групі з урахуванням останніх дій цього користувача.

У Web Mining можна виділити такі напрями, як Web Content Mining і Web Usage Mining, Opinion Mining.

*Web Content Mining* забезпечує автоматичний пошук і здобуття корисної інформації з різноманітних інформаційних ресурсів Web, що, зазвичай, містять велику кількість «інформаційного шуму». Аналізується зміст документів: знаходяться схожі за змістом слова та виявляється їх кількість. Потім розв'язується задача *кластеризації* або *класифікації*, що дає змогу згрупувати IP за змістовною близькістю [191, 112]. З погляду онтологічного аналізу це дає змогу знаходити Web-ресурси, що релевантні певній ПрО й здобувати з них потрібні знання, а також поновлювати відомості про екземпляри онтології. Приміром, це дає можливість знайти відповідні сторінки Вікіпедії й здобути з них найбільш актуальні відомості про певних осіб, організації або публікації.

*Web Usage Mining* аналізує інформаційні потоки в глобальних комп'ютерних мережах, виявляє закономірності в діях користувача Web-сайту або групи сайтів. Для цього робиться аналіз того, які сторінки переглядав користувач і якою була послідовність їх перегляду. Аналізується також, які групи користувачів можна виділити серед загальної їх кількості на основі історії перегляду Web-вузла [75]. В інтелектуальному аналізі даних ці технології виконують такі завдання:

- *опис відвідувачів сайту* (кластеризація, класифікація);
- *опис відвідувачів, які здійснюють покупки* в Інтернет-магазині (кластеризація, класифікація);
- *визначення типових сесій* і навігаційних шляхів користувачів сайту (пошук популярних наборів, асоціативних правил);
- визначення груп або сегментів відвідувачів (кластеризація);
- *виявлення залежностей* під час користування послугами сайту (пошук асоціативних правил).

Цей напрям розглядає взаємозв'язки між Web-сторінками, ґрунтуючись на зв'язках між ними. Побудовані моделі можуть бути використані для категоризації Web-ресурсів, пошуку схожих і розпізнавання авторських сайтів.

Залежно від поставленого завдання структура сайту моделюється з певним рівнем деталізації. У найпростішому випадку гіперпосилання

подають у вигляді спрямованого графа G=(D, L), де D – це набір сторінок, вузлів або документів; L – набір посилань.

В онтологічному аналізі здобуття Web-структур може бути використано як підготовчий етап для пошуку онтологічних зв'язків між онтологічними моделями Web-контенту цих сайтів.

Метод аналізу структури посилань між різними Web-сторінками, внутрішніми й зовнішніми сайтами у виділеному мережному сегменті дає змогу виконувати завдання, що постають у процесі аналізу соціальних мереж або специфічних галузей людської діяльності, або знань, наприклад, в аналізі цитування авторів. Результатом такого аналізу може слугувати виявлений набір специфічних сторінок таких типів [74, 99]:

- *хаби* – з такої сторінки посилання йдуть на найбільш значущі ресурси в цій галузі знань або на «знайомства» з найбільш значущими користувачами соціальної мережі;
- *авторитети* – сторінки, на які посилається велика кількість авторів з цієї тематики або користувачі соціальної мережі, до «дружби» з якими прагне велика кількість користувачів.

Топологія структури посилань має вигляд спрямованого графа з зазначеними вузлами відповідно до їх функціональної класифікації й дуг з вагами, що відображають, наприклад, частоти переходів за посиланнями. Для моделювання топології Web-посилань використовуються різні алгоритми, зокрема HITS.

*Opinion Mining* – це аналіз емоційної тональності тексту, що передає ставлення автора тексту до об'єктів, відображених у тексті, яка визначається за допомогою методів контент-аналізу. Емоційний складник аналізується на рівні лексем і їх сукупностей. Це дає змогу поповнювати онтології додатковими властивостями для інформаційних ресурсів і встановлювати нові відношення між групами користувачів зі спільними рисами й групами IP.

Щодо онтологічного аналізу, то такі методи забезпечують поповнення онтологій відомостями про різні групи користувачів і можуть стати основою для класифікації як самих користувачів, так і ресурсів, до яких вони звертаються. Така класифікація відображається в онтології через відношення між класами й належність елементів до класу, а відповідні знання забезпечують персоніфіковану роботу інтелектуальних Web-застосувань.

*Text Mining* використовує методи Data Mining для семантичного аналізу великих масивів неструктурованої інформації – природномовних текстів. Саме ці методи є основним джерелом для автоматизованого створення й поповнення онтологій.

*Opinion Mining* – це напрям у Web Mining для аналізу думок людей, які наведені в рецензіях, коментарях, оглядах, соціальних мережах тощо [384, 385]. Обсяг ресурсів, з яких здобувається така інформація, дуже великий і зростає з кожним днем: створюються нові блоги, коментарі та ЖЖ, споживачі постійно обмінюються думками про товари й послуги.

Крім перерахованих вище, наявні технології Call Mining, Audio Mining і Video Mining, що дають змогу здобувати знання зі спеціалізованих видів ІР, але нині їх практичне застосування лише розпочинається.

## 6. 3. Здобуття онтологічних знань з контенту природномовних Web-ресурсів

Значна частина ресурсів Web – це природномовні тексти, які містять велику кількість інформації – фактів, визначень, правил і зв'язків між ними тощо. За даними аналітиків, понад 80% інформації, яка зберігається в документах, подається в текстовій формі. Але через складність і неоднозначність розуміння природної мови (ПМ) обробка таких ресурсів потребує великих обчислювальних потужностей і не може бути автоматизована повністю. Інформаційно-пошукові системи дають змогу отримати набір Web-сторінок, релевантних певному запиту, проте отримання знань з цих ІР потребує спеціалізованих засобів і технологій.

Сучасні інформаційні системи мають бути придатними для розв'язання всього комплексу задач, які виникають у процесі керуванням потоком вхідних «сирих даних», тобто вони повинні підтримувати автоматичну класифікацію й автоматичне індексування таких текстів, виконувати оперативний і адекватний розподіл нової інформації серед користувачів, забезпечувати передачу й збереження даних в електронному архіві й пошук у цьому архіві за змістом.

Для розв'язання цих задач розробляються різноманітні технології автоматичного аналізу тексту ділових і наукових документів в інформаційних системах з обмеженою ПрО. Ці технології орієнтовані на підтримку роботи з БЗ інформаційної системи як на момент її створення, так і в ході її експлуатації [77].

У технології аналізу текстових документів можна виділити три основні компоненти знань [174]:

- онтологія, що містить поняття й відношення ПрО, які використовуються для опису даних, які необхідно здобути з тексту й розмістити в БД системи;

- предметний словник (тезаурус), що містить терміни, за допомогою яких у тексті можуть подаватися поняття й відношення онтології;
- інформаційне наповнення системи або БД.

Дані в системі подаються як набір різнотипних інформаційних об'єктів (ІО), що являють собою опис об'єктів предметної області й у сукупності утворюють інформаційне наповнення системи. Кожен ІО пов'язаний з деяким терміном онтології й може розглядатися як екземпляр відповідного класу, тобто мати задану експертом структуру з фіксованим набором атрибутів.

Кожний ІО може бути розглянутий у трьох різних аспектах – структура, контент і контекст. Структура об'єкта може характеризуватися як набором власних атрибутів і зв'язків, так і описом формальної структури його змісту. Контент відображає інформаційний зміст об'єкта за допомогою понять і відношень, заданих в онтології ПрО, і являє собою набір інформаційних об'єктів.

Розглянуті вище технології аналізу орієнтовані на роботу з тими ІО, зміст яких визначається текстом. Такі ІО називають текстовими ресурсами. Для того, щоб презентувати в інформаційній системі всі три наведені вище аспекти текстового ресурсу, потрібно:

- здійснити опис понять (класів), яким відповідають текстові ресурси;
- визначити формальну структуру змісту для кожного класу текстових ресурсів;
- задати схеми фактів, що задають правила здобуття змістовних об'єктів з тексту.

У процесі аналізу документа використовується формальна презентація структури його тексту, що залежить від типу чи жанру документа.

Відповідно до [91], текст в електронній формі має принаймні три рівні формальної структури – фізичний, логічний і жанровий. Перший становить презентацію тексту на сторінці, наприклад, за допомогою тегів або таблиці стилів. До другого рівня належать такі елементи, як текст, параграф, рядок, речення тощо. Третій рівень – розбивка тексту на жанрові частини. Наприклад, текст наукової публікації [23] має такі жанрові розділи: УДК, заголовок, автори, основний розділ і список посилань.

### 6. 3. 1. Технології аналізу природномовних текстів

Розв'язання частини проблем, що пов'язані з аналізом природномовних текстів, пропонує *Text Mining* – технологія автоматичного здобуття знань з великих за обсягом природномовних

текстів, що базується на поєднанні лінгвістичних, семантичних, статистичних і машинних навчальних методик [212, 224].

Серед найбільш поширених сфер застосування Text Mining можна виділити:

- поповнення онтологічних баз знань всеосяжних і детальних формалізацій у вигляді концептуальної схеми;
- діалогові системи й фактографічний пошук;
- визначення семантичної близькості текстів для аналізу даних і обробки природної мови.

Технологія Text Mining – це використання методів Data Mining для здобуття знань з природномовних інформаційних ресурсів [1, 212, 224]. Така технологія призначена для глибинного аналізу великих за обсягом текстів і дає змогу виявляти в неструктурованій інформації важливі для користувача закономірності. Ця технологія базується на поєднанні лінгвістичних, семантичних, статистичних методик і машинного навчання [13]. Text Mining розширює технології Data Mining за рахунок уведення додаткового етапу, на якому здійснюється трансформація неструктурованих текстових масивів у структуровані. Після цього дані можуть оброблятися за допомогою стандартних методів.

Text Mining дає змогу проаналізувати не синтаксис, а семантику текстів. Здобуття інформації з тексту передбачає її лінгвістичний аналіз для виявлення частоти входжень різних слів, виявлення шаблонів, встановлення тегів розмітки й анотування, а також застосування таких методів аналізу інформації, як аналіз зв'язків і асоціацій, візуалізацію і прогностичний аналіз.

*Контент-аналіз* – методика об'єктивного аналізу вмісту інформаційних ресурсів [72]. Це формалізований підхід до вивчення текстової і графічної інформації, що полягає в трансформації досліджуваної інформації в кількісні показники та її статистичній обробці для виявлення й вимірювання різних фактів і тенденцій, що відображаються в цих документах.

Контент-аналіз базується на збиранні кількісних даних про інформаційний об'єкт – досліджуване явище чи процес, відомості про які містяться в документах. Під документом при цьому розуміють довільний природномовний ІР.

Розвиток технології Text Mining припадає на роки правління президента США Ричарда Ніксона (1969–1974 рр.). Тоді було виділено десятки мільйонів доларів на розвиток наукових напрямів, пов'язаних з автоматизацією перекладу. Це відбувалося в епоху холодної війни, коли, зокрема, дуже актуальним було завдання комп'ютерного перекладу з російської мови на англійську найрізноманітніших

документів, починаючи з наукових доповідей і закінчуючи технічною документацією. Цей проект носив закритий характер.

У той же самий час з'явилася нова галузь знань – Natural Language Processing (NLP), у країнах СНД – комп'ютерна лінгвістика. У 90-х роках XX ст. у відкритих джерелах почали з'являтися не тільки доповіді з наукових конференцій, а й програмні коди, що дало змогу залучити до розробок більш широке міжнародне наукове співтовариство. Найбільш активними в цій галузі є вчені зі США, Великобританії, Франції та Німеччини.

У нашій країні розвиток комп'ютерної лінгвістики мав свою специфіку. Вона розвивалася в основному в інтересах оборонних підприємств і служб безпеки й не орієнтувалася на розв'язання конкретних бізнес-завдань. Позначився й брак цільового фінансування цієї галузі останніми роками. Однак бурхливий розвиток ЗМІ та Інтернету породжує попит як з боку державних служб, так і з боку комерційних організацій (наприклад, конкурентна розвідка).

Класична схема обробки текстів передбачає кілька послідовних етапів:

- *нормалізація слів* з урахуванням морфології мови (морфологічний аналіз).
- *семантичний аналіз* тексту для визначення конкретного змісту слова залежно від контексту.
- створення *семантичного образу* вихідного документа, на основі якого робляться інтелектуальні запити на аналіз текстів.

Важливим компонентом технології Text Mining є здобуття з тексту характерних для нього елементів або властивостей, які можуть використовуватися як метадані документа, ключові слова, анотації. Іншим завданням є зарахування документів до категорії, відповідно до заданої схеми їх систематизації.

Розглянемо конкретні компоненти процесів Text Mining.

Типові стадії й завдання цієї технології:

- Пошук інформації й виявлення вихідних даних це підготовчий крок, що передбачає збирання або виявлення набору текстових матеріалів для аналізу. Такі матеріали можуть міститися в Інтернеті, базах даних, файлових системах або системах керування контентом.
- Використання суто статистичних методів аналізу, а також процесів обробки природної мови й засобів лінгвістичного аналізу.
- Виявлення змістів використання статистичних та інших технік для виявлення пойменованих ознак тексту згадувань людей, організацій, місць, символів, абревіатур тощо. Контекст

допомагає визначити, що саме позначає те чи інше слово в конкретному входженні.

- Виявлення шаблонів можна виявити, у яких шаблонах у тексті презентовані ті чи інші змісти.
- Виявлення перехресних посилань виявлення визначень та інших ознак, що стосуються тих самих об'єктів.
- Виявлення взаємозв'язків, фактів і подій пошук зв'язків між різними змістами, укладеними в текстовій інформації.
- Значеннєвий аналіз розпізнавання суб'єктивного (а не фактичного) матеріалу й виявлення різних форм оцінної інформації змістів, думок, настроїв, емоцій. Технології аналізу дають змогу вивчати змісти на рівні тем, концептів, а також виділяти законодавців думок і об'єкти думок.
- Кількісний аналіз тексту використання набору технік, запозичених з соціальних наук, які полягають у тому, що людина або комп'ютер «витягає» семантичні або граматичні зв'язки між окремими словами, щоб зрозуміти зміст стилістичних шаблонів, провести психологічне профілювання тощо.

Технології Text Mining використовуються для керування знаннями в різних напрямах і галузях, а тому в кожному випадку мають свою специфіку – це може бути використання в урядових, дослідницьких, корпоративних цілях.

Технологія глибинного аналізу тексту – Text Mining – це той самий інструментарій, що дає змогу аналізувати великі обсяги інформації в пошуках тенденцій, шаблонів і взаємозв'язків, здатних допомогти в ухваленні стратегічних рішень. Крім того, Text Mining – це новий вид пошуку, що, на відміну від традиційних підходів, не тільки знаходить списки документів, формально релевантних запитам, а й допомагає виконати прохання: «Допоможи мені, будь ласка, зрозуміти зміст, розібратися з цією проблемою». Клод Фогель, один із засновників і головний технолог компанії Semio, пояснює: «Використовуючи аналогію з бібліотекою, технологія Text Mining подібна до розгортання книги перед читачем з підкресленою необхідною інформацією. Порівняйте це з видачею читачеві стосу документів і книг, у яких де-небудь міститься інформація, потрібна читачу, однак знайти її складно». Процес осмисленого пошуку далеко не тривіальний, часто в колекції документів є лише натяк на необхідну інформацію. Необхідні могутні інтелектуальні можливості, щоб знайти те, що потрібно. У назві технології слово «mining» (видобування руди) є метафорою, що вказує на наявність глибоко «заритої» інформації.

Підкреслимо, що технології глибинного аналізу тексту історично передувала технологія здобуття даних (Data Mining), методологія й підходи якої широко використовуються й у методах Text Mining.

### 6. 3. 2. Основні види застосування технологій Text Mining

Можна виділити такі основні види застосунків технологій Text Mining:

- *Класифікація текстів,* у якій використовуються статистичні кореляції для побудови правил розміщення документів у визначені категорії. Класифікація, що базується на ознаках документів і використовує лінгвістичні й математичні методи без використання визначених категорій. Результат таксономія чи візуальна карта, що забезпечує ефективне охоплення великих обсягів даних; семантичні мережі чи аналіз зв'язків, що визначають появу дескрипторів (ключових фраз) у документі для забезпечення навігації. Здобуття фактів призначено для отримання деяких фактів з тексту для поліпшення класифікації, пошуку та кластеризації.

У сучасних системах класифікація застосовується, наприклад, у таких завданнях: групування документів в Intranet-мережах і на Web-сайтах, розміщення документів у визначені папки, сортування повідомлень електронної пошти, виборче поширення новин передплатникам.

- *Кластеризація об'єктів* – виділення компактних підгруп об'єктів з близькими властивостями. Система повинна самостійно знайти ознаки й розподілити об'єкти за підгрупами. Цей процес, як правило, передує завданню класифікації, оскільки дає змогу визначити групи об'єктів.

Розрізняють два основні типи кластеризації – ієрархічну й бінарну. Ієрархічна кластеризація полягає в побудові дерева кластерів, у кожному з яких міститься невелика група документів.

Бінарна кластеризація забезпечує групування й перегляд документальних кластерів за посиланнями подібності. В одному кластері розміщуються найближчі за своїми властивостями документи. У процесі кластеризації будується базис посилань від документа до документа, в основі якого – значущість і спільне використання визначених ключових слів. Кластеризація сьогодні застосовується для реферування великих документальних масивів, визначення взаємозалежних груп документів, спрощення процесу перегляду під час пошуку необхідної інформації, віднайдення унікальних документів у колекції, виявлення дублікатів чи дуже близьких за змістом документів.

- *Знаходження виключень*, тобто пошук об'єктів, що за своїми характеристиками значно виділяються з загальної маси. Для цього спочатку з'ясовуються середні параметри об'єктів, а потім досліджуються ті об'єкти, параметри яких найбільш сильно відрізняються від середніх значень. Як відомо, пошук виключень широко застосовується, наприклад, у роботі спецслужб. Подібний аналіз часто проводиться після класифікації для того, щоб з'ясувати, наскільки остання була точною.

Окремо від задачі кластеризації стоїть задача пошуку зв'язаних ознак (полів, понять) окремих документів. Від прогнозування ця задача відрізняється тим, що заздалегідь не відомо, за якими саме ознаками реалізується взаємозв'язок; мета саме в цьому й полягає, щоб знайти зв'язок між ознаками. Ця задача подібна до кластеризації, але не за безліччю документів, а за безліччю властивих їм ознак.

- *Візуалізація даних* передбачає обробку структурованих числових даних, однак вона також є ключовою ланкою у процесі подання схем неструктурованих текстових документів. Так, наприклад, сучасні системи класу Text Mining можуть здійснювати аналіз великих масивів документів і формувати предметні покажчики понять і тем, висвітлених у цих документах. Візуалізація, зазвичай, використовується як засіб презентації контенту всього масиву документів, а також для реалізації навігаційного механізму, що може застосовуватися під час дослідження документів та їхніх класів.

### 6. 3. 3. Системи автоматичного реферування

Поряд з традиційним індексуванням IP сьогодні широко застосовуються системи автоматичного реферування.

*Автоматичне реферування* (анотування) – це процес побудови короткого змісту документа чи множини пов'язаних документів за його повним текстом.

Обсяг анотації має становити від 5 до 30% вихідного тексту (підготовка анотацій кількох джерел інформації потребує ще більшого коефіцієнта стискання). Є два типи таких систем. Головна розбіжність між ними полягає в тому, що вони формують – коротку анотацію чи набір цитат. Текст, отриманий шляхом з'єднання окремих оригінальних фрагментів, позбавлений гладкості, важко читати. Крім того, такі системи призначені тільки для обробки текстової інформації й не можуть працювати одночасно з кількома її джерелами. До подібних систем належать AutoSummarise (Microsoft Office), Oracle Context, Insight Summarizer (AltaVista).

Формування короткого змісту тексту вимагає потужних обчислювальних ресурсів для семантичної обробки природномовної інформації. Для цього, звичайно, застосовують один з двох основних підходів. Перший спирається на традиційний лінгвістичний метод синтаксичного розбору речень, а семантична інформація застосовується для анотування дерев розбору. Другий підхід ґрунтується на системах штучного інтелекту й спирається на розуміння природної мови (синтаксичний розбір є складником цього методу).

Нині розробляються найрізноманітніші методи й відповідне ПЗ, призначені для синтаксичного й семантичного аналізу природномовних текстів з різних ПрО. Зокрема, програмний комплекс КОНТЕНТ призначений для здобуття з текстів основних фактів і подій на основі концептуального аналізу семантичних фреймів. Знання ПрО подаються як множина фреймів. Лінгвістичні знання презентуються у вигляді словника. Визначення слова складається з його значення й синтаксичної інформації про це слово. Аналіз полягає в співвідношенні тексту й заданої ПрО [69, 206].

Деякі програмні засоби здатні автоматично анотувати повнотекстові документи, з якими працює користувач, щоб використовувати цю інформацію для визначення контексту пошуку. Наприклад, система Watson моделює контекст інформації, що потрібна користувачеві, на основі вмісту документів, які він редагує засобами Microsoft Word чи переглядає в Internet Explorer. Ці документи аналізуються за допомогою евристичного алгоритму, який виявляє слова, характерні для вмісту документів. Потім знайдені слова автоматично додаються до запиту користувача. Крім того, Watson у фоновому режимі шукає у Web документи, пов’язані з матеріалами, що редагує й переглядає користувач. Недоліком системи є непрозорість алгоритмів, використовуваних системою, для кінцевого користувача.

Альтернатива індексації природномовної інформації – технологія, розроблена компанією Excalibur Technologies, яка поєднує метод адаптивного розпізнавання образів APRP (Adaptive Pattern Recognition Processing) і семантичні мережі. Вона дає змогу працювати з інформацією будь-якого типу: текстом, графікою, відео тощо. Метод APRP використовує теорію нейронних мереж і дає змогу здійснювати бінарну індексацію, за якою розмір індексу навіть за обробки неструктурованої інформації не перевищує 30% від обсягу вихідних даних. Проте ця технологія потребує значних обчислювальних ресурсів.

Text Mining забезпечує такі основні дії:

а) здобуття інформації щодо ключових слів – *Feature (Entity) Extraction* – знаходження слів чи груп слів, які, з погляду користувача, є важливими для опису змісту документа. Це можуть бути згадки

про осіб, організації, географічні місця, терміни певної предметної області тощо. Видобуті сутності також можуть бути найбільш значущими словосполученнями, що характеризують документ за його основною темою.

З погляду онтологічного аналізу це пошук фрагментів тексту, що пов'язані з термінами онтології.

б) Здобуття інформації щодо зв'язків між термінами – *Feature (Entity) Association Extraction* – більш складне з технологічного погляду. Простежуються різного роду зв'язки між видобутими сутностями. Наприклад, якщо про обраних суб'єктів згадується в різних документах і вони мають певну спільну характеристику (час, місце тощо), то можна прогнозувати, чи є між ними якийсь зв'язок.

З погляду онтологічного аналізу – це перевірка відомостей про онтологічні відношення та їх відображення в текстах, що аналізуються.

в) Здобуття інформації про факти та події – *Relationship, Event and Fact Extraction* – найскладніший варіант здобуття інформації (Information Extraction), що передбачає здобуття сутностей, розпізнавання фактів і подій, а також здобуття інформації з цих фактів.

З погляду онтологічного аналізу – це пошук фрагментів тексту, що пов'язані з екземплярами класів онтології – певними пойменованими сутностями, що задовольняють конкретним умовам. При цьому враховуються як властивості класів, так і відношення між ними. Наприклад, можна знайти всіх домашніх тварин на ім'я «Бася», що живуть у місті Києві, і визначити, до якого підкласу вони належать. Звичайні пошукові системи можуть виконати пошук таких сутностей, але не здатні класифікувати їх за контекстом.

Сьогодні пропонується досить багато інструментів Text Mining – від відносно простих програм, що спираються на статистичний аналіз окремих термінів у текстах, таких, як WordStat, до найскладніших застосувань типу Aerotext і Businessobjects Text Analysis.

Приміром, *EPAM Systems* пропонує власний підхід до аналізу неструктурованої інформації для розв'язання бізнес-завдань, заснований на досвіді найбільших компаній світу й можливостях сучасних технологій (Text Mining, Opinion Mining, Business Intelligence, OLAP, Data Mining), забезпечуючи підвищення ефективності різноманітних маркетингових досліджень і розробки продуктів і послуг.

*TextAnalyst* ™ – це інструмент, що забезпечує аналіз змісту текстів, пошуку інформації на рівні семантики й формування електронних архівів. Він дає змогу аналізувати зміст тексту, автоматично формувати семантичну мережу з гіперпосиланнями, тобто отримувати змістовний портрет тексту в термінах основних понять і їх

змістовних зв'язків; будує тематичне дерево з гіперпосиланнями, яке відображає семантичну структуру тексту у вигляді ієрархії тем і підтем. Крім того, система підтримує семантичний пошук з урахуванням прихованих змістовних зв'язків слів запиту зі словами тексту й забезпечує автоматичне реферування тексту – формування його змістовного портрету в термінах найбільш інформативних фраз.

Система *PolyAnalyst* дає змогу практично здобути корисні знання з великої кількості текстових і структурованих даних і передати ці знання в доступній для розуміння формі у вигляді оперативно застосовуваних моделей для ухвалення відповідальних бізнес-рішень.

Наголосимо, що методи й програмні засоби Text Mining повинні бути адаптовані до відповідної предметної області. Деякі типи текстів (наприклад, художня література, професійний сленг тощо) погано піддаються машинній обробці.

## 6. 4. Лінгвістичні методи аналізу природномовних текстів

### 6. 4. 1. Лінгвістичні бази знань

Сьогодні в лінгвістичному аналізі широко застосовується онтологічне подання знань як щодо природної мови, так і щодо предметної області, до якої належить аналізована інформація. Крім того, онтології застосовуються для створення й аналізу метаданих, що супроводжують природномовні документи.

Лінгвістичні методології використовуються для ПМ-текстів довільної, чітко не виділеної структури. Вони значною мірою залежать від мови, якою написаний текст, і потребують великих обчислювальних потужностей, а також не завжди уможливлюють однозначну ідентифікацію семантики тексту.

Для подання лінгвістичних знань щодо природної мови сьогодні використовуються три підходи:

- інформаційно-пошукові тезауруси;
- тезаурус WordNet;
- спеціалізовані формальні онтології.

Традиційні інформаційно-пошукові тезауруси розроблялися для ручного індексування документів [313], а тому використання їх для автоматизованої обробки інформації в електронній формі пов'язано з багатьма проблемами.

Тезаурус WordNet орієнтований на автоматизовану обробку інформації [387]. WordNet містить відомості про загальну лексику тієї чи іншої мови (хоча можна будувати тезауруси типу WordNet і для конкретних ПрО). Нині він містить 155 тисяч лексем і словосполучень, що організовані в 117 тисяч понять для багатьох

мов [176]. Базовим поняттям у WordNet є *лексема*, а базовим відношенням – відношення *синонімії*. Основні структурні одиниці WordNet – *синсети* (набори синонімів). Лексеми вважаються у WordNet синонімами, якщо заміна однієї іншою у висловлюванні не змінює значення його істинності.

Але принциповим недоліком WordNet є те, що його структура не пристосована до опису термінології ПрО – у цьому тезаурусі окремо відображаються різні частини мови, більшість лексем не пов'язані одна з одною, слабко підтримується обробка словосполучень. Тому, незважаючи на велику користь від WordNet, виникає потреба в більш інтероперабельних засобах подання лінгвістичних знань.

Таким засобом є спеціалізовані лінгвістичні онтології [338]. Це, як правило, легкі онтології, у яких принципи розробки інформаційно-пошукових тезаурусів і лінгвістичних ресурсів типу WordNet інтегровані з методологіями онтологічного аналізу [453]. Приміром, у [107] запропонована модель лінгвістичної онтології для автоматичної обробки текстів ПрО, до складу яких увіходять класи сутностей, що мають між собою необмежені типи відношень і ситуацій. Формальна модель лінгвістичної онтології для широкої ПрО містить такі елементи, як множина понять онтології, що володіють однаковими властивостями й пов'язані однаковими відношеннями з іншими класами сутностей; множина унікальних імен понять і екземплярів в онтології; множина екземплярів понять онтології; набір відношень між поняттями що спеціально сформований для автоматичної обробки текстів; множина правил виведення, що базуються на властивостях транзитивності й спадкування відношень; множина багатозначних слів і виразів тощо. Така *лінгвістична онтологія* ПрО являє собою БЗ онтологічного типу, що містить знання щодо понять і лексико-термінологічного складу цієї ПрО.

У застосуваннях, яким потрібно використовувати лінгвістичні знання, залежно від специфіки застосування, можуть використовуватися окремі випадки таких подібних лінгвістичних онтологій – накладання додаткових обмежень на лінгвістичну онтологію може значно спростити алгоритми її обробки, не впливаючи на загальний результат роботи. Якщо інші знання в застосуванні (приміром, профіль користувача) теж подані через онтології, то онтологічний підхід дає можливість більш ефективно інтегрувати з ними потрібні лінгвістичні знання.

Наявність лінгвістичної БЗ є необхідною умовою лінгвістичного аналізу ПМ-тексту, але, щоб аналізувати зміст ПМ-тексту й здобувати з нього потрібну інформацію, потрібні ще й відповідні *лінгвістичні*

*методи,* що можуть використовувати такі лінгвістичні знання в інтересах користувача.

### 6. 4. 2. Методи лінгвістичного аналізу

Традиційно лінгвістичний аналіз складається з таких етапів, як морфологічний, синтаксичний і семантичний аналізи.

*Морфологічний* аналіз стосується окремих слів. Він застосовується для виділення в ПМ-тексті *лексем* – груп словоформ, основи яких тотожні за значенням, а однойменні морфи основ фонематично близькі чи тотожні один одному. В одну лексему поєднуються різні словоформи одного слова (наприклад, «собака, собаками, собаку» тощо). На етапі морфологічного аналізу для кожного слова визначається, якою частиною мови є це слово, а потім визначаються його морфологічні характеристики.

На етапі *синтаксичного аналізу* аналізуються речення: здійснюється розпізнавання синтаксичної структури речення на основі морфологічної інформації та синтаксичних правил поєднання слів і словосполучень певної мови. Синтаксична структура відображає лінгвістичні зв'язки між словами в реченні.

Метою *семантичного аналізу* є розпізнавання змісту тексту в цілому, залежно від мети аналізу. Це дає змогу здобути з ПМ-тексту знання, закладені в нього автором, і перетворити їх для автоматизованої обробки.

## 6. 5. Використання онтологій для семантичної розмітки природномовних текстів

### 6. 5. 1. Семантична розмітка природномовних текстів

Семантична розмітка ПМ-тексту дає змогу встановити зв'язки елементів тексту – слів, словосполучень, речень – з деякими поняттями відповідної ПрО. Текст може бути розмічений різними способами залежно від того, яка онтологія обрана за основу його розмітки. Вона може використовуватися як джерело знань для формування й поповнення онтологій (ПрО, ІР, користувачів).

*Теги розмітки* – елементи, що застосовуються для розмітки (як структурної, так і семантичної). Звичайна розмітка тексту, за аналогією до гіпертексту, використовує фіксований набір тегів з певними значеннями, приміром, «новий параграф» або «текст виділено зеленим кольором». У семантичній розмітці враховуються й семантичні зв'язки між окремими тегами. Наприклад, якщо для розмітки використані терміни онтології, у якій клас «собака» є підкласом класу «тварина», то

фрагмент, позначений тегом «собака», автоматично буде належати до класу «тварина».

Якщо ПМ-текст має семантичну розмітку, тобто його фрагменти явно пов'язані з певними поняттями, то це значно спрощує аналіз змісту такого тексту й підвищує точність виконання інформаційних запитів до нього. Семантична розмітка дає змогу обробляти текст на рівні даних і оперувати при цьому не всім документом, а його окремими фрагментами, а це має велике значення для обробки ІР великого обсягу. Наприклад, недостатньо повідомити користувачеві, що опис потрібного йому ІО вміщено в певному документі, доречніше вказати, де саме в цьому документі міститься опис цього ІО.

Формально *семантична розмітка* довільного тексту X може бути визначена в такий спосіб [102]: текст X, який потрібно розмітити, розглядається як скінчена послідовність символів, що належать скінченій множині A, частина з яких є роздільниками, приміром, пробіл, крапка, кома. Семантична розмітка здійснюється для певної ПрО, яка характеризується набором термінів зі скінченої множини T, які можуть використовуватися як теги семантичної розмітки. Довільний фрагмент тексту може бути пов'язаний з терміном.

Розмітку називають *семантичною*, якщо для множини T описана певна семантика (приміром, зв'язки між термінами представлено за допомогою таксономії або онтології).

Під час формування семантичної розмітки потрібно використовувати не тільки знання ПрО (чи хоча б її термінологічну базу), а й правила тієї конкретної природної мови, якою написаний текст, – лінгвістичну БЗ. Семантична розмітка текстів дає змогу виконувати над ними різні логічні операції, здобувати з них нові знання тощо [103]

Семантична розмітка залежить і від того, які саме засоби (онтології, тезауруси, таксономії) використовуються для опису Про. Так, якщо для розмітки були використані дві різні онтології однієї й тієї ж ПрО, то результати розмітки того самого тексту, виконані тими самими методами, можуть значно відрізнятися.

Основні сфери застосування семантичної розмітки:

- візуалізація інформації;
- індексація й пошук інформації;
- керування знаннями, зокрема можливість одержання, збереження й використання знань;
- взаємодія з Web-сервісами й агентними системами.

На жаль, мови семантичної розмітки, розроблені в Semantic Web (RDF, RDFS),є досить складними, а відсутність значної кількості застосувань Semantic Web не створює достатньої мотивації для їхнього

освоєння, тому велика частина ресурсів Web, як і раніше, не має метаописів. Але саме брак таких описів є перешкодою для створення застосувань Semantic Web. Виникає замкнене коло, виходом з якого може стати автоматична або автоматизована семантична розмітка документів. Незважаючи на недосконаліть якості такої розмітки, подібні метадані могли б стати ядром глобальної розподіленої бази знань Web, і, подібно до Вікіпедії, надалі вдосконалюватися зусиллями самих користувачів IP.

### 6. 5. 2. Етапи семантичної розмітки ПМ-тексту термінами онтології

У процесі семантичної розмітки необхідно розв'язати такі задачі [104]:

1) визначити, які теги повинні використовуватися для семантичної розмітки тексту в певній ПрО;

2) виявити в тексті ті фрагменти, що відповідають класам та екземплярам класів з обраної онтології;

3) розмітити обраними тегами ПМ-тексти відповідної ПрО.

Прикладом системи, у якій використовується семантична розмітка, є *Semantic Wiki*. Це розширення традиційної Wiki-технології надає користувачу можливість для додавання семантичної розмітки до контенту, що дає змогу краще структурувати інформацію [57]. Наприклад, якщо деякий об'єкт, опис якого міститься у Вікіпедії, класифікується як «Людина», тоді можна відтворювати такі його атрибути, як «Рік народження», «Ім'я», «Професія», а потім використовувати ці атрибути та їхні значення для пошуку й формування нових списків, наприклад, для пошуку всіх людей, згаданих у Вікіпедії, день народження яких збігається з поточною датою.

Є різні засоби семантичної розмітки тексту, які різняться наборами тегів, що використовуються для розмітки, і складністю відношень, які можна встановлювати між різними тегами: GML (Generalized Markup Language), XML, RDF (Resource Description Framework), мікроформати (hAtom – дає змогу розмічати канали Atom безпосередньо в HTML; hCalendar – дає можливість розмічати події в HTML; hCard – дає змогу розмічати контактну інформацію в HTML; hReview – дає змогу розмічати огляди в HTML; hResume – призначено для розмітки резюме й CV; rel-directory – призначено для створення й внесення до розподіленого каталогу; rel-nofollow – створено як ініціативу по боротьбі зі спамом у контенті; xFolk – призначено для позначення посилання на теги; XFN – призначено для позначення соціальних контактів; XOXO – призначено для розмітки списків і схем). Найбільшу виразність як джерело тегів семантичної розмітки мають онтології ПрО.

Якщо як теги семантичної розмітки використовуються терміни онтології, то це дає змогу встановити явний зв'язок між фрагментами цього тексту та поняттями ПрО, що відображена в онтології, тобто визначити семантичну структуру цього тексту.

Для автоматизації семантичної розмітки природномовного тексту термінами онтології відповідної ПрО доцільно використовувати лінгвістичні методи, розглянуті вище. Семантична розмітка ПМ-текстів для ПрО передбачає такі етапи:

- навчання (які слова та словосполучення ПМ-тексту пов'язані на змістовному рівні з поняттями ПрО), що міститься в онтології, і в ПС;
- використання отриманих правил для автоматизованої розмітки інших ПМ-текстів.

Повністю автоматизувати обидва етапи неможливо (через нечіткість, неоднозначність, динамічність природної мови, а отже, принципову неможливість її повної формалізації), а тому потрібно використовувати досвід і знання користувача для усунення неоднозначностей, які неможливо зняти іншими засобами.

Під час розробки онтології ПрО ($О_{ПрО}$), що цікавить користувача, пропонується використовувати спеціалізовану лексичну онтологію (ЛО) цієї ПрО, що створюється й поповнюється паралельно з розробкою $О_{ПрО}$ на основі аналізу корпусу ПМ-текстів, які користувач вважає релевантними до досліджуваної ПрО. Така ЛО є інструментом, на основі якого ітеративно виконується семантична розмітка ПМ-текстів і вдосконалення онтології ПрО, на якій базується ця розмітка.

Принципова відмінність ЛО від звичайних лексичних онтологій полягає в тому, що як екземпляри класу «лексема» до неї вносяться тільки ті поняття, що пов'язані з поняттями $О_{ПрО}$, а не всі поняття, що трапляються в ПМ. Це значно зменшує обсяг онтології і, відповідно, скорочує час її обробки.

*Лексична онтологія* – це онтологія, що містить два основні класи: «поняття ПрО» і «словоформа», екземпляри яких можуть бути пов'язані відношенням «відповідає». Призначення цієї онтології – зафіксувати зв'язки між поняттями ПрО й відповідними фрагментами природномовного тексту.

Лексична онтологія поповнюється ітеративно: під час кожного додавання нового класу до онтології $O_1$ до $O_L$ також уводиться цей новий клас, для кого тим чи іншим способом формується словоформа.

Семантична розмітка ПМ-текстів для визначеної користувачем ПрО створюється в два етапи. На першому етапі здійснюється *навчання*. Основним результатом цього етапу є формування лексичної онтології

ПрО, яка дає змогу зв'язати кожне з понять онтології з множиною фрагментів ПМ-тексту, що семантично пов'язані з цим поняттям. У процесі навчання в корпусі текстів потрібно знайти пойменовані сутності (ПС) – власні назви, яким відповідають слова, написані з великої літери, що не входять до загального словника, і слова, узяті в лапки. Потім у реченнях, що містяться в ПС, потрібно визначити імена класів, до яких належать ці ПС. Якщо таке ім'я класу наявне, то здійснюється спроба виділити слова, що зв'язують синтаксично ПС та ім'я її класу. Якщо це вдається, то для цих слів будується шаблон.

Результатом такої роботи є зіставлення кожного терміна онтології ПрО зі словоформами, що відповідають у ПМ цьому поняттю. Словоформи здобуваються з навчальної множини ПМ-текстів, зарахованих користувачем до певної ПрО, відображеної в О з погляду інформаційних інтересів користувача [138].

Після навчання здійснюють використання лексичної онтології ПрО для семантичної розмітки ПМ-тексту. Лексична онтологія дає змогу в новому ПМ-тексті виділити словоформи, пов'язані з поняттями й відношеннями онтології ПрО, а також слова, що можуть бути іменами ПС. Для цього потрібно знайти в ПМ-текстах ці словоформ й приписати на початку їх і в кінці теги, що відповідають поняттям та елементам онтології ПрО [102].

Виконавши цей процес, отримуємо семантично розмічений ПМ-текст, придатний для автоматичного аналізу. Наприклад, користувач легко може отримати всі абзаци, у яких містяться фрагменти, пов'язані з вибраним поняттям онтології та його підкласами.

Лексичну онтологію можна використовувати для поповнення онтології ПрО.

Розширення онтології ПрО здійснюється в процесі аналізу семантично розмічених текстів. Онтологія поповнюється новим класом, якщо в одному абзаці тексту є пов'язані з ним фрагменти, а також словоформи іншого класу онтології та відношення. Для цього лексична онтологія має зберігати імена відношень ПрО і відповідні їм словоформи. Наприклад, відношенню «складається з» відповідають такі фрагменти ПМ-тексту, як ««виготовлений з», «містить у собі». Подібні словоформи можуть бути сформовані так: якщо в одному реченні ПМ-тексту виявлені фрагменти, що відповідають екземплярам класів лексичної онтології, таким, що між цими класами в онтології ПрО є відношення х, тоді фрагмент ПМ-тексту, що знаходиться між фрагментами, може бути екземпляром словоформи х.

Якщо в одному реченні є фрагменти – екземпляри класів лексичної онтології, але в онтології ПрО не зафіксовані відношення

між цими класами, то необхідно запитати користувача про доцільність поповнення онтології ПрО новим відношенням.

Онтологія ПрО і лексична онтологія, побудовані на попередньому етапі, потім можуть використовуватися для пошуку та семантичної розмітки документів, близьких за тематикою до цієї ПрО. Після цього така нова множина документів може стати основою для подальшого поповнення онтології ПрО.

**Висновки**

Методи лінгвістичного аналізу й Text Mining дають змогу здобувати знання, що містяться в природномовних документах. Використання в такому пошуку елементів онтологічного аналізу значно підвищує ефективність і гнучкість роботи, дає змогу формалізувати сферу індивідуальних інформаційних потреб користувача й подати йому семантично розмічену інформацію.

Наявність семантичної розмітки уможливлює автоматизоване виконання складних запитів з використанням знань ПрО, що зберігаються в онтології. Інтероперабельне подання й наявність посилань на онтологію дають змогу для використання цієї розмітки іншими застосуваннями, зокрема й сервісами Semantic Web.

# РОЗДІЛ VII.
# ВИКОРИСТАННЯ ВІДКРИТИХ ДЖЕРЕЛ WEB ДЛЯ ПОПОВНЕННЯ ОНТОЛОГІЧНИХ МОДЕЛЕЙ ПРЕДМЕТНИХ ОБЛАСТЕЙ

Одним з визначальних факторів ефективності застосування онтологій як джерела знань у різноманітних інтелектуальних застосуваннях Web є їх актуальність. Через надзвичайно великий обсяг інформації, доступ до якої надає Web, розробники багатьох сайтів зазвичай не можуть гарантувати, що подані ними відомості відповідають сучасному стану й відображають усі зміни в навколишньому середовищі. Але колективна праця всієї спільноти користувачів Web на базі сучасних інформаційних технологій дає змогу значно покращити ситуацію – наявність відкритих (як для читання, так і для редагування) джерел Web забезпечує актуальне і вчасне поновлення інформації й відображення знань багатьох осіб та організацій.

Для того, щоб використовувати ці відкриті джерела з метою поповнення й удосконалення різноманітних онтологій, потрібно розуміти їх структуру, особливості створення й потенційні проблеми функціювання. Тому розглянемо докладніше як самі відкриті джерела Web, так і засоби здобуття з них корисних онтологічних знань.

Для цього проаналізуємо технології колективного збирання й аналізу інформації на основі технології Open Source Intelligence [213]. Сьогодні ці інформаційні ресурси – соціальні мережі, Wiki-ресурси, блоги тощо, які створюють та поповнюють мільйони користувачів Web, стають усе більш значущим джерелом знань [377]. Інформація в таких джерелах має певну структурованість, але вона досить часто буває суперечливою й неповною. Open Source Intelligence є складником Open Source [375], що стосується лише створення й використання інформаційних ресурсів.

Останнім часом значної популярності набула ідея поширення вільного й відкритого програмного забезпечення, тобто програм, що поширюються разом з вихідним кодом. Це означає, що будь-який програміст може корегувати таку програму відповідно до власних потреб і навіть поширювати виправлений варіант. Такий підхід дає змогу через Web працювати над однією програмою сотням осіб, що, зазвичай, дає свої плоди. Спочатку Open Source стосувалися тільки розроблення програмного забезпечення, але зі збільшенням кількості користувачів мережі Інтернет і зростанням їхньої кваліфікації цей підхід поширився й на інші галузі. Однією з таких галузей є колективне

збирання й аналіз інформації, що в практичному застосуванні отримала назву «Інтелект Відкритих Джерел» (Open Source Intelligence – OS-INT).

Рух «Відкритих Джерел» (Open Source) є новим проявом колективної праці, унікально пристосований до мережі Інтернет і розроблення високоякісних інформаційних продуктів для Web [464].

## 7. 1. Концепція відкритих джерел у Web

У 70-ті роки минулого століття програмне забезпечення спільно використовувалося розробниками, але згодом компанії-розробники почали обмежувати його вільне поширення, у зв'язку з дотриманням ліцензійних угод. Це спричинило негативну реакцію багатьох програмістів і користувачів. У 1983 р. Сталлман увів терміни «вільне програмне забезпечення» й «copyleft» і почав працювати над проектом GNU, спрямованим на те, щоб дати користувачам незалежність і стримати приватизацію інформації.

Сьогодні забезпечити на практиці спільне, вільне й відкрите використання продуктів досить важко. Потрібний новий спеціалізований підхід, який би досяг балансу між відкритістю інформації й безконтрольністю її поширення. Унаслідок цього, у процесі розвитку Інтернет-технологій у рамках руху «Відкриті Джерела Програмного Забезпечення» (Open Source Software) виник новий напрям – «Інтелект Відкритих Джерел» (Open Source Intelligence – OS-INT), метою якого є колективне збирання й аналіз інформації. OS-INT базується на таких принципах спільної праці, як експертна оцінка, практика взаємного оцінювання експертами власної роботи, превалювання незалежної репутації над адміністративним контролем, вільний обіг продуктів і гнучкі рівні залежності й відповідальності. Оцінки тих спеціалістів, що мають високу репутацію, мають більшу вагу. Знання здобуваються в процесі уточнень і досягнення консенсусу.

OS-INT складається з великої кількості незалежних проектів (приміром, Nettime, Wikipedia та Web-сайт NoLogo.org), які мають різну історію, різну технічну й соціальну стратегію для реалізації принципів спільного використання відкритих джерел.

В OS-INT є вагоме теоретичне й технологічне підґрунтя. Багато з базових Інтернет-технологій були створені для сприяння вільному й доступному обміну інформацією між експертами, у процесі якого інформація не тільки ефективно поширюється, а й спільно оцінюється. Найпростіші платформи OS-INT – поштові списки розсилання (e-mail lists) – з'явилися в середині 70-х рр. минулого століття. У 80-ті дошки FidoNet та Usenet надавали користувачам деякі можливості OS-INT, але

вони були досить складними у використанні. У 90-х рр. багато з цих платформ було трансформовано у зв'язку з появою Word Wide Web. Створення Web-стандартів було спрямовано на співробітництво між науковцями, що живуть у різних країнах по всіх континентах, але сутність Інтернету в процесі його розвитку значно комерціювалася. Принципи вільного спільного використання, що характеризували його початок, змінилися на товарно-орієнтовані контрольні структури, які традиційно домінували в індустрії вироблення контенту.

Передумови для ПЗ з відкритим кодом з'явилися одночасно з виникненням руху за лібералізацію. У 1984 році його очолив співробітник Масачусетського технологічного інституту Ричард Столман. Ідея freeware з'явилася в нього в процесі роботи над проектом GNU (Gnu Not Unix), метою якого було створення вільно поширюваної Unix-сумісної операційної системи. Філософія Столмана виражена в маніфесті GNU Manifesto девізом «Програмне забезпечення хоче бути вільним» (Software wants to be free). У 1985 році була створена асоціація Free Software Foundation (FSF). Столман запропонував концепцію copyleft, відповідно до якої всі авторські права повинні були бути відкинутими (AM rights reversed), а також новий тип ліцензування GNU General Public License (GPL). Прямим продовжувачем робіт Столмана став Лінус Торвальдс, що випустив у 1991 році ядро операційної системи Linux за ліцензією GPL. Торвальдс використовував цілий ряд пакетів із GNU, тому спочатку цю ОС нерідко називали GNU/Linux. Недоліком Free Software було те, що цей підхід не сприяв тому, що тепер уважають однією з найважливіших переваг відкритого ПЗ, а саме – можливості виникнення неформальних об'єднань розроблювачів.

Вирішальну роль у формуванні концепції відкритого програмного забезпечення (Open Source Software, OSS) зіграв Брюс Перенс. На основі його проекту Debian Free Software GuideHnes були вироблені визначення Open Source Definition. Незабаром після утворення Open Source Initiative (OSI) усередині руху почалися помітні розбіжності.

Реальна цінність OSS полягає в кваліфікації спільноти, що бере участь у тій чи іншій ініціативі, у тому, що її члени здатні спроектувати й побудувати. Як приклад, можна навести такі ініціативи OSS, як Linux, Apache і Grid, до роботи над якими долучилися видатні програмісти, учені й технічні експерти, спільна праця яких стала можливою завдяки наявності Internet і відкритих стандартів.

Особливий вплив на ставлення до OSS чинить висока якість продуктів, створених у відкритих кодах. Воно стало несподіванкою для тих, хто був вихований на класичних корпоративних традиціях, де все чітко структуровано – від початкових етапів досліджень і розробок

до реалізації, тестування окремих компонентів і систем загалом, аж до підтримки й обслуговування.

На користь OSS звичайно наводять такі аргументи:

- найбільш популярні продукти OSS збирають довкола себе велику кількість кваліфікованих експертів, тому вони більш ефективні, а швидкість їхньої розробки та внесення виправлень може бути ще вищою;
- відкритість кодів дає змогу їх повторно використовувати й адаптувати до потреб задачі;
- велика прихильність до стандартів у процесі розробки забезпечує високу здатність до взаємодії;
- конкуренція постачальників OSS-рішень веде до високої якості обслуговування.

Але при цьому є й недоліки:

- якщо проект не надто великий і популярний, то немає гарантії, що в ньому візьме участь висококваліфіковане співтовариство й буде забезпечена достатня якість продукту;
- орієнтація на рівних собі призводить до того, що недостатньо враховуються інтереси користувачів, і якість інтерфейсів в OSS-продуктах, зазвичай, є нижчою;
- істотно нижчим є і контроль над процесом появи нових версій: якщо немає формально організованого адміністрування, то з'являється ймовірність фрагментації кодів і появи рівнобіжних версій (code forking).

Порівняно з вільним програмним забезпеченням, приєднання до вільного проекту інформації є набагато простішим. Різноманітні помилки в програмному забезпеченні не можуть спричинити поломку комп'ютера й втрату цінних даних.

Для того, щоб додати інформацію до Wiki, не потрібно бути експертом. Будь-яка частина людського знання, подана в енциклопедичній формі, у проекті Wikipedia вітається й поважається. Інформація щодо подорожей більше стосується проекту Wikitravel. Книги й навчальні посібники містяться у Wikibooks. Майже кожен охочий, хто має здатність друкувати й доступ до Internet, може додавати свої знання.

Проекти, подібні до Wikipedia, що базуються на вільному програмному забезпеченні, підтверджують: є набагато більше людей, які можуть виправити орфографічну помилку, ніж програмістів, які можуть усунути помилку в програмі. Постачальників контенту можна поділити на три групи: коректори записів помилок у літерах і орфографічних помилок, користувачі, які час від часу додають деякі рядки, і професійні редактори вмісту (контенту).

Open Source Intelligence – ще досить молода галузь, що потребує деяких змін.

По-перше, це необхідність масштабування. Порівняно з традиційними широкомовними медіа, проекти OS-INT досить малі. Оскільки масштаб і відкритість тісно пов'язані із соціальною динамікою, то зростання не може бути легким для багатьох проектів.

По-друге, це економічна проблема. Більшість проектів OS-INT є суто добровільними. Ресурси для них – це пожертва. Наприклад, Wikipedia залежить в апаратній частині й смузі пропуску від Bomis Inc. NoLogo.org фінансується з ліцензійних платежів від продажу книг. Більшість проектів OS-INT не має свого достатнього прибутку для покриття деяких неминучих витрат. Тому сьогодні вони досить вдало покладаються на пожертвування (від співчутливих окремих осіб, корпорацій чи фондів), але тривала криза Інтернет-економіки не полегшує зростання капіталу, який стає дедалі важливим зі зростанням проектів в обсягах і збільшенням інфраструктури й смуги пропускання.

Порівняно з традиційними моделями виробництва й видавництва, проекти OS-INT здебільшого беруть участь за межами традиційної товарно-грошової економіки. Вкладники проектів загалом не мотивовані безпосередньою фінансовою вигодою. Однак не всі ресурси можуть бути забезпечені безкоштовно, тому повинні бути винайдені нові моделі фінансування таких проектів. Ймовірно, що проекти OS-INT, з економічного погляду, будуть розвиватися в гібрид, який передбачатиме прямі доходи (наприклад, передплата, реклама), добровільні пожертвування й волонтерські зусилля. Співвідношення цих різних елементів буде змінюватися від проекту до проекту. Для творчого експерименту є потреби й багато простору.

Незважаючи на ці зміни, є серйозні підстави, щоб будувати оптимістичний прогноз на майбутнє. По-перше, соціотехнологічний процес навчання поглиблюється. Платформи та методи OS-INT стають більш зрозумілими, а отже, перешкоди для користувачів, як і для постачальників, поступово усуваються.

З боку користувачів, досвід вивчення роботи в спільноті зростає. Їх відмінний характер розвивається, відшліфовується й оцінюється.

Для постачальників досвід вивчення OS-INT вкладається в складне, але вільно доступне програмне забезпечення за ліцензією GPL. Кошти для запуску нових проектів мінімальні, а можливості адаптування платформи до особливих потреб кожного проекту максимальні. Результат цього розмаїття, так само, збагачує спільний навчальний процес.

По-друге, оскільки засоби масової інформації (ЗМІ) змінюються на все меншу кількість конгломератів, які неухильно просувають

і контролюють безліч своїх медіа-продуктів, підвищується потреба в альтернативних каналах інформації, принаймні серед людей, що інвестують час та енергію пізнання в те, щоб бути критично інформованими.

З огляду на економіку керованих рекламою засобів масової інформації, стає зрозуміло, що можливості «альтернативної газети» достатньо обмежені. Платформи OS-INT, розподіляючи робочу силу всюди серед співтовариства, надають можливість створення більш широкої аудиторії, не піддаючись тим економічним тискам, яких зазнають широкомовні й друковані ЗМІ, особливо враховуючи той факт, що передплата забезпечує доступ до змісту без реклами.

Чим більше ЗМІ стає гомогенним, тим більше простору відкривається для альтернатив. І якщо ці альтернативи є життєздатними, то вони не повинні бути обмежені альтернативним змістом, але обов'язково повинні досліджувати структуру своєї продукції. Це перспектива й потенціал OS-INT.

Обсяг технологій OS-INT настільки ж широкий, як і діапазон співтовариств, і між ними наявні близькі зв'язки. Технології відкривають можливості й відбирають їх у тому ж сенсі, що роблять і соціальні співтовариства. Як зазначав Лоренс Лессінг (Lawrence Lessig), як код стосується онлайнового світу, так архітектура – світу фізичного. Шлях, яким ми йдемо й структури, у яких ми живемо – глибоко пов'язані. Культура технології все більше й більше стає культурою нашого суспільства.

## 7. 2. Технологія Wiki

Сьогодні одним з найпоширеніших напрямів OS-INT є технологія Wiki [468]. Саме ця технологія нині активно використовується для розробки систем менеджменту знань у різних сферах [394, 456].

Wiki усе частіше сприймається як новий тип колаборативної технології. Технологія Wiki може вплинути на управління знаннями, а також може підтримувати створення й спільне використання знань.

Перші Wiki-системи, реалізовані в 1995 році, давали змогу користувачам створювати, редагувати й організовувати контент у Web-форматі.

### 7. 2. 1. Історія створення Wiki

**Wiki** (від «Wiki Wiki», що гавайською мовою означає «швидко») – це гіпертекстове середовище (зазвичай – Web-сайт), яке дає змогу користувачам Інтернету відносно легко створювати й модифікувати його контент. Історія свідчить, що автор цієї концепції –

У. Канінгем [331] почув слово «wiki» в аеропорту Гонолулу (місцеві жителі називали так автобуси). Він вирішив назвати так свою технологію, тому що це слово дійсно відображає її сутність – адже Wiki здатна значно спростити життя компанії й прискорити обмін інформацією між співробітниками.

Термін Wiki може також стосуватися спільного програмного забезпечення (*collaborative software*), яке створюється для такого сайту.

**Wiki-технологія** – це технологія побудови Web-сайту, що дає змогу відвідувачам брати участь у редагуванні його вмісту – виправляти помилки, додавати нові матеріали, не маючи при цьому необхідності використовувати спеціальні програми, реєструватися на сервері й знати HTML [456].

Для створення Wiki-середовища необхідно спеціальне ПЗ – Wiki-двигун. Wiki-двигун забезпечує роботу такого Web-сайту. Інформація, презентована у Wiki, має нелінійну навігаційну структуру. Кожна сторінка, зазвичай, містить велику кількість гіперпосилань на інші сторінки.

Оригінальна система Wiki була винайдена в 1995 р. У. Канінгемом. Вона була створена для web-вузла Pattern Languages Community з метою спростити спільне створення й документування програмних зразків (software patterns).

У таких системах написання й редагування є колективним процесом. Читач, який бачить у статті помилку чи недолік, може негайно її виправити чи додати інформацію, якої бракує. Оскільки процес перегляду й уточнення є публічним і безперервним, то немає принципової різниці між попередніми й фінальними версіями інформації. Використання принципу експертного перегляду відкритих джерел дає змогу проекту зростати не тільки за кількістю статей, а й за глибиною, унаслідок колективного внеску читачів, що є експертами в цій галузі.

Спочатку Wiki-системи являли собою просто сайти, сторінки яких міг редагувати будь-хто безпосередньо з web. У сучасному варіанті – це система для збирання й структурування інформації, що характеризується такими ознаками:

- багато авторів, якими, зазвичай, можуть бути всі користувачі Wiki-ресурсу.
- багатокористувацький режим роботи усе редагування здійснюється через web-інтерфейс, є центральний сервер (чи кластер), що зберігає весь масив даних.
- можливість багаторазово редагувати текст за допомогою самого Wiki-середовища (web-сайту) без застосування особливих пристосувань з боку користувача.

- поява змін одразу після їх внесення.
- поділ інформації на однозначно ідентифіковані документи сторінки, кожна з яких має власну назву.
- нескладна мова розмітки, що дає змогу легко відокремити контент від оформлення.
- облік змін (версій) тексту й можливість повернення до попередньої версії.

У [456] міститься опис основних принципів, які регулюють розробку Wiki-застосувань (таблиця 7. 1).

***Таблиця 7. 1.***

**Основні принципи розробки Wiki-застосувань**

| Принцип | Пояснення |
|---|---|
| Відкритість Open) | Якщо виявляється, що сторінка є неповною або погано організована, будь-який читач може редагувати її так, як він вважає за потрібне. Wiki базується на технології з відкритим вихідним кодом (open-source). |
| Поетапність (Incremental) | Сторінки можуть містити посилання на інші сторінки, зокрема й на сторінки, що ще не були написані. |
| Органічність (Organic) | Структура й текстовий контент сайту відкриті для редагування й еволюції. |
| Простота (Mundane) | Невелика кількість (нерегулярних) текстових конвенцій має забезпечити доступ до найбільш корисної (але обмеженої) розмітки сторінки. |
| Універсальність (Universal) | Механізми редагування й організації такі самі, як і ті, що використовуються для написання, тому будь-який автор автоматично стає редактором і організатором. |
| Відкритість (Overt) | Відформатована (і опублікована) вихідна інформація запропоновує вхідні дані, що потрібні для її відтворення (наприклад, розміщення на сторінці.) |
| Уніфікація (Unified) | Імена сторінок отримуються з плаского простору (flat space), тому жодного додаткового контексту не потрібно, щоб інтерпретувати їх. |
| Точність (Precise) | Сторінки отримують імена з достатньою точністю, щоб уникнути конфліктів між більшістю імен, як правило, шляхом формування іменників. |
| Толерантність (Tolerant) | Інтерпретована (навіть якщо небажана) поведінка має переваги порівняно з повідомленням про помилки. |
| Спостережність (Observable) | За активністю в межах сайту може спостерігати будь-який інший відвідувач сайту. Wiki-сторінки розроблені на основі довіри. |

| Принцип | Пояснення |
|---|---|
| Конвергентність (Convergent) | Дублювання може видалятися шляхом пошуку й за посиланнями на аналогічний або пов’язаний контент. |

### 7. 2. 2. Характерні риси Wiki

Використання технології Wiki забезпечує такі переваги:

1) Іменування та ідентифікація. Wiki являє собою колекцію довільних документів, єдиний спосіб доступу до яких – ідентифікатор (у самому документі посилання на інший документ створюється автоматично).

2) Шаблони надають можливість збереження й подання структурованих даних.

3) Автоматичне створення різноманітних списків і класифікацій. Один з прикладів реалізації таких механізмів – це механізм категорій, який полягає в тому, що документ позначається як такий, що належить до якоїсь категорії. Після цього під час звернення до документа "Категорія: Задане ім’я" (у просторі імен "Категорія") буде виведений список документів, позначених заданим ім’ям.

4) Цілісність посилань: Wiki надає можливість відстежити як усі посилання з поточної сторінки, так і всі посилання звідкись на поточну сторінку.

Програмне забезпечення й контент, що міститься на Wiki-сторінках, найчастіше використовують ліцензію вільного поширення документації – GNU Free Documentation License (GFDL), що гарантує сумісність ліцензії з наявною сукупністю текстів GFDL. Це дає можливість мати один Wiki зі сторінками, що мають різну ліцензію.

### 7. 2. 3. Платформа розробки Wiki-ресурсів

Wiki-сайти – перспективний шлях розвитку приватних і публічних баз знань. Простота й зручність у підтримці колективної творчості, притаманні технології Wiki, дали змогу засновникам енциклопедичного проекту Nupedia – Д. Уельсу й Л. Сенгері використати її як основу електронної енциклопедії. Унаслідок цього в січні 2001 року було створено найпопулярнішу Інтернет-енциклопедію "Wikipedia" (спочатку вона працювала на базі програмного забезпечення UseMod, але потім перейшла на власну відкриту базу кодів, яку тепер використовують і багато інших Wiki) [468].

Платформа Wiki реалізувала одну з базових концепцій Бернерса Лі, надавши користувачам можливість бачити відправний код і вільно редагувати контент сторінок, які вони бачать. Зміни в контенті, внесені користувачами, набувають чинності відразу, без будь-якої перевірки

й перегляду навіть автором чи модератором, але обов'язково підтримуються функції, що дають змогу користувачам переглядати зміни й, за необхідності, повертати сторінки до попередніх версій.

Всесвітньо відомий приклад застосування технології Wiki – створена в 2001 році Вікіпедія – найбільша з безкоштовних онлайн-енциклопедій. Ця «суспільна» енциклопедія, нараховує сьогодні 1,5 млн. статей 100 мовами (для порівняння: Encyclopedia Britannica 2007 року містить близько 100 тис. статей). Невдоволення повільністю створення й виходу паперових енциклопедій було очевидним уже давно, але створення оперативних енциклопедій потребує багато людей і є економічно невиправданим. У Вікіпедії платою авторам стають не гроші, а соціальний статус. Особливість цієї енциклопедії в тому, що всі відвідувачі сайту wikipedia.org. можуть її редагувати й доповнювати власними статтями. За коректністю наведеної в енциклопедії інформації стежать тисячі осіб, тому окремі вандали не можуть зашкодити репутації порталу, і його наповнення залишається на досить високому рівні.

Інші цікаві проекти, що базуються на технології Wiki – словники Wiktionary, колекції книг Wikibooks, цитат Wikiquotes, документів Wikisource, новин Wikinews і матеріалів для самоосвіти Wikiversity.

На початку використання Wiki поширювався на підприємствах як спільне програмне забезпечення для комунікації в рамках проектів, intranet і обміну документацією. У грудні 2002 р. Socialtext випустив перше комерційне рішення Wiki з відкритим кодом. Після цього відкритий код програмного забезпечення Wiki швидко поширювався, легко завантажувався й встановлювався. Сьогодні для деяких компаній Wiki стало єдиним спільним програмним забезпеченням і повністю замінило статичний intranet.

Створення й редагування Wiki-сторінок. Кожна окрема сторінка Wiki називається «Wiki-сторінка» (*Wiki page*). Створення й редагування Wiki-сторінок не потребує від користувачів знання мови HTML: механізм Wiki надає можливість написання документів за допомогою простої мови розмітки й Web-браузеру. Всі сторінки Wiki-сайту – це статті, які мають назву, на яку можна посилатися з інших статей, і вміст (рис. 7. 1).

Скориставшись спеціальним посиланням або кнопкою, відвідувач Wiki-сайту може відредагувати текст сторінки, яку він розглядає, безпосередньо всередині свого web-браузера й зберегти змінений варіант на сервері.

Усі виправлення статей на Wiki-сайтах зберігаються в базі даних. Кожен користувач будь-коли може зробити запит щодо будь-якого попереднього варіанта статті чи проглянути різницю між будь-якими

двома минулими варіантами статей (за допомогою посилання «Останні виправлення», що дає змогу легко й швидко відстежувати випадки вандалізму.

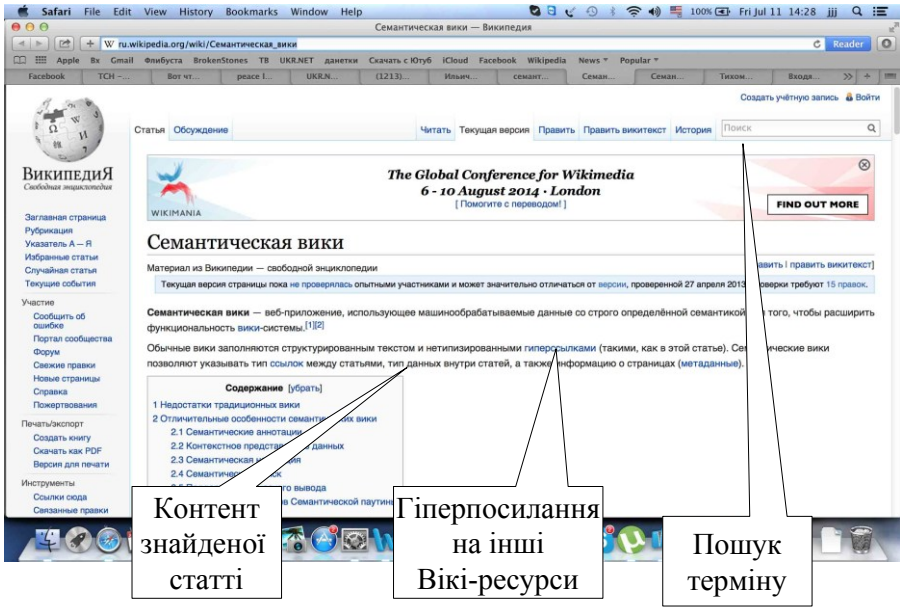



Рис. 7. 1. Сторінка Вікіпедії

Формат Wiki-сторінок (Wiki-text) – це спрощена мова розмітки, що використовується для того, щоб виділити в тексті різні структурні й візуальні елементи або вказати на них. Кожна Wiki-система має власний стиль і синтаксис залежно від реалізації. У багатьох (але не в усіх) реалізаціях Wiki гіперпосилання показується таким, яким воно є насправді, на відміну від HTML, де невидиме гіперпосилання може мати довільний видимий текст або зображення.

Характерною ознакою Wiki-технології є легкість, з якою сторінки можуть бути створені й модифіковані. Загалом для того, щоб модифікації були прийняті (з'явилися на сторінці), жодного механізму попереднього перегляду (коректором, модератором тощо) публікації немає. Більшість Wiki відкрита для широкої публіки навіть без необхідності реєстрування, але деякі приватні Wiki-сервери вимагають авторизації користувача. Однак значна частина редагувань може бути зроблена в режимі реального часу й миттєво з'являтися на сторінці. Це може часто призводити до зловживання системою або виведення її з ладу (автоматичне генерування спаму тощо).

Двигун Wiki-сайту робить запис змін так, щоб у будь-який час сторінку можна було б повернути до кожного з її попередніх станів. Wiki-система може також містити різні інструментальні засоби для простого контролювання стану Wiki, що постійно змінюється, а також для обговорювання й розв'язання багатьох неминучих проблем,

що виникають через різні погляди на зміст Wiki-сторінок. Контент Wiki може містити неправдиву й некоректну інформацію, що вводиться користувачами навмисно або випадково.

Більшість Wiki надає користувачам необмежений доступ так, щоб була можливість доступу до сторінок і їх редагування без обов'язкової реєстрації, що, зазвичай, потрібно в процесі користування іншими типами інтерактивних сайтів, наприклад, інтернет-форумів або чатів. Для багатьох споживачів цей факт є дуже важливим.

Деякі останні версії Wiki-двигунів використовують різні методи: вони дають змогу редагувати методом WYSIWYG (що видно, те і є), що реалізується за допомогою JavaScript або ActiveX, які перетворюють графічно введені команди форматування, наприклад, шрифти «жирний» («грубий») і «курсив» на відповідні теги HTML. У таких реалізаціях HTML-розмітка Wiki-сторінки генерується й постачається на сервер автоматично, таким чином, користувач не бачить жодних технічних деталей. Споживачі, які не мають необхідного плагіну, можуть редагувати сторінку, набираючи безпосередньо код HTML. Останні Wiki-двигуни генерують синтаксис Wiki замість HTML.

Хоча протягом років стандартом де-факто був синтаксис першого WikiWikiWeb, нині команди форматування змінюються залежно від Wiki-двигуна. Простий Wiki дає змогу здійснювати тільки основне форматування тексту, тоді як більш складні системи забезпечують підтримку таблиць, зображень, формул і навіть інтерактивні елементи типу голосувань та ігор. Загального стандарту для розмітки Wiki на сьогодні немає.

Пошук у Wiki. Більшість Wikis забезпечує пошук за заголовками, а деякі – повнотекстовий пошук (full-text search) з усієї бази. Сфера пошуку залежить від використовуваної Wiki-двигуном бази даних, реалізації індексованого доступу до бази даних, що необхідно для високошвидкісного пошуку на великих Wikis. Для пошуку в множині Wikis створений пошуковий механізм MetaWiki, який здійснює пошук за ключовими словами (keyword-based).

Контроль за змінами в контенті Wiki-сторінок. Wiki-система надає можливість перегляду історії сторінок, що видає зміни між двома ревізіями (версіями) сторінки.

Усі Wikis спроектовані за основним принципом – краще, щоб помилку було легко виправити, ніж щоб її було важко зробити. Так якщо Wikis є відкритими, вони мають засоби для перевірки правильності нових здійснених редагувань наявних сторінок. Для цього майже в кожному Wiki є сторінка «Останні Зміни» (Recent Changes), яка показує пронумерований список усіх редагувань сторінок, що зроблені за визначений інтервал часу. Деякі Wikis мають можливість фільтрувати

список, щоб видалити старі редагування й редагування, зроблені автоматично, за допомогою скриптів імпортування ("bots").

Здебільшого у Wikis, окрім журналу змін, реалізуються такі функції: хронологія ревізій (*Revision History*), що показує попередні версії сторінки; функція пошуку розбіжностей (diff), яка висвічує рядки, що є різними в двох ревізіях. Редактор Wiki за допомогою Хронології ревізій може переглядати попередню версію статті й відновлювати її. Постійний користувач Wiki може дивитися розбіжності між редагуваннями, що зазначені на сторінці «Останні Зміни», й у разі, якщо це редагування є неприпустимим, має змогу перевірити хронологію й відновити попередню ревізію. Цей процес є більш-менш спрощеним, залежно від використаного програмного забезпечення Wiki.

Постійні користувачі, що користуються логіками, часто перевіряють, які були внесені зміни й ким. Анонімні редагування розглядаються більш прискіпливо.

У тому разі, якщо на сторінці «Останні Зміни» пропущені неприпустимі виправлення, деякі Wiki-двигуни мають додатковий механізм для керування змістом (контентом), щоб гарантувати збереження «якості» сторінки чи набору сторінок. Користувач-автор, який розмістив сторінки, буде попереджений щодо модифікацій його сторінок, що дає змогу йому швидко перевірити коректність нових редагувань.

### 7. 2. 4. Вандалізм у Wiki-середовищі

Відкрита філософія більшості Wiki – давати змогу будь-якому користувачеві змінювати вміст сторінок. Але наміри таких редакторів не завжди добрі. Більшість публічних Wiki уникає обов’язкових реєстраційних процедур, але багато з найбільших Wiki-систем включно з MediaWiki, MoinMoin, UseModWiki й TWiki мають певні методи для обмеження доступу до написання тексту. Деякі Wiki-системи мають можливість забороняти редагування певним індивідуальним користувачам, блокуючи конкретні IP-адреси або імена користувачів, які відомі своїми шкідливими діями.

Загальновживаний спосіб захисту від настирливих «вандалів» – просто дозволити їм зіпсувати стільки сторінок, скільки вони бажають, знаючи, що ці сторінки легко відстежити й повернути назад після того, як вандал піде. Однак ця стратегія швидко може стати непрактичною, оскільки злість або почуття власної неповноцінності таких осіб можуть змусити їх систематично псувати чужі статті.

У разі надзвичайних ситуацій деякі Wiki мають можливість змінювати режим баз даних так, що вони стають доступними тільки для читання. Інші застосовують політику, що дозволяє продовжувати

редагування лише давнім користувачам, які зареєструвалися до якоїсь довільно обраної дати. Однак, загалом кажучи, будь-яку шкоду, завдану «вандалом», можна швидко й легко виправити. Більш проблематичними є непомітні помилки, які вставляють у сторінки, наприклад, зміни в датах випусків альбомів співаків, їхньої дискографії.

Багато Wiki мають змогу захистити певні сторінки від редагування: захищені сторінки Вікіпедії можуть редагувати тільки адміністратори, які здатні встановлювати й знімати такий захист. Але така практика суперечить основній філософії Wiki, а тому її, як правило, уникають. Наприклад, англійська Вікіпедія водночас має близько тридцяти захищених сторінок з понад трьохсот тисяч.

### 7. 2. 5. Wiki-спільноти

Сьогодні найбільшою з Wiki є англійська версія Wikipedia. Деякі інші великі Wiki – це WikiWikiWeb, Wikitravel, World66. Також є WikiNodes – вузли Wiki-сторінки, розміщені на Wiki, які містять описи пов'язаних за темою Wikis. Вони звичайно організовані (і мають відповідну назву) як родичі й делегати. *Родинний Wiki* – це просто Wiki, який може мати обговорення подібного змісту (контенту) або має можливість якось інакше входити до кола питань, що пов'язані з предметом обговорення. *Делегат Wiki* – Wiki, що погоджується делегувати деякий зміст своїх сторінок до родинного Wiki.

Частина Wiki-спільнот – приватні й вузько спеціалізовані (приміром, у межах підприємств, де вони використовуються як спільне програмне забезпечення). Такі Wiki часто використовуються для ведення внутрішньої документації, для локальних систем і прикладних програм. Вони зростають і розвиваються повільніше за загальнопоширені, такі, як Wikipedia. Чим більше користувачів має Wiki-спільнота, тим вище темп її зростання й удосконалення.

### 7. 2. 6. Відмінності Wiki-систем від систем керування контентом

Wikis мають спільне використання й підтримують деякі функції, що подібні до відомих систем керування контентом (content management systems – CMS), які використовуються підприємствами й спільнотами, орієнтованими на практичне застосування. Порівнюючи CMS з технологією Wiki, необхідно зазначити такі базові особливості:

1. Назва статті одночасно є гіперпосиланням.

2. Статті можуть створюватися або редагуватися в будь-який час кожним користувачем (з деякими обмеженнями для захищених статей).

3. Статті доступні для редагування у web-браузері.

4. Кожна стаття одним кліком надає доступ до перегляду й редагування хронологій /версій текстових варіантів сторінки, що підтримує пошук розбіжностей і видобування попередніх версій.

5. Кожна стаття одним кліком надає доступ до сторінки обговорення, що стосується цієї статті.

6. Останні додавання та інші модифікації статей можуть бути перевірені (відстежені) в активному або пасивному режимі.

Усі ці особливості не специфічні для Wiki, деякі з них розвинулися незалежно. Однак поняття Wiki недвозначно стосується цього основного набору характерних особливостей. Поєднані разом, вони відповідають породжувальній, змінній природі Інтернету й спрямовані на те, щоб «підштовхнути» кожного користувача сприяти розбудові Інтернету.

### 7. 2. 7. Теоретичний базис Wiki

Незважаючи на простоту використання, технологія Wiki ґрунтується на серйозному теоретичному базисі й використовує здобутки з інших галузей знань. Термін «наука Wiki» (Wiki Science) означає набір відомостей, поглядів і рекомендацій щодо створення й розвитку Wiki-застосувань. Наука Wiki використовує результати, отримані в рамках таких наукових напрямків, як (рис. 7. 2):

- теорія хаосу (Chaos teory);
- теорія емерджентної поведінки (Emergent behaviour; автор Е. Бонаб'ю);
- штучне життя (Artificial life);
- когнітивна нейробіологія (Cognitive neuroscience);
- загальне управління якістю (Total Quality Management).

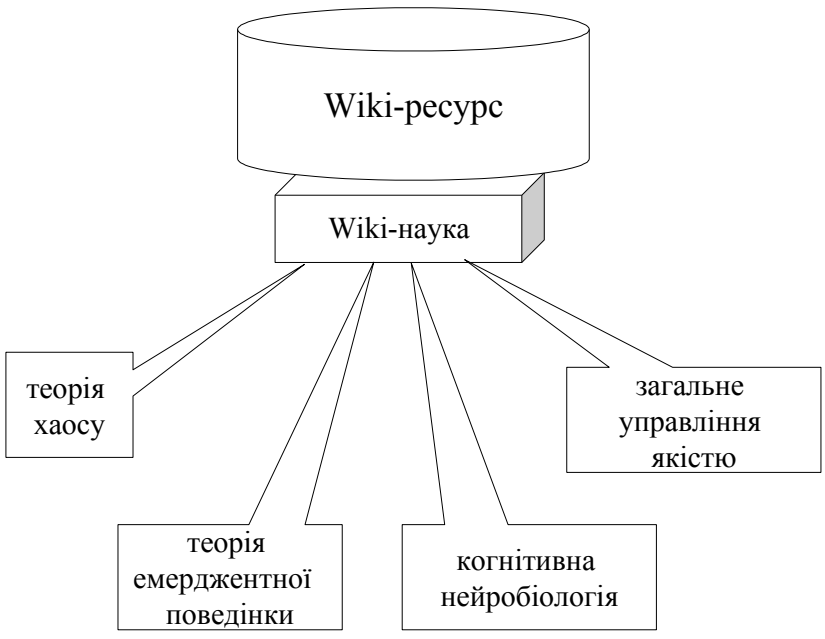



Рис. 7. 2. Теоретичний базис науки Wiki

Технологія Wiki є основою для групової взаємодії й управління знаннями [456] (рис. 7. 3).


Рис. 7. 3. Технології, що застосовуються у Wiki-системах

### 7. 2. 8. Еволюція Wiki-ресурсів

«Ефект брунькування» (budding effect) – термін, уведений для опису явища, коли один Wiki-ресурс породжує інший за аналогією до рослини, що містить зародок. Великий батьківський Wiki-ресурс може «народити» кілька менших дочірніх, створюючи своєрідне генеалогічне дерево.

Wiki може брунькуватися як повільно, так і швидко. Вони можуть виникати як результат зіткнення на батьківському Wiki або ж можуть поступово формуватися з членів батьківського Wiki, що вважають себе належними й до батька, й до дитини. Є багато технологій, що підтримують процес брунькування, починаючи з близького гіперпосилання й завершуючи сторінкою кластера з простору найменувань (name spaces).

Еволюція статей Wiki – це процес, коли стаття Wiki розвивається, починаючи з кореня (*stub)* або спеціально виділеного місця для майбутньої статті з невеликою кількістю інформації. Такий корінь уводиться тим, хто хоче ідентифікувати потребу в статті, але не має часу або знань для створення її повної версії. З часом користувачі додають, розширюють цю статтю, поки вона не стане актуальною. Якщо корінь оригіналу не створити, то й інші користувачі не будуть зацікавлені у написанні кінцевої версії статті. Це є співробітництвом

серед нестачі, де невелике початкове зусилля змушує інших виконувати певні дії й створювати кінцевий продукт, цікавий для всіх, навіть якщо кожен з учасників розробки не має всіх необхідних складників.

Самозагоєння (self healing) – це здатність Wiki швидко зцілитися після акту вандалізму або іншого пошкодження.

### 7. 2. 9. Використання технологій Wiki в освіті

Вікіпедія – дуже вдалий інструмент, що спонукує активно працювати з інформацією замість того, щоб пасивно її сприймати. Користувачі можуть не тільки читати й аналізувати статті, а й брати участь у їх написанні.

Проект Wikiversity є найбільш потенційним у рамках WikiMedia (Wikimedia), він спрямований на навчання за допомогою Wiki. Уведено термін Wiki-студент (*WikiU* student) (рис. 7. 4).


Рис. 7. 4. Сторінка проекту Wikiversity у Вікіпедії

Організовуються світові конференції та симпозіуми стосовно досліджень, які реалізуються в рамках WikiResearch, створюється система для проведення експертного оцінювання академічних публікацій.

На базі Wiki розробляється цікава наукова теорія процесу здобуття знань, що базується на теорії «емерджентної поведінки» (Emergent behaviour). Ця нова теорія освіти припускає, що використання Wiki спрощує здобуття знань індивідуумами. Процес здобуття знань розглядається як шлях між двома точками – точкою, що характеризує знання, які індивідуум уже має, і точкою нових знань. Зазвичай, перехід з однієї точки до іншої, нової здійснюється з докладанням великих

зусиль і траєкторія переходу має форму кривої. Скорочення шляху між цими точками відбувається завдяки тому, що Wiki має певну структуру, складену за певною логікою, а групова творчість поступово викристалізовує найчіткіші й найзрозуміліші вирази й образи, найбільш прийнятні для засвоєння. Отже, траєкторія переходу між точками набуває вигляду прямої, відстань між точками зменшується, а кількість необхідних зусиль зменшується. Ця теорія сьогодні перебуває в стані зародження.

### 7. 2. 10. Програмні реалізації Wiki

Найбільш поширеними є серверні реалізації системи Wiki. Це означає, що функції редагування, відображення й керування забезпечуються на сервері за допомогою програмного забезпечення Wiki, яке відтворює зміст сторінки у форматі HTML для відображення у Web-браузері.

З боку клієнта Wiki-системи необхідно тільки, щоб сервер «обслуговував» Wiki за допомогою протоколу HTTP. У цьому типі Wiki-систем усе виконання складається з необхідності конвертування первинного Wiki-тексту в екранне відформатоване зображення сторінки у web-браузері клієнта. Аналогічно, інструментальні засоби редагування та інших функціональних можливостей містяться в програмі браузера.

Wiki-система, що реалізується з боку клієнта, відповідає HTML-коду, у якому сторінка містить команди для виконання, які інтерпретуються web-браузером. Такі системи можуть бути небагато більшими ніж код плагіну для найбільш традиційних браузерів.

На сьогодні про Wiki потрібно говорити саме як про **технологію**, а не про програмний продукт чи систему, оскільки є багато реалізацій Wiki. Загальною рисою всіх реалізацій є швидкість уведення контенту й простота керування ним. Багато редакторів контенту Wiki мають функції WYSIWYG, а замість досить складної HTML-розмітки використовують спрощену Wiki-розмітку, яка, на жаль, не зовсім збігається в різних реалізаціях. Крім оформлення тексту, редактори Wiki надають засоби введення математичних формул, символів інших кодувань, а також інші корисні функції. Уведення контенту прискорюється також за рахунок автоматизації деяких задач і використання шаблонів, що дають змогу швидко перетворити вид документа.

Популярність технологій Wiki сприяла появі великої кількості реалізацій практично для всіх можливих платформ і конфігурацій ПЗ. Крім того, різні реалізації Wiki неоднаково розширюють основні можливості технології, додаючи, наприклад, убудовані графічні редактори чи сервіси WAP.

**TikiWiki** має всі необхідні функції, реалізується за допомогою PHP і використовує СКБД MySQL. TikiWiki підтримує блоги, передачу даних мобільними пристроями й управління голосом. За допомогою спеціального Java-скрипту можна створювати прості векторні малюнки. Система підтримує поштові списки розсилання, публікацію новин у форматі RSS, містить чат. Рядові користувачі системи володіють значним особистим простором, що дає їм змогу не тільки вибирати персональні налаштування для перегляду сторінок, а й надавати особистий блокнот для записів і календар для нагадувань.

Систему **MediaWiki** використовує проект Вікіпедія. MediaWiki побудована на основі PHP і MySQL. MediaWiki відрізняється якісним редактором контенту. Крім того, серед модулів розширення MediaWiki є редактор FCEditor, інтерфейс якого виконаний у стилі Ms Word (рис. 7. 5).

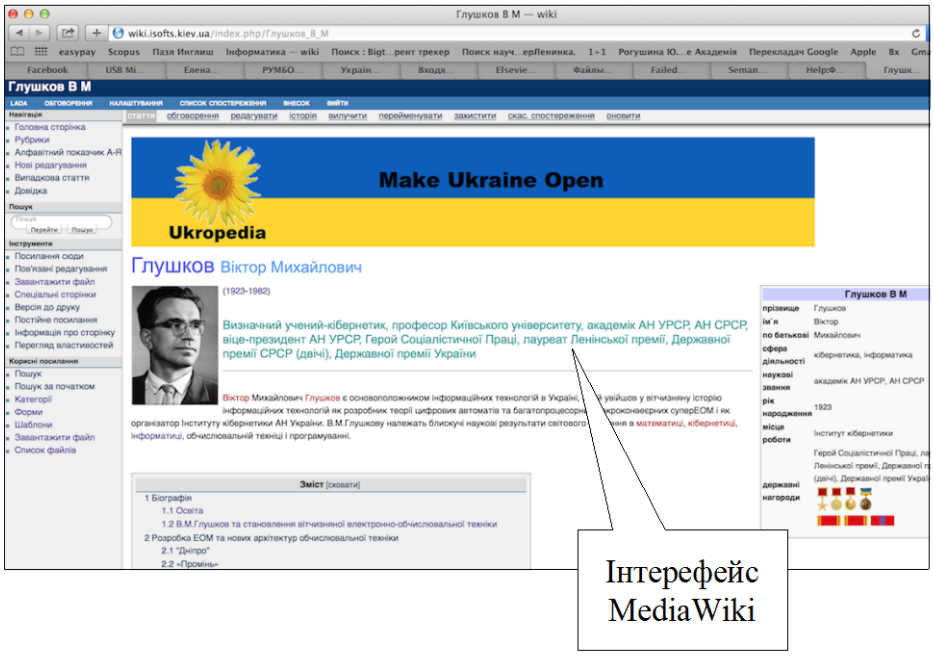



Рис. 7. 5. Інтерфейс сторінки на базі MediaWiki

Реалізація Wiki **UseModWiki** написана на Pearl. UseModWiki не використовує бази даних, а тому може бути розміщена на серверах, що не пропонують таку послугу. Цікавою функцією UseModWiki є наявність «під-сторінок», що можуть функціонувати як самостійні Wiki.

**FlexWiki** написана на C# і призначена для роботи в середовищі Ms Windows +NET. FlexWiki може обійтися без баз даних, а може використовувати SQL Server чи інші СКБД, доступні через MSDE. Цікавий інструмент FlexWiki – FlexWikiPad – редактор контенту, що підтримує підсвічування синтаксису й автоматичне завершення введення.

**SushiWiki** також реалізована на C# і орієнтована на SQL Server 2000 та інші СКБД, доступні через MSDE. Якщо в інших редакторах контенту Wiki можливості WYSIWYG обмежені чи необхідні спеціальні модулі для їх під'єднання, то редактор SushiWiki дає можливість редагувати сторінку, не відриваючись від її зовнішнього вигляду.

### 7. 3. Українська Вікіпедія

Українська Вікіпедія – це україномовний розділ Вікіпедії – багатомовного Інтернет-проекту зі створення Wiki-енциклопедії, яку може редагувати кожний охочий користувач Web. На сьогодні кількість статей української Вікіпедії становить 572 676 (станом на 04. 06. 2015 р., 13:27). За цим показником вона посідає 16-е місце серед усіх мовних розділів, 11-е місце серед європейських вікіпедій і 3-є місце серед вікіпедій слов'янськими мовами. Доступ до цієї сторінки можна отримати, набравши в пошуковому запиті до Google «Українська Вікіпедія» (рис. 7. 6).

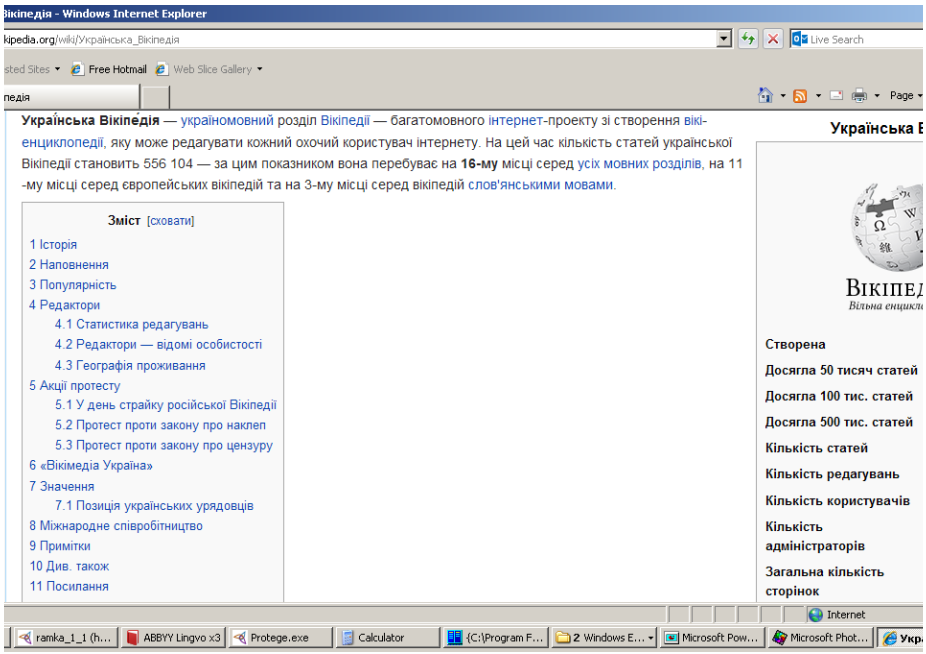



Рис. 7. 6. Українська Вікіпедія

Ця стаття Вікіпедії містить відомості про історію, контент та інструментарій українського сегмента Вікіпедії.

У режимі «Редагувати код» можна проглядати й змінювати контент цієї сторінки (рис. 7. 7).

У режимі «Перегляд історії» можна дізнатися про попередні версії цієї сторінки. Щоб довідатися, як правильно редагувати сторінки, слід ознайомитися з відповідною сторінкою, перейшовши до неї

за допомогою пункту меню в лівій частині екрану «Довідка». Наприклад, можна довідатися, як створити нову сторінку або як посилатися на інші сторінки Вікіпедії.

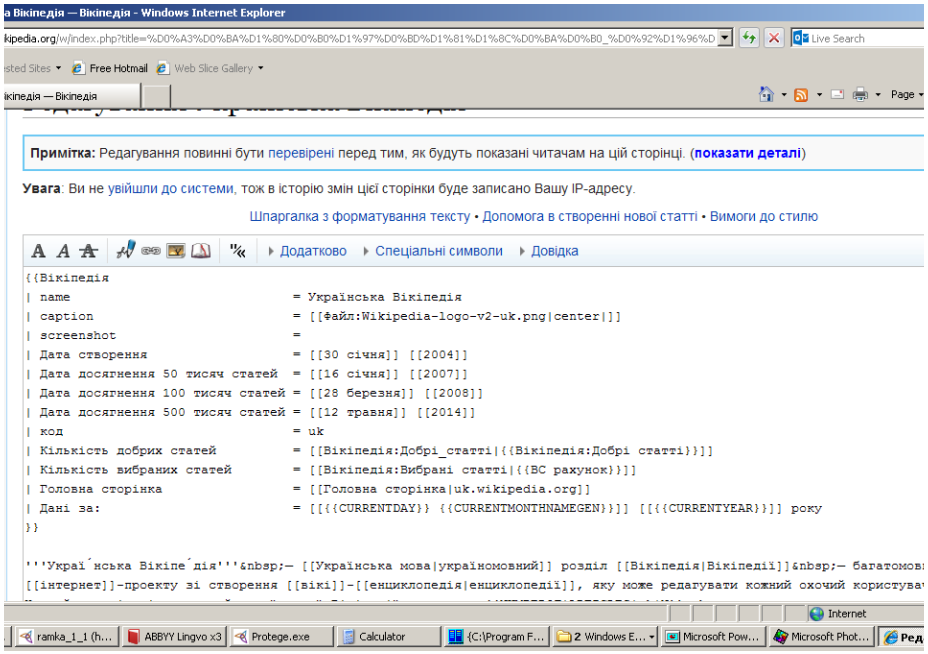



Рис. 7. 7. Редагування коду в українській Вікіпедії

## 7. 4. Семантизація Wiki-ресурсів та онтології

Одна з основних проблем IP у Web, що найбільш яскраво виявляється для Wiki-ресурсів, – узгодженість вмісту. Постійне дублювання даних зумовлює можливість наявності подібної інформації на декількох різних сторінках, а внесення змін на одній сторінці спричиняє необхідність відновлення на всіх інших сторінках. Виникають і проблеми повторного використання знань. Неможливість використання типізованих властивостей породжує величезну кількість тегів або категорій.

Усунути такі недоліки може допомогти семантизація Wiki, що пропонують користувачеві засоби для додавання семантичної розмітки інформації [425].

Одним з найвідоміших прикладів застосувань Semantic Web є Semantic Wiki. Це семантичне розширення традиційної Wiki-технології, що надає користувачеві можливість для додавання семантичної розмітки до контенту. В цьому розширенні для запитів використовується мова SPARQL [438]. Система Semantic MediaWiki написана за допомогою механізму розширень MediaWiki. Це спрощує її

інтеграцію з наявними застосуваннями MediaWiki. Необхідні колекції семантичних даних у SMW поповнюються користувачами, вони можуть додати анотації до Wiki-текстів статей за допомогою спеціальної розмітки.

У семантичній Вікіпедії модель її знань відображена на її ж сторінках. Семантична Вікіпедія пропонує такі елементи для розмітки статей: 1) категорії; 2) типізовані посилання; 3) атрибути – властивості вмісту статей.

Семантична розмітка дає змогу значно спростити всю структуру Wiki, допомагає користувачам швидше знаходити потрібну інформацію.

### 7. 4. 1. Напрямки семантизації Wiki

Значна частина інформації у Вікіпедії може бути презентована у вигляді списків, вихідними даними для яких є різні статті. У процесі відновлення таких статей Семантична Вікіпедія автоматично оновлює всі похідні від них списки й результативні дані. Наприклад, якщо з'явилися нові відомості про населення якого-небудь міста, то автоматично оновиться інформація про населення країни, у якій розташовано це місто. Запити, за якими створюються самі списки, повинні формуватися спочатку користувачами, а підтримка їх в актуальному стані – це вже проблема, яку розв'язує сама Вікіпедія. Такі списки завжди актуальні та їх легко налаштувати для одержання додаткової інформації [162].

Використання метаданих дає змогу краще структурувати інформацію. Наприклад, якщо деякий об'єкт, опис якого міститься у Вікіпедії, класифікується як «Людина», то можна вказати на такі його атрибути, як «Рік народження», «Ім'я», «Професія», а потім використовувати ці атрибути та їхні значення для пошуку й формування нових списків, наприклад, знайти всіх людей, згаданих у Вікіпедії, день народження яких збігається з поточною датою.

Метадані дають змогу встановити зв'язки між аналогічними статтями, написаними різними мовах, що уможливить актуалізацію даних і здатність знаходити невідповідності між ними. Наприклад, якщо ми введемо останні відомості про населення Києва до української Вікіпедії, то автоматично оновляться відповідні статті в усіх Вікіпедіях, де згадується це число.

У такий спосіб Вікіпедія перетворюється з простого сховища даних на розподілену базу знань і забезпечує розподілений доступ до цієї бази знань через Web.

Структура статей Вікіпедії може бути подана у вигляді онтології, класами якої є елементи розмітки статей, а контент самих статей – джерелом для структурованого здобуття знань методами Data Mining.

Семантичні Wiki здатні автоматизовано обробляти дані з чітко визначеною семантикою, а це дає змогу істотно розширити функціональність таких систем. Наприклад, можна обробляти типізовані посилання між Wiki-статтями й усередині статей [57].

Структуровані дані зберігаються в семантичних Wiki-ресурсах або безпосередньо в тексті сторінки за рахунок розширення Wiki-розмітки (зокрема, так зберігає метадані Semantic MediaWiki), або ж окремо в спеціальній формі (приміром, в Ontowiki).

Посилання між статтями містять у своєму імені інформацію про тип зв'язку. Приміром, у KiWi можна пов'язувати структуровані дані за допомогою засобів RDF, а потім співвідносити RDF-терміни з фрагментами тексту.

Більшість семантичних Wiki дають змогу здобувати додаткову інформацію за посиланням, що підтримує базовані на семантиці способи навігації та семантичний пошук. Так, користувач може побудувати багатокритеріальний запит, подібний до SPARQL, а використання семантичних анотацій дасть змогу відфільтрувати результати пошуку.

Семантичні Wiki орієнтовані на використання форматів чи технологій Semantic Web для подання й збереження знань. Приміром, багато з них підтримують імпорт /експорт знань у RDF і OWL, а для користувацьких запитів до Wiki слугує SPARQL [428], що забезпечує підтримку логічного виведення в системі.

У перших семантичних Wiki (Platypus Wiki і Rhizome Wiki ) RDF-дані презентувалися як текст, що вільно редагується й не пов'язується з неструктурованим контентом.

Найчастіше семантизація розмітки досягається шляхом збагачення посилань (анотацією тексту), що містять опис їх значень. Приміром, у статті про Київ, посилання на статтю Україна можна іменувати як «is capital» чи «located in». Таке анотування дає змогу значно поліпшити візуалізацію інформації (демонстрація контекстуальної інформації), навігацію (доступ до релевантної інформації), пошук (пошук за контекстом, а не лише за текстом).

Є два види взаємозбагачення онтологій і Wiki: системи, у яких Wiki використовуються для побудови онтологій, і системи, що використовують онтології для роботи Wiki. Часто Wiki-системи використовуються як зовнішній інтерфейс колаборативної розробки онтологій (Semantic MediaWIki, PlatypusWiki). В іншому підході онтології дають змогу поліпшити функціювання самої Wiki-системи (приміром, IkeWiki, SweetWiki,SWIM).

*Таблиця 7. 2.*

**Найпоширеніші Wiki-системи**

| Назва | Посилання | Платформа /Мова |
|---|---|---|
| AwkiAwki | http://awkiawki.bogosoft.com/ | Awk |
| CitiWiki | http://wiki.cs.cityu.edu.hk/citiwiki | PHP |
| CLiki | http://www.cliki.net/ | CommonLisp |
| DidiWiki | http://didiwiki.org/ | C |
| DotNetWiki | http://www.dotnetwiki.org/ | C# |
| EddiesWiki | http://c2.com/cgi-bin/wiki?EddiesWiki | C++ |
| FlexWiki | http://www.flexwiki.com/ | C# |
| FreeWiki | http ://sourceforge.net/projects/freewiki/ | Java |
| JavaWiki | http://c2.com/cgi-bin/wiki?JavaWiki | Java |
| JspWiki | http ://www.j spwiki.org | Java |
| MediaWiki | http ://www.mediawiki.org/ | PHP |
| NoodleWiki | http://flangy.com/dev/asp/noodle/ | ASP |
| OpenWiki | http ://www. OpenWiki.com | ASP |
| PmWiki | http://www.pmichaud.com/pmwiki | PHP |
| TiddlyWiki | http://www.tiddlywiki.com | JavaScript |
| WackoWiki | http ://wackowiki.org/WackoWiki | PHP |
| WebMacro Wiki | http ://www.webmacro.org/WebMacro Wiki | Java |
| WikiAsp | | ASP |
| WikiCpp | http://wikicpp.sourceforge.net/ | C++ |
| WikicWeb | http://c2.com/cgi-bin/wiki?WikicWeb | C |

На сьогодні є багато різноманітних реалізацій семантичних Wiki для різних платформ.

### 7. 4. 2. Реалізації семантичних Wiki

Одна з найбільш поширених Wiki-систем – **Semantic Media Wiki** (SMW) – є розширенням двигуна MediaWiki й базується на використанні семантичних анотацій. Вона дає змогу користувачам додавати власні елементи до Wiki-розмітки. Wiki-посилання супроводжуються семантичними анотаціями мовою OWL DL у форматі OWL/RDF. Розмітка здійснюється за допомогою Web-інтерфейсу. Кожна Wiki-стаття відповідає одному онтологічному елементу, а кожна анотація у статті містить твердження тільки про один елемент. Таке

обмеження має ключове значення для експлуатації, оскільки знання використовуються повторно, й кінцеві споживачі повинні знати, звідки ця інформація була отримана. Таким чином, SMW створює додатковий семантичний шар, що дає можливість системі виглядати як Wiki, а функціонувати як Linked Data.

Основні онтологічні поняття:

- *категорія* – анотація, що дає змогу користувачам класифікувати Web-сторінки;
- *відношення* – зв'язок між статтями за допомогою імен посилання;
- *атрибут* – зв'язок між статтею і сутністю (дата, кількість тощо).

Для реалізації атрибутів є кілька типів даних: ціле число, дата, температура, географічні координати, електронна пошта тощо.

Система подання даних SMW дає змогу позбутися як деяких проблем недосконалості текстового пошуку, так і порушення цілісності даних у традиційних Wiki-системах. Wiki-статті містять багато інформації, однак найчастіше користувач бажає одержати конкретну відповідь на поставлене питання. Для розв'язання цієї проблеми у MW були введені статті-списки. Вони складаються з посилань на інші статті, об'єднані однією тематикою. Однак і цей підхід має багато недоліків, оскільки користувач під час створення статті повинен власноруч вносити зміни до списків, яких вони стосуються. Аналогічно, змінюючи інформацію в статті зі списку, потрібно вручну правити статтю-список. SMW завдяки механізму динамічного створення сторінок і великої кількості форматів висновку контенту допомагає розв'язати цю проблему.

Одним з головних недоліків SMW є брак убудованої підтримки моделювання правил. Тому вона не дає можливості з явних тверджень Wiki одержувати метадані. Однак використання шаблонів і розширень дає змогу змоделювати деякі найбільш уживані й корисні типи правил у SMW: моделювання логічного виведення в OWL, логічні програми, перевірка на обмеження цілісності. Triple Store Connector дає змогу поєднати Wiki з RDF-сховищем і використовувати SPARQL для запитів. Версія 1. 6 працює з будь-яким RDF-сховищем, що підтримує SPARQL і SPARQL Update.

Дані, створені у SMW, можуть легко передаватися у форматах CSV, JSON і RDF назовні. Це дає можливість Wiki бути джерелом даних для зовнішніх застосувань, виконуючи роль, що, зазвичай, виконують реляційні бази даних.

**IkeWiki** використовує онтології для поліпшення самої системи. У системі реалізоване логічне виведення для підтримки користувача у виконанні завдань. Система пропонує і WYSIWYG-редактор, що

призначений для користувачів, які не мають досвіду роботи з Wiki-редактором (рис. 7. 8).

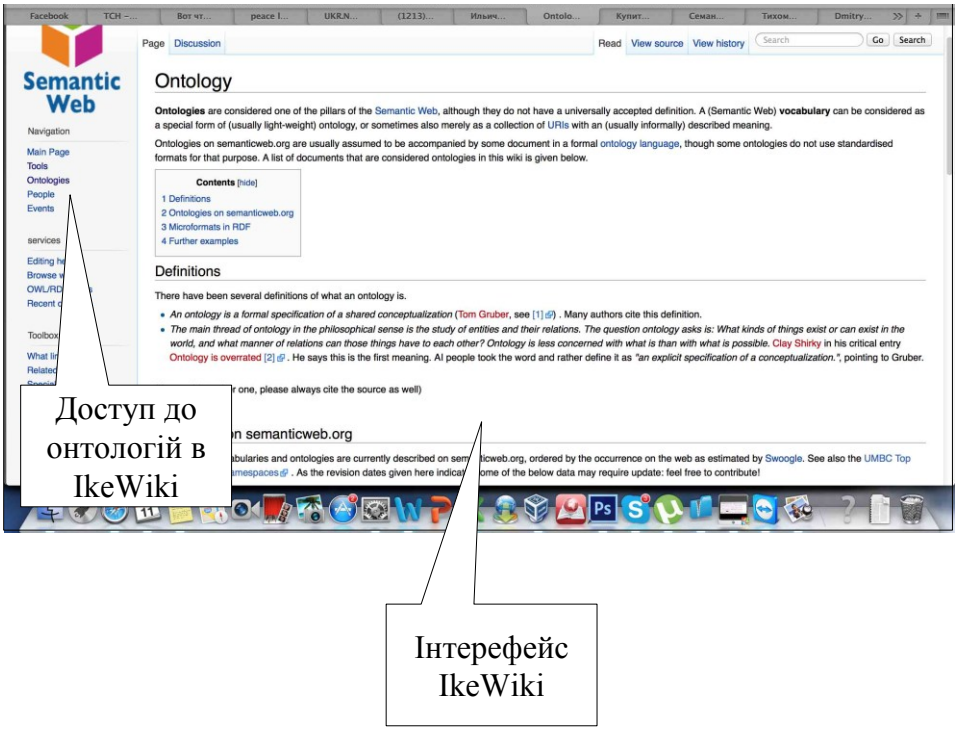



Рис. 7. 8. Інтерфейс IkeWiki

Платформа IkeWiki написана мовою Java. Дані зберігаються у СКБД Postgres, однак є розмежування тексту й структури документів. За необхідності, вони повертаються користувачеві в зручному форматі XML (для тексту) чи RDF (для структури).

База знань у системі презентована RDF фреймворком Jena. Частина RDF-сховища є SPARQL-двигуном, що забезпечує пошук у системі й базі.

Для анотацій є три види редакторів. Редактор метаданих дає змогу заповнювати текстуальні метадані, що стосуються сторінки (дані Dublin Core чи RDF-коментарі). Редактор типу дає змогу асоціювати сторінки з одним чи декількома типами, упровадженими в систему. Редактор посилань забезпечує керування анотаціями посилань. У IkeWiki доступні анотації, що визначаються логічним висновком. Приміром, якщо посилання зі статті «Kyiv» до «Ukraine» іменувати як «capitalOf», то система автоматично асоціює тип «Capital» зі сторінкою, що містить опис Києва, і цей тип не може бути вилучений користувачем.

IkeWiki більше не розвивається як самостійна система. Її продовженням стала система KiWi, що успадкувала більшість характеристик. Можна зробити KiWi-систему частиною Linked Open Data, що, так само, уможливлює інтеграцію контенту з іншими сервісами Semantic Web.

Серед особливостей системи KiWi слід назвати сторінку користувача Dashboard. Вона дає змогу відстежувати й застосовувати

в роботі: потік діяльності (приміром, відновлення елементів контенту), рекомендації (пропонує користувачеві інший контент за допомогою різних рекомендаційних алгоритмів), історію (список елементів, які переглядав чи редагував користувач), теги (список тегів, що використовувалися користувачем). Крім цього, Dashboard – це і профіль користувача, і список його друзів.

Система **OntoWiki** призначена для розробки баз знань і спирається на подання даних у форматі RDF. Для машинної обробки система підтримує різні RDF-формати, RDFa, Linked Data і SPARQL-інтерфейси. Знання в системі презентовані за допомогою «інформаційної карти», збагаченої зручними інтерфейсами для візуалізації й редагування контенту (WYSIWYG редактор для RDF, контроль версій, статистика, підтримка співтовариства тощо). Кожен вузол, представлений сторінкою системи, в інформаційній карті зв'язаний з відповідним цифровим джерелом (рис. 7. 9).

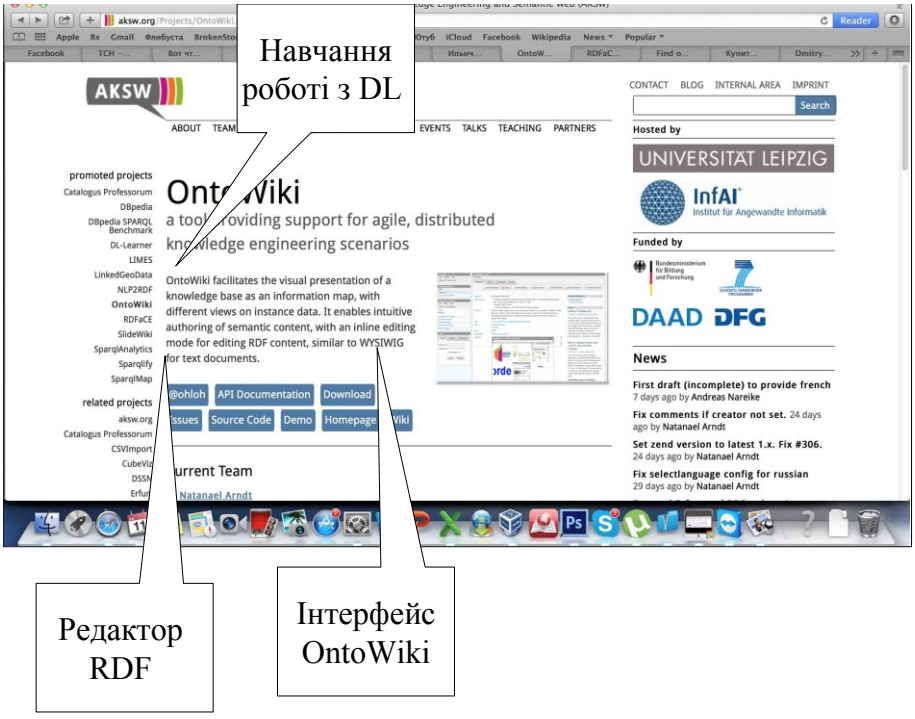



Рис. 7. 9. Інтерфейс OntoWiki

OntoWiki спроектована для роботи з онтологіями будь-якого обсягу й намагається підтримувати створення онтологій «з нуля». Ця платформа не тільки дає змогу застосовувати частково визначені шаблони з репозитарію шаблонів, а й створювати власні. Вона підтримує колаборативну розробку шляхом відстеження змін, а також завдяки можливості коментувати й обговорювати кожну частину бази знань, оцінюючи й обмежуючи кількість контенту й заохочуючи користувальницьку активність.

Семантичний пошук системи презентований як пошук у локальному RDF-сховищі за допомогою SPARQL-запитів.

Для навігації пропонується кілька способів: таксономічний та ієрархічний пошук, фасетний пошук і текстовий пошук.

Система **Freebase** дає змогу користувачам визначати власні схеми для моделювання різних типів даних і керувати зв'язаною структурованою інформацією. Це велика колаборативна база знань (рис. 7. 10).

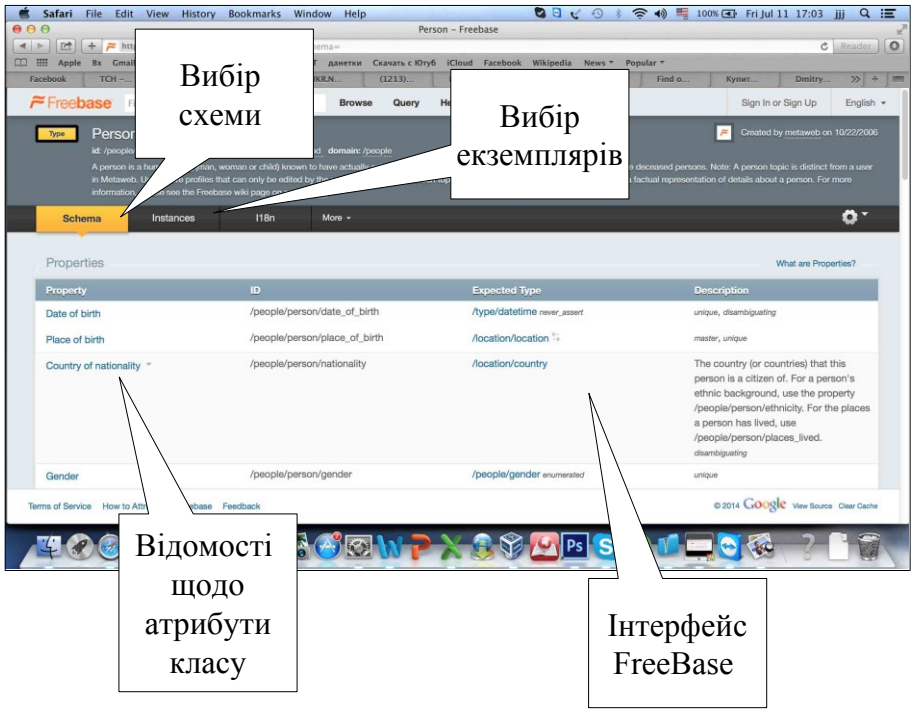



Рис. 7. 10. Інтерфейс FreeBase

Платформа намагається інтегрувати різні підходи до формування бази знань. Крім Wiki-стилю, вона дає змогу збирати інформацію автоматично з різних ресурсів, таких, як Wikipedia і MusicBrainz. Деякі сутності можуть бути змодельовані й додані до системи людьми. Однак у ній не просто створювати власні типи. Всі атрибути повинні мати типи й межі, визначені самою системою. Web-інтерфейс надає користувачам зручну можливість шукати, приймати, редагувати й організовувати інформацію. Недоліком платформи є складність приєднання до зовнішнього ресурсу.

**Висновки**

Використання та розвиток семантичних Wiki-ресурсів дозволяє одночасно вирішувати дві задачі: з одного боку, Wiki-ресурси, що містять семантичну розмітку, стають джерелом структурованих знать щодо предметної області для побудови онтологій, а з іншого – ці ресурси самі використовують такі онтології в якості основи для семантичної розмітки. Їх застосування тісно пов'язані з різноманітними засобами здобуття й подання знань, а також з технологіями Data Mining та онтологічним аналізом.

# РОЗДІЛ VIII.
# ЗАСТОСУВАННЯ ОНТОЛОГІЧНОГО АНАЛІЗУ ДЛЯ ДОСЛІДЖЕННЯ КОМПЕТЕНЦІЙ В ОСВІТІ Й НАУЦІ

Комп'ютерні технології значно змінюють практику освіти та науки. Підтримка індивідуального й дистанційного навчання на основі Web та інтелектуального аналізу інформації мають значний потенціал для революційних удосконалень освітньої та дослідницької діяльності [137].

## 8. 1. Використання онтологій для контролю навичок студентів у мультиагентних системах е-навчання

Концепція е-навчання принципово відрізняється від традиційної системи навчання. *E-learning* – навчання, що базується на використанні обчислювальної техніки й відповідного програмного забезпечення. Навчальний процес стає більш індивідуалізованим, слухачі одержують інформацію з тією швидкістю, що забезпечує ефективне засвоєння матеріалу. Між студентами й викладачем у середовищі е-навчання безпосередніх контактів немає, але вони мають можливість спілкуватися за допомогою засобів телекомунікацій [289].

Через те, що процес навчання безпосередньо пов'язаний з обміном знаннями між викладачем і студентом, потрібно використовувати формалізоване інтероперабельне подання знань. При цьому під знаннями розуміють сукупність відомостей, фактів, понять, подань про що-небудь, накопичених унаслідок навчання, досвіду, у процесі діяльності, а також різних видів залежностей між ними.

Тому в навчанні доцільно застосовувати методи подання, обробки й аналізу знань, запозичені зі штучного інтелекту й керування знаннями [141].

Саме для цього потрібно використовувати онтології, що містять семантичну інформацію про предметну область навчання [281, 292]. Онтологічні системи слугують гнучкою платформою для керування знаннями. Це дає змогу забезпечити повторне використання атомарних одиниць навчання й фіксувати їх характеристики в загальноприйнятих, формалізованих описах метаданих, що уможливлює автоматизоване анотування курсів відповідно до загальновизнаних стандартів (що має полегшити студентам відбір необхідних курсів).

Одним з ключових питань процесу розробки курсу є ідентифікація абстрактного інформаційного домену, у межах якого буде цей курс [289]. Онтологія ПрО відіграє центральну роль як ресурс, що структурує контент навчання. Викладачу необхідно подати опис

основних термінів, із яких конструюється курс, і визначити відношення між ними.

**Зворотний зв'язок у системах е-навчання.** Одним з важливих атрибутів ефективного навчання є наявність зворотного зв'язку між студентами й викладачем.

Зворотний зв'язок використовується в багатьох навчальних парадигмах. Поняття зворотного зв'язку дуже важливо в освітній психології. Це один з її основних психологічних принципів. Інформація про перевірку результатів навчання необхідна для того, щоб оцінити його ефективність, виправити помилки й поліпшити роботу системи. Зворотний зв'язок відображає будь-які комунікаційні процедури, призначені для того, щоб інформувати студента про правильність його відповідей. Інформація, що здобувається студентом зі зворотного зв'язку, містить не тільки виправлені відповіді, а й іншу інформацію, наприклад, оцінку точності та вчасності відповіді, інструкції щодо подальшої роботи, мотиваційні повідомлення, додатковий матеріал, обговорення послідовності навчання й напрям навчання. Управління навчанням потребує засобів порівняння БЗ, яка сформувалася в процесі навчання в того, хто навчається, з БЗ предметної області курсу. Для цього необхідні потужні й інтероперабельні інструменти подання й аналізу знань.

За традиційного навчання студенти й викладач можуть взаємодіяти безпосередньо: студенти мають змогу ставити питання викладачу, а викладач за поведінкою студентів здатен визначити, чи розуміють студенти термінологію, методи розв'язання проблем і зв'язок між ними. Очевидно, що в е-навчанні, де студент значну частину часу працює самостійно під керівництвом навчальної програми, викладач слабше контролює процес навчання, ніж у традиційному навчанні, де саме він визначає контент і послідовність досліджуваних модулів.

У системах е-навчання проблема зворотного зв'язку є набагато складнішою й має різні технологічні й соціальні аспекти. У наявних системах е-навчання зворотний зв'язок реалізується, як правило, лише у вигляді запитань і відповідей. Тому потрібно впроваджувати сучасні інтелектуальні методи (технології розпізнавання облич і мовлення, персональні агенти студентів і викладачів, інтелектуальний аналіз і узагальнення передісторії взаємодії) для підтримки зворотного зв'язку засобами інформаційно-комунікаційних технологій (ІКТ).

Значна частина матеріалів, що використовуються в навчальному процесі з метою контролю за його результатами, є повнотекстовими документами (наприклад, контрольні роботи, звіти, реферати), які сьогодні подаються в електронній формі. Але їх автоматизована обробка потребує розв'язання складних задач розпізнавання семантики,

оскільки знання, відображені в таких документах, не структуровані й подаються природною мовою.

*Онтологічна модель ПрО*, яку будує викладач, з одного боку, є засобом інтероперабельного подання його знань, а з іншого – забезпечує об'єктивний автоматизований контроль за навичками студентів, яких вони набули на семантичному рівні в процесі навчання. Для цього потрібно порівняти онтологічну модель ПрО, створену викладачем, з онтологічною моделлю, яку будує студент.

Специфіка запропонованого нами підходу полягає в тому, щоб студент самостійно побудував онтологію ПрО досліджуваної дисципліни (користуючись фіксованим набором понять і відношень, що йому пропонуються), а потім порівняти її з еталонною онтологією, сформованою викладачем у процесі розробки електронного курсу [289].

Для розробки онтології домену досліджуваної дисципліни студентам необхідно, згідно з методологією IDEF5 [211], виконати такі дії (рис. 8. 1):

- визначити основні класи й терміни домену, навести опис їх значень;
- сконструювати таксономію термінів домену;
- визначити синонімічні та інші види відношень між цими термінами;
- здійснити опис зразків сконструйованих класів.

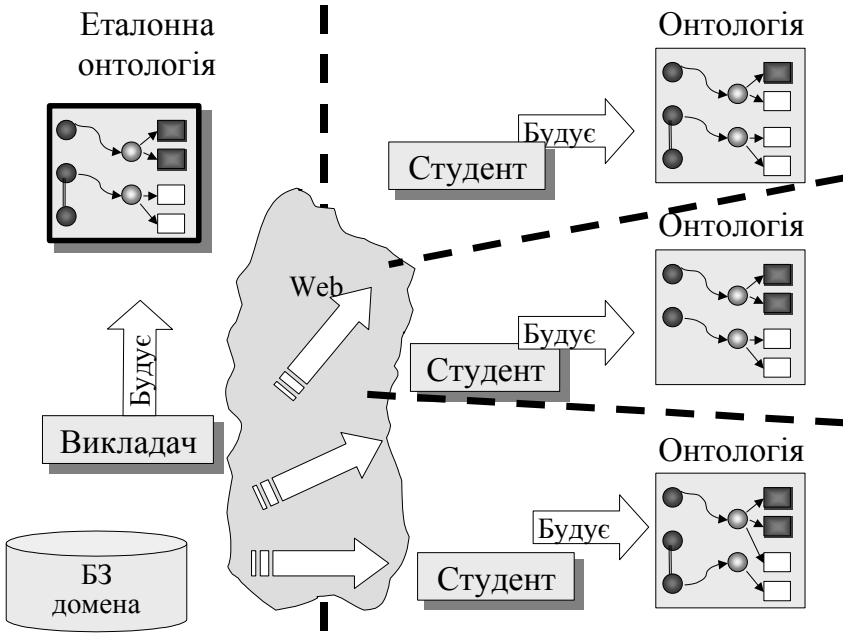



Рис. 8. 1. Побудова онтології домену як результат навчання

Коли студент формує онтологію домену, то відповідна програма порівнює її з еталонною онтологією, сконструйованою викладачем. Використовується оригінальний алгоритм для автоматичного порівняння онтологій, що забезпечує відповідність ієрархічних рівнів

у таксономії термінів (якщо клас А є підкласом В в еталонній таксономії й В – підклас А в таксономії, розробленій студентом, то це є помилкою – рис. 8. 2) і керує належністю екземплярів до класів (якщо екземпляр *а* належить до класу А в еталонній таксономії, а студент визначає *а* як екземпляр класу В, тоді це є помилкою – рис. 8. 3).

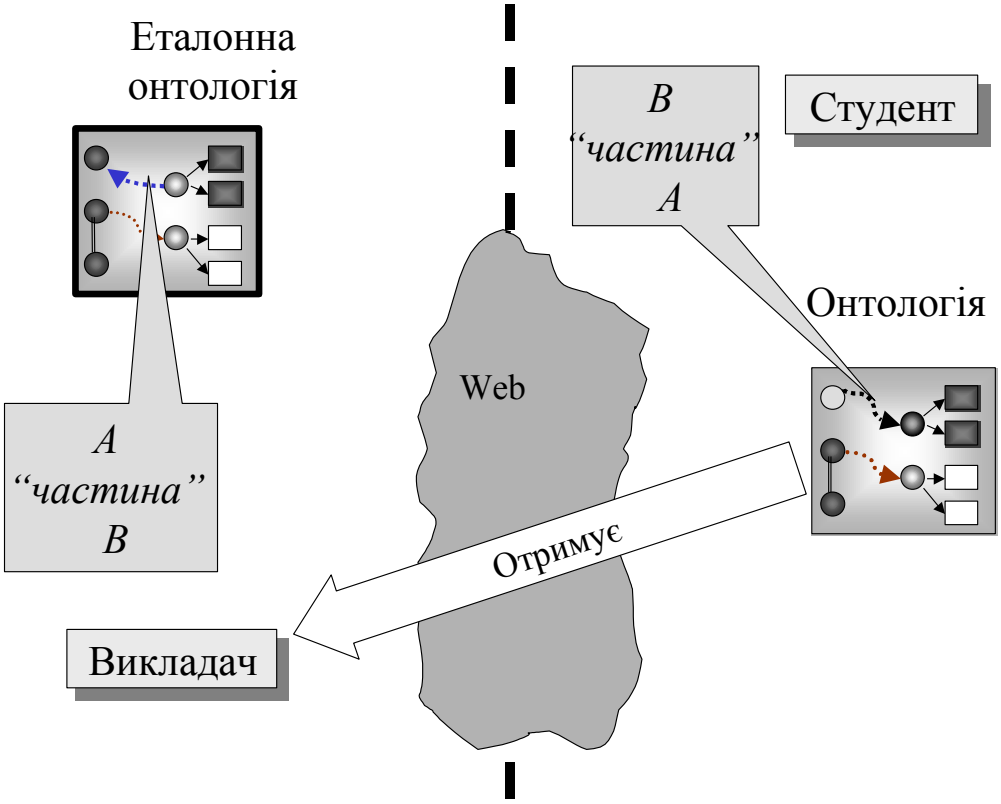



Рис. 8. 2. Помилка в напрямі ієрархії класів

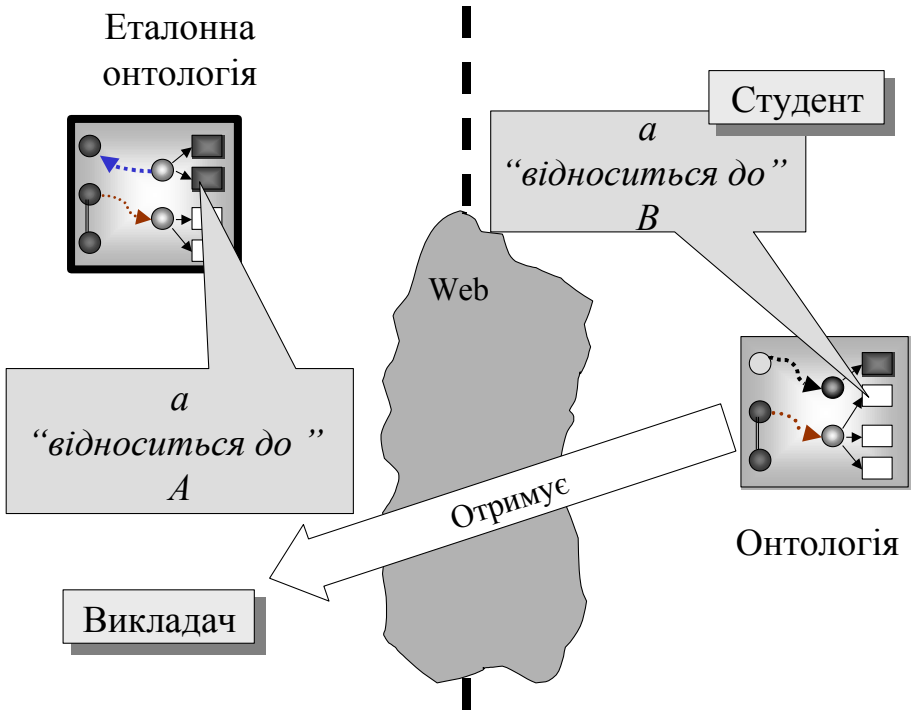



Рис. 8. 3. Помилка класифікації екземпляра

Ми розрізняємо помилки за їх важливістю для розуміння студентом матеріалів курсу (нерозуміння базових принципів призводить до найсуттєвіших помилок, а неправильна інтерпретація окремих елементів – до менш значущих помилок). Якщо студент використовує невідповідне відношення між класами, але з групи ієрархічних відношень (наприклад, вказує, що *«А є частиною В»* замість того, щоб вказати, що *«А є підкласом В»* – рис. 8. 4), то це краще, ніж він би використав синонімічне відношення замість ієрархічного (наприклад, «*А є частиною В»* замість «*А і В – синоніми»* – рис. 8. 5). Суттєвою помилкою є неправильний напрям ієрархічних відношень (рис. 8. 2), оскільки це свідчить про більш глибоке нерозуміння студентом структури домену.

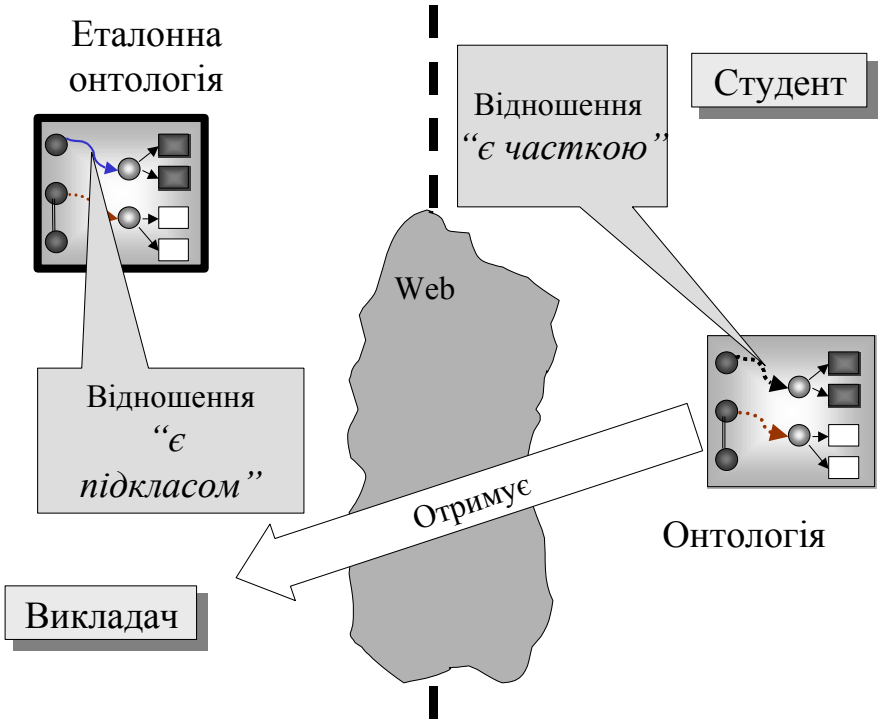



Рис. 8. 4. Помилка в імені відношення між термінами онтології

На базі цього алгоритму ми впорядковуємо оцінки результатів роботи студентів за 100-бальною системою (це доцільно у зв'язку з переходом до кредитно-модульної системи оцінювання знань студентів у рамках процесу зближення й «гармонізації» систем освіти країн Європи, відомого як Болонський процес).

На рис. 8. 6 зображено інтерфейс системи, що порівнює розроблені студентами онтології з еталонною й класифікує знайдені помилки.

Основними перевагами запропонованого підходу до контролю за знаннями, які студенти отримують у процесі е-навчання, є:

- результати тестування знань студентів аналізуються автоматично, без допомоги викладача;

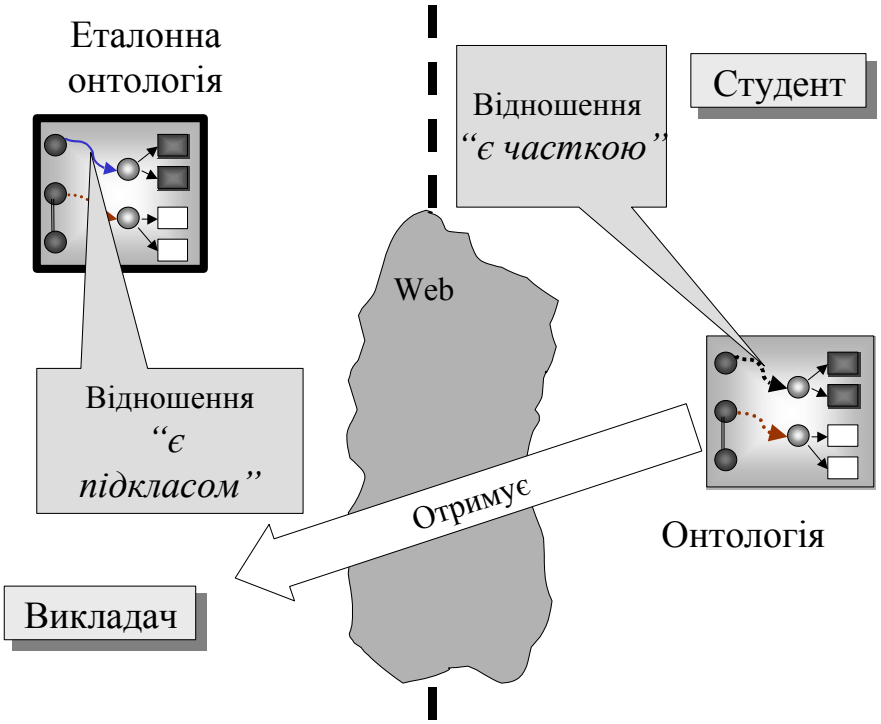



Рис. 8. 5. Помилка в ієрархії термінів онтології

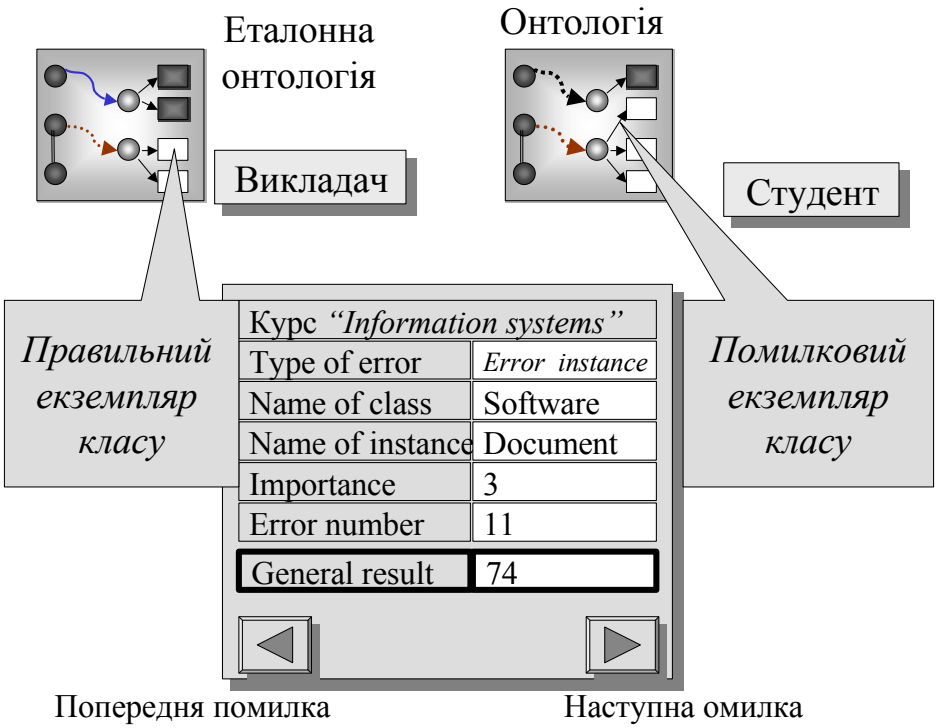



Рис. 8. 6. Оцінка коректності розроблених студентами онтологій домену

- персональні ПА дають змогу виявляти типові помилки, яких припускаються студенти, й на підставі цього вдосконалювати курс е-навчання;
- результати аналізуються об’єктивно;

- наявність онтологічного подання знань курсу е-навчання в інтероперабельній формі спрощує пошук придатного курсу;
- структуризація знань домену спрощує процес вивчення;
- викладачі можуть обмінюватися своїми знаннями на основі еталонних онтологій доменів курсів;
- подання знань курсів е-навчання у вигляді онтології забезпечує їх інтероперабельність і спрощує перехід до кредитно-модульної системи навчання в рамках Болонського процесу;
- використання стандартів і технологій Semantic Web забезпечує пертинентний пошук Web-сервісів, що відповідають курсам е-навчання, іншими застосуваннями.

Надалі ми плануємо розвивати більш потужні алгоритми аналізу онтології, що передбачають інтеграцію онтологій різних доменів і їх розподілене відновлення на основі мультиагентних технологій. Використання персональних ПА студентів і викладачів забезпечує персоналізацію розподіленого процесу навчання. Ці агенти можуть використовувати передісторію навчання для зворотного зв'язку між студентом і навчальною програмою.

## 8. 2. Онтології як база знань персональних агентів у МАС е-навчання

Онтологічне подання знань про домен може бути автоматично оброблено інтелектуальними програмними агентами [273], що представляють інтереси студентів у МАС, призначеній для е-навчання [282].

Мета таких персональних ПА в е-навчанні полягає в тому, щоб кожен студент міг знайти необхідний йому курс (з одного боку такий, що відповідає його інформаційним потребам, а з іншого – що базується на раніше вивчених цим студентом курсах і поданий йому в найбільш зручній для засвоєння формі). Наприклад, студенти різних спеціальностей одного навчального закладу, звичайно, вчаться за різними програмами й часто мають різну теоретичну та практичну підготовку. Їхні персональні агенти можуть ураховувати це й пропонувати їм не тільки універсальну програму курсу, а й додаткові факти й посилання на родинні курси, яких вони не вивчали. Порівняння матеріалів дисциплін з наявними у студента знаннями відбувається із застосуванням аналізу компетенцій [66, 287] і шляхом зіставлення онтологій дисциплін.

Застосування агентно-орієнтованих технологій в е-навчанні дає змогу враховувати в процесі навчання персоніфіковану інформацію про студентів і викладачів і позбавляє всіх користувачів від рутинних

операцій (наприклад, один раз визначивши тип інтерфейсу чи найбільш зручну форму контролю, студент завжди буде одержувати за замовчуванням курс саме в такому вигляді).

З погляду сучасних ІКТ, персональний ПА студента звертається до інших ПА, які підтримують пошук відповідних Web-сервісів, знаходить Web-сервіс, що пов'язаний з потрібним курсом е-навчання [39] і відповідає визначеним вимогам, і встановлює інтерфейс студента з цим курсом. Такий курс може формуватися з окремих модулів, презентованих через різні Web-сервіси.

Для реалізації цих принципів розроблено прототип МАС для е-навчання, що базується на онтології й автоматично здійснює семантичний контроль за уявленнями й навичками студентів про домен досліджуваного курсу [287] – система M(e)L. Онтології використовуються в цій системі для того, щоб здійснити опис досліджуваних матеріалів і подати знання студентів про домен, отримані в процесі е-навчання. Архітектура цієї МАС наведена на рис. 8. 7.

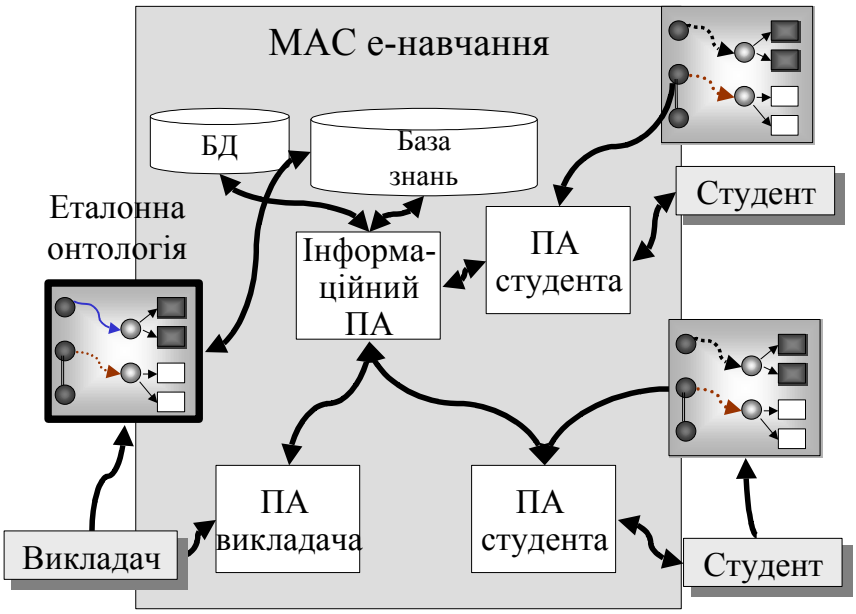



Рис. 8. 7. Архітектура мультиагентної системи е-навчання

Ця МАС охоплює персональних ПА студентів і викладачів, а також агентів-посередників. Використання агентів-посередників підвищує ефективність цієї системи й допомагає користувачам у пошуках необхідної інформації [45, 286]. Агенти студентів і викладачів не взаємодіють безпосередньо, а за необхідності обміну інформацією звертаються до посередників. Наприклад, персональні ПА студентів надсилають онтологічну інформацію про курс

інформаційному агенту, що аналізує її й повертає результати студентам і викладачам.

Еталонна онтологія заноситься до бази знань МАС персональним ПА викладача, який розробляє відповідний курс е-навчання. Коли студент формує онтологію домену у форматі OWL, його персональний ПА з'єднується не безпосередньо з ПА викладача курсу, а з посередником – інформаційним ПА, який надсилає цю онтологію для порівняння з еталонною (її останньою версією). Після того, як інформаційний агент здійснить порівняння, він надсилає його результати персональним ПА студента й викладача [289].

Теоретичні засади, на яких базується наведена вище система, більш детально проаналізовані в [281].

Якщо студент надає перевагу певному режиму подання інформації (конфігуруванню системи), то його персональний ПА, зазвичай, забезпечує всі ці вимоги для нового курсу без безпосередньої команди. Студент одержує інформацію в зручній для нього формі й з урахуванням передісторії його навчання. Наприклад, якщо студент регулярно робить однотипні помилки в різних онтологіях, то він одержує повідомлення про це й поради, пов'язані з відповідними матеріалами курсу.

Надалі ми плануємо використовувати методи Data Mining [141] та індуктивного виведення [143, 156] для формування найбільш придатних персональних стратегій навчання для кожного студента шляхом узагальнення передісторії взаємодії студентів з системою е-навчання (наприклад, деякі студенти надають перевагу теоретичним матеріалам, а інші – навчанню на прикладах практичних задачах або графічному чи текстовому поданню інформації тощо).

Ще одна важлива перевага використання мультиагентних технологій в е-навчанні пов'язана із застосуванням персональних ПА викладача. Якщо багато студентів у результатах тестування припускається однотипних помилок, то персональний ПА викладача одержує повідомлення про це, і викладач може на підставі цієї інформації внести доповнення чи уточнення до навчальної програми.

Наведена вище МАС використовувалася під час контролю знань студентів з таких курсів: «Системне програмування у Visual C++» (Європейський університет, Київ, www.eufimb.edu.ua) і «Сучасні Інтернет-технології» (Київський славістичний університет, www.mgi_ksu.edu.ua). Створені детальні моделі ПрО цих курсів. Модель курсу «Системне програмування у Visual C++» містить понад 150 термінів і використовує 8 типів відношень між ними. Курс супроводжується 16 online-лекціями й 10 практичними вправами. Модель курсу «Сучасні Інтернет-технології» містить близько

70 термінів і використовує 10 типів відношень між ними. Курс супроводжується 18 лекціями й 6 практичними вправами. Онтології, побудовані студентами, оцінюються за формулою: $K = (K_{term} * m_{term} + K_{rel} * m_{rel} + K_{type} * m_{type}) / (m_{term} + m_{rel} + m_{type})$, значення коефіцієнтів у якій визначаються викладачем залежно від специфіки предметної області. Приклади оцінки онтологій, складених студентами під час вивчення двох предметів – «Системне програмування у Visual C++» і «Сучасні Інтернет-технології», наведені в таблиці 8. 1.

Таблиця 8. 1.

Параметри онтології домену, сформовані студентами

| Назва курсу | «Системне програмування у Visual C++» | «Сучасні Інтернет-технології» |
|---|---|---|
| Кількість студентів, що вивчають цей курс | 22 | 16 |
| Кількість термінів в онтології | 153 | 68 |
| Кількість відношень між термінами в онтології | 8 | 10 |
| Кількість термінів, коректно використаних у розроблених студентами онтологіях $K_{term}$ | 94.2% | 91.6% |
| Кількість відношень між термінами, коректно використаних у розроблених студентами онтологіях $K_{rel}$ | 72.0% | 66,3% |
| Кількість відношень між термінами, тип яких коректно використаний у розроблених студентами онтологіях $K_{type}$ | 89.1% | 81.5% |
| Вага $m_{term}$ | 0.9 | 0.7 |
| Вага $m_{rel}$ | 0.3 | 0.5 |
| Вага $m_{type}$ | 0.7 | 0.8 |
| Загальна оцінка коректності онтології домену, K | 88,51% | 80,14% |

## 8. 3. Використання онтологій для аналізу компетенцій експертів у наукових дослідженнях

### 8. 3. 1. Проблема пошуку експертів у нових ПрО

Один з важливих напрямів розвитку інтелектуальних ІС пов’язаний з підтримкою ухвалення управлінських рішень з метою забезпечувати *осіб, що ухвалюють рішення* (ОУР), знаннями відповідної

ПрО і засобами їх аналізу. Досить поширеним на сьогодні є таке завдання: ОПР потрібно розв'язати задачу в новій для неї ПрО, а для цього – знайти спеціалістів (експертів), яким можна доручити розв'язання проблеми. Для цього ОПР потрібно оцінити їх компетентність саме в цій галузі. Прикладами подібних задач є експертиза проектів, рецензування наукових статей, оцінка об'єктів.

Актуальність таких задач зростає в сучасному суспільстві завдяки збільшенню обсягу інформації, що обробляється, ускладненню її структури й унаслідок цього постійного виникнення нових технологій виробництва, наукових напрямів, складних інформаційно-ємних товарів, об'єктів і послуг тощо. Через це в багатьох сферах, що потребують експертних рішень, немає офіційно визнаних експертів, кваліфікація яких підтверджена відповідними документами. Загалом *експертом* називають особу, що має спеціальні знання стосовно проблем, які безпосередньо пов'язані з певною ПрО [61]. Якщо нова ПрО тільки формується, то в ній можуть працювати спеціалісти з суміжних областей, але невідомо, хто саме з таких спеціалістів працює в цій ПрО, розширивши свою кваліфікацію в напрямі інших суміжних областей, від яких походить ця ПрО. Якщо важко знайти спеціалістів, сфера компетенції яких повністю покриває межі ПрО, для якої здійснюється експертиза, то потрібно підібрати групу експертів, об'єднання сфер компетенції яких перекриває межі ПрО. При цьому бажано знаходити таких спеціалістів, сфери компетенції яких хоча б частково перетинаються: це дає змогу забезпечити більш ефективну взаємодію між ними (спільний понятєвий апарат, загальновизнані методи формулювання висновків, зрозумілі методики отримання оцінок тощо). Пошук експертів має базуватися на стратегії евристичного пошуку [131], тобто, крім визначення самого завдання, необхідно використовувати знання, які належать до конкретної ПрО. Приміром, у деяких ПрО свідченням компетентності експерта є наявність спеціального сертифіката, в інших – наукове звання або звіти про виконані роботи тощо.

Сьогодні, формуючи команди експертів і добираючи рецензентів, організатори експертизи в основному орієнтуються на власний досвід роботи з певними спеціалістами, на враження від їхньої роботи й на переконання інших спеціалістів у цій ПрО. Такий підхід на практиці довів свою ефективність, але виникає багато динамічних нових галузей, що розвиваються на перетині кількох дисциплін, для яких ще не сформовані співтовариства авторитетних експертів (приміром, е-навчання, економіка знань, образний комп'ютер), класифікатори не відповідають сучасному стану знань про домен і тому важко визначити, хто саме зі спеціалістів має розв'язувати конкретну

проблему. Важко не стільки визначити рейтинг конкретного експерта, скільки сформувати початкову множину спеціалістів, серед яких є ймовірність знайти спеціалістів, кваліфікація яких дає змогу ефективно розв'язувати поставлену задачу. Для того, щоб дізнатися думку спеціаліста щодо інших спеціалістів, потрібно якось знайти першого спеціаліста, з якого можна почати опитування. Крім того, є ймовірність, що спеціаліст, недостатньо компетентний у ПрО поставленої задачі, порекомендує як експертів ще менш компетентних осіб: навмисно, щоб підвищити власний рейтинг, або через неправильне розуміння поставленої задачі.

У роботі [93] запропоновано метод автоматичного визначення компетентності наукових співробітників шляхом аналізу змісту їхніх публікацій. Метод базується на спеціальній онтології галузі наукового знання, ядром якої є ієрархія тем наукової галузі, пов'язаних таксономічним відношенням «тема – підтема». Кожній темі відповідає набір термінів, що її характеризують.

Релевантність документа (публікації) темі обчислюється за п'ятибальною шкалою, виходячи з трьох параметрів:

- загальна кількість згадувань основних термінів теми в документі;
- кількість фрагментів документа (наприклад, абзаців), у яких траплялися основні терміни теми;
- розмаїтість основних термінів теми (кількість різних термінів) у документі.

Профіль документа визначається як вектор його релевантності кожній з тем онтології. Профіль компетентності співробітника визначається як профіль групи його публікацій.

Компетенція може бути схарактеризована як сукупність таких ознак:

- найменування компетенції;
- категорія чи сфера, до якої належить компетенція;
- опис компетенції, який пояснює, що може робити власник компетенції;
- свідчення компетенції, за наявністю яких можна визначити, чи володіє співробітник цією компетенцією.

Слід розрізняти *компетенцію* та *компетентність*. Компетенція – це поняття, загалом не пов'язане з конкретною особою; компетентність – це відношення між особою і компетенцією, яке означає, що певна людина володіє цією компетенцією. Відповідно, до поняття компетенції належать її найменування, категорія та опис, а свідчення компетенції має підтверджувати наявність відношення компетентності.

Є багато робіт з автоматичного визначення компетентностей на основі документів [335, 476].

Усі вони використовують текстовий пошук для знаходження в документі свідчень щодо компетенції його автора, якими є слова й словосполучення. Здебільшого описом компетенції є запит користувача. Свідченням володіння компетенцією є згадування слів запиту в текстах документів, автором яких був користувач. При цьому можуть використовуватися методи розширення запитів, наприклад, уведення синонімів або інших пов'язаних термінів з онтології відповідної ПрО. Фактично цей підхід являє собою визначення релевантності документа запиту.

Джерелами свідчень компетентності для науковців можна вважати публікації в журналах, що рецензуються. Вимога рецензування є істотною, оскільки пропонований підхід не ставить за мету оцінити якість публікацій (коректність, новизна, актуальність, стиль тощо). Тому передбачається, що якість уже позитивно оцінена рецензентами роботи.

Аналогічно можна визначати компетентність викладачів – за допомогою аналізу текстів підручників, навчальних посібників та інших методичних матеріалів, авторами яких вони є (або хоча б використовують у навчальному процесі). Також цей підхід можна застосовувати до визначення компетентності навчальних закладів або наукових установ – як сукупність розробок їхніх співробітників.

Релевантність документа темі характеризується використанням основних термінів цієї теми; компетентність співробітника в темі – наявністю публікацій, релевантних темі.

Розглянемо *окремий випадок* задачі пошуку експертів: ОПР має *вперше* організувати експертизу в певній ПрО. В організації подальших експертиз у цій ПрО ОПР може використовувати накопичений раніше досвід, власні знання про потенційних експертів, результати й умови виконання попередніх експертиз, проте якщо експертиза проводиться вперше, тоді ОПР доступні тільки формальні відомості стосовно ПрО та потенційних експертів. Якщо ОПР звертається за порадами й консультаціями до інших осіб і організацій, то це зводить задачу до попередньої – у кожному співтоваристві хтось має організувати експертизу в певній ПрО вперше. Дотримуються таких умов:

Особа, що має ухвалювати рішення щодо пошуку експертів, сама не є експертом у цій ПрО (ОПР, що вважає себе компетентною у ПрО задачі, може оцінювати проекти самостійно або знаходити експертів на основі власного досвіду, а не тільки формальних відомостей).

ОПР має доступ до відомостей щодо потенційних експертів, які характеризують їх спеціалізацію, кваліфікацію і здатність до здійснення

експертизи (це можуть бути відомості щодо їх місця роботи, посади, проектів, які вони виконують, опублікованих ними наукових праць, наявності дипломів і сертифікатів тощо), тобто завдання зводиться до такого: за наявними відомостями серед тих осіб, що відомі ОПР і відповідають певним формальним вимогам, знайти тих, сфера компетенції яких релевантна експертній задачі, тобто спеціальні знання й досвід яких дають змогу найбільш ефективно розв'язати цю задачу.

Для того, щоб підбір групи експертів міг здійснюватися об'єктивно (був формалізований), необхідно, щоб інформація щодо кваліфікації експертів була надана в певній стандартизованій формі (тобто відомості, недоступні ОПР, не враховуються, приміром, якщо немає переліку публікацій особи А, то вважається, що А взагалі не має публікацій).

ОПР має певні відомості стосовно ПрО, до якої належать проекти (це можуть бути звіти про раніше виконані проекти в цій ПрО, паспорт певної спеціальності, державні й міжнародні стандарти й нормативи), але не має відповідної кваліфікації, щоб самостійно виділити в цих інформаційних ресурсах найважливіше.

ОПР має набір формальних вимог щодо осіб, серед яких потрібно обрати експертів (приміром, це можуть бути працівники певної установи або міністерства, громадяни певної держави, особи, що мають науковий ступінь у певній галузі); такі вимоги можуть бути не пов'язані безпосередньо з кваліфікацією осіб у конкретній ПрО (так, доктор технічних наук може бути експертом у розподілених обчисленнях, але не розумітися на автомобілебудуванні).

Інформація щодо експертів і ПрО подається в електронній формі, придатній для автоматизованої обробки.

Для того, щоб особа А була визнана експертом у ПрО В, потрібно формалізувати як K(A) – досвід і сферу компетенції А, так і S(B) – межі ПрО В, а потім визначити відповідність між K(A) і S(B). При цьому важливі як виразні засоби, що застосовуються до формалізації K(A) і S(B), так і ступінь об'єктивності у формуванні S(B). Крім того, під час пошуку відповідностей між ними потрібно враховувати семантику й специфічні властивості ПрО В.

При цьому похідна задача містить у собі декілька підзадач:

– знайти інформаційні ресурси, що містять відомості про осіб, які можуть бути експертами в певній ПрО, і оцінити ступінь достовірності запропонованих відомостей;

– з відомостей про потенційних експертів здобути знання про їх кваліфікації;

– створити формальний опис ПрО експертизи: з інформаційних ресурсів, що стосуються цієї ПрО, здобути відповідні знання –

властивості, за якими можуть оцінюватися потенційні експерти, термінологію ПрО тощо;

– знайти експертів, кваліфікація яких релевантна ПрО експертизи, тобто побудувати критерій, за яким можна кількісно оцінити подібність між кваліфікацією особи й формалізованим описом ПрО експертної задачі.

*Компетентність* – це рівень досягнень (досвіду, знань, навичок) особи в певній ПрО. Компетентність може бути визначена на основі аналізу діяльності фахівця, рівня й широти обізнаності з досягненнями науки й техніки, розуміння досліджуваних проблем, можливих шляхів їхнього розвитку. Для кількісної оцінки рівня компетентності використовується *коефіцієнт компетентності,* з урахуванням якого зважуються висновки експерта.

Загалом коефіцієнт компетентності R є функцією від характеристик знань і досвіду експерта (якісних і кількісних) і від опису експертної задачі: R(A,B)=f(K(A), S(B)). Особа A може бути експертом, якщо для задачі B її коефіцієнт компетентності R(A,B)>p, де p – константа, яка визначає мінімальні вимоги ОПР до експерта. Цей коефіцієнт визначається за різноманітними апріорними й апостеріорними даними, склад і відносна важливість яких визначаються відповідно до специфіки ПрО. У процесі використання апріорних даних оцінка коефіцієнта компетентності виробляється до проведення експертизи на основі самооцінки експерта й взаємної оцінки інших експертів. Під час використання апостеріорних даних оцінка коефіцієнта компетентності виробляється на основі обробки результатів експертизи. Кожен експерт на підставі персональних даних одержує кваліфікаційну категорію, що визначається як функція від його персональних даних. Приміром, у [114] персональні дані кожного експерта відображає трійка (s, r, h), де $s \in S, r \in R, h \in H$, $S = \{s_1, s_2, s_3\}$ – вища освіта експерта ($s_1$ – збігається з профілем пріоритетного напряму, $s_2$ – базова освіта за суміжною спеціальністю, $s_3$ – базова освіта за іншою спеціальністю); $R = \{r_1, r_2, r_3\}$ – наукова підготовка ($r_1$ – академік НАНУ або член-кор., академік галузевої академії, $r_2$ – професор, доктор наук, $r_3$ – кандидат наук, с. н. с., доцент); $H = \{h_1, h_2, h_3\}$ – стаж роботи за цим пріоритетним напрямом ($h_1$ – не менше десяти років, $h_2$ – не менше п’яти років, $h_3$ – не менше року). Якщо оцінки r і h визначити досить просто, то оцінити s для нових ПрО досить складно (більшість експертів отримає оцінку $s_2$, і це не дасть змогу встановити пріоритет щодо їх кваліфікації). Крім того, якщо новий напрям діяльності виник порівняно недавно, то жоден спеціаліст не має великого стажу в цій галузі, а висока оцінка, отримана за рівень наукової підготовки, може мати негативне значення – спеціаліст отримував знання тоді, коли нової ПрО

ще не було, а тому, ймовірно, взагалі не обізнаний з її специфікою. З іншого боку, за великий проміжок часу, присвячений науковій діяльності, спеціаліст міг кілька разів принципово змінювати напрям своїх досліджень, а висока оцінка r свідчить про високий рівень інтелекту, наполегливість у науковій роботі й здатність до отримання цікавих і корисних результатів. Тому, крім трійки (s, r, h), потрібно враховувати відомості про поточні наукові й професійні досягнення спеціаліста, напрям його роботи й працездатність. Для працівників багатьох спеціальностей ці відомості відображаються в їх наукових публікаціях, звітах про виконану роботу, описах розроблених продуктів, підготовлених навчально-методичних матеріалах. Те, наскільки доступні ці відомості (приміром, опубліковані в Інтернеті, презентовані в бібліотеках), визначає певною мірою ступінь авторитетності спеціаліста та його здатність і готовність віддавати власні знання. Це безпосередньо пов'язано зі здатністю спеціаліста бути експертом.

### 8. 3. 2. Web як джерело інформації про потенційних експертів

Значна частина підприємств і установ (насамперед, найбільш успішних, розвинутих і авторитетних) має власні сайти й портали у Web. У мережі містяться відомості про більшість людей, які зайняті певною виробничою діяльністю. У ресурсах Інтернету презентовані проекти й завдання, які виконують люди, наведені результати їхньої роботи.

Щоб вийти за рамки індивідуального обмеженого простору знань і переконань, потрібно мати доступ до знань про інших суб'єктів інформаційного простору ПрО. Крім того, дуже важливо, щоб ці знання були подані в певній стандартизованій формі, придатній для автоматичної обробки. Це зумовлюється надзвичайно великим обсягом інформації стосовно ПрО, що потенційно доступна сьогодні кожній людині за допомогою інформаційно-комунікаційних технологій. Приміром, нереально продивитися особові справи всіх лікарів або інженерів певного міста чи країни, які формально відповідають надзвичайно слабким вимогам, що можна задати апріорно.

Для того, щоб зрозуміти, чи можна використовувати відомості, доступні через Web, для пошуку експертів, потрібно дослідити структуру Web і визначити, чи буде будь-яка інформація, опублікована в мережі, потенційно доступна в разі коректно сформульованого запиту до пошукових систем. Для цього потрібно проаналізувати структуру зв'язків між інформаційними ресурсами Інтернету й досяжність їх через ланцюг гіперпосилань.

Web можна розглядати як інформаційну модель реального світу, що відображає відношення та взаємозв'язки між різними його

компонентами. Web-простір характеризується великою кількістю експертних оцінок, реалізованих через гіперпосилання: автори ресурсів посилаються на ті ресурси Інтернету, які вони вважають важливими, цікавими, авторитетними. Такі гіперпосилання є базою для побудови моделі Web-простору.

Структура Web-простору – це орієнтований граф, що має топологію Bow Tie, у якому вершини відповідають сторінкам, а ребра – гіперпосиланням, які поєднують сторінки. Аналіз структури зв'язків між окремими Web-сторінками, виконаний у рамках цієї моделі, дав змогу виявити такі компоненти [99]:

– *центральне ядро* – Web-сторінки, за гіперпосиланнями кожної з яких можна потрапити на будь-яку іншу;

– «*відправні*» Web-сторінки – сторінки, гіперпосилання з яких ведуть до ядра, але з ядра до них потрапити не можна;

– «*кінцеві*» Web-сторінки – сторінки, до яких можна прийти за гіперпосиланнями з ядра, але не можна повернутися назад до ядра;

– «*відростки*» – цілком ізольовані від центрального ядра сторінки: або «*миси*», пов'язані гіперпосиланнями зі сторінками будь-якої іншої категорії, або «*перешийки*», які з'єднують дві Web-сторінки, що не входять до ядра. Крім цього, у Web є й «*острови*», що не перетинаються з іншими ресурсами Інтернету. Єдиний спосіб отримати доступ до ресурсів цієї групи – знати їхні адреси. Пошукові машини не здатні знаходити ці острови. Пропорції цих категорій сторінок залишаються незмінними в часі.

Топологія і характеристики моделі виявилися приблизно однаковими для різних підмножин Web-простору, підтверджуючи тим самим твердження про те, що Web є фракталом, тобто властивості структури всього Web-простору правильні й для його окремих підмножин. Тому можна припустити, що й у реальному світі є «*острови*» – співдружності інформаційно пов'язаних людей і організацій, що практично не мають відомостей про інші аналогічні співдружності. Причинами цього можуть бути мовна й географічна роз'єднаність, підтримка режиму секретності тощо.

З погляду пошуку експертів більш цікавими є сторінки ядра й кінцеві Web-сторінки: гіперпосилання Інтернету відображають авторитетність певних організацій та їхніх співробітників: якщо на думку якоїсь людини, викладену у Web, посилається багато інших спеціалістів (навіть з негативними оцінками), то можна вважати таку людину авторитетною в певному співтоваристві. Тому, якщо на якусь сторінку є багато гіперпосилань з інших сторінок, то така сторінка може містити відомості про широко визнаних спеціалістів.

Велике значення має множина вузлів, з яких починається пошук. Приміром, результати виконання одного й того ж пошукового запиту до різних ІПС будуть суттєво різнитися як за обсягом, так і за порядком подання знайдених ресурсів, незважаючи на те, що всі ці ІПС намагаються проіндексувати один інформаційний об'єкт – контент Web. Тому в процесі пошуку потрібно надавати перевагу ІПС, що пов'язані з користувачем територіально (наприклад, використовувати ІПС Meta для пошуку спеціалістів в Україні) і за специфікою ПрО.

Інше важливе питання: чи можна довіряти відомостям, поданим у мережі Інтернет про потенційних експертів. Якщо ці відомості не супроводжуються документами, то людина може дезінформувати зацікавлених осіб (на жаль, електронний підпис в Україні ще не реалізований для широкого загалу). Але якщо відомості подаються на офіційному сайті певної організації, то вони мають бути достовірними. Приміром, відомості на сайті організації про кваліфікацію її співробітників забезпечують доступ до достовірної інформації з більшою вірогідністю, ніж дані, наведені на особистій сторінці експерта. Тому серед знайдених ІР доцільно надавати перевагу офіційним сайтам.

Розв'язавши питання про те, у яких ІР можна знайти відомості про потенційних експертів, слід перейти до другого етапу – як формалізувати відомості про експертів і ПрО експертизи так, щоб можна було встановлювати відповідність між ними.

### 8. 3. 3. Організаційні онтології

*Організаційна онтологія* – це онтологія, що відображає знання про організаційну й функціональну структуру певного суб'єкта економічної діяльності, тобто його основні компоненти й зв'язки між ними. Вона містить інформацію про працівників підприємства, ієрархію виробничих відносин між ними; ресурси, що використовуються на підприємстві в процесі виробництва; продукцію, створення якої є наслідком функціонування підприємства, і структурні одиниці підприємства та зв'язки між ними [45].

Хоча наука про організації сьогодні досить розвинена, цілий ряд питань знайшов тільки часткові розв'язання. Наприклад, є багато різних визначень організацій, але так і не знайдено однозначну відповідь на питання про те, що ж таке організація і які вона має властивості, дії, обмеження й поводження. Тому велику увагу вчені приділяють моделюванню роботи організації.

*Організація* – це стійка система відносин між суб'єктами, заснована на сукупності досягнутих ними угод. Будь-яка організація являє собою винятково складний для вивчення об'єкт, повний обсяг

властивостей і параметрів функціонування якого принципово непізнаваний. Дослідник завжди має справу з якоюсь абстракцією, моделлю організації, що відображає окремі аспекти «життя» реального об'єкта. Подання організації як системи відносин є однією з можливих моделей організації. У контексті організації між індивідами формується багато типів відносин – від виробничих до соціально-психологічних. Використовуються три види заснованих на онтологіях моделей організацій, що структурують і організовують інформацію [182]:

- організаційна онтологія;
- онтологія ПрО діяльності організації;
- онтологія користувацької діяльності.

Організаційна онтологія забезпечує семантичну інформацію про структуру організації. Ця складна структура часто використовує декомпозицію в окремих ієрархічних модулях.

*Онтологія ПрО діяльності організації* проектується для того, щоб організувати й структурувати бізнес-функції та дії, що мають місце в певній ПрО. Така доменна онтологія діяльності забезпечує ієрархічну структуру для класифікації записів, що документують бізнес-функції та дії для класифікації й індексації цілей.

*Онтологія користувацької діяльності* пов'язана з діями для пошуку інформації, здійснюваними кінцевими користувачами. З погляду аналізу, що базується на знаннях, потрібно визначити, яким користувачам і що саме потрібно знати про інформаційні об'єкти та дії, що використовуються в задачі пошуку інформації, і як ці знання мають бути організовані. Дії користувачів звичайно можуть бути відображені через виконувані ними дії та інформаційні об'єкти, зв'язані з інформаційними потребами. Розвиток онтології користувацької діяльності починається з формування таксономічної класифікації знань про задачу та інформаційні об'єкти. Знання про задачу містять словник для подання процесу виконання дій, наприклад, пошук, перегляд, збереження. Такі онтології можуть використовуватися:

- для виконання складних інформаційних запитів, пов'язаних з обробкою контенту на семантичному рівні (наприклад, знайти всі проекти, які за певний період часу виконувалися працівниками, що контактували зі співробітниками певного підрозділу й мали певний рівень повноважень, тобто мали доступ до відповідних корпоративних знань);
- для формування групи експертів, що мають достатні знання й досвід роботи, щоб виконувати експертизу в галузі, презентованій у вигляді повнотекстового документа;

- для пошуку співробітника, який відповідає за певне коло питань (незалежно від того, як у цій організації називається його посада і як сформульовані його посадові обов'язки);
- для встановлення відповідності між інформаційними потребами користувача та інформаційними ресурсами, презентованими в ІС.

Наприклад, організаційна онтологія наукової організації (рис. 8. 8) містить такі класи й підкласи, що характеризуються певними атрибутами:

людина (ім'я (STRING), по батькові (STRING), прізвище (STRING), рік народження (INTEGER)):

співробітник (…, посада (STRING), працює в (підрозділ), ідентифікаційний код (INTEGER)):

науковий співробітник (…, науковий ступінь (STRING), працює над темою (тема), науковий стаж (DATE), публікації (публікація));

інженер(…, має кваліфікацію (STRING));

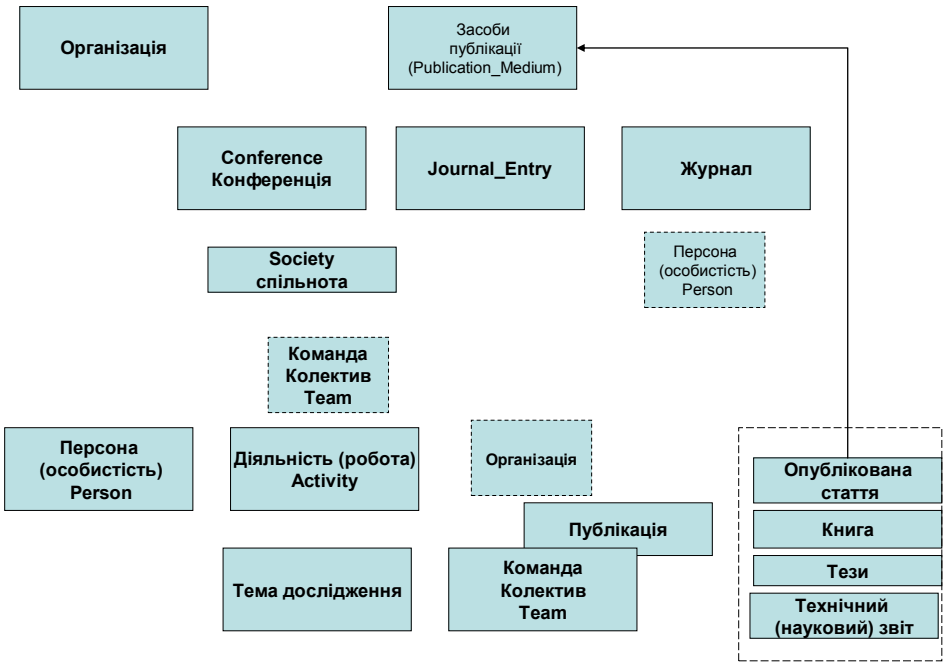



Рис. 8. 8. Фрагмент структури організаційної онтології

аспірант (…, рік вступу (DATE), науковий керівник (науковий співробітник), публікації (публікація));

підрозділ (назва (STRING), керівник (співробітник)):

інститут (…, адреса (STRING));

відділ (…, належить до (інститут));

лабораторія (…, належить до (відділ));

тема (шифр (STRING), назва (STRING), керівник (науковий співробітник), дата початку (DATE), дата завершення (DATE), виконавці (співробітник)):

комплексна тема (…, складається з (тема)) ;

публікація (назва (STRING), автори (людина), рік публікації (STRING), мова (STRING), кількість сторінок (INTEGER)):

наукова стаття (…, УДК (STRING), анотація (STRING), назва видання (STRING));

монографія (…, рецензент (науковий співробітник), назва видавництва (STRING));

тези конференції (назва конференції (STRING), дата проведення (DATE), місце проведення (STRING)).

Будемо вважати, що інформація, яка стосується тієї ПрО, у якій певна особа може вважатися експертом, міститься в назвах і УДК її публікацій і наукових доповідей, у паспорті спеціальності за класифікацією ВАК (якщо особа має науковий ступінь), у назві організації та підрозділу, де вона працює, а найголовніше – у назвах і анотаціях проектів, які ця особа виконувала. Крім того, можна оцінювати рівень кваліфікації особи за загальним і науковим стажем, кількістю публікацій (загальною й за останні роки), наявністю аспірантів тощо. Ще одним важливим джерелом інформації можуть бути відомості про співробітників, у колективі з якими виконувалися проекти або друкувалися статті.

Одним із засобів моделювання ПрО є створення її тезауруса. Можна сформувати тезауруси як окремої людини, так і тезаурус цілої ПрО. Часто терміни «онтологія» і «тезаурус» використовують як синоніми, але в ІТ тезаурус частіше застосовують для опису лексики в проекції на семантику, а онтологію – для моделювання семантики та прагматики в проекції на мову подання. Для такої задачі тезаурус можна розглядати як окремий випадок онтології.

**Область застосування аналізу компетенцій**

Останнім часом, завдяки комплексним дослідженням з психології, менеджменту та теорії навчання, набули поширення такі теорії керування персоналом, як *керування компетенціями* (competence management) і *керування талантами* (talent management).

У загальному значенні *компетенція* – здатність успішно діяти на основі наявних знань і практичного досвіду під час розв’язання задач. Елементи компетенції – знання й навички, життєвий досвід, фізичний потенціал, здібності, риси характеру, інтелект тощо – об’єднані в різні конфігурації для розв’язання людиною конкретних проблем. У керуванні персоналом під компетенцією, зазвичай,

розуміють формальний опис вимог до особистісних, професійних та інших якостей співробітника.

На сьогодні поняття «компетенція» є нечітким і дуже залежить від специфіки конкретної ПрО.

Поняття компетентності передбачає сукупність фізичних, інтелектуальних і психологічних (соціальних) якостей і властивостей людини, необхідних їй для самостійного або спільного з іншими ефективного виходу з різних професійних і життєвих ситуацій.

Часто компетенції об'єднують у чотири групи: 1) можливість розв'язувати проблеми; 2) лідерство; 3) міжособистісна впливовість; 4) особиста й корпоративна ефективність.

Уперше вивчення компетенцій для прогнозування рівня ефективності виконання роботи було запропоновано американським психологом Д. МакКлеландом [5], який доводив, що традиційні тестування здібностей і знань, а також наявність яких-небудь дипломів у потенційних виконавців робіт не забезпечують ні ефективного виконання таких робіт, ні успіхів у професійній діяльності. Подібні висновки змусили Д. МакКлеланда розпочати пошук таких характеристик – *компетенцій,* що дають змогу віднайти конкретних виконавців, здатних до виконання певних видів робіт. Для цього він вивчав характеристики успішних виконавців робіт і порівнював їх з характеристиками менш успішних виконавці тієї ж роботи.

Р. Бояцис, що досліджував характеристики, пов'язані з ефективністю роботи менеджерів незалежно від специфіки діяльності організацій, у яких вони працюють, з'ясував, що компетенції є необхідними для ефективного виконання роботи, але вони не мають вирішального значення, якщо не відповідають функціональним вимогам самої роботи й культурі організації [17]. Саме це твердження пояснює той факт, що найчастіше, змінюючи місце роботи, а разом з ним і умови організаційного оточення, ефективні виконавці можуть не досягати високих результатів під час виконання аналогічної роботи, а зі зміною функціональних обов'язків у межах тієї ж організації (наприклад, посадове підвищення) працівники, що досягли значної ефективності в минулому, не завжди забезпечують її на інших посадах.

Керування компетенціями є складником керування персоналом з погляду системного підходу поряд з функціями підприємства й корпоративною культурою [179].

Незважаючи на те, що керування компетенціями на сьогодні є самостійною сферою досліджень, системи керування компетенціями, з погляду інформаційних технологій, можна розглядати і як окремий випадок системи керування знаннями (СКЗ), і як інформаційно-пошукову систему, оскільки, попри власну специфіку, ці системи

розв’язують задачу подання й зіставлення знань про два типи об’єктів – завдань і виконавців, здатних впоратися з цими завданнями.

Задача керування компетенціями бачиться перспективною сферою для впровадження технологій Semantic Web. Причинами цього є, з одного боку, актуальність і комплексність самої задачі (і брак її автоматизованих розв’язків для загальних випадків), а з іншого – необхідність використовувати для її розв’язання гетерогенні знання (як щодо виконавців, так і щодо організації, у якій вони мають працювати, і предметної області, до якої належить задача, яку вони мають розв’язати), розподілено подані у Web.

Ми розглядаємо окремий випадок цієї задачі, пов’язаний з керуванням компетенціями у сфері наукових досліджень і освіти. Ця підзадача уявляється нам найбільш складною й цікавою з трьох причин. Перша з них – слабка формалізованість профілів компетенцій наукових співробітників (практично кожна окрема наукова тема вимагає розробки власного профілю) і великий вплив специфіки домену досліджень, що припускає необхідність автоматизованої обробки зовнішніх баз знань, що містяться у Web.

Друга причина – діяльність наукових співробітників і результати їхньої роботи, зазвичай, досить чітко формалізовані й відкриті для аналізу: це наукові публікації, патенти, звіти, описи розробок та інші матеріали, презентовані у вигляді природномовних текстів з елементами структурованих даних (таблиць і графіків) і мультимедіа, здебільшого подані в електронній формі, що припускає створення засобів їх автоматизованого аналізу. У сфері освіти результатами роботи є підручники, наукові посібники, робочі програми та інші методичні матеріали, які також можна автоматизовано обробляти засобами, орієнтовними на аналіз ПМ-текстів.

Третя причина – це потреба в знаходженні співвідношення між здобутою освітою й накопиченим досвідом і наявною компетентністю працівників.

Для керування компетенціями в ході планування наукових досліджень такі критерії, як освіта, кваліфікація і досвід, можуть бути виявлені вже на самому початку добору. Компетенції виявляться пізніше, уже в самому процесі добору. Але компетенції здатні зробити важливий внесок у всі етапи добору. Проте зараз такі дослідження досить ефективні лише для чітко формалізованих і схарактеризованих професій і не придатні для добору й атестації персоналу, що займається науковими дослідженнями.

Це пов’язано з тим, що, по-перше, для оцінки роботи науковців потрібно використовувати знання про ту предметну область, яку вони досліджують. Ця ПрО постійно змінюється, доповнюється й часто

загалом може бути суперечливою з погляду різних теоретичних концепцій. По-друге, необхідно об'єктивно оцінювати результати роботи науковців, презентовані в основному у вигляді природномовних документів (статей, доповідей, звітів тощо). Крім того, досить часто важливо оцінити не загальний науковий потенціал конкретної особи, а доцільність її участі в розв'язанні конкретної проблеми (наприклад, чи може цей науковець дати кваліфіковану рецензію на певну статтю) і співробітництва в певному колективі.

Усе це зумовлює необхідність використання в таких системах онтологій різного рівня (предметних областей, проблем, організаційних онтологій тощо). Крім того, виникає проблема інтерпретації семантики наукових праць (автоматизація семантичної розмітки ПМ-тексту, створення й обробка їх метаописів, вибір способів їх адекватного структурування й збереження).

З одного боку, тільки фахівець у конкретній ПрО може оцінити знання й навички кандидата на певну роботу та його професійний рівень. Але якщо виконується спроба автоматизувати цю роботу, то знання такого експерта лягають в основу онтології ПрО й можуть потім застосовуватися без залучення самого експерта. Це, по-перше, звільняє експерта від рутинної роботи й дає змогу йому займатися безпосередньо своїми дослідженнями, а по-друге, забезпечує деяку об'єктивність і уможливлює інтегрування знань кількох експертів.

Модель компетенцій дає змогу створити набір критеріїв, що пов'язують конкретні види діяльності з керуванням персоналом.

Така модель забезпечує:

- розробку загальної термінологічної бази (онтології) для опису роботи організації загалом, тобто однозначної інтерпретації організаційної інформації співробітниками організації;
- досягнення більш високого рівня узгодженості й об'єктивності в оцінці працівника.

Щоб створити модель компетенцій, необхідно:

- визначити найбільш складні, «критичні» робочі завдання цієї діяльності;
- визначити компетенції, що будуть потрібні для виконання кожного з таких завдань;
- упорядкувати ці компетенції відповідно до їх важливості для виконання роботи;
- з'ясувати, чи зможуть особи, що володіють конкретним набором компетенцій, швидко виконувати роботу на прийнятному рівні якості (якщо ні, то ці компетенції треба доповнити).

Важливий клас в онтології наукових досліджень – резюме науковців. Вони містять такі відомості, як публікації, досвід викладання

й наукової діяльності, ступінь освіти й історію дослідницької діяльності [66].

Стандарт Resume RDF Schema може використовуватися як основа для опису структури резюме. Цей стандарт є досить ґрунтовним і багатогранним. Однак, на жаль, розроблювачі цієї онтології не приділили належної уваги особливостям резюме науковців, які, крім загальної інформації, містять специфічну структуровану інформацію – список публікацій різного рівня (зокрема в журналах ВАК, у рецензованих журналах, у базах даних Scopus, Web of Science, Google Академія тощо), досвід дослідницької й викладацької роботи, участь у конференціях тощо. Отже, їх неможливо використовувати без змін для побудови онтології науковців.

Під час формування екземплярів в онтології академічних компетенцій значну частину знань можна здобути з аналізу різних відкритих публікацій дослідників, тобто інформаційних ресурсів, доступних через Web і презентованих природною мовою (ПМ). Розглядаючи ПМ-тексти як джерело знань, на основі якого будується онтологія відповідної ПрО, доцільно використовувати тезауруси.

### 8. 3. 4. Алгоритм визначення рейтингів відповідності компетенцій фахівців поставленій задачі

Будемо вважати, що для певної проблеми:

- визначена ПрО (тобто задана її формалізація через відповідну онтологію ПрО);
- наявний документ (ПМ-текст), який містить опис тієї проблеми, яку потрібно розв’язати (це може бути технічне завдання, подана на рецензію стаття, опис проекту тощо);
- створена організаційна онтологія, яка відображає відомості про тих осіб, що потенційно можуть розв’язати порушену проблему (відомості побудовані на основі аналізу публікацій та іншої природномовної наукової продукції);
- є методи для аналізу ПМ-текстів та їх зіставлення з онтологіями й тезаурусами [103].

Визначення оцінки компетентності фахівців з порушеної проблеми передбачає такі етапи (рис. 8. 9) [25]:

- побудувати тезауруси потенційних експертів (за організаційною онтологією) і тезаурус документа, що ОПР подає на експертизу (за контентом документа);
- нормалізувати ці тезауруси за допомогою онтології відповідної ПрО;
- порівняти термінологію нормалізованих тезаурусів.

Методи побудови тезаурусів за ПМ-документами й алгоритми їх порівняння розглянуті вище, а відповідне наукове обґрунтування цих методів міститься в [42, 54, 167].

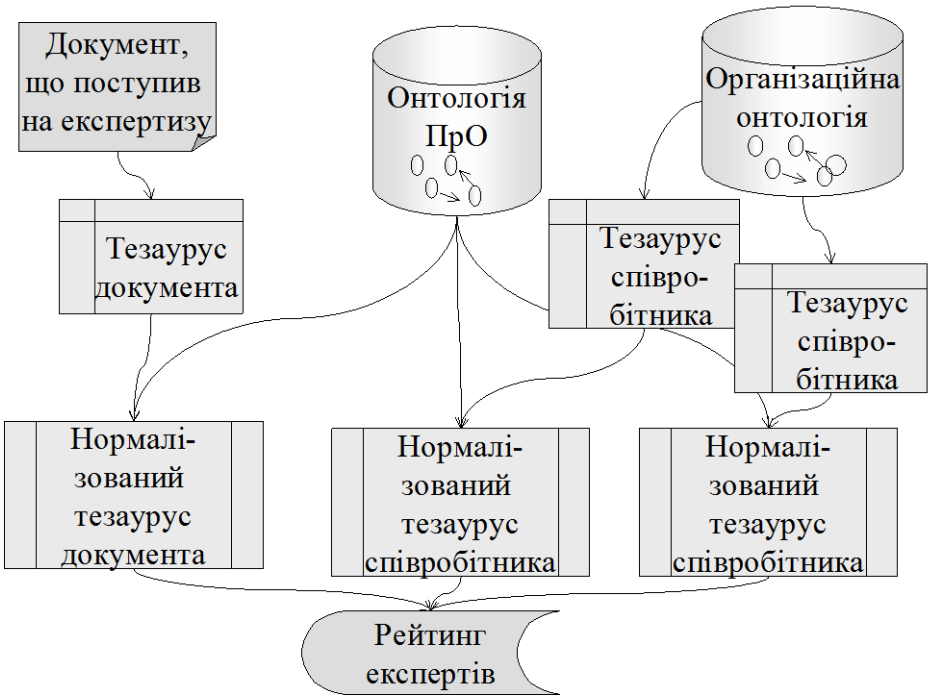



Рис. 8. 9. Побудова рейтингів потенційних експертів

За контентом документа, для якого потрібно здійснити експертизу, будується тезаурус документа $\mathrm{Th(d)}$. Передбачається, що цей документ належить до визначеної ПрО, що відповідає сфері діяльності організації. Знання ПрО подані у вигляді доменної онтології $\mathrm{O_{domain}}$, $\mathrm{O_{domain}} = \langle \mathrm{T, R, F} \rangle$, онтологія ПрО містить n термінів $|\mathrm{T}| = \mathrm{n}$.

На другому етапі роботи алгоритму потрібно нормалізувати тезаурус документа, що надійшов на експертизу (проекція тезауруса на онтологію ПрО) $\mathrm{ThN_n(d)} = \{\mathrm{t_i} : \mathrm{t_i} \in \mathrm{T(O_{domain})}\}$.

Нехай в організаційній онтології містяться відомості про s співробітників.

За організаційною онтологією будуються тезауруси цих співробітників. Інформація для побудови тезаурусів співробітників здобувається з організаційної онтології науково-дослідного інституту, у якому вони працюють і адміністрацією якого можуть бути притягнуті як експерти.

На множину співробітників, яких можна залучати як експертів, можуть бути накладені початкові обмеження (наявність ученого ступеня, кількість публікацій за останній рік і десятиліття, науковий стаж тощо).

Тезаурус співробітника – це множина слів, що містяться в назвах його публікацій, паспорті ВАК його спеціальності, назві науково-дослідних проектів, що він виконує тощо – $Th_i(p)$. Нормалізований тезаурус співробітників $ThN_i(p)$ – це проекція його тезауруса на множину термінів онтології ПрО $O_{domain}$.

Одержуємо для кожного терміна онтології домену $O_{domain}$ $t_j$ набір значень $m_{doc}(t_j)$ і $m_i(t_j), i = \overline{1,s}$ – ваги термінів онтології у відповідних нормалізованих тезаурусах. Для кожного співробітника будується його рейтинг для експертизи конкретного документа $r_i = \sum_{j=1}^{n} m_{doc}(t_j) * m_i(t_j), i = \overline{1,s}$. Ті співробітники, що мають найвищий рейтинг, потенційно є найбільш компетентними для проведення експертизи розглянутого документа (рис. 8. 10).

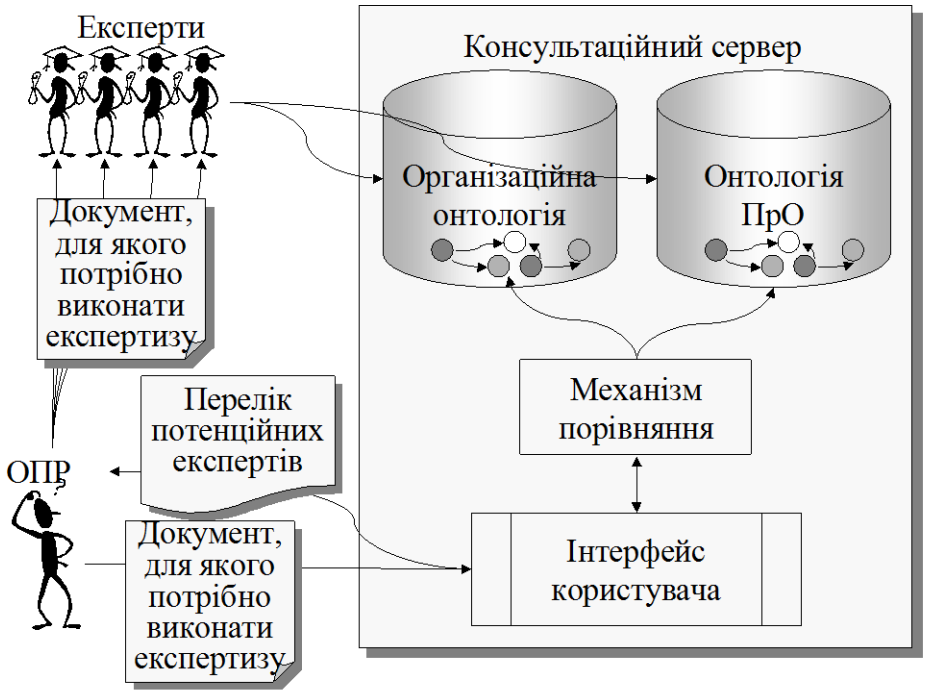



Рис. 8. 10. Процес формування початкової множини експертів за організаційною онтологією

Якщо ж перед експертами стоїть завдання не тільки дати оцінку окремому документу (наприклад, написати рецензію на статтю до наукового журналу, рецензію на дипломний проект, дисертацію тощо), а й зробити порівняльний аналіз і впорядкувати документи з деякої сукупності (наприклад, експертиза проводиться для конкурсу дослідницьких проектів у визначеній галузі), тоді доцільно замість тезауруса окремого документа використовувати об’єднання тезаурусів усіх документів, що підлягають експертизі.

Запропонований підхід дає змогу за описом експертної задачі й формальними відомостями про спеціалістів, поданими у Web через організаційні онтології, побудувати тезаурус задачі й тезауруси потенційних експертів, порівняти їх і на основі цього порівняння визначити коефіцієнт компетентності кожного з цих спеціалістів – кількісну оцінку придатності цього спеціаліста для розв'язання конкретної експертної задачі. Це дає змогу для кожної задачі сформувати висхідну групу експертів, для вдосконалення складу якої можна використовувати інші методики формування експертних комісій.

## 8. 4. Онтологічна модель як основа адаптивності відкритої освіти дорослих

В інформаційному суспільстві одним з ключових питань стає підтримка процесу неперервної освіти, важливим компонентом якої є освіта дорослих. Причиною цього є орієнтованість суспільства на обробку й аналіз знань у всіх сферах діяльності, що потребує отримання нових знань усіма учасниками економічних процесів. Освіта дорослих має власну специфіку й потребує значно вищої персоніфікації засобів освіти й більшого застосування знань відповідних ПрО.

Створення й функціонування системи неперервної освіти об'єктивно пов'язано з розвитком і ефективністю освіти дорослих. Саме освіту дорослих у багатьох країнах світу стали вважати запорукою соціально-економічної стабільності, перспективою розвитку суспільства загалом і професійної життєтворчості особистості.

Однією зі стратегій розвитку освіти дорослих є побудова системи відкритої освіти [83]. Найбільш придатною технологією підтримки відкритої освіти дорослих є *дистанційне навчання*, що базується на сучасних інформаційно-комунікаційних технологіях, а саме – на Web-ресурсах. Але широкий розвиток освіти дорослих стримує недостатньо чітке наукове обґрунтування рекомендацій щодо того, як на практиці створювати, адаптувати й упроваджувати системи дистанційного навчання [11].

Аналіз публікацій свідчить про великий інтерес дослідників до цієї проблеми [113, 121]. Так, у [10] з системних позицій розглядаються основи теорії моделювання організаційних систем відкритої освіти, пропонується теоретико-методологічний апарат системного подання й дослідження організаційних систем, проектуються моделі організаційних систем відкритої освіти, аналізуються особливості їх побудови, проектування, реалізації та впровадження. Особливості адаптивності функціонування освітніх систем у Web на основі рекомендацій розглянуто в [220].

### 8. 4. 1. Специфіка створення систем дистанційної освіти для дорослих у Web

Важливе значення в дослідженні освіти дорослих належить розумінню принципів андрагогіки.

*Андрагогіка* – це науковий напрям, що об'єднує знання про специфіку навчання дорослої людини з урахуванням її віку, освітніх і життєвих потреб, реальних можливостей, індивідуальних особливостей і досвіду, психіки та фізіології. Андрагогіка має свій науковий апарат, власні теоретико-методологічні засади, особливості тощо [3].

Важливим аспектом андрагогіки є наявність індивідуального підходу до навчання на основі особистих потреб, з урахуванням соціально-психологічних характеристик особи й тих обмежень, які накладаються її діяльністю, наявністю вільного часу, фінансових ресурсів.

Більшість дослідників у цій сфері вважають принцип адаптивності навчання одним з основних факторів, що визначають ефективність освіти дорослих.

*Принцип адаптивності навчання* у відкритих системах освіти дорослих спрямований на побудову індивідуальних освітніх програм, забезпечує психологічне корегування стереотипу дії особистості, її мислення й механізми самореалізації.

Аналіз проектів дистанційного навчання, реалізованих на основі сучасних інформаційно-комунікаційних технологій, дає змогу дійти висновку про те, що принцип адаптивності навчання в них реалізований частково або не реалізований зовсім. Одна з причин, на нашу думку, криється в намаганні підлаштуватися під інформаційно-комунікаційні технології, домінування технологічного (а можливо, і технократичного) підходу до проектування зазначених систем. Більшість дистанційних курсів, що розроблювалися для забезпечення якісної освіти дорослих, використовують застарілі інформаційно-комунікаційні технології, в основі яких – ідеї Web 1. 0 і Web 2. 0, що не враховують сучасного стану відкритого інформаційного середовища Web і не застосовують сучасні Web-технології та стандарти.

У перших проектах відкритої освіти, орієнтованих на Web 1. 0, не враховувалася динамічність інформаційних ресурсів Web. Наприклад, проект дистанційного навчання OpenCourseWare – це яскравий приклад реалізації ідей Web 1. 0: ієрархічно організовані інституційні репозиторії надають доступ до ресурсів у форматах HTML чи PDF – дистанційних курсів, навчальних планів і програм разом

з матеріалами до них. Матеріали з-поза меж інституцій не приймаються й перед публікацією проходять ретельну перевірку щодо якості.

Концепція Web 2. 0 тісно пов'язана з таким поняттям, як «соціальний Web», і активно використовує соціальні мережі, XML-структурування й відкриті Wiki-ресурси, зміст і структуру яких користувачі можуть спільно змінювати, переробляти й доповнювати за допомогою відповідних інструментів. Відповідно, проекти відкритої освіти, реалізовані за принципами Web 2. 0, зосереджували свою увагу на розвитку спільноти паралельно з розвитком відкритих ресурсів, вони почали використовувати контент, який створювали самі користувачі. Проект дистанційного навчання Connexions [199] є типовим прикладом такої моделі відкритої освіти, що базується на трьох принципах – більша доступність, більший доступ і більша участь.

Проте всі ці підходи не забезпечували повною мірою принцип адаптивності навчання у відкритих системах освіти дорослих, тому що не враховували семантику інформації, що в них оброблялася, і не містили засобів її автоматизованого аналізу. Тому виникає потреба в розробленні нових підходів до проектування й функціонування таких систем, що враховуватимуть як специфіку сучасного орієнтованого на знання Web, так і використання методів штучного інтелекту для обробки інформації [342].

Основою для цього можуть стати технології наступного покоління Web – Web 3. 0 і Semantic Web.

Основними перевагами систем дистанційної освіти на основі технологій Semantic Web є персоніфіковане й колаборативне навчання, коли кожен студент отримує індивідуальну програму навчання й контролю, побудовану з урахуванням як його особистих переваг і можливостей, так і досвіду інших користувачів цієї системи.

Дистанційна освіта для дорослих через Web-середовище пов'язана з обміном великими обсягами даних в електронній формі між студентами та системою. Крім того, для неї використовуються більш складні та спеціалізовані ресурси. Саме ці дані стають основою для аналізу методами Data Mining.

Новою тенденцією в методології аналізу даних, що дасть змогу ефективно виконувати завдання знаходження й структурування знань у значною мірою хаотично організованій інформаційній мережі, є Web Mining.

Важливо підкреслити, що у Web як навчальні ресурси й знання предметних областей, які вони відображають, так і закономірності, пов'язані з навчанням, – динамічні й постійно змінювані. Це зумовлює потребу в засобах Web Mining для адаптації системи дистанційної освіти до цих змін. Наприклад, якщо в навчальному курсі вказується

остання версія якогось програмного продукту, то потрібно відстежувати посилання саме на нову версію. Інший приклад: з часом певні відомості стають загальновідомими (приміром, робота з Web-сторінками) або застарілими й не цікавлять студентів, а тому їх доцільно прибирати з відповідного курсу.

### 8. 4. 2. Використання Semantic Web для створення інтелектуальних адаптивних освітніх систем

*Інтелектуальні адаптивні освітні системи* (ІАОС) повинні мати достатньо розвинені механізми для підтримки персоналізації навчання. Такі системи активно застосовують результати, отримані в такому науковому напрямі, як штучний інтелект [141]. Як зазначає дослідник В. Деведзич, з погляду учня система виступає як тьютор, що організовує його навчальну сесію. Система використовується як презентаційний планер, що добирає, готує й адаптує навчальний матеріал для учня; поступово будує модель учня під час навчальної сесії для відстеження прогресу в навчанні й оперативного виявлення та виправлення помилок в освітній стратегії учня [241].

Персоналізація навчання є важливим складником адаптивності навчальної системи. Інтелектуальні адаптивні навчальні системи повинні будувати освітню стратегію учня з урахуванням персоналізації, активно допомагати учневі й взаємодіяти з ним у ході всього процесу навчання.

Як правило, персоналізація навчання передбачає:

- адаптивність взаємодії (користувацького інтерфейсу);
- адаптивну доставку курсу;
- адаптацію контенту навчального матеріалу до індивідуальних потреб і можливостей користувача (приміром, пропонувати матеріали певною природною мовою);
- адаптивну підтримку співпраці (наприклад, формування нових завдань і тестів на основі оцінювання попередніх) [319].

Однією з нових форм підтримки адаптивного навчання є надання рекомендацій учневі [398]. *Рекомендації* в ІАОС – це проактивні дії навчальної системи, що надають учневі певну інформацію на основі свого досвіду взаємодії як з ним самим, так і з іншими, подібними до нього учнями.

Створення рекомендацій в ІАОС, зазвичай, базується на узагальненні колаборативного досвіду взаємодії системи з користувачами: закономірності, знайдені для одних користувачів, поширюються на інших користувачів з подібними властивостями. Це дає змогу зменшувати період адаптації системи до персональних потреб кожного з користувачів. Чим вище й точніше інформованість

рекомендувальної системи про персональні потреби користувача та його індивідуальну специфіку сприйняття й обробки інформації, тим ефективнішими є результати надання рекомендацій.

Система рекомендацій може бути складена як на підставі аналізу анкетних даних самого учня, так і в процесі навчання. З погляду використання інтелектуальних адаптивних навчальних систем, інтерес становить система автоматичної персоналізації.

Процедура складання рекомендацій передбачає такі етапи:

- Аналіз моделі учня на підставі використаних Web-ресурсів. На цьому етапі аналізуються Web-сесії учня й застосовується кластеризація. Таким чином, визначаються групи учнів зі схожими освітніми інтересами, рівнем підготовки тощо.
- Попередній обхід та індексація навчальних ресурсів. На цьому етапі складається перелік необхідних ключових слів і Web-сторінок, що містять ці слова.
- Отримання налаштувань користувача з активних навчальних сесій.
- Складання посилань на рекомендації для можливості їхнього використання під час реалізації освітніх стратегій.

Такий підхід базується на двох компонентах – *фазі моделювання* й *фазі рекомендацій*.

Для того, щоб обробляти знання в системі навчання, потрібно побудувати *формальні моделі* основних суб'єктів навчання (учень, група учнів, навчальний ресурс тощо) і встановити відношення між їх елементами.

Фаза моделювання ґрунтується на концепції Web Usage Mining і складається з розробки: 1) моделі (профілю) учня; 2) моделі (профілю) групи учнів і 3) моделі (профілю) змісту (контенту) навчального матеріалу.

Розробка моделі учня передбачає створення профілю учня на основі наявної інформації через явні/неявні засоби зворотного зв'язку. Тут використовується два підходи до моделювання – на підставі спільної діяльності й автоматичний.

Побудова моделі групи відбувається після побудови моделі учня. На основі кластеризації формуються вподобання групи учнів щодо навчальних ресурсів.

На етапі моделювання змісту (контенту) навчального матеріалу застосовується індексація й отримання даних з тексту, що є частиною Web Content Mining. Аналіз робиться на підставі Web-ресурсів, які відвідав учень.

Фаза рекомендацій завершується формуванням освітньої стратегії учня.

Загальна схема взаємодії учня з інтелектуальною адаптивною освітньою системою наведена на рис. 8. 11. Як свідчить схема, на початку відбувається активна взаємодія учня з інтелектуальною адаптивною навчальною системою.

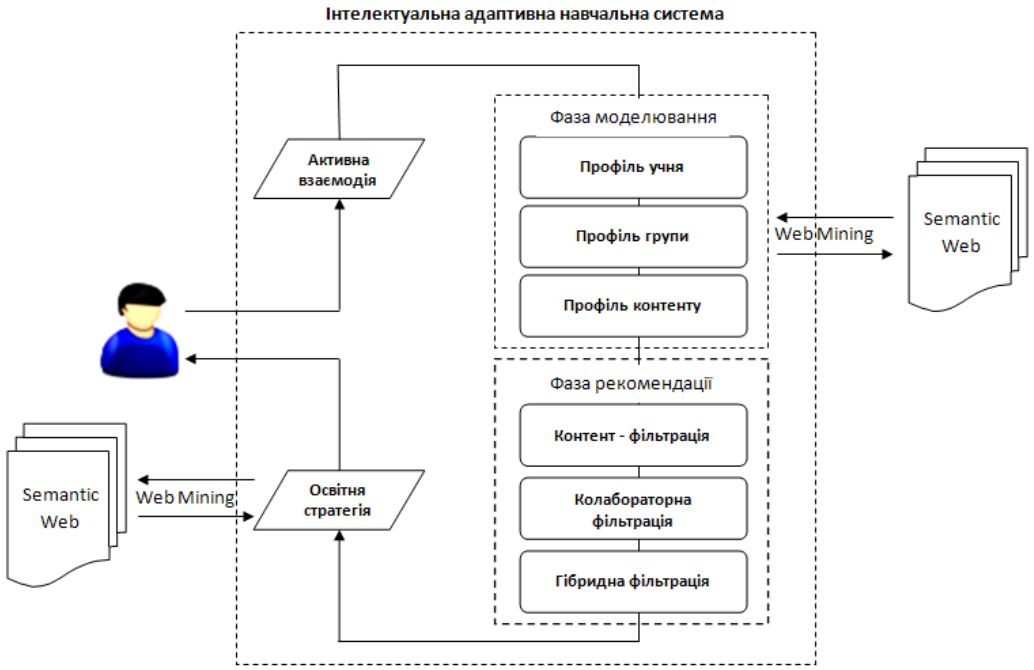



Рис. 8. 11. Загальна схема взаємодії з інтелектуальною адаптивною навчальною системою

На цьому етапі система презентує свої навчальні можливості й намагається отримати максимальну інформацію про учня. На підставі аналізу персональних даних відбувається формування профілю учня, групи й навчального контенту – фаза моделювання. Закінчується фаза моделювання виявленням освітніх потреб учня. На наступному етапі – фазі рекомендацій – відбувається відбір навчальних матеріалів і надання учневі рекомендацій. Завершується цей етап формуванням освітньої стратегії учня. Маючи освітню стратегію учня, інтелектуальна адаптивна навчальна система готова запропонувати йому навчальні матеріали з використанням ресурсів Semantic Web.

### 8. 4. 3. Онтологічна модель дистанційного навчання дорослих

Наведені вище матеріали дають змогу дійти висновку, що створення моделі дистанційного навчання дорослих є складним завданням, що потребує багато часу й відповідних знань. Тому доцільно зробити цю модель придатною для повторного використання, а також інтероперабельною. Крім того, важливо, щоб така модель була чітко формалізована й не припускала розбіжностей в її розумінні [153].

Однією з вимог до відкритих освітніх систем дорослих є забезпечення високого рівня інтероперабельності, що передбачає

можливість взаємодії з різними системами в умовах створення розподілених навчальних систем в Інтернеті. Здебільшого наявні навчальні системи реалізують цю вимогу за рахунок відкритості інтерфейсу, доступу до своїх сервісів шляхом використання єдиної форми для обміну даними, зокрема XML, і об'єктної моделі презентації документів DOM. Проте для відкритих освітніх систем дорослих самої лише синтаксичної інтероперабельності недостатньо.

Для забезпечення семантичної інтероперабельності відкритих освітніх систем дорослих необхідно розробити такий спосіб подання знань, що дасть змогу автоматично опрацьовувати їх програмними агентами і Web-сервісами. Такий спосіб базується на використанні онтологічного підходу.

З урахуванням того, що ця модель орієнтована на застосування, що використовують для взаємодії середовище Web, доцільно, щоб ця модель підтримувалася сучасними Web-стандартами й програмними засобами для колективної роботи.

Ще одна важлива умова, що забезпечить цінність побудованої моделі, – це її виразна здатність. Потрібно, з одного боку, мати досить виразні засоби для відображення семантики взаємодії суб'єктів навчання, а з іншого – забезпечити гарантії того, що алгоритми обробки такої інформації будуть скінченими, а сама модельне не міститиме протиріч.

Усім цим вимогам відповідає онтологічне подання знань для систем дистанційної освіти [283]. Як мову для опису такої онтології доцільно використовувати OWL 2. 0, а для створення й редагування онтології – редактор онтологій Protégé [202]. Крім того, плагіни Protégé дають змогу візуалізувати зв'язки між класами та екземплярами (рис. 8. 12).

Онтологічна модель дистанційної освіти містить класи, що відповідають основним суб'єктам навчального процесу, – «студент», «група студентів», «навчальний курс», «викладач», «послідовність курсів» тощо. Екземпляри класів – це конкретні дисципліни й особи. Властивості дають змогу відображати відношення між екземплярами, приміром, «Студент Петренко вивчає навчальний курс "Програмна інженерія"», «Навчальний курс "Програмна інженерія" вивчається після навчального курсу "Матаналіз"», а також властивості окремих екземплярів, наприклад, «Студент Петренко народився в 1995 році».

Наявність онтологічної моделі дистанційної освіти дасть змогу інтегрувати різні застосовні системи дистанційної освіти; легко переносити з однієї системи до іншої як навчальні ресурси, так і персональні дані студентів; встановлювати зв'язок з іншими онтологіями, які формалізують пов'язані предметні області, та відкритими Wiki-ресурсами.

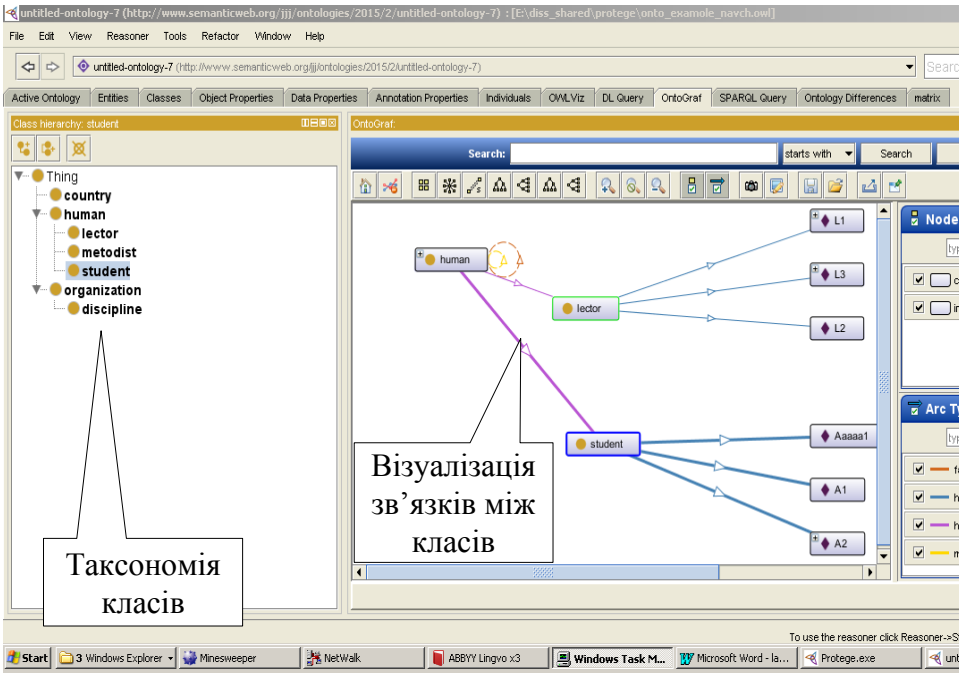



Рис. 8. 12. Онтологічна модель дистанційного навчання дорослих

Це особливо важливо для підтримки безперервної освіти, оскільки персоніфікація дистанційної освіти дорослих є значно вищою порівняно з дистанційною освітою дітей. Це зумовлено тим, що дорослі студенти значно більше різняться досвідом роботи, кваліфікацією, віком, здатністю до сприйняття інформації та прийнятним режимом отримання нових знань.

Адаптація системи до навчання дорослого студента потребує значно більшого обсягу знань, який доцільніше імпортувати з іншої системи, за допомогою якої він навчався раніше, ніж наново вводити всі персональні дані, а потім накопичувати досвід для врахування його особистих переваг.

### 8. 4. 4. Використання Web-сервісів у відкритих системах дистанційної освіти

Парадигма сервіс-орієнтованого програмування забезпечує значно швидший шлях створення нових освітніх програмних продуктів, у яких можна використовувати раніше створені фрагменти програм, що відображені у відповідних стандартах.

Для інтеграції онтологічних систем дистанційної освіти дорослих доцільно схарактеризувати функції цих систем як сервісів, що посилаються на терміни відображеної вище онтології дистанційної освіти; організаційних онтологій установ, що розробляють ці системи; і онтологій верхнього рівня, що містять потрібні поняття (приміром,

географічні відомості або національну систему кваліфікації). Для того, щоб пов'язати сервіси, потрібно зіставити вхід одного з виходом іншого, виконавши операцію вирівнювання над відповідними онтологіями.

Отже, ефективне функціонування системи відкритої освіти дорослих можливе за умови використання інтелектуальних адаптивних навчальних систем, що базуються на сучасних інтелектуальних Web-технологіях. Системи, в основу яких покладено принцип адаптивного навчання, спрямовані на побудову індивідуальних освітніх стратегій і забезпечують психологічне корегування стереотипу дії особистості, її мислення й механізми самореалізації. Функціонування інтелектуальних адаптивних навчальних систем передбачає використання можливостей Semantic Web – нового покоління Web, що уможливлює подання інформації в придатній для автоматичного опрацювання формі. В основі інтелектуальних адаптивних навчальних систем – онтологічна рекомендаційна модель, яка передбачає успішний вибір освітньої стратегії на основі фази моделювання й фази рекомендацій. Фаза моделювання передбачає складання профілю учня, групи й контенту. Фаза рекомендацій базується на фільтрації контенту й аналізі спільної діяльності групи. Функціонування системи на кожному етапі передбачає активне використання методології аналізу даних Web Mining.

## 8. 5. Використання онтологій як основи для розвитку єдиного Європейського освітнього простору

Сьогодні створення єдиного європейського освітнього простору, формування Європейської системи стандартів якості навчання й професійної підготовки, що сприятиме мобільності студентів і науковців, забезпечується Конвенцією про визнання кваліфікацій, що стосуються вищої освіти в європейському регіоні (Лісабон, 1997 р.). Цю концепцію розроблено й ухвалено під егідою Ради Європи та ЮНЕСКО. Але для її ефективного використання на практиці виникає потрібні засоби та методи ухвалення рішень щодо встановлення еквівалентності академічних кваліфікацій, навчальних програм, дипломів тощо.

Відповідно, це потребує порівняння тих знань, що містяться у відповідних нормативно-правових і методичних документах, які регламентують діяльність систем освіти й професійної підготовки [94, 226].

Структура системи стандартів вищої освіти в Україні – це ієрархія взаємопов'язаних компонентів, які встановлюють вимоги до змісту, обсягу й рівня вищої освіти на трьох рівнях – держави, галузі й вищого навчального закладу. Однак стандартизація національної вищої освіти

стикається з великими труднощами. Її незавершеність і нехтування чинною нормативно-правовою базою в галузі освіти призводять до непорозуміння зі сферою праці, знижують соціальний рівень випускників, перешкоджають можливостям їх працевлаштування [231].

Ще на початку 2006 року Міністерство освіти і науки України для розв'язання зазначеної проблеми підписало спільний з Міністерством праці та соціальної політики наказ про затвердження методики створення Державного стандарту вищої освіти, тобто гармонізованого Переліку кваліфікацій і Переліку напрямів і спеціальностей вищої освіти. У грудні 2006 року Постановою Кабінету міністрів затверджено Перелік напрямів, за якими здійснюється підготовка фахівців у вищих навчальних закладах за освітньо-кваліфікаційним рівнем «бакалавр» (далі – Перелік-2006). Уже в січні 2007 року Міністерством освіти і науки цей перелік набув чинності (наказ МОНУ № 58 від 27. 01. 07 р.). Цим же наказом передбачалося до 1 червня 2007 року визначити базові ВНЗ і створити робочі групи з розроблення галузевих стандартів відповідно до Переліку-2006.

Але виникли певні проблеми. Як можна розроблювати стандарт (освітньо-кваліфікаційну характеристику, освітньо-професійну програму, засоби діагностики якості вищої освіти) без розуміння кваліфікації фахівця? Крім того, міжнародна практика розробки кваліфікацій свідчить про нагальну необхідність співпраці академічних і професійних кіл, сфери освіти й ринку праці. Тільки стратегія партнерства соціальних учасників може забезпечити адекватне визначення конкурентоспроможних професій і посад. Слід ураховувати світовий досвід, визначаючи перелік кваліфікацій на етапі розробки галузевих стандартів.

Здійснений огляд дає загальне уявлення про основні складники такої галузі знань, як комп'ютинг. Наочні зображення, що були використанні під час супроводу опису кожного з напрямів, дійсно є ефективними для формування цілісної картини. Проте для більш детального опису кожної з галузей скористаємося наведеними нижче табличними даними (див. табл. 8. 2), що містять інформацію як про перелік дисциплін комп'ютингу, так і про необхідний рівень оволодіння фахівцями всіх п'яти напрямів цими дисциплінами. Лівий стовпчик таблиці містить перелік з 40 дисциплін комп'ютингу. В інших стовпчиках – числові значення необхідного рівня вивчення дисципліни для фахівця кожного з напрямів комп'ютингу (значення знаходяться в діапазоні від 0 (найнижчий) до 5 (найвищий)).

**Таблиця 8. 2.**

**Порівняльна характеристика п’яти напрямів комп’ютингу**

| № п\п | Найменування галузі знань | CE | | CS | | IS | | IT | | SE | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | min | max | min | max | min | max | min | max | min | max |
| 1. | Основи програмування (Programming Fundamentals) | 4 | 4 | 4 | 5 | 2 | 4 | 2 | 4 | 5 | 5 |
| 2. | Компонентно-базоване програмування (Integrative Programming) | 0 | 2 | 1 | 3 | 2 | 4 | 3 | 5 | 1 | 3 |
| 3. | Алгоритми і складність (Algorithms and Complexity) | 2 | 4 | 4 | 5 | 1 | 2 | 1 | 2 | 3 | 4 |
| 4. | Архітектура й організація комп’ютерів (Computer Architecture and Org.) | 5 | 5 | 2 | 4 | 1 | 2 | 1 | 2 | 2 | 4 |
| 5. | Розробка та принципи операційних систем (Operating Systems Principles & Design) | 2 | 5 | 3 | 5 | 1 | 1 | 1 | 2 | 3 | 4 |
| 6. | Конфігурування й використання операційних систем (Operating Systems Configuration & Use) | 2 | 3 | 2 | 4 | 2 | 3 | 3 | 5 | 2 | 4 |
| 7. | Розробка та принципи мережних технологій (Net | 1 | 3 | 2 | 4 | 1 | 3 | 3 | 4 | 2 | 4 |

| № п\п | Найменування галузі знань | CE | | CS | | IS | | IT | | SE | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | min | max | min | max | min | max | min | max | min | max |
| | Centric Principles and Design) | | | | | | | | | | |
| 8. | Використання й конфігурування мережних технологій (Net Centric Use and configuration) | 1 | 2 | 2 | 3 | 2 | 4 | 4 | 5 | 2 | 3 |
| 9. | Платформні технології (Platform technologies) | 0 | 1 | 0 | 2 | 1 | 3 | 2 | 4 | 0 | 3 |
| 10. | Теорія мов програмування (Theory of Programming Languages) | 1 | 2 | 3 | 5 | 0 | 1 | 0 | 1 | 2 | 4 |
| 11. | Людино-машинна взаємодія (Human-Computer Interaction) | 2 | 5 | 2 | 4 | 2 | 5 | 4 | 5 | 3 | 5 |
| 12. | Графіка й візуалізація (Graphics and Visualization) | 1 | 3 | 1 | 5 | 1 | 1 | 0 | 1 | 1 | 3 |
| 13. | Інтелектуальні системи (Intelligent Systems) | 1 | 3 | 2 | 5 | 1 | 1 | 0 | 0 | 0 | 0 |
| 14. | Теорія баз даних (Information Management (DB) Theory) | 1 | 3 | 2 | 5 | 1 | 3 | 1 | 1 | 2 | 5 |
| 15. | Використання баз даних (Information Management (DB) Practice) | 1 | 2 | 1 | 4 | 4 | 5 | 3 | 4 | 1 | 4 |
| 16. | Чисельні методи | 0 | 2 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 |

| № п\п | Найменування галузі знань | CE | | CS | | IS | | IT | | SE | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | min | max | min | max | min | max | min | max | min | max |
| | (Numerical mthds) | | | | | | | | | | |
| 17. | Соціальні й етичні питання IT (Legal / Professional / Ethics / Society) | 2 | 5 | 2 | 4 | 2 | 5 | 2 | 4 | 2 | 5 |
| 18. | Розробка інформаційних систем (Information Systems Development) | 0 | 2 | 0 | 2 | 5 | 5 | 1 | 3 | 2 | 4 |
| 19. | Аналіз бізнес-вимог (Analysis of Business Requirements) | 0 | 1 | 0 | 1 | 5 | 5 | 1 | 2 | 1 | 3 |
| 20 | Е-бізнес (E-business) | 0 | 0 | 0 | 0 | 4 | 5 | 1 | 2 | 0 | 3 |
| 21. | Аналіз технічних вимог (Analysis of Technical Requirements) | 2 | 5 | 2 | 4 | 2 | 4 | 3 | 5 | 3 | 5 |
| 22. | Основи програмної інженерії (Engineering Foundations for SW) | 1 | 2 | 1 | 2 | 1 | 1 | 0 | 0 | 2 | 5 |
| 23. | Економіка програмної інженерії (Engineering Economics for SW) | 1 | 3 | 0 | 1 | 1 | 2 | 0 | 1 | 2 | 3 |
| 24. | Моделювання й аналіз програмного забезпечення (Software | 1 | 3 | 2 | 3 | 3 | 3 | 1 | 3 | 4 | 5 |

| № п\п | Найменування галузі знань | CE | | CS | | IS | | IT | | SE | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | min | max | min | max | min | max | min | max | min | max |
| | Modeling and Analysis) | | | | | | | | | | |
| 25. | Проектування програмного забезпечення (Software Design) | 2 | 4 | 3 | 5 | 1 | 3 | 1 | 2 | 5 | 5 |
| 26. | Верифікація й випробування програмного забезпечення (Software Verification and Validation) | 1 | 3 | 1 | 2 | 1 | 2 | 1 | 2 | 4 | 5 |
| 27. | Супровід програмного забезпечення (Software Evolution (maintenance)) | 1 | 3 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 4 |
| 28. | Процеси програмного забезпечення (Software Process) | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 5 |
| 229. | Якість програмного забезпечення (Software Quality) | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 4 |
| 330. | Технологія обчислювальних систем (Comp Systems Engineering) | 5 | 5 | 1 | 2 | 0 | 0 | 0 | 0 | 2 | 3 |
| 331. | Схемотехніка (Digital logic) | 5 | 5 | 2 | 3 | 1 | 1 | 1 | 1 | 0 | 3 |
| 332. | Вбудовані системи (Embedded Systems) | 2 | 5 | 0 | 3 | 0 | 0 | 0 | 1 | 0 | 4 |
| 333. | Розподілені | 3 | 5 | 1 | 3 | 2 | 4 | 1 | 3 | 2 | 4 |

| № п\п | Найменування галузі знань | CE | | CS | | IS | | IT | | SE | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | min | max | min | max | min | max | min | max | min | max |
| | системи (Distributed Systems) | | | | | | | | | | |
| 334. | Основи безпеки IT (Security: issues and principles) | 2 | 3 | 1 | 4 | 2 | 3 | 1 | 3 | 1 | 3 |
| 335. | Керування безпекою IT (Security: implementation and mgt) | 1 | 2 | 1 | 3 | 1 | 3 | 3 | 5 | 1 | 3 |
| 336. | Системне адміністрування (Systems administration) | 1 | 2 | 1 | 1 | 1 | 3 | 3 | 5 | 1 | 2 |
| 337. | Організаційне керування IT (Management of Info Systems Org.) | 0 | 0 | 0 | 0 | 3 | 5 | 0 | 0 | 0 | 0 |
| 338. | Системна інтеграція (Systems integration) | 1 | 4 | 1 | 2 | 1 | 4 | 4 | 5 | 1 | 4 |
| 339. | Технології мультимедіа, розробка (Digital media development) | 0 | 2 | 0 | 1 | 1 | 2 | 3 | 5 | 0 | 1 |
| 440. | Технічна підтримка (Technical support) | 0 | 1 | 0 | 1 | 1 | 3 | 5 | 5 | 0 | 1 |

Для пояснення методики використання даних з таблиці 8. 2. розглянемо дві дисципліни «Розробка і принципи операційних систем (Operating Systems Principles & Design)» і «Конфігурування і використання операційних систем (Operating Systems Configuration &

Use)». Обидві дисципліни стосуються операційних систем. Дисципліна «Розробка і принципи операційних систем (Operating Systems Principles & Design)» надає студенту можливість зрозуміти основні принципи й задачі операційних систем, їх стратегію й тактику, різні механізми, популярні підходи до проектування операційних систем. Після вивчення теоретичного матеріалу програмою підготовки передбачається, що студент або розробить проект операційної системи, або внесе доповнення до вже наявної ОС.

Дисципліна «Конфігурування і використання операційних систем (Operating Systems Configuration & Use)» передбачає практичне використання операційних систем і не зосереджує увагу на основних принципах проектування й розробки ОС. Проте значна увага приділяється формуванню в студента компетентностей ефективного використання операційних систем, розуміння впливу властивостей системи на розподіл ресурсів, на задоволення потреб користувача.

Аналіз таблиці дав змогу дійти висновку про те, що фахівці з інформаційних систем і технологій значно менше приділяють уваги дисципліні «Розробка і принципи операційних систем (Operating Systems Principles & Design)» порівняно з фахівцями з комп'ютерних наук, комп'ютерної та програмної інженерії. Максимальні значення для двох зазначених галузей становлять 1 і 2 відповідно, що свідчить про ознайомлення майбутніх фахівців лише з загальною термінологією та принципами. Мінімальні показники для галузей CE, CS, і SE наочно підтверджують, що студенти цих напрямів ґрунтовно вивчають основні принципи й положення ОС. Проте й тут можна спостерігати цікавий факт стосовно того, що для фахівців з CS і SE мінімальне значення становить 3 одиниці, тоді як для CE – лише 2. Це означає, що програми підготовки для CS і SE, зазвичай, приділяють цій темі більше уваги. Максимальні показники для CS (5) вищі, ніж у CE й SE (4), що свідчить про те, що в програмах підготовки CS, як правило, виділяється більше часу на цю тему для поглибленого вивчення матеріалу.

Дещо інша картина з дисципліною «Конфігурування і використання операційних систем (Operating Systems Configuration & Use)». Тоді, як усі програми забезпечують набуття достатніх навичок у використанні й конфігуруванні операційних систем, програмою підготовки IT-фахівців передбачені найвищі показники діапазону – 3 і 5 відповідно. Це означає, що від студентів вимагаються ґрунтовні знання й навички з цієї теми. Інші програми підготовки припускають значно нижчі показники вимог.

Як переконуємося, використання наведеної інформації дає змогу чітко розподілити «сфери інтересів» кожного напряму комп'ютингу,

що, сподіваємося, стане незамінним інструментом для розробників українського стандарту з комп'ютингу.

Зауважимо, що останніми роками зусиллями представників декількох найбільших російських університетів була проведена значна робота з підвищення якості підготовки й забезпечення уніфікації обсягу знань випускників вищої школи в галузі інформаційних технологій. Її результатом стало впровадження Міністерством освіти РФ у 2002 році нового науково-освітнього напрямку «Інформаційні технології». У 2003 році Міністерством були затверджені також документи, що визначали вимоги до змісту й рівня підготовки випускників вищої школи зі спеціалізацій цього напрямку й фактично отримали статус експериментального освітнього стандарту. Зазначені документи значною мірою спиралися на Computing Curricula-2001. Навіть більше, у 2002 році Санкт-Петербурзький університет за підтримки асоціації АПКІТ і декількох комп'ютерних компаній Росії й України випустив переклад заключного звіту об'єднаної комісії ACM і IEEE-CS «Computing Curricula 2001: Computer Science». І хоча його наклад лише 600 екземплярів, електронна версія перекладу звіту має вільний доступ за адресою: se.math.spbu.ru/cc2001. Використання вказаних документів як методичних матеріалів у процесі розробки конкретних навчальних програм, як зазначають російські фахівців, безсумнівно, сприяє забезпеченню відповідності знань випускників сучасному стану комп'ютингу, їх підготовленості до актуальних наукових досліджень, практичних розробок, використання продуктів інформаційних технологій, затребуваності фахівців на ринку праці.

Проведений аналіз світового бачення проблеми вивчення такої галузі знань, як комп'ютинг, дає можливість дещо по-іншому подивитися на українські реалії й спробувати зрозуміти еволюцію бачення цієї проблеми в українській системі вищої освіти. Так, згідно з Постановою КМУ № 507 від 24 травня 1997 р. набув чинності Перелік напрямів і спеціальностей, за якими здійснюється підготовка фахівців у вищих навчальних закладах за відповідними освітньо-кваліфікаційними рівнями (далі – Перелік-1997). Комп'ютинг як галузь знань був презентований напрямами підготовки 0802 «Прикладна математика» і 0804 «Комп'ютерні науки». Проте протягом 2004-2005 рр. на підставі змін, внесених згідно з наказами Міністерства освіти і науки, до переліку включені такі напрями підготовки, як: 0915 «Комп'ютерна інженерія», 0914 «Комп'ютеризовані системи, автоматика і управління» й 0925 «Автоматизація та комп'ютерно-інтегровані технології». Отже, наприкінці 2006 року комп'ютинг в українському варіанті передбачав п'ять напрямів підготовки в межах

галузей «Математика та інформатика» й «Інженерія». Звичайно ж, наведений перелік був далеким від бачення світової спільноти.

Так, наприклад, напрям 0802 «Прикладна математика» з узагальненим об'єктом діяльності «математичне моделювання, розробка алгоритмів, проектування, розробка та експлуатація комп'ютерних програмних засобів» передбачав здобуття кваліфікації 3114 «Технік обчислювального (інформаційно-обчислювального) центру». Фахівець був підготовлений до виконання таких видів роботи: 72.10 «Консультації з питань інформатизації», 72.20 «Створення програмного забезпечення», 72.30 «Обробка даних», 72.40 «Робота з базами даних». Відповідно до Державного класифікатора професій, такий фахівець здатний був виконувати таку професійну роботу: 3114 – Технік обчислювального (інформаційно-обчислювального) центру, 3115 – Технік з автоматизації виробничих процесів, 3121 – Технік-програміст, 3139 – Технік-оператор електронного устаткування, 3491 – Лаборант наукового підрозділу, що може обіймати первинну посаду «технік-програміст».

Натомість напрям 0804 «Комп'ютерні науки» з узагальненим об'єктом діяльності – «комп'ютерні інформаційні системи й технології» – передбачав присвоєння кваліфікації «інженер-програміст». Чи не дивно? І знову ж фахівець був підготовлений до виконання таких видів роботи: 72.10 «Консультації з питань інформатизації», 72.20 «Створення програмного забезпечення», 72.30 «Обробка даних», 72.40 «Робота з базами даних», хоча вже був здатний виконувати професійну роботу як техніка-програміста, так і інженера-програміста.

Напрям підготовки 0915 «Комп'ютерна інженерія» з узагальненим об'єктом діяльності «Технічні (апаратні) засоби та системне програмне забезпечення комп'ютерних систем і мереж універсального та спеціального призначення та їх компонентів», здавалося б, не викликає запитань. Проте кваліфікації 3121 «Технік-програміст», 3122 «Оператор електронно-обчислювальної (комп'ютерної техніки)», 3114 «Технік обчислювального (інформаційно-обчислювального) центру» і 3119 «Лаборант (комп'ютерна техніка)» навряд чи відповідають змісту підготовки.

Узагальнений об'єкт діяльності напряму 040301 «Прикладна математика» передбачає як математичне моделювання, так і розробку алгоритмів, проектування, розробку та експлуатацію комп'ютерних програм. На нашу думку, широта об'єкта програми підготовки зазначеного напряму пов'язана з інертним розумінням змісту комп'ютингу в пострадянській вищій школі й не відображає сучасного стану галузі знань, де розробкою алгоритмів, проектуванням, розробкою

та експлуатацією комп'ютерних програм займаються напрями «Computer Science» і «Software Engineering». Узагальнений об'єкт діяльності напряму 040301 «Прикладна математика», за аналогією, наприклад, до напряму 030502 «Економічна кібернетика», можна було б сформулювати як «Математичне моделювання». Такий підхід дав би змогу вилучити напрям підготовки 040301 «Прикладна математика» з галузі комп'ютингу й внести його до галузі 0402 «Фізико-математичні науки».

Ситуація з напрямом 040302 «Інформатика» ще парадоксальніша. Виділившись в окремий напрям підготовки в «Переліку-2006», цей напрям ще більше загострив термінологічну дискусію. Підсилює дискусію й така галузь знань, як 0501 «Інформатика і обчислювальна техніка», що як напрям підготовки виділяє напрям 050101 «Комп'ютерні науки».

На окрему увагу заслуговують два напрями підготовки – 040303 «Системний аналіз (процеси системного відображення дійсності та проектування систем керування)» і 050201 «Системна інженерія (системи планування й управління технічними, технологічними та організаційними об'єктами)». На нашу думку, напрям 040303 «Системний аналіз» становить теоретичну основу напряму «Information Systems», адже фахівці в галузі інформаційних систем відіграють ключову роль у визначенні вимог до інформаційних систем організації, проводять активну політику у визначенні специфікацій, проектуванні й розробці інформаційних систем. Натомість напрям 050201 «Системна інженерія» наближається до напряму «Information Technology», адже, на відміну від фахівців з інформаційних систем, IT-фахівці більше розглядають технологічний аспект комп'ютингу.

Напрям підготовки 050101 «Комп'ютерні науки» з узагальненим об'єктом діяльності «комп'ютерні інформаційні системи і технології» дещо не відповідає світовому баченню «Computer Science», однак органічно розчиняється як в «Information Systems», так і в «Information Technology». Цей факт необхідно врахувати у процесі розробки галузевих стандартів.

Узагальнений об'єкт діяльності напряму 050102 «Програмна інженерія», що виділений у «Переліку-2006», на нашу думку, потребує уточнення.

Напрям підготовки 050202 «Автоматизація та комп'ютерно-інтегровані технології (системи автоматизації та комп'ютерно-інтегровані технології), на нашу думку, дещо надумано виділений в окремий напрям, адже в його основі – розуміння напряму 050102 «Комп'ютерна інженерія».

Критичний аналіз «Переліку-2006» свідчить про необхідність доповнення його галуззю знань на зразок «Комп'ютинг». Розуміючи можливі наслідки використання транскрипції в нормативних документах, можна зупинитися на варіанті «Інформатика і комп'ютерна техніка». Проте й наявна на сьогодні галузь «Системні науки і кібернетика» за певної реструктуризації також може бути аналогом до комп'ютингу.

Проведений аналіз світового бачення й українських реалій становлення й розвитку такої галузі знань, як комп'ютинг, свідчить про необхідність напруженої роботи в напрямку гармонізації національних освітніх стандартів зі світовими аналогами. Системі професійної освіти в Україні потрібен критичний перегляд і узгодження її з рекомендаціями Computing Curricula щодо наявних освітніх стандартів, пов'язаних з підготовкою фахівців з комп'ютингу. Такий підхід дасть змогу розширити міжнародне співробітництво з закордонними університетами з питань підготовки кадрів для комп'ютингу, зберегти й розвинути сильні сторони вітчизняної вищої школи, що виявляються у фундаментальній математичній підготовці випускників, залучаючи тим самим закордонних студентів не стільки нижчою вартістю навчання, скільки більш якісною та фундаментальною освітою, посилити позиції національної вищої школи на міжнародному ринку освітніх послуг в одній з найбільш актуальних і перспективних галузей знань – комп'ютингу. Зазначимо, що цей процес не повинен негативно вплинути на якість математичної підготовки фахівців. Сама ж математична підготовка має акцентувати увагу на вивченні таких дисциплін, як дискретна математика, математична логіка й чисельні методи, що безпосередньо використовуються у формуванні науково-методичних основ галузі комп'ютингу.

У подальших дослідженнях планується проаналізувати співвідношення між структурами національних галузевих стандартів у галузі комп'ютінгу і світовими аналогами для визначення можливостей щодо адаптації українських нормативних документів до рекомендацій світового комп'ютерного співтовариства.

## 8. 6. Забезпечення прозорості Європейської та національних рамок кваліфікацій за допомогою комп'ютерних онтологій

В Україні вже затверджено Національну рамку кваліфікацій (Постанова Кабінету Міністрів України «Про затвердження Національної рамки кваліфікацій» за № 1341 від 23. 11. 2011 р.). Однак

у документі йдеться тільки про визначення рівнів кваліфікацій, а їх зміст, співвідношення з ЄРК, стратегії розвитку національної системи кваліфікацій потребують подальшої розробки. Немає сьогодні і єдиного розуміння та бачення вимог щодо інструментального засобу, який дав би можливість встановлювати співвідношення між рівнями Європейської та національних рамок кваліфікацій, забезпечуючи їх міжнародне порівняння й визнання.

Отже, розробка інструментального засобу забезпечення прозорості Європейської та національних рамок кваліфікацій є актуальним і вчасним науковим завданням [129].

Концептуальні засади та методичні аспекти впровадження національних рамок кваліфікацій сьогодні активно обговорюються академічною спільнотою України [106, 181]. Вітчизняні науковці аналізують можливість використання досвіду створення Європейської та національних рамок кваліфікацій [12, 116]. Метою такого аналізу є розроблення й упровадження нових освітніх стандартів як основи трансформації навчальних програм та інших складників системи навчально-методичного забезпечення підготовки фахівців, принципового оновлення методів і засобів діагностування результату навчання. Однак розробка інструментального засобу забезпечення прозорості Європейської та національних рамок кваліфікацій, що полегшує встановлення співвідношення рівнів кваліфікацій, забезпечуючи їх прозорість і міжнародне визнання, нині перебуває на периферії наукового пошуку, а окремі роботи не дають загального розуміння проблеми.

Сьогодні встановити співвідношення між рівнями кваліфікацій Європейської та національних рамок допомагає спеціальний механізм – інтерактивні таблиці, що розміщені на порталі Єврокомісії, присвяченому європейській рамці кваліфікацій [http://ec.europa.eu/eqf/compare_en.htm].

Зазначені інтерактивні таблиці дають змогу порівнювати національні рівні як з ЄРК, так і між собою. Важливим є той факт, що в цих інтерактивних таблицях міститься доступ до дескрипторів, за допомогою яких здійснюється опис рівнів кваліфікацій. Саме дескриптори дають можливість розглянути результати навчання через призму таких категорій, як знання, навички й компетентності. Однак наведений механізм позбавлений можливості аналізувати рівні кваліфікацій, встановлювати співвідношення між освітніми й професійними кваліфікаціями.

Інструментальний засіб, розроблений в рамках проекту DISCO II (the European DIctionary of Skills and COmpetences) Єврокомісії (http://disco-tools.eu/) забезпечує порівняння результатів навчання,

термінологічну підтримки процесу перекладу й зіставлення документів про освіту в рамках проекту Європейської прозорості документів.

Сьогодні в Євросоюзі реалізується проект TRACE (TRAnsporentCompetenceinEurope), метою якого також є забезпечення прозорості між європейською рамкою кваліфікацій і національними рамками країн, що входять до ЄС [334]. На відміну від інтерактивних таблиць порталу Єврокомісії, розроблені в межах проекту TRACE комп'ютерні онтології дають змогу пов'язувати освітні й професійні кваліфікації, що значно полегшує процес встановлення співвідношення між рівнями кваліфікацій. Проте освітні кваліфікації, що використовуються в проекті TRACE, побудовані на основі навчальних програм, так званих курикулумах (Curriculum). Такий підхід ускладнює процес визначення відповідних кваліфікацій, адже національні освітні кваліфікації базуються на галузевих освітніх стандартах, що також повинні бути подані у вигляді семантичного значення предметної області. Саме таким поданням можуть стати комп'ютерні онтології.

Європейська рамка кваліфікацій містить вісім взаємопов'язаних рівнів, на яких кваліфікація визначається за результатами навчання через тріаду професійних якостей – знань, навичок і компетентностей. Такий підхід допомагає порівняти кваліфікації та спрощує процедуру їх визнання.

Національні системи кваліфікацій призначені не тільки для опису системи кваліфікацій, а й для модернізації системи професійної освіти й підготовки кадрів, збільшення доступу громадян до кваліфікацій. Роль національних рамок кваліфікацій у модернізації полягає в тому, що професійна освіта повинна перейти до результатів навчання. Для цього необхідно розвивати співробітництво у сфері праці, розробити професійні стандарти, нові технології оцінки компетентностей, що покладені в основу кваліфікацій, і визнавати результати навчання незалежно від того, чи вони були досягнуті у сфері формальної чи неформальної освіти. Роль НРК у розширенні доступу до кваліфікації полягає в тому, що завдяки рамці люди здатні визначити свої власні компетенції, не проходячи для цього навчання в межах обов'язкових освітніх програм, що, між іншим, уможливлює оптимізацію ресурсів на навчання й формування гнучких траєкторій освіти.

Україна також долучилася до процесу розроблення й упровадження національної системи кваліфікацій. Як зазначено в проекті Концепції розвитку національної системи кваліфікацій (станом на 16. 10. 2012 р.), процес формування й розвитку національної системи кваліфікацій України спрямований на реалізацію політики навчання впродовж життя й ґрунтується на загальних європейських принципах і рекомендаціях із забезпечення якості у сфері освіти

й професійної підготовки [130]. Національна система кваліфікацій передбачає участь соціальних партнерів у процесах, пов'язаних з визнанням навчання, розробленням, забезпеченням якості та присвоєнням кваліфікацій. Визнання результатів навчання здійснюється незалежно від способу їх отримання (як шляхом визнання формальної, так і неформальної та інформальної (спонтанної) освіти).

Підкреслимо, що пріоритетними завданнями розвитку національної системи кваліфікацій є:

- забезпечення відповідності кваліфікацій потребам ринку праці, розвитку економіки, суспільства й громадян;
- розширення участі соціальних партнерів у процесах, пов'язаних з визнанням результатів навчання, розробленням, забезпеченням якості та присвоєнням кваліфікацій;
- створення механізмів визнання результатів навчання, незалежно від способу їх отримання;
- забезпечення гнучкості кваліфікацій, зокрема різноманітності траєкторій (шляхів) їх здобуття й підвищення;
- підвищення рівня компетентності працівників;
- визнання цінності кваліфікацій соціальними партнерами (соціальне визнання кваліфікацій);
- забезпечення міжнародної порівнянності / прозорості й визнання кваліфікацій, здобутих в Україні.

Основним елементом національної системи кваліфікацій є Національна рамка кваліфікацій (НРК), що охоплює всі рівні та підсистеми кваліфікацій і співвідноситься з європейською рамкою кваліфікацій навчання впродовж життя. Національна рамка кваліфікацій містить опис рівнів для всіх підсистем кваліфікацій – як кваліфікацій формальної освіти, так і професійних кваліфікацій.

Зіставлення кваліфікацій з кваліфікаційними рівнями НРК здійснюється на основі співвіднесення результатів навчання за кваліфікацією певного типу з описом відповідного кваліфікаційного рівня НРК. Проте сьогодні є ряд труднощів, що унеможливлюють упровадження національної рамки кваліфікацій в Україні:

- наявні кваліфікаційні характеристики професійної сфери й освітні стандарти не враховують систему компетентностей НРК і, як правило, їх не можна порівняти з Національною та європейськими рамками кваліфікацій;
- сучасна структура галузевих стандартів вищої освіти є надмірно ускладненою й регламентованою, обмежує можливості навчальних закладів щодо модифікації програм підготовки відповідно до запитів ринку праці;

- кваліфікації вищої освіти не можна формально зіставляти з кваліфікаціями Європейського простору вищої освіти (ЄПВО);
- стандарти компетентності для значної кількості класів і підкласів професій не сформовані, унаслідок чого наявні певні труднощі, пов'язані з присвоєнням професійних кваліфікацій;
- переліки напрямів і спеціальностей вищої освіти надмірно деталізовані й не відповідають потребам ринку праці.

Для розв'язання окреслених проблем рекомендується здійснити такі кроки:

- розробити характеристики вітчизняних освітніх кваліфікацій з урахуванням дескрипторів Національної рамки кваліфікацій;
- провести формальне зіставлення вітчизняних освітніх кваліфікацій з Національною рамкою кваліфікацій (за рівнями);
- зіставити вітчизняні кваліфікації вищої освіти з Рамкою кваліфікацій Європейського простору вищої освіти;
- ужити комплекс заходів щодо впровадження компетентнісного підходу в освітні стандарти й навчальні програми, практику викладання й оцінювання;
- формувати професійні стандарти з урахуванням дескрипторів Національної рамки кваліфікацій і зіставлення професійних кваліфікацій з кваліфікаційними рівнями НРК;
- застосувати нові підходи до розробки галузевих стандартів вищої освіти, визначивши, що:

a) галузеві стандарти вищої освіти розробляються за галузями освіти, перелік яких доцільно сформувати відповідно до Міжнародної стандартної класифікації освіти (ISCED);

b) галузевий стандарт вищої освіти є цілісним документом, який повинен містити опис соціально-особистісних, загальнонаукових, інструментальних і загальнопрофесійних компетентностей, а також методи демонстрації та критерії оцінювання результатів навчання;

- визнати невіддільним академічним правом вищих навчальних закладів здатність до визначення спеціальних професійних компетентностей (результатів навчання) випускників і формування освітньо-професійної програми підготовки.

Ще одним необхідним кроком, на нашу думку, є дослідження питання розроблення й використання інструментальних засобів встановлення співвідношення між рівнями кваліфікацій для забезпечення прозорості Європейської та Національної рамок кваліфікацій.

Таким інструментальним засобом можуть слугувати RCD (Reasable Competency Definition) і SRCM (Simple Reasable Competency Mapping) [334]. RCD розроблювався як стандарт для послідовного

й структурованого опису компетентностей. Цей стандарт містить не тільки опис компетентностей, а й дає змогу здійснювати обмін інформацією про них між різними автоматизованими системами. Однак компетентності, схарактеризовані за допомогою природної мови, не несуть семантичного навантаження. У деяких випадках дві практично ідентичні компетентності через брак можливості їх семантичного аналізу розпізнавалися системою як абсолютно різні. Альтернативою стандарту RCD став стандарт SRCM, що доповнив RCD логічними зв'язками. Це уможливило покращення рівня розуміння компетентностей та їх ідентифікацію. Проте гарантувати якісний аналіз без повноцінного семантичного наповнення стандарт SRCM не міг. Саме тому найбільш придатним інструментальним засобом подання кваліфікацій та опису результатів навчання вбачаються комп'ютерні онтології.

Зазначимо, що ідея використання комп'ютерних онтологій для семантичного подання певної предметної галузі не нова. Застосування мультиагентного онтологічного підходу до створення розподілених систем дистанційного навчання розглядалося в [85]. Питання використання онтологій предметних областей на базі технологій Semantic Web як основи для функціонування відкритих освітніх систем висвітлено в [283]. У [404] розкрито стратегію управління навчальними програмами (Curriculum) на основі онтологічного підходу. Дослідження [416] присвячено потенціалу онтологій у сфері неформальної та інформальної освіти.

У зазначених роботах були розглянуті окремі аспекти застосування онтологічного підходу в освітніх системах. Однак використання комп'ютерних онтологій як інструментального засобу забезпечення прозорості рамок кваліфікацій не стало предметом окремого дослідження.

Продемонструємо можливості комп'ютерних онтологій для реалізації цього завдання на прикладі онтології, що дає змогу здійснити опис кваліфікації 3121– «Фахівець з інформаційних технологій». Предметна галузь вибрана не випадково. Підготовка фахівців з інформаційних технологій відбувається в такій галузі знання, яка визначена світовою спільнотою як «Computing». Ця галузь позначає узагальнену галузь знань, до складу якої входить комп'ютерна інженерія, комп'ютерні науки, програмна інженерія, інформаційні системи та інформаційні технології.

Розглянемо це детальніше на прикладі ситуації з переліком напрямів підготовки фахівців, що пов'язані з інформатикою і комп'ютерною технікою, а також з інформаційними технологіями.

На нашу думку, наведений перелік не тільки далекий від світового бачення, а й не може бути реалізований у сучасній системі професійної освіти України, що намагається бути конкурентоспроможною.

Зауважимо, що проблема гармонізації національних стандартів у галузі комп'ютингу вже була предметом дослідження як зарубіжних, так і українських освітян. Зокрема, у [192] подано загальний огляд становлення освітніх програм у галузі інформаційних систем і технологій. У [184] розглянуто основні принципи розробки освітніх стандартів для підготовки бакалаврів і магістрів за напрямом «Інформаційні технології». Конкретний приклад адаптації рекомендацій документа «Computing Curricula: Software Engineering» до умов російської освіти наведено в роботі [187].

Серед вітчизняних досліджень заслуговує на увагу робота [173], у якій розкрито особливості розвитку та становлення напрямів підготовки, пов'язаних з комп'ютерними дисциплінами, проаналізовано спектр напрямів комп'ютингу й зроблено огляд програм підготовки спеціальностей у межах зазначеної галузі знань. Проблемі впровадження в навчальний процес національних вищих закладів освіти рекомендацій з вивчення комп'ютерних дисциплін присвячена публікація [196].

На жаль, вказані дослідження позбавлені комплексного бачення проблеми в інформаційно-комп'ютерній галузі. У них немає аналізу українських реалій, бракує конкретних рекомендацій розробникам стандартів, які б ураховували особливості національної системи вищої освіти.

Термін «комп'ютинг» (від англійського «Computing»), який увів у науковий обіг керівник АСМ П. Деннінг, позначає узагальнену галузь знань, до складу якої входить комп'ютерна інженерія, комп'ютерні науки, програмна інженерія, інформаційні системи та інформаційні технології. Комп'ютинг охоплює такі проблеми, як проектування й розробка апаратного забезпечення, обробка, структуризація й управління різними видами інформації, виконання наукових досліджень на комп'ютерах, підвищення інтелектуальності комп'ютерних систем, створення й використання комунікаційних і мультимедійних засобів, пошук і збирання інформації, релевантної будь-якій специфічній діяльності тощо [228].

Переклад терміна «комп'ютинг» став дискусійним у середовищі науковців і професіоналів, адже він, як і термін «computer science», в українській мові адекватно не перекладений [133]. Цей термін можна перекласти як «обчислення», «обчислювальні науки» або «обчислювальна техніка». Проте такі визначення надто звужуються галузь комп'ютингу, що охоплює і науку, і техніку, й інженерні

дисципліни. Широко вживаний у російській науковій літературі й нормативних документах Російської Федерації варіант перекладу «інформаційні технології» є ще більш невдалим і відображає лише один з окремих напрямів підготовки в межах ширшої галузі знань під назвою комп'ютинг. Саме тому доцільніше використовувати термін «комп'ютинг», що є транслітерацію відповідного англійського слова «computing».

Тривалий час різні галузі комп'ютингу не мали суттєвих точок перетину. Студенти, які хотіли бути розробниками програмного забезпечення або вивчати теоретичні аспекти комп'ютингу, вчилися на спеціальності «Computer Science» (CS). Для студентів, які б хотіли працювати з комп'ютерним обладнанням, пропонувалася спеціальність «Electrical Engineering», а для тих, хто бажав використовувати апаратне й програмне забезпечення для розв'язання проблем виробництва й бізнесу, – спеціальність «Information Systems».

Кожна із зазначених спеціальностей мала свою сферу впливу, й випускники цих спеціальностей претендували на цілком різні зони ринку праці. Студенти, що навчалися за спеціальностями «Computer Science» і «Electrical Engineering», могли працювати разом над різними аспектами однієї проблеми; натомість фахівці галузі Information Systems, частіше співробітничали з бізнес-школами та випускниками менеджерських спеціальностей. До 90-х років минулого сторіччя спеціальність «Computer Engineering» була спеціалізацією спеціальності «Electrical Engineering».

У 90-ті роки XX століття відбулися серйозні зміни в галузі комп'ютингу, що й спричинило зміну номенклатури спеціальностей. Так, зокрема, у цей період мікропроцесорна техніка стала основною компонентою електричних систем, різного обладнання. Проектуванням і програмуванням комп'ютерних чипів і зайнялися комп'ютерні інженери. Унаслідок цього спеціальність «Computer Engineering» вийшла зі спеціальності Electrical Engineering. Спеціальність «Computer Science» в 90-х роках набула популярності. У цей час уже припинилися дискусії щодо її легітимності й «Computer Science» була визнана самостійною галуззю знань (противники цього процесу називали «Computer Science» і платформою для математичних досліджень, і псевдонаукою для програмістів). Паралельно з «Computer Science» почала формуватися й така галузь знань, як «Software Engineering». Необхідність появи такої спеціальності зумовлена розробкою складного, проте якісного програмного забезпечення. Тоді, як фахівці з «Computer Science» займалися створенням нових знань, фахівці з «Software Engineering» вивчали методи проектування й розробки надійного програмного забезпечення. Така галузь знань, як «Information

Systems», у 90-х роках минулого століття також адекватно реагувала на потреби як комп'ютингу, так і всього суспільства. І якщо спочатку «Information Systems» були інструментарієм для технічних спеціалістів, то незабаром вони стали складниками робочого середовища будь-якої організації чи установи. Програми підготовки за спеціальністю «Information Technology» з'явилися наприкінці 90-х років XX ст. У цей час стало зрозуміло, що наявні академічні програми не готують фахівців, які б гарантували адекватне використання обчислювальної техніки для розв'язання проблем підприємства й ефективного керування організаційними процесами. Підсилювали зацікавленість в «Information Technology» й розвиток мережі Інтернет, і необхідність розв'язання проблеми Y2K, і, як не дивно, поява євро в Європі.

У Радянському Союзі дослідження в галузі навчання інформатики й обчислювальної техніки розвивалися значною мірою самостійно й у деякому відриві від світових розробок. Але інтеграція світового науково-технічного потенціалу комп'ютингу, що базується на основі діяльності міжнародної системи стандартизації, характеризується масштабом, аналогів якого ще не знала історія науки й техніки [183]. Основні ж зусилля світового педагогічного співтовариства зосереджені навколо створення й постійного оновлення документа «Computing Curricula» (CC), що містить рекомендації для розробки стандартів з комп'ютингу. До речі, у 2002 році російський переклад CC2001 отримав назву «Рекомендації щодо викладання інформатики в університетах» (як робочі варіанти редактори перекладу розглядали такі назви, як: «Рекомендації щодо складання навчальних планів з інформатики» і «Рекомендації до навчальних планів з інформатики» [186]), що, на нашу думку, дещо не відповідає змістові документа.

Перша версія «Computing Curricula» була розроблена Комітетом з освіти в рамках проекту Асоціації з обчислювальної техніки (Association for Computing Machinery, ACM) і вийшла у світ у 1968 році. У 70-х роках аналогічний документ був підготовлений комп'ютерною спільнотою Інституту інженерів з електротехніки та електроніки (IEEE Computer Society).

Наприкінці 90-х років минулого століття стало зрозумілим, що сфера знань комп'ютингу дуже стрімко розвивається і її важко, якщо взагалі можливо, повністю висвітлити в рамках одного університетського курсу. У зв'язку з цим було ухвалено рішення про поділ «Computing Curricula» на чотири основні галузі – інформатика, або комп'ютерні науки (computer science), програмна інженерія (software engineering), комп'ютерна інженерія (computer engineering) та інформаційні системи (information systems). Перший

варіант серії «Computing Curricula 2001» був випущений наприкінці 2001 року. Як офіційні рекомендації щодо навчання інформаційних систем був затверджений документ «Information Systems 2002», розроблений у межах спільного проекту ACM, AIS (Association for Information Systems) і AITP (Association of Information Technology Professionals). Рекомендації щодо навчання програмної інженерії були випущені в серпні 2004 року. Нарешті, документ з рекомендаціями щодо комп'ютерної інженерії був затверджений у грудні 2004 року.

У вересні 2005 року випущено оглядовий том для всього проекту «Computing Curricula». У ньому була вперше сформульована потреба виділення ще однієї самостійної галузі під назвою «Інформаційні технології» (Information Technology). Найближчими роками очікується початок наступної ітерації відновлення стандартів серії «Computing Curricula», що, можливо, приведе до подальшого розширення списку дисциплін.

Підкреслимо, що, незважаючи на американське походження й підтримку проекту «Computing Curricula» Національним науковим фондом США (National Science Foundation, NSF), у документі CC2005 враховані глобалістичні тенденції, без особливої прив'язки до освіти США. Над розробкою проекту SE2004 («Програмна інженерія 2004»), що є основою відповідного розділу CC2005, працювали представники Австралійської комп'ютерної спільноти (Australian Computer Society), Британської комп'ютерної спільноти (British Computer Society), а також Японської спільноти з обробки інформації (Information Processing Society of Japan). Отже, проект CC2005 став світовим баченням проблем, пов'язаних з викладанням комп'ютингу.

Для кращого розуміння змісту кожної з програм підготовки комп'ютингу наведемо їх коротку характеристику.

Напрям *Комп'ютерна інженерія* (Computer Engineering, CE) пов'язаний з проектуванням і конструкцією комп'ютерів і апаратної частини комп'ютерних систем. Програма підготовки з CE передбачає вивчення технічних засобів, програмного забезпечення, комунікацій, розглядає особливості взаємодії між ними. Курс навчання зосереджується на питаннях теорії, принципах і методах традиційної радіотехніки й математики, на можливостях застосування їх до проблем проектування комп'ютерів і комп'ютерних пристроїв. Студенти – майбутні комп'ютерні інженери – вивчають проектування цифрових апаратних систем, способи розробки програмного забезпечення, що безпосередньо стосується цифрових пристроїв та їх інтерфейсів, взаємодії з користувачем та іншими пристроями. Напрям CE має виражену інженерну спрямованість і зосереджується більше на апаратному забезпеченні, ніж на програмному.

Напрям *Комп'ютерні науки* (Computer Science, CS) охоплює широкий діапазон знань – від теоретичних і алгоритмічних основ до сучасних питань робототехніки, інтелектуальних систем, біоінформатики тощо. Сфери діяльності фахівців з «Computer Science» можна поділити на три категорії. До першої категорії належить діяльність, що пов'язана з розробкою і впровадженням програмного забезпечення, з контролем за роботою інших програмістів, ознайомленням їх з новими підходами. У межах другої категорії фахівці винаходять нові шляхи використання комп'ютерної техніки, зокрема для розшифрування таємниці ДНК. Третя категорія діяльності пов'язана з розробкою ефективних шляхів розв'язання проблем комп'ютингу. Так, фахівці з CS розробляють можливі кращі способи зберігання інформації в базах даних, транспортування даних у мережах тощо. Отже, напрям Computer Science охоплює широкий діапазон знань – від теорії до програмування. І хоча програми підготовки з Computer Science дуже часто зазнають критики за вказану широту, вони все-таки дають змогу випускникам швидко адаптуватися до нових ідей і технологій.

Фахівці з інформаційних систем (Information Systems, IS) концентрують увагу на інтеграції інформаційних систем і бізнес-процесів, щоб відповідати інформаційним потребам бізнес-структур, надаючи їм можливість досягати ефективного розв'язання задач. Напрям Information Systems акцентує увагу на інформації й розглядає технологію як інструмент для генерації, обробки й поширення інформації. Проте фахівці з інформаційних технологій повинні розуміти як технічні, так і організаційні аспекти процесу обробки інформації, мають бути здатні допомогти організації визначитися в тому, як інформація в бізнес-процесах може забезпечити конкурентну перевагу. Фахівець з інформаційних систем відіграє провідну роль у визначенні вимог до інформаційних систем організації, проводить активну політику у визначенні специфікації, проектуванні й реалізації.

Напрям *Інформаційні технології* (Information Technology, IT) – галузь, що постала як реакція на практичні, щоденні потреби бізнесу та інших галузей виробництва і яка швидко зростає. Сьогодні більшість організації повністю залежить від інформаційних технологій, що змушує їх мати відповідні системи, які повинні працювати належним чином, бути безпечними, здатними до модернізації, підтримки й заміни. Фахівці з інформаційних технологій мають володіти необхідною комбінацією теоретичних знань і практичних умінь, що давало б їм змогу дбати про інфраструктуру інформаційних потоків організації. IT-фахівці відповідають за вибір апаратного й програмного забезпечення для організації чи установи й займаються впровадженням його у виробничий процес.

*Програмна інженерія* (Software Engineering, SE) займається розробкою і супроводом надійних та ефективних систем програмного забезпечення. Програмна інженерія відрізняється від інших інженерних напрямів неосяжною природою програмного забезпечення й дискретною природою операцій з розробки програмного забезпечення. Вона прагне об'єднати принципи математики й комп'ютерних наук з інженерними методами, що застосовуються для інженерії матеріальних, фізичних тіл. Майбутні програмні інженери вивчають питання надійності програмного забезпечення, розглядають методи його розробки та супроводу.

Така розбіжність у презентації галузі знань «Computing» практично унеможливлює зіставлення програм підготовки, позбавляє перспектив навчання за програмами «подвійних» дипломів. Цілком логічним у такому разі є уніфіковане подання інформації про стандарти вищої освіти з певної галузі знань у вигляді онтологічної моделі.

Розвиток і подальше використання онтології в галузі знань «Computing» дасть змогу: забезпечити розробку складників системи галузевих стандартів вищої освіти на єдиній методологічній основі; гармонізувати національні галузеві стандарти вищої освіти з відповідними світовими аналогами; виконувати порівняння програм навчання для забезпечення «подвійних» дипломів; гарантувати прозорість рамок кваліфікацій.

Як зазначено в галузевому стандарті вищої освіти, узагальненим об'єктом діяльності для кваліфікації 3121 – «Фахівець з інформаційних технологій» є процеси обробки інформації алгоритмічними методами з використанням комп'ютерної техніки й навчання інформатики в навчальних закладах I-II рівнів акредитації. Фахівець підготовлений до таких видів робіт у галузі економіки (за ДК 009: 2005), як «Діяльність у сфері інформатизації» та «Освіта», і до таких професійних робіт (за класифікаційним угрупованням з класифікатора ДК 003: 2005), як 3121 – «Фахівець з інформаційних технологій і 3340 – «Викладач-стажист».

Поступово здійснюючи опис усіх класів і властивостей об'єктів предметної галузі відповідно до галузевого стандарту вищої освіти, одержуємо онтологію, яка є семантичним поданням кваліфікації 3121 – «Фахівець з інформаційних технологій».

На наступному етапі онтологія доповнюється (рис. 8. 13) описом рівнів Європейської та Національної рамок кваліфікацій і їх складників (знаннями, уміннями, компетенціями).

Редактор онтологій Protégé дає змогу подати розроблену онтологію в графічному вигляді (рис. 8. 14).

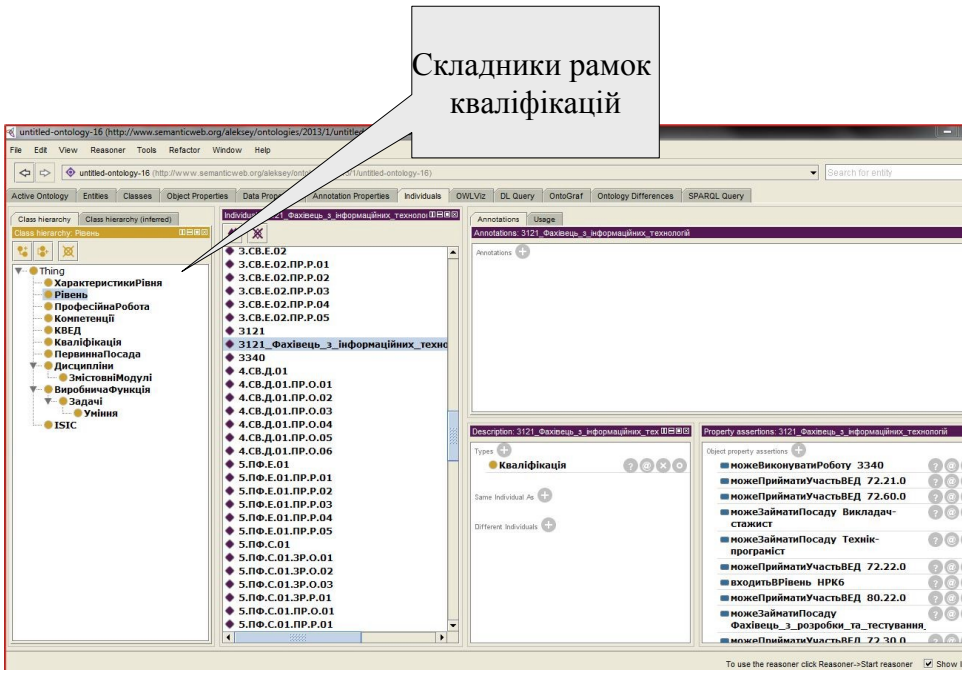



Рис. 8. 13. Опис рівнів рамок кваліфікацій та їх складників

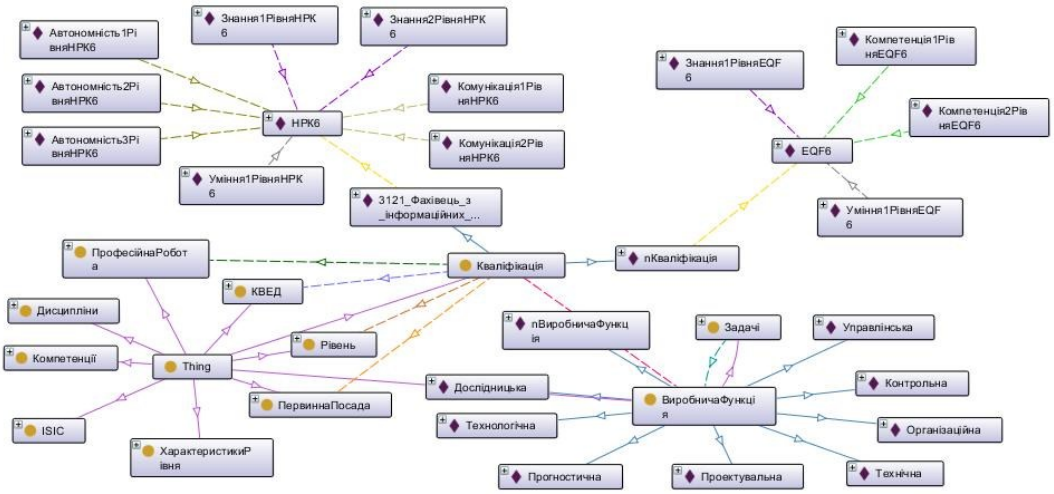



Рис. 8. 14. Графічне подання онтології

У Protege є можливість формувати запити до онтології мовою SPARQL. Наведемо приклади запитів до онтології.

Запит на рис. 8. 15 дає змогу вивести перелік знань з певного рівня рамок кваліфікацій.

У запиті на рис. 8. 16 з онтології виведено перелік змістовних модулів.

Наступний запит (рис. 8. 17) дає змогу вивести перелік кваліфікацій (з різних рамок кваліфікацій) і поєднаних з ними змістових

модулів, які належать до того ж рівня рамки кваліфікацій, що й кваліфікація 3121 – «Фахівець з інформаційних технологій».

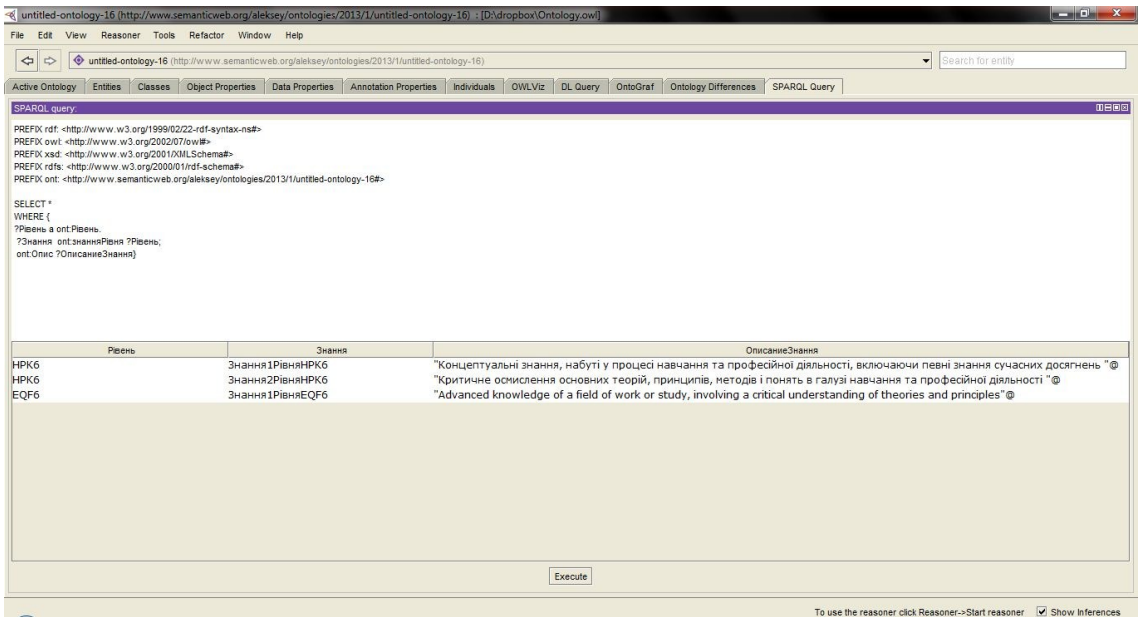



Рис. 8. 15. Результат запиту щодо знань з певного рівня рамок кваліфікацій

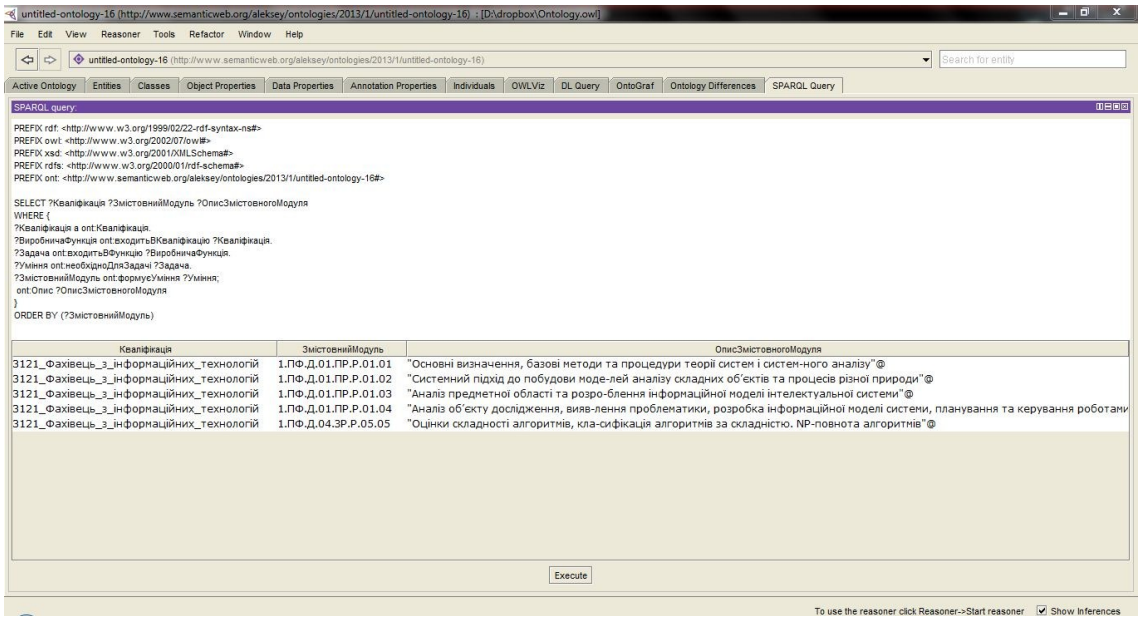



Рис. 8. 16. Результат запиту щодо змістовних модулів

Зауважимо, що можливості редактора Protege-OWL дають змогу інтегрувати онтології з іншими онтологіями. Наприклад, з наведеною вище онтологією можна інтегрувати онтології інших рамок кваліфікацій або ж онтології галузевих освітніх стандартів, що робить онтологію масштабованою та динамічною.

Отже, розроблений на основі комп'ютерних онтологій інструментальний засіб є ефективним механізмом забезпечення прозорості європейської та національних рамок кваліфікацій. Саме

за допомогою комп'ютерних онтологій здійснюватиметься встановлення співвідношення між рівнями кваліфікацій Європейської та національних рамок, стане легшим процес порівняння кваліфікацій і спроститься процедура їх визнання. Надалі планується розробити зручне для користування програмне забезпечення, що уможливило б використання комп'ютерних онтологій Європейської та національних рамок кваліфікацій усіма соціальними партнерами.

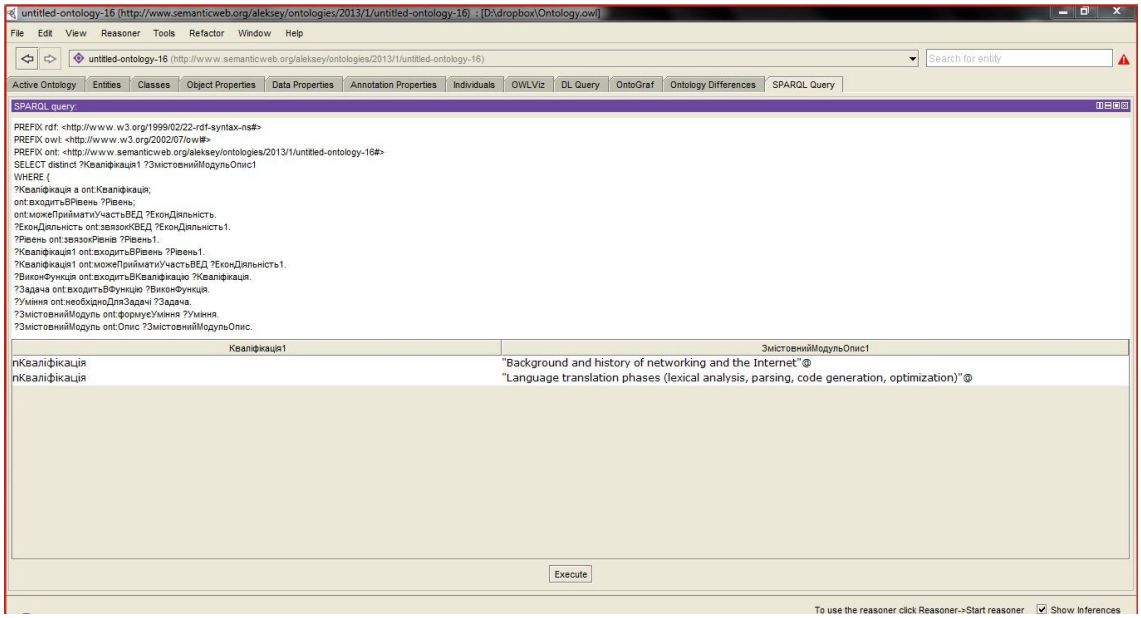



Рис. 8. 17. Результат запиту щодо кваліфікацій (з різних рамок кваліфікацій) і поєднаних з ними змістових модулів

**Висновки**

Наведені вище приклади використання онтологічного підходу в розвитку єдиного європейського освітнього простору, а також застосування його з метою аналізу й порівняння компетенцій (як для пошуку експертів у сфері наукових досліджень, так і для підтримки навчального процесу), сприяння мультиагентним персоніфікованим системам дистанційної освіти тощо доводять високу ефективність такого підходу.

Аналіз публікацій у цій сфері підтверджує великий інтерес наукової спільноти до подібних досліджень і свідчить про їх актуальність.

# ДОДАТКИ

## ДОДАТОК А.
## Онтологічна модель взаємодії між користувачами та інформаційними ресурсами в семантичній пошуковій системі

Онтологічна модель взаємодії між користувачами та інформаційними ресурсами в семантичній пошуковій системі МАІПС створена в редакторі онтологій Protégé й подана мовою OWL. Ця модель застосовується як приклад у наведених вище матеріалах. Детальний опис цієї моделі входить до опису системи МАІПС. Онтологія, презентована нижче, відображає тільки класи й не містить опису екземплярів. На рисунку А. 1 наведена візуалізація цієї онтології в Protégé, зроблена за допомогою плагіна OWL Viz.

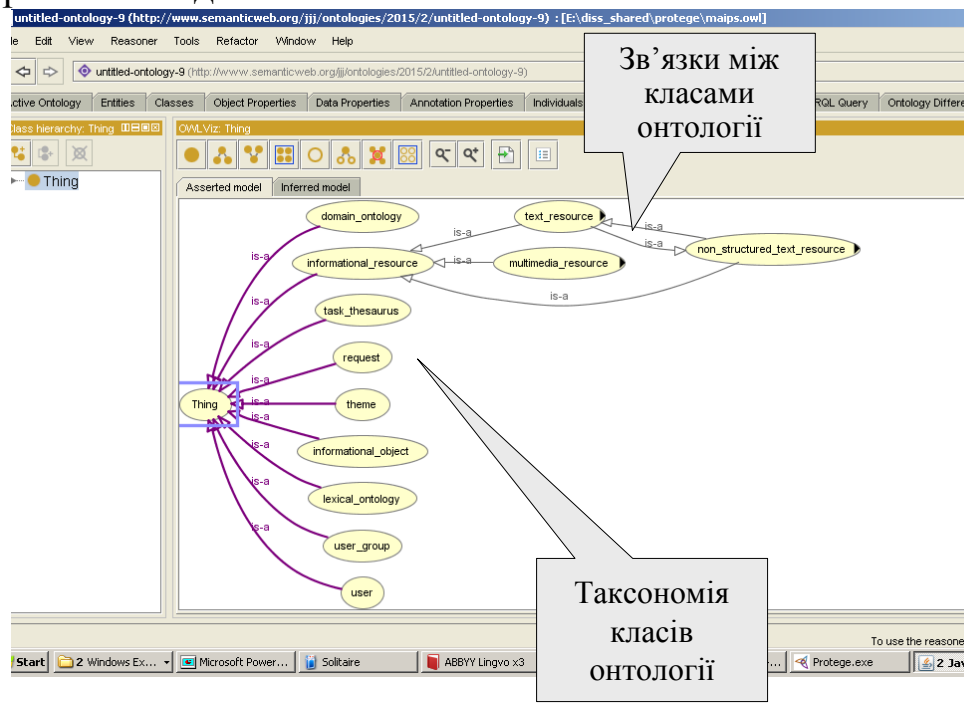



Рис. А. 1. Візуалізація онтології в Protégé, зроблена за допомогою плагіна OWL Viz

Нижче наведено опис цієї онтології мовою OWL.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
 <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
 <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
 <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
 <!ENTITY untitled-ontology-9
"http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#" >
]>
```

<rdf:RDF xmlns="http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#"
xml:base="http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:untitled-ontology-9="http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#">
<owl:Ontology rdf:about="http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9"/>

<!--
//
// Object Properties
//
-->
<!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#based_on_ontology -->

<owl:ObjectProperty rdf:about="&untitled-ontology-9;based_on_ontology">
<rdfs:range rdf:resource="&untitled-ontology-9;domain_ontology"/>
<rdfs:domain rdf:resource="&untitled-ontology-9;task_thesaurus"/>
</owl:ObjectProperty>

<!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#contain_lexics -->

<owl:ObjectProperty rdf:about="&untitled-ontology-9;contain_lexics">
<rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
<rdfs:range rdf:resource="&untitled-ontology-9;domain_ontology"/>
<rdfs:domain rdf:resource="&untitled-ontology-9;lexical_ontology"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#include_people -->

<owl:ObjectProperty rdf:about="&untitled-ontology-9;include_people">
<rdfs:range rdf:resource="&untitled-ontology-9;user"/>
<rdfs:domain rdf:resource="&untitled-ontology-9;user_group"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#interest_in -->

<owl:ObjectProperty rdf:about="&untitled-ontology-9;interest_in">
<rdfs:range rdf:resource="&untitled-ontology-9;domain_ontology"/>
<rdfs:domain rdf:resource="&untitled-ontology-9;user"/>

```
        </owl:ObjectProperty>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#is_a_domain -->

        <owl:ObjectProperty rdf:about="&untitled-ontology-9;is_a_domain">
        <rdfs:domain rdf:resource="&untitled-ontology-9;domain_ontology"/>
        <owl:inverseOf rdf:resource="&untitled-ontology-9;interest_in"/>
        <rdfs:range rdf:resource="&untitled-ontology-9;user"/>
        </owl:ObjectProperty>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#is_a_goal -->

        <owl:ObjectProperty rdf:about="&untitled-ontology-9;is_a_goal">
        <rdfs:domain rdf:resource="&untitled-ontology-9;informational_object"/>
        <rdfs:range rdf:resource="&untitled-ontology-9;request"/>
        </owl:ObjectProperty>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#is_an_ontology_class -->

        <owl:ObjectProperty rdf:about="&untitled-ontology-9;is_an_ontology_class">
        <rdfs:range rdf:resource="&untitled-ontology-9;domain_ontology"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;informational_object"/>
        </owl:ObjectProperty>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#is_charecterd_by -->

        <owl:ObjectProperty rdf:about="&untitled-ontology-9;is_charecterd_by">
        <rdfs:range rdf:resource="&untitled-ontology-9;text_resource"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;user"/>
        </owl:ObjectProperty>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#require -->

        <owl:ObjectProperty rdf:about="&untitled-ontology-9;require">
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
        <rdfs:range rdf:resource="&untitled-ontology-9;request"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;user"/>
        </owl:ObjectProperty>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#semantically_marked_by -->

        <owl:ObjectProperty                                rdf:about="&untitled-ontology-
9;semantically_marked_by">
        <rdfs:range rdf:resource="&untitled-ontology-9;lexical_ontology"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;text_resource"/>
        </owl:ObjectProperty>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#use_thesaurus -->
```

```
        <owl:ObjectProperty rdf:about="&untitled-ontology-9;use_thesaurus">
        <rdfs:domain rdf:resource="&untitled-ontology-9;request"/>
        <rdfs:range rdf:resource="&untitled-ontology-9;task_thesaurus"/>
        </owl:ObjectProperty>
        <!--
        //
        // Data properties
        //
       //////////////////////////
        -->

        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#id -->

        <owl:DatatypeProperty rdf:about="&untitled-ontology-9;id">
        <rdfs:domain rdf:resource="&untitled-ontology-9;informational_object"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;informational_resource"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;lexical_ontology"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;request"/>
        <rdfs:range rdf:resource="&xsd;integer"/>
        <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>

        <!--  http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#name
-->

        <owl:DatatypeProperty rdf:about="&untitled-ontology-9;name">
        <rdfs:domain rdf:resource="&untitled-ontology-9;domain_ontology"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;task_thesaurus"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;theme"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;user"/>
        <rdfs:domain rdf:resource="&untitled-ontology-9;user_group"/>
        <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
        //
        // Classes
        //
        -->
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#data_base -->

        <owl:Class rdf:about="&untitled-ontology-9;data_base">
        <rdfs:subClassOf                                rdf:resource="&untitled-ontology-
9;structured_text_resource"/>
        </owl:Class>
```

```
        <!--          http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#domain_ontology -->

        <owl:Class rdf:about="&untitled-ontology-9;domain_ontology"/>
        <!--          http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#informational_object -->

        <owl:Class rdf:about="&untitled-ontology-9;informational_object"/>

        <!--          http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#informational_resource -->

        <owl:Class rdf:about="&untitled-ontology-9;informational_resource"/>
        <!--          http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#lexical_ontology -->

        <owl:Class rdf:about="&untitled-ontology-9;lexical_ontology"/>
        <!--          http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#metadata -->

        <owl:Class rdf:about="&untitled-ontology-9;metadata">
        <rdfs:subClassOf                                    rdf:resource="&untitled-ontology-
9;structured_text_resource"/>
        </owl:Class>
        <!--          http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#multimedia_resource -->

        <owl:Class rdf:about="&untitled-ontology-9;multimedia_resource">
        <rdfs:subClassOf rdf:resource="&untitled-ontology-9;informational_resource"/>
        </owl:Class>
        <!--          http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#non_structured_text_resource -->

        <owl:Class rdf:about="&untitled-ontology-9;non_structured_text_resource">
        <rdfs:subClassOf rdf:resource="&untitled-ontology-9;informational_resource"/>
        <rdfs:subClassOf rdf:resource="&untitled-ontology-9;text_resource"/>
        </owl:Class>
        <!--          http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#picture -->

        <owl:Class rdf:about="&untitled-ontology-9;picture">
        <rdfs:subClassOf rdf:resource="&untitled-ontology-9;multimedia_resource"/>
        </owl:Class>
        <!--  http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#rdf --
>

        <owl:Class rdf:about="&untitled-ontology-9;rdf">
```

```
        <rdfs:subClassOf                                    rdf:resource="&untitled-ontology-
9;structured_text_resource"/>
        </owl:Class>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#request -->

        <owl:Class rdf:about="&untitled-ontology-9;request"/>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#sound
-->

        <owl:Class rdf:about="&untitled-ontology-9;sound">
        <rdfs:subClassOf rdf:resource="&untitled-ontology-9;multimedia_resource"/>
        </owl:Class>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#structured_text_resource -->

        <owl:Class rdf:about="&untitled-ontology-9;structured_text_resource">
        <rdfs:subClassOf rdf:resource="&untitled-ontology-9;text_resource"/>
        </owl:Class>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#task_thesaurus -->

        <owl:Class rdf:about="&untitled-ontology-9;task_thesaurus"/>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#text_resource -->

        <owl:Class rdf:about="&untitled-ontology-9;text_resource">
        <rdfs:subClassOf rdf:resource="&untitled-ontology-9;informational_resource"/>
        <rdfs:subClassOf                                    rdf:resource="&untitled-ontology-
9;non_structured_text_resource"/>
        </owl:Class>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#theme -->

        <owl:Class rdf:about="&untitled-ontology-9;theme"/>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#user -
->

        <owl:Class rdf:about="&untitled-ontology-9;user"/>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#user_group -->

        <owl:Class rdf:about="&untitled-ontology-9;user_group"/>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#user_publication -->

        <owl:Class rdf:about="&untitled-ontology-9;user_publication">
```

```
        <rdfs:subClassOf                                rdf:resource="&untitled-ontology-
9;non_structured_text_resource"/>
        </owl:Class>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#user_site -->

        <owl:Class rdf:about="&untitled-ontology-9;user_site">
        <rdfs:subClassOf                                rdf:resource="&untitled-ontology-
9;non_structured_text_resource"/>
        </owl:Class>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#video
-->

        <owl:Class rdf:about="&untitled-ontology-9;video">
        <rdfs:subClassOf rdf:resource="&untitled-ontology-9;multimedia_resource"/>
        </owl:Class>
        <!--            http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#wiki_resorce -->

        <owl:Class rdf:about="&untitled-ontology-9;wiki_resorce">
        <rdfs:subClassOf                                rdf:resource="&untitled-ontology-
9;structured_text_resource"/>
        </owl:Class>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#xml -
->

        <owl:Class rdf:about="&untitled-ontology-9;xml">
        <rdfs:subClassOf                                rdf:resource="&untitled-ontology-
9;structured_text_resource"/>
        </owl:Class>
        <!--
        //
        // Individuals
        //
        -->

        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#Aaa -
->

        <owl:NamedIndividual rdf:about="&untitled-ontology-9;Aaa">
        <rdf:type rdf:resource="&untitled-ontology-9;user"/>
        </owl:NamedIndividual>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#Basia
-->

        <owl:NamedIndividual rdf:about="&untitled-ontology-9;Basia">
        <rdf:type rdf:resource="&untitled-ontology-9;user"/>
        <name>Hamster</name>
```

```
        </owl:NamedIndividual>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#Bbb -
->

        <owl:NamedIndividual rdf:about="&untitled-ontology-9;Bbb">
        <rdf:type rdf:resource="&untitled-ontology-9;user"/>
        </owl:NamedIndividual>
        <!--  http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-9#Jjj  --
>

        <owl:NamedIndividual rdf:about="&untitled-ontology-9;Jjj">
        <rdf:type rdf:resource="&untitled-ontology-9;user"/>
        <name>Julia</name>
        <interest_in rdf:resource="&untitled-ontology-9;ramka1.owl"/>
        </owl:NamedIndividual>
        <!--             http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#aaa.owl -->

        <owl:NamedIndividual rdf:about="&untitled-ontology-9;aaa.owl">
        <rdf:type rdf:resource="&untitled-ontology-9;domain_ontology"/>
        </owl:NamedIndividual>
        <!--             http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#example.owl -->

        <owl:NamedIndividual rdf:about="&untitled-ontology-9;example.owl">
        <rdf:type rdf:resource="&untitled-ontology-9;domain_ontology"/>
        <rdfs:comment>додаткова інформація про елемент онтології</rdfs:comment>
        <name>Ed_example</name>
        </owl:NamedIndividual>
        <!--             http://www.semanticweb.org/jjj/ontologies/2015/2/untitled-ontology-
9#ramka1.owl -->

        <owl:NamedIndividual rdf:about="&untitled-ontology-9;ramka1.owl">
        <rdf:type rdf:resource="&untitled-ontology-9;domain_ontology"/>
        <rdfs:comment>РїСЂРёРјРµСЂ                                    РґР»СЏ
РњРµР»РёС‚РsРїРsР»СЏ</rdfs:comment>
        </owl:NamedIndividual>
       </rdf:RDF>
       <!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net -->
```

# ДОДАТОК Б.
## Онтологічна модель аналізу компетенцій

Онтологічна модель порівняння компетенцій створена в редакторі онтологій Protégé й подана мовою OWL. Ця модель застосовується як приклад у наведених вище матеріалах.

Більш детальний опис елементів цієї моделі вміщено в розділі 8. Наведена нижче онтологія відображає класи та екземпляри класів. На рисунку Б. 1 презентована візуалізація цієї онтології в Protégé, зроблена за допомогою плагіна OntoGraf, який завдяки різним кольорам дає змогу відтворити різні типи онтологічних відношень і візуалізувати екземпляри класів.

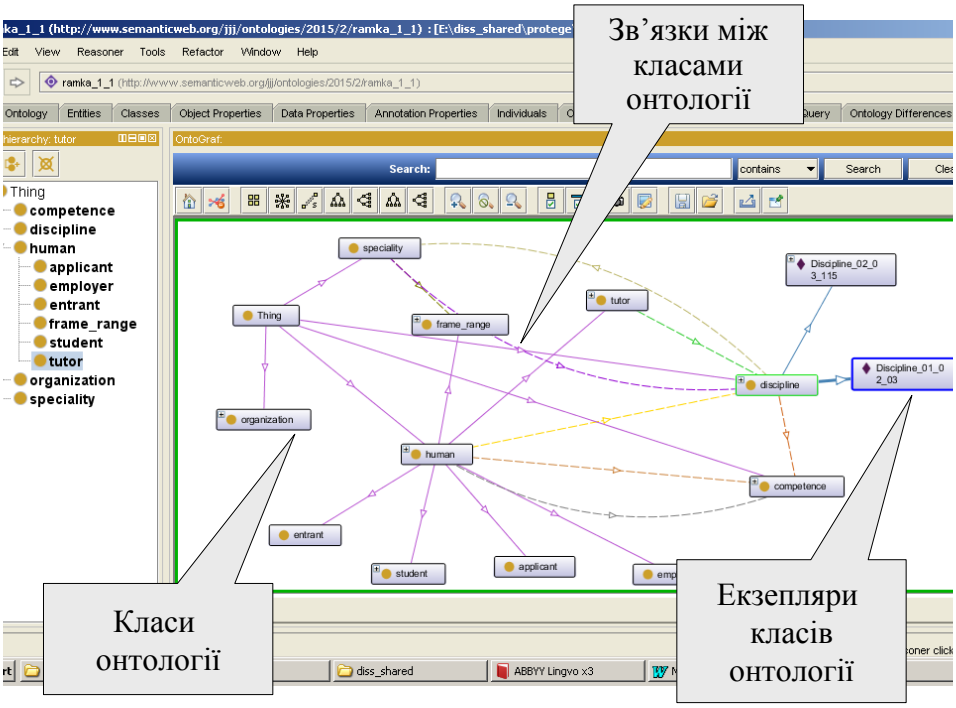



Рис. Б. 1. Візуалізація цієї онтології в Protégé, зроблена за допомогою плагіна OntoGraf

Нижче наведено опис цієї онтології мовою OWL.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
 <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
 <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
 <!ENTITY ramka_1_1
"http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#" >
```

```
]>


<rdf:RDF
xmlns="http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#"
     xml:base="http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:ramka_1_1="http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
     <owl:Ontology
rdf:about="http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1">
     <rdfs:isDefinedBy>РђPSC,PsP»PsPiPëCЏ, PsPїPëCЃC‹PIP°CЂC‰P°CЏ
CЃC,CЂCѓPєC,CѓCЂCѓ PїPsPSCЏC,PëP№ PrP»CЏ CЃCЂP°PIPSPµPSPëCЏ
PєPsPjPїPµC,PµPSC†PëP№ PrPëCЃC†PëPїP»PëPS</rdfs:isDefinedBy>
     </owl:Ontology>
     <!--
     //
     // Object Properties
     //
     -->
     <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#can_teach -->

     <owl:ObjectProperty rdf:about="&ramka_1_1;can_teach">
     <rdfs:range rdf:resource="&ramka_1_1;discipline"/>
     <rdfs:domain rdf:resource="&ramka_1_1;tutor"/>
     </owl:ObjectProperty>
     <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#consist_of_conpetences --
>

     <owl:ObjectProperty rdf:about="&ramka_1_1;consist_of_conpetences">
     <rdfs:domain rdf:resource="&ramka_1_1;discipline"/>
     <rdfs:range rdf:resource="&ramka_1_1;discipline_competence"/>
     </owl:ObjectProperty>

     <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#has_competence -->

     <owl:ObjectProperty rdf:about="&ramka_1_1;has_competence">
     <rdfs:range rdf:resource="&ramka_1_1;competence"/>
     <rdfs:domain rdf:resource="&ramka_1_1;human"/>
     </owl:ObjectProperty>
     <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#has_status -->
     <owl:ObjectProperty rdf:about="&ramka_1_1;has_status">
     <rdfs:range rdf:resource="&ramka_1_1;frame_range"/>
```

```
        <rdfs:domain rdf:resource="&ramka_1_1;speciality"/>
        </owl:ObjectProperty>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#include -->
        <owl:ObjectProperty rdf:about="&ramka_1_1;include">
        <rdfs:range rdf:resource="&ramka_1_1;discipline"/>
        <owl:inverseOf rdf:resource="&ramka_1_1;is_a_component"/>
        <rdfs:domain rdf:resource="&ramka_1_1;speciality"/>
        </owl:ObjectProperty>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#include_competence -->

        <owl:ObjectProperty rdf:about="&ramka_1_1;include_competence">
        <rdfs:range rdf:resource="&ramka_1_1;atom_competence"/>
        <rdfs:range rdf:resource="&ramka_1_1;competence"/>
        <rdfs:domain rdf:resource="&ramka_1_1;discipline"/>
        <rdfs:range rdf:resource="&ramka_1_1;discipline_competence"/>
        </owl:ObjectProperty>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#is_a_component -->

        <owl:ObjectProperty rdf:about="&ramka_1_1;is_a_component">
        <rdfs:domain rdf:resource="&ramka_1_1;discipline"/>
        <rdfs:range rdf:resource="&ramka_1_1;speciality"/>
        </owl:ObjectProperty>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#learn_discipline -->

        <owl:ObjectProperty rdf:about="&ramka_1_1;learn_discipline">
        <rdfs:range rdf:resource="&ramka_1_1;discipline"/>
        <rdfs:domain rdf:resource="&ramka_1_1;human"/>
        </owl:ObjectProperty>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#need_competence -->

        <owl:ObjectProperty rdf:about="&ramka_1_1;need_competence">
        <rdfs:range rdf:resource="&ramka_1_1;competence"/>
        <rdfs:domain rdf:resource="&ramka_1_1;human"/>
        </owl:ObjectProperty>
        <!--   http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#proposed_by
-->

        <owl:ObjectProperty rdf:about="&ramka_1_1;proposed_by">
        <rdfs:domain rdf:resource="&ramka_1_1;discipline"/>
        <rdfs:range rdf:resource="&ramka_1_1;learning_organization"/>
        </owl:ObjectProperty>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#ranges_to -->

        <owl:ObjectProperty rdf:about="&ramka_1_1;ranges_to">
```

<rdfs:range rdf:resource="&ramka_1_1;frame_range"/>
<rdfs:domain rdf:resource="&ramka_1_1;learning_organization"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#work_in -->
<owl:ObjectProperty rdf:about="&ramka_1_1;work_in">
<rdfs:range rdf:resource="&ramka_1_1;learning_organization"/>
<rdfs:domain rdf:resource="&ramka_1_1;tutor"/>
</owl:ObjectProperty>
<!--
//////////////////////
//
// Data properties
//
-->
<!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#discipline_code -->

<owl:DatatypeProperty rdf:about="&ramka_1_1;discipline_code">
<rdfs:domain rdf:resource="&ramka_1_1;discipline"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#discipline_name -->
<owl:DatatypeProperty rdf:about="&ramka_1_1;discipline_name">
<rdfs:domain rdf:resource="&ramka_1_1;discipline"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#frame_country -->
<owl:DatatypeProperty rdf:about="&ramka_1_1;frame_country">
<rdfs:domain rdf:resource="&ramka_1_1;frame_range"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#frame_name -
->
<owl:DatatypeProperty rdf:about="&ramka_1_1;frame_name">
<rdfs:domain rdf:resource="&ramka_1_1;frame_range"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#frame_number -->
<owl:DatatypeProperty rdf:about="&ramka_1_1;frame_number">
<rdfs:domain rdf:resource="&ramka_1_1;frame_range"/>
<rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>
<!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#has_Engl_name -->
<owl:DatatypeProperty rdf:about="&ramka_1_1;has_Engl_name">

```
        <rdfs:domain rdf:resource="&ramka_1_1;speciality"/>
        <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#has_UA_name -->
        <owl:DatatypeProperty rdf:about="&ramka_1_1;has_UA_name">
        <rdfs:domain rdf:resource="&ramka_1_1;speciality"/>
        <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#has_code -->
        <owl:DatatypeProperty rdf:about="&ramka_1_1;has_code">
        <rdfs:domain rdf:resource="&ramka_1_1;speciality"/>
        <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--  http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#human_ID  --
>
        <owl:DatatypeProperty rdf:about="&ramka_1_1;human_ID">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="&ramka_1_1;human"/>
        <rdfs:range rdf:resource="&xsd;integer"/>
        </owl:DatatypeProperty>
        <!--  http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#human_name
-->
        <owl:DatatypeProperty rdf:about="&ramka_1_1;human_name">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="&ramka_1_1;human"/>
        <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
        //
        // Classes
        //
        -->
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#applicant -->
        <owl:Class rdf:about="&ramka_1_1;applicant">
        <rdfs:subClassOf rdf:resource="&ramka_1_1;human"/>
        </owl:Class>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#atom_competence -->
        <owl:Class rdf:about="&ramka_1_1;atom_competence">
        <rdfs:subClassOf rdf:resource="&ramka_1_1;competence"/>
        </owl:Class>
        <!--  http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#competence  -
->
        <owl:Class rdf:about="&ramka_1_1;competence"/>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#discipline -->

        <owl:Class rdf:about="&ramka_1_1;discipline"/>
```

```
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#discipline_competence  --
>

        <owl:Class rdf:about="&ramka_1_1;discipline_competence">
        <rdfs:subClassOf rdf:resource="&ramka_1_1;competence"/>
        </owl:Class>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#employer -->
        <owl:Class rdf:about="&ramka_1_1;employer">
        <rdfs:subClassOf rdf:resource="&ramka_1_1;human"/>
        </owl:Class>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#entrant -->
        <owl:Class rdf:about="&ramka_1_1;entrant">
        <rdfs:subClassOf rdf:resource="&ramka_1_1;human"/>
        </owl:Class>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#frame_range -
->
        <owl:Class rdf:about="&ramka_1_1;frame_range"/>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#human -->
        <owl:Class rdf:about="&ramka_1_1;human"/>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#learning_organization -->

        <owl:Class rdf:about="&ramka_1_1;learning_organization">
        <rdfs:subClassOf rdf:resource="&ramka_1_1;organization"/>
        </owl:Class>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#organization -
->
        <owl:Class rdf:about="&ramka_1_1;organization"/>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#speciality -->
        <owl:Class rdf:about="&ramka_1_1;speciality"/>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#speciality_competence  --
>

        <owl:Class rdf:about="&ramka_1_1;speciality_competence">
        <rdfs:subClassOf rdf:resource="&ramka_1_1;competence"/>
        </owl:Class>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#student -->
        <owl:Class rdf:about="&ramka_1_1;student">
        <rdfs:subClassOf rdf:resource="&ramka_1_1;human"/>
        </owl:Class>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#tutor -->
        <owl:Class rdf:about="&ramka_1_1;tutor">
        <rdfs:subClassOf rdf:resource="&ramka_1_1;human"/>
        </owl:Class>
        <!--
        //
```

```
        // Individuals
        //
        -->
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#AAA -->
        <owl:NamedIndividual rdf:about="&ramka_1_1;AAA">
        <rdf:type rdf:resource="&ramka_1_1;human"/>
        </owl:NamedIndividual>
        <!-- http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#BBB -->
        <owl:NamedIndividual rdf:about="&ramka_1_1;BBB">
        <rdf:type rdf:resource="&ramka_1_1;student"/>
        <human_ID rdf:datatype="&xsd;integer">123467</human_ID>
        <has_competence rdf:resource="&ramka_1_1;Competence_1"/>
        <has_competence rdf:resource="&ramka_1_1;Competence_44"/>
        <learn_discipline rdf:resource="&ramka_1_1;Discipline_02_03_115"/>
        </owl:NamedIndividual>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#Competence_1 -->
        <owl:NamedIndividual rdf:about="&ramka_1_1;Competence_1">
        <rdf:type rdf:resource="&ramka_1_1;atom_competence"/>
        </owl:NamedIndividual>

        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#Competence_2 -->

        <owl:NamedIndividual rdf:about="&ramka_1_1;Competence_2">
        <rdf:type rdf:resource="&ramka_1_1;atom_competence"/>
        <human_ID rdf:datatype="&xsd;integer">123456</human_ID>
        <human_name rdf:datatype="&xsd;string">Bbbbb Bbbb Bbbb</human_name>
        <has_competence rdf:resource="&ramka_1_1;Competence_1"/>
        <has_competence rdf:resource="&ramka_1_1;Competence_44"/>
        </owl:NamedIndividual>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#Competence_44 -->

        <owl:NamedIndividual rdf:about="&ramka_1_1;Competence_44">
        <rdf:type rdf:resource="&ramka_1_1;atom_competence"/>
        </owl:NamedIndividual>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#Discipline_01_02_03 -->
        <owl:NamedIndividual rdf:about="&ramka_1_1;Discipline_01_02_03">
        <rdf:type rdf:resource="&ramka_1_1;discipline"/>
        </owl:NamedIndividual>
        <!--
http://www.semanticweb.org/jjj/ontologies/2015/2/ramka_1_1#Discipline_02_03_115  --
>
        <owl:NamedIndividual rdf:about="&ramka_1_1;Discipline_02_03_115">
        <rdf:type rdf:resource="&ramka_1_1;discipline"/>
        </owl:NamedIndividual>
```

```
 <!--
г
 //
 // General axioms
 //
 -->
 <rdf:Description>
 <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
 <owl:members rdf:parseType="Collection">
 <rdf:Description rdf:about="&ramka_1_1;competence"/>
 <rdf:Description rdf:about="&ramka_1_1;discipline"/>
 <rdf:Description rdf:about="&ramka_1_1;frame_range"/>
 <rdf:Description rdf:about="&ramka_1_1;human"/>
 <rdf:Description rdf:about="&ramka_1_1;speciality"/>
 </owl:members>
 </rdf:Description>
</rdf:RDF>
<!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net -->
```

## ПЕРЕЛІК СКОРОЧЕНЬ

| | |
|---|---|
| *ABox* | Assertional box |
| ALC | Attributive Language with Complements |
| CBR | Case Based Reasoning, |
| DAML | DARPA Agent Markup Language |
| DAI | Distributed Artificial Intelligence |
| DBMS | Database Management System, |
| DL | Description Logics |
| DM | Data Mining |
| DOM | Document Object Model |
| DPS | Distributed Problem Solver |
| FOL | First Order Logic |
| GGG | Giant global graph |
| IRI | Internationalized Resource Identifier |
| KDD | knowledge discovery in databases |
| KIF | Knowledge Interchange Format |
| KM | Knowledge Management |
| KR | Knowledge representation |
| MOF | Meta-Object Facility |
| OBDA | Ontology-Based Data Access Systems |
| OLAP | On-Line Analytical Processing |
| OS-INT | Open Source Intelligence |
| OSS | Open Source Software |
| OWL | Web Ontology Language |
| OWL-S | Web Ontology Language for Services |
| RDF | Resource Definition Framework |
| RDF(S) | RDF Schema |
| RIF | Rule Interchange Format |
| SMIL | Synchronized Multimedia Integration Language |
| SOAP | Simple Object Access Protocol |
| SWRL | Semantic Web Rule Language |
| *TBox* | Terminological box |
| UDDI | Universal Description, Discovery and Integration |
| UML | Unified Modeling Language |
| URN | Uniform Resource Name |
| XML | Xtensible Markup Language |
| АОП | Агентно-орієнтоване програмування |
| БД | База даних |
| БЗ | База знань |
| ВіЙм | Відношення йменування |

| | |
|---|---|
| ЖЦ | Життєвий цикл |
| ІАД | Інтелектуальний аналіз даних |
| ІАОС | Інтелектуальні адаптивні освітні системи |
| ІІС | Інтелектуальна інформаційна система |
| ІКТ | Інформаційно-комунікаційні технології |
| ІО | Інформаційний об’єкт |
| ІПА | Інтелектуальний програмний агент |
| ІПС | Інформаційно-пошукові системи |
| ІР | Інформаційний ресурс |
| ІС | Інформаційна система |
| ІТ | Інформаційна технологія |
| МАС | Мультиагентна система |
| МЗ | Менеджмент знань |
| МПЗ | Модель подання знань |
| ОУР | Особа, що ухвалює рішення |
| ПА | Програмний агент |
| ПЗ | Програмне забезпечення |
| ПМ | Природна мова |
| ПрО | Предметна область |
| ПС | Пойменована сутність |
| РЕ | Рекомендований елемент |
| РС | Рекомендувальна система |
| СБЗ | Система, що базується на знаннях |
| СКБД | Система керування базою даних |
| СКЗ | Система керування знаннями |
| СОА | Сервіс-орієнтована архітектура |
| ШІ | Штучний інтелект |

# ГЛОСАРІЙ

## ЛІТЕРАТУРА


1. Алыгулиев Р.М. Роль технологии интеллектуального анализа текстов в обеспечении национальной безопасности // İnformasiya Texnologiyaları Problemləri, 2013, No 1(7).
2. Андон Ф.И., Яшунин А.Е., Резниченко В.А. Логические модели интеллектуальных информационных систем. – Киев: Наукова думка, 1999.
3. Андрагогика: принципи практичного навчання для дорослих. – http://toplutsk.com/articles-article_398.html.
4. Андриченко А.Н. Тенденции и состояние в области управления справочными данными в машиностроении // Онтология проектирования, 2012, №2. – С. 25-35.
5. Армстронг М. Практика управления человеческими ресурсами. — 8-е изд. / Пер. с англ. под ред. С. К. Мордовина. — СПб.: Питер, 2008. — 832 с.
6. Барсегян А.А. Технологии анализа данных: Data Mining, Visual Mining, Text Mining,. OLAP / А. А. Барсегян, М. С. Куприянов, В. В. Степаненко,.И.И.Холод. – СпБ: БХВ-Петербург, 2007. – 384 с.
7. Бахрушин В.Є. Методи аналізу даних: навчальний посібник для студентів, Запоріжжя: КПУ, 2011.
8. Бениаминов Е. М. , Болдина Д. М. Система представления знаний Ontolingua – принципы и перспективы. 22 с. – http://www.masters.dgtu.donetsk.ua/2010/fknt/bolotova/library/stanford.pdf.
9. Бизнес-анализ на базе IBM Cognos BI. – http://www.columbusglobal.com/ ru-RU/Shared/Technology/ RU_technonology/IBM-Cognos.
10. Биков В.Ю. Моделі організаційних систем відкритої освіти: Монографія / В.Ю. Биков. – К.: Атіка, 2008. – 684 с.
11. Богачков Ю.М. Системоутворювальні фактори системи дистанційного навчання в загальноосвітніх навчальних закладах: огляд [Електронний ресурс] / Ю.М. Богачков // Інформаційні технології і засоби навчання. – 2011. – №6(26). – Режим доступу до журналу: http://www.journal.iitta.gov.ua.
12. Болонский процесс: европейские и национальные структуры квалификаций (Книга-приложение 2) / Под науч. ред. д-ра пед. наук, профессора В.И. Байденко. – М.:Исследовательский центр проблем качества подготовки специалистов , 2009. – 220 с.
13. Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие –

М.: МИЭМ, 2011. – 272 с.
14. Боргест Н.М., Коровин М.Д. Онтологии: современное состояние, стандарты, средства поддержки. Уч. пособие. СРАУ.- Самара, 2013. – 84 с.
15. Боргест Н.М. , Коровин М.Д. Онтологии: современное состояние, краткий обзор. // Онтология проектирования, №2 (8), 2013. – С.49-55.
16. Боргест Н.М. Роль онтологии в проектировании информационных систем // Материалы IV международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» OSTIS-2014, Минск, БГУИР, 2014. – С.151-160.
17. Бояцис Р. Компетентный менеджер. Модель эффективной работы. / Пер. с англ.— М.: HIPPO, 2008. — 352 с.
18. Браславский П. И., Гольдштейн С. Л., Ткаченко Т. Я. Тезаурус как средство описания систем знаний // Информационные процессы и системы, 1997, № 11, Серия 2. – С.16-22.
19. Васильев С.Н. От классических задач регулирования к интеллектуальному управлению // Известия Академии наук. Теория и системы управления. – 2001, № 2. – С.5-21.
20. Величко В. Автоматизированное создание тезауруса терминов предметной области для локальных поисковых систем [Электронный ресурс] / Величко В., Волошин П., Свитла С. – Режим доступа : – www.foibg.com/ibs_isc/ibs-15/ibs-15.pdf
21. Гаврилова Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. – СПб. : Питер, 2001.
22. Гаврилова Т. А. Интеллектуальные технологии в менеджменте: инструменты и системы: учебное пособие / Т. А. Гаврилова, Д. М. Муромцев. – СПб. : Изд-во «Высшая школа менеджмента»; Издат. дом С.-Петерб. гос. ун-та, 2007. – 488 с.
23. Гершензон Л. М. Система извлечения и поиска структурированной информации из больших текстовых массивов СМИ. Архитектурные и лингвистические особенности / Гершензон Л. М., Ножов И. М., Панкратов Д. В. // Труды международной конференции Диалог’2005 «Компьютерная лингвистика и интеллектуальные технологии». – М. : Наука, 2005. – С. 97–101
24. Гладун А. Я. Архитектурная концепция интеллектуальных сетей / А. Я. Гладун, В. Л. Плескач // Искусственный интеллект. — 1999. – – № 2 (спец. выпуск). — Крым-Кацивелли. — С. 413—421.
25. Гладун А. Я. Використання організаційних онтологій для пошуку експертів у нових предметних областях / А. Я. Гладун, Ю. В. Рогушина // Проблеми програмування. — 2007. — № 1. — С. 73—84.

26. Гладун А. Я. Интеллектуальные поисковые системы в контексте технологий Semantic Web / А. Я. Гладун, Ю. В. Рогушина // Сборник трудов VIII Международной конференции «Интеллектуальный анализ информации», ИАИ-2008. — Киев, 2008. — С. 388—398.
27. Гладун А. Я. Использование агентно-ориентированных технологий в телекоммуникационных сетях / А. Я. Гладун, В. Л. Плескач // Проблеммы программирования. — 2000. — № 1. — С. 43—59.
28. Гладун А. Я. Использование инфраструктуры Semantic Web для исследования функционирования возможностей Web-сервисов / А. Я. Гладун, Ю. В. Рогушина // XVII науково-практична конференція «Інформаційні технології в економіці, менеджменті та бізнесі. Проблеми науки, практики і освіти». — Київ, 2011. — С. 206—208.
29. Гладун А. Я. Использование Semantic Web для интеллектуализации GRID-инфраструктур / А. Я. Гладун, Ю. В. Рогушина // Тези доповідей Міжнародної наукової конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку». — Киев, 2013. — С. 186—188.
30. Гладун А. Я. Использование тезауруса предметной области как инструмента представления знаний при повышении эффективности проблемно-ориентированного поиска в Web [Электронный ресурс] / А. Я. Гладун, Ю. В. Рогушина // Искусственный интеллект. — 2010. — № 3. — С. 462—472. — Режим доступа : http://dspace.nbuv.gov.ua/bitstream/handle/ 123456789/ 56565/55-Gladun.pdf?sequence=1.
31. Гладун А. Я. Использование технологии Semantic Web для интеллектуального управления в динамических распределенных системах / А. Я. Гладун, Ю. В. Рогушина // International Book Series «Information Sciences and Computing». — Varna, Bulgarien, 2009. — P. 143—153.
32. Гладун А. Я. Использование технологии Semantic Web для управления знаниями в системах Business Intelligence / Гладун А. Я., Рогушина Ю. В., Петрухина Л. В. // Искусственный интеллект. — 2010. — № 2. — С. 15—23.
33. Гладун А. Я. Інтеграція технологій Semantic Web з системами Business Intelligence 2. 0 / А. Я. Гладун, Ю. В. Рогушина // Проблеми програмування. — 2010. — № 1. — С. 79—87.
34. Гладун А. Я. Інтелектуальні мережі. Навчальний посібник / А. Я. Гладун, О. М. Пєчкурова. — К. : НаУКМА – Видавничий дім «Академія», 2005. — 221 с.
35. Гладун А. Я. Інтелектуальні мережі та агентно-орієнтовані

технології / А. Я. Гладун // Наукові записки НАУКМА, Києво-Могилянська Академія. — 1999. — Том 11. — С. 234—242.
36. Гладун А. Я. Комп'ютерна програма «Інтелектуальна мультиагентна для електронної комерції "MIMCO"» / А. Я. Гладун, М. В. Несен // Свідоцтво про реєстрацію авторського права на твір № 23407.
37. Гладун А. Я. Моделювання комп'ютерних мереж на основі інструментально-програмних засобів. Навчальний посібник / А. Я. Гладун. — К. : Видавничий дім «Європейський університет», 2006. — 146 с.
38. Гладун А. Я. Мультиагентна система для інформаційної системи кадрового агентства корпорації / А. Я. Гладун, В. С. Білан // Наукові записки. Сер. Комп'ютерні науки. — К. : НаУКМА, 2004. – – Том 28. — С. 116—112.
39. Гладун А. Я. Мультиагентні системи для електронної комерції / А. Я. Гладун, С. В. Даниленко // Збірник наукових праць IX Міжнародної науково-практичної конференції «Інформаційні технології в менеджменті та бізнесі: проблеми науки, практики і освіти», Київ: ЄУФІМБ, 2004. – С.98-112.
40. Гладун А. Я. Онтологии в корпоративных системах. Часть 1. / А. Я. Гладун, Ю. В. Рогушина // Корпоративные системы. — 2005. — № 6. — С. 23
41. Гладун А. Я. Онтологии в корпоративных системах. Часть 2. / А. Я. Гладун, Ю. В. Рогушина // Корпоративные системы. — 2006. — № 1. — С. 23
42. Гладун А. Я. Онтологии и мультилингвистические тезаурусы как основа семантического поиска информационных ресурсов Интернет / А. Я. Гладун, Ю. В. Рогушина // Proc. of XII-th Intern. Conf. KDS'2006, Varna, Bulgaria. — P. 115– 121.
43. Гладун А. Я. Онтологический анализ Web-сервисов в интеллектуальных сетях / Гладун А. Я., Рогушина Ю. В., Штонда В. М. // Proc. of The XIII-th International Conf. «Knowledge-Dialogue-Solution», ITHEA, Sofia, 2007. — V. 2. — С. 451—459.
44. Гладун А. Я. Онтологический подход к поиску веб-сервисов в распределенной среде Интернет / А. Я. Гладун, Ю. В. Рогушина // Информатика. – 2006. — № 4. — Минск. — С. 116—127.
45. Гладун А. Я. Організаційні онтології та управління знаннями у системах підтримки прийняття рішень // Міжнародна наукова конференція «Інтелектуальні системи прийняття рішень та прикладні аспекти інформаційних технологій» ISDMIT`2006: зб.наук. праць. — Херсон : Видавництво Херсонського морського інституту, 2006. — Т. 1. — С. 363—366.

46. Гладун А. Я. Основи методології формування тезаурусів з використанням онтологічного та мереологічного аналізу / А. Я. Гладун, Ю. В. Рогушина // Искусственный интеллект. — 2008. — № 5. — С. 112—124.
47. Гладун А. Я. Перспективные направления и тенденции интеллектуализации grid-инфраструктур / А. Я. Гладун, Ю. В. Рогушина // Зб. наук. праць XVI Міжнар. наук.-практ. конф. «Інформаційні технології в економіці, менеджменті і бізнесі — К. : Видавничий дім «Європейський університет», 2010.— Т. 1. — С. 128—133.
48. Гладун А. Я. Применение технологий Semantic Web в телемедицине / А. Я. Гладун, Ю. В. Рогушина // Труды 2-й Международной конференции «Продвинутые информационные технологии и технологии по телемедицине», AITTH-2008. — Минск, Белорусь, 2008. — Том 2. — С. 76—80.
49. Гладун А. Я. Проблеми управління знаннями у середовищі Semantic Web: перспективи та технології / А. Я. Гладун, Ю. В. Рогушина // Матеріали XV Міжнар. наук.-практ. конференції «Інформаційні технології в економіці, менеджменті і бізнесі. Проблеми науки, практики і освіти». — К. : Вид-во Європейського університету, 2010. — Т. 1. — С. 129—131.
50. Гладун А. Я. Реалізація мультиагентної системи для створення інтелектуальних сервісів в розподілених інформаційних системах / Гладун А. Я., Журавльов Ю. Д., Несен М. В. // Збірник наукових праць міжнародної наукової конференції «Інформаційна інфраструктура рекреаційно-туристичної галузі». — Трускавець, 2003. — С. 124—131.
51. Гладун А. Я. Репозитории онтологий как средство повторного использования знаний для распознавания информационных объектов / А. Я. Гладун, Ю. В. Рогушина // Онтология проектирования. — 2013. — № 1(7). — С. 35—50.
52. Гладун А.Я., Семантичні технології: принципи та практики / А.Я. Гладун, Ю.В. Рогушина – К.:ТОВ "ВД "АДЕФ-Україна", 2016. – 308 с
53. Гладун А. Я. Формирование и применение онтологий предметных областей для поиска Web-сервисов на семантическом уровне / А. Я. Гладун, Ю. В. Рогушина // Труды Всероссийской конференции с международным участием «Знания-Онтологии-Теории», 2007. — Т. 2. — С. 176–185.
54. Гладун А. Я. Формирование тезауруса предметной области как средства моделирования информационных потребностей пользователя при поиске в Интернете / А. Я. Гладун,

Ю. В. Рогушина // Вестник компьютерных и информационных технологий. — Москва, 2007. — № 1.
55. Гладун В. Структурирование онтологии ассоциаций для конспектирования естественно-языковых текстов [Электронный ресурс] / Гладун В., Величко В., Святогор Л. — Режим доступа : http://www.foibg.com/ibs_isc/ibs-02/IBS-02-p20.pdf.
56. Гладун В. П. Процессы формирования новых знаний / В. П. Гладун. — София, 1994. — 189 с.
57. Глибовец А. Н. Семантическая паутина и Wiki-системы / Глибовец А. Н., Глибовец Н. Н., Покопцев Д. Е., Сидоренко М. О. // Проблеми програмування. – 2013. – № 1. – С. 45–67.
58. Глибовець М. М. Штучний інтелект / М. М. Глибовець, О. В. Олецький. – К. : Вид.дім «КМ Академія». – 2002. – 366 с.
59. Голенков В. В. Семантическая технология компонентного проектирования систем, управляемых знаниями / В. В. Голенков, Н. А. Гулякина // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем». – Минск, 2015. – С. 57–78.
60. Городецкий В. И. Многоагентные системы (обзор) [Электронный ресурс] / Городецкий В. И., Грушинский М. С., Хабалов А. В. – Режим доступа : http://www.raai.org/library/ainews/1998/2/GGKHMAS.ZIP.
61. Григорьев С. И. Основы современной социологии [Электронный ресурс] / С. И. Григорьев, Ю. Е. Растов. – Режим доступа : http://irbis.asu.ru/mmc/grig/.
62. Гриценко В. И. Агентно-ориентированные WEB-технологии в интеллектуальных сетях: новые области реализации интеллектуальных услуг / В. И. Гриценко, А. Я. Гладун // Труды междунар. научн.-практ. конф. «Современные и будущие информационные технологии Украины». – К. : УкрНИИсвязи, 15-17 марта 2000 г. – С. 63–68.
63. Гриценко В. И. Агентно-ориентированные системы для компьютерной поддержки принятия решений / В. И. Гриценко, А. Я. Гладун // Материалы XI Междунар. конф. По автоматическому управлению «Автоматика – 2004». – Киев, 2004. – С. 56–60.
64. Гриценко В.И. Модель мультиагентной системы для электронного бизнеса и технология ее программной реализации / Гриценко В. И., Гладун А. Я, Несен М. В., Журавлев Ю. Д. // Проблемы программирования. – 2004. – № 2-3. – С. 510–520.
65. Гриценко В. И. Семантическое распознавание информационных

объектов на основе онтологического представления знаний о предметной области в задачах интелектуального управления / Гриценко В. И., Гладун А. Я., Рогушина Ю. В. // Кибернетика и вычислительная техника. – 2014. – № 4. – Вып. 178. – С. 5–20.
66. Гришанова И. Ю. Использование методов онтологического анализа для управления компетенциями персонала как составной части планирования научных исследований / И. Ю. Гришанова, Ю. В. Рогушина // X международная конференция им. Т. А. Таран «Интеллектуальный анализ информации ИАИ-2010» : сб.научн. трудов. – К., 2010. – С. 62–73.
67. Гришанова И. Ю. Средства интеллектуализации поиска информационных ресурсов в сети Интернет / И. Ю. Гришанова, Ю. В. Рогушина // Тезисы VI Международной конференции «Интеллектуальный анализ информации ИАИ-2007» : сб.научн. трудов. – К., 2007. – С. 279–289.
68. Гришанова І. Ю. Комп'ютерна програма «Мультиагентна інформаційно-пошукова система «МАІПС»(«МАІПС») / І. Ю. Гришанова, Ю. В. Рогушина. – Свідоцтво про реєстрацію авторського права на твір № №32015.
69. Дерецкий В. А. Об одном подходе к обработке естественно-языковых данных на основе анализа семантических сетей [Електронний ресурс] / В. А. Дерецкий // Перша міжнар. наук.-практ. конф. з програмування УкрПРОГ'98 / НАН України, 1998. – С. 405–411. – Режим доступу : http:// www.dl99.nw.ru/PDF/14.pdf.
70. Джанетто К. Управление знаниями: Руководство по разработке и внедрению корпоративной системы управления знаниями [Электронный ресурс] / К. Джанетто, Э. Уилер. – М. : Добрая книга, 2005. – 192 с. – Режим доступа : http://bulletinsite.net/index. php?id1= 6&category= business&author=djannetto-k&book=2005.
71. Джексон П. Введение в экспертные системы / П. Джексон. – М. : Издательский дом «Вильямс», 2001.
72. Дмитриев И. Контент-анализ: сущность, задачи, процедуры [Электронный ресурс] / И. Дмитриев. – Режим доступа : http://www. psyfactor. org/lib/ka. htm.
73. Добров Б. В. Онтологии и тезаурусы: модели, инструменты, приложения [Электронный ресурс] / Добров Б. В., Иванов В. В., Лукашевич Н. В., Соловьев В. Д. – Электронная книга, 2006. – 220 с. – Режим доступа : http:// catscpp.googlecode.com/svnhistory/ r146/trunk/diploma/materials/ontologies_tesaurusespdf.
74. Додонов А. Г. Конкурентная разведка в компьютерных сетях / Додонов А. Г., Ландэ Д. В., Прищепа В. В., Путятин В. Г. – К. : ИПРИ НАН Украины, 2013. – 248 с.

75. Додонов О. Г. Інформаційні потоки в глобальних комп'ютерних мережах / Додонов О. Г., Ланде Д. В., Путятін В. Г. – К. : Наукова думка, 2009. – 295 с.
76. Дюк В. Data Mining. Учебный курс / В. Дюк, А. Самойленко. – СПб. : Питер, 2001. – 386 с.
77. Загорулько Ю. А. Подход к интеллектуализации документооборота / Загорулько Ю. А., Кононенко И. С., Сидорова Е. А., Костов Ю. В. // Информационные технологии. – 2004. – № 11. – С. 2–11.
78. Загорулько Ю. А. Технологии разработки интеллектуальных систем, основанные на интегрированной модели представления знаний // Материалы III Междунар. научн.-техн. конф. OSTIS-2013 (Минск, 21–23 февраля 2013 г.). – Минск : БГУИР, 2013. – С. 31–42.
79. Захарова О. В. Використання дескриптивних логік в проблематиці WEB-сервісів / О. В. Захарова // Проблеми програмування. – 2015. – № 1. – С. 38–50.
80. Захарова О. В. Підходи до вирішення проблеми ефективності виявлення семантичних веб-сервісів на функціональному рівні [Електронний ресурс] / О. В. Захарова // Інженерія програмного забезпечення. – 2014. – № 4(20). – С. 38–44. – Режим доступу : http://jrnl.nau.edu.ua/ index.php/ IPZ/ article/download/7625/8760.
81. Золин Е. Дескрипционная логика [Электронный ресурс] / Е. Золин. – Режим доступа : http://lpcs.math.msu.su/~zolin/dl/.
82. Ивашенко В. П. Семантические модели и средства интеграции и отладки баз знаний / В. П. Ивашенко // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем». – Минск, 2012. – С. 193–204.
83. Іщенко А. Відкрита освіта на сучасному етапі: сутність, ідеї та головні тенденції (Передмова до українського видання) / А. Іщенко // Відкрита освіта: колективний розвиток освіти через відкриті технології, відкритий контент і відкрите знання / за редакцією Тору Іійосі та М. С. Віджая Кумара; пер. з англ. А. Іщенко, О. Носика. — К. : Наука, 2009. — С. 7–13.
84. Катков Ю. В. Мастер-класс по SPARQL [Электронный ресурс] / Ю. В. Катков. – Режим доступа : http://test.wikivote.ru/index.php/Мастер-класс_по_SPARQL.
85. Келеберда И. Н. Использование мультиагентного онтологического подхода к созданию распределенных систем дистанционного обучения / Келеберда И. Н., Лесная Н. С., Репка В. Б. // Educational Technology & Societe. – 2004. – № 7(2). – С. 190–205.
86. Клещев А. С. Математические модели онтологий предметных

областей. Часть 1.: Существующие подходы к определению понятия «онтология» / А. С. Клещев, И. Л. Артемьева // Научно-техническая информация. – 2001. – Серия 2. – С. 20–27.
87. Клещев А. С. Отношения между онтологиями предметных областей. Ч. 1.: Онтологии, представляющие одну и ту же концептуализацию. Упрощение онтологии / А. С. Клещев, И. Л. Артемьева // Информационный анализ. – 2002. – В. 1. – Серия 2. – С. 4–9.
88. Клещев А.С. Математические модели онтологий предметных областей. Часть 3. Сравнение разных классов моделей онтологий [Электронный ресурс] / А. С. Клещев, И. Л. Артемьева. – Режим доступа : https://www.google. com.ua/ url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved =0CCAQFjAA&url=http%3A%2F%2Fwww.iacp.dvo.ru%2Fes%2Fpubl %2F104_3.rtf&ei=i5tkVfTmOKTvywPH94DoDw&usg=AFQjCNH_Hr ORBfpzGglNArtQBnULxEnE_Q&sig2=fm-oTIshJEn0qESfJn7KwA.
89. Когаловский М. Р. Системы доступа к данным, основанные на онтологиях / М. Р. Когаловский // Программирование. – 2012. – № 4. – С. 55– 77.
90. Коммюнике онтологического саммита 2013: оценка онтологии в течение всего жизненного цикла / гл. ред. Неухаус Ф., Вайздом А. // Онтология проектирования. – 2013. – № 2(8). – С. 66–74.
91. Кононенко И. С. Обработка делового письма в системе документооборота / И. С. Кононенко, Е. А. Сидорова // Труды международного семинара Диалог’2002 по компьютерной лингвистике и ее приложениям. – М. : Наука, 2002. – Т. 2. – С. 299–310.
92. Котельникова И. А. Базы данных на основе Microsoft SQL Server 2008 / И. А. Котельникова. – Специальные книги, 2009. – 245 с.
93. Крюков К. В. О понятии формальной компетентности научных сотрудников [Электронный ресурс] / Крюков К. В., Кузнецов О. П., Суховеров В. С. // Открытые семантические технологии проектирвоания интеллектуальных систем» OSTIS-2014: материалы III международной научно-технической конференции. – Минск : БГУИР, 2013. – С. 143–146. – Режим доступа : http://libeldoc.bsuir.by/bitstream/123456789/4248/ 1/О% 20понятии% 20формальной%20компетентности.PDF.
94. Кудін А. Реалізація в Україні принципів і завдань Болонського процесу: забезпечення мобільності громадян з можливістю їх працевлаштування / А. Кудін // Вища школа. – 2006. – № 1. – С. 27–33.

95. Кудрявцев Д. В. Практические методы отображения и интеграции онтологий [Электронный ресурс] / Д. В. Кудрявцев // Семинар Знания и онтологии Elsewhere, КИИ-2008, Дубна, 2008. – Режим доступа : http://raai.org/ conference/cai-08/files/cai-08_paper_226.doc.
96. Кузьменко Г. Є. Прагматичний підхід до оцінки рівня інтелекту інтелектуалізованих систем / Г. Є. Кузьменко, В. А. Литвинов // Математичні машини і системи. – 2003. – № 1. – С. 3–9.
97. Кузьмина К. И. Семейная медицина сегодня и проблема ее дальнейшей интеллектуализации с помощью информационных технологий и компьютерных систем / Кузьмина К. И., Оноприенко В. Н., Козак Н. С., Семик Т. М., Андон Т. А. // Теорія і практика управління соціальними системами. – 2012. – № 2. – С. 56–67.
98. Кутукова Е. С. Технологія Text Mining [Электронный ресурс] / Е. С. Кутукова. – Режим доступа : http://www.sworld.com.ua/simpoz3/3.pdf
99. Ландэ Д. В. Поиск знаний в Internet. Профессиональная работа / Д. В. Ландэ. – М. : Изд. дом Вильямс, 2005. – 272 с.
100. Лапшин В. А. Онтологии в компьютерных системах [Электронный ресурс] / В. А. Лапшин. – Режим доступа : http://www.rsdn.ru/ article/philosophy/ what-is-onto.xml.
101. Лапшин В. А. Система Cyc и ее библиотека онтологий [Электронный ресурс] / В. А. Лапшин // Искусственный интеллект и принятие решений. – 2010. – № 2. – Режим доступа : http://www.isa.ru/aidt/images/documents/2010-02/ 42_53.pdf.
102. Лесько О. Н. Автоматизация семантической разметки естественно-языковых текстов / О. Н. Лесько, Ю. В. Рогушина // Интеллектуальный анализ информации ИАИ-2009: сборник научных трудов IX международной конференции имени Т. А. Таран. – Киев, 2009. – С. 247–253.
103. Лесько О. Н. Использование онтологий для анализа семантики естественно-языковых текстов / О. Н. Лесько, Ю. В. Рогушина // Проблемы программирования. – 2009. – № 3. – С. 59–65.
104. Лесько О. Н. Использование специализированной лексической онтологии для автоматизации формирования онтологии предметной области по естественно-языковым текстам / О. Н. Лесько, Ю. В. Рогушина. – Information Models of Knowledge. Edited by K. Markov, V. Velychko, O. Voloshin. – IT H E A, Kiev-Sofia, 2010. – P. 93–100.
105. Лесько О. Н. Формирование онтологических знаний о предметной области для повышения пертинентности семантического поиска / О. Н. Лесько, Ю. В. Рогушина // Интеллектуальный анализ

информации ИАИ-2011: сборник научных трудов XI международной конференции им. Т. А. Таран. – Киев, 2011. – С. 39–45.
106. Луговий В. Концептуальні засади розроблення національної рамки кваліфікацій / В. Луговий // Вища школа : Науково-практичне видання. – 2010. – № 9. – С. 15–24.
107. Лукашевич Н. В. Проектирование лингвистических онтологий для информационных систем в широких предметных областях / Н. В. Лукашевич, Б. В. Добров // Онтология проектирования. – 2015. – Т. 5. – № 1(15). – С. 47–69.
108. Лукашевич Н. В. Тезаурусы в задачах информационного поиска / Н. В. Лукашевич. – М. : Изд-во Московского университета, 2011. – 512 с.
109. Любич А. А. О выборе критериев оценки интеллектуальности информационной системы / Любич А. А., Плескач В. Л., Рогушина Ю. В. // УсиМ. – 2005. – № 1. – С. 3–7.
110. Люгер Д. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем / Д. Ф. Люггер; пер. с англ. – [4-е издание]. – К. : Издательский дом «Вильямс», 2003. – 864 с.
111. Мангалова Е. С. О проблеме выделения информативных признаков в задаче классификации текстовых документов / Е. С. Мангалова, Е. Д. Агафонов // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – 2013. – Выпуск № 1(22). – С. 190– 212.
112. Маннинг К. Д. Введение в информационный поиск / К. Д. Маннинг, П. Рагхаван, Х. Шютце. – М. : Вильямс, 2011. – 528 с.
113. Марон А. Е. Концепция развития открытых систем образования взрослых / А. Е. Марон, Л. Ю. Монахова // Человек и образование. — 2008. – № 1(14). — С. 75–82.
114. Методология оценки эффективности вузовской науки. Практическое пособие. Выпуск 3. Часть первая / под ред. проф., д. э. н. Э. Н. Яковлева. – М. : Государственное научное учреждение «Экспертно-аналитический центр Минобразования РФ», 2002. – 20 с.
115. Мигас С. С. Интеллектуальные информационные системы: конспект лекций / С. С. Мигас. – СПб : СПбГИЭУ, 2009. – 160 с.
116. Муравьева А. А. Принципы и процедуры разработки национальной рамки квалификаций / Муравьева А. А., Олейникова О. Н., Коулз М. – М. : Центр изучения проблем профессионального образования, 2006. – 160 с.
117. Нариньяни А. С. Кентавр по имени ТЕОН: Тезаурус + Онтология [Электронный ресурс] / А. С. Нариньяни. – Режим доступа :

http://www.artint.ru/ articles/ narin/ teon.htm.
118. Никоненко А. А. Обзор баз знаний онтологического типа [Электронный ресурс] / А. А. Никоненко // Штучний інтелект. – 2009. –№ 4. – С. 208–219. – Режим доступа : http://dspace.nbuv.gov.ua/ bitstream/ handle/ 123456789/8144/27-Nikonenko.pdf.
119. О формальных основах OWL [Электронный ресурс]. – Режим досупа : http:// shcherbak.net/2009/03/o-formalnyx-osnovax-owl.
120. Овдей О. М. Обзор инструментов инженерии онтологий [Электронный ресурс] / О. М. Овдей, Г. Ю. Проскудина // Труды Шестой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». – 2004. – Т. 7. – Вып. 4. – С. 3–19. – Режим доступа : http://www.elbib.ru/ index.phtml? page= elbib/rus/journal/2004/part4/op.
121. Овдій О. М. Онтології у контексті інтеграції інформації: представлення, методи та інструменти побудови [Електронний ресурс] / О. М. Овдій, Г. Ю. Проскудіна // Проблеми програмування. – 2004. – № 2-3. – С. 353–365. – Режим доступу : http://dspace.nbuv.gov.ua/ bitstream/ handle/ 123456789/ 1683/48%20-%20Ovdiy.pdf.
122. Огієнко О. І. Інформаційні технології як засіб адаптивного навчання дорослих [Електронний ресурс] / О. І. Огієнко // Інформаційні технології і засоби навчання. – 2010. – № 6(20). – Режим доступу : http://www.ime.edu-ua.net/em.html.
123. Осадчий В. В. Використання технологій Semantic Web для підвищення адаптивності інтелектуальних навчальних систем відкритої освіти дорослих / Осадчий В. В., Рогушина Ю. В., Прийма С. М. // Інженерія програмного забезпечення. – 2015. – № 1(21). – С. 37–47.
124. Павлишенко Б. М. Групування текстових даних на основі моделі семантичного контексту / Б. М. Павлишенко // Восточно-Европейский журнал передовых технологий. – 2011. – Выпуск 2(53). – Том 5. – С. 234–247.
125. Плескач В. Л. Агентні технології / В. Л. Плескач, Ю. В. Рогушина. – К. : Київ. нац. торг.-екон. ун-т, 2005. – 338 с.
126. Плескач В. Л. Інтелектуалізація інформаційних ресурсів як базовий напрямок сучасного етапу розвитку ринку ІТ Ринок технологій: проблеми та шляхи вирішення: тези доповіді / В. Л. Плескач, Ю. В. Рогушина. – К. : УкрІНТЕІ, 2007. – С. 105–111.
127. Плескач В. Л. Інформаційні технології та системи: навч.посібник / Плескач В. Л., Рогушина Ю. В., Кустова Н. П. – К. : КНТЕУ, 2003.

128. Плескач В. Л. Управління знаннями у web-середовищі / В. Л. Плескач, Ю. В. Рогушина // Теоретичні та прикладні аспекти побудови програмних систем TAAPSD'2008: збірник тез доповідей Міжнародної конференції. – Київ, 2008. – С. 100– 109.
129. Прийма С. М. Забезпечення прозорості європейської і національних Рамок кваліфікацій за допомогою комп'ютерних онтологій [Електронний ресурс] / С. М. Прийма, О. В. Панін // Інформаційні технології і засоби навчання. — 2013. — Том 33. — № 1. — Режим доступу : http:// journal.iitta. gov.ua/index.php/itlt/article/view/789/588.
130. Проект Концепції розвитку національної системи кваліфікацій (станом на 16.10.2012 р.) [Електронний ресурс]. — Режим доступу : http://www.ihed.org.ua/images/pdf/conseption.pdf.
131. Рассел С. Искусственный интеллект: Современный подход / С. Рассел, П. Норвиг; пер. с англ. – К. : Издательский дом «Вильямс», 2006. – 1407 с.
132. Рекомендации по преподаванию информатики в университетах [Электронный ресурс] // Computing Curricula 2001: Computer Science; пер. с англ. /под. ред. В. Л. Павлова и А. А. Терехова. – СПб. : Издательство Санкт-Петербургского государственного университета, 2002. – 372 с. – Режим доступа : http://se.math.spbu.ru/cc2001.
133. Рекомендации по преподаванию программной инженерии и информатики в університетах Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering; Computing Curricula 2001: Computer Science; пер. с англ. — М. : ИНТУИТ.РУ «Интернет-Университет Информационных Технологий», 2007. – 462 с.
134. Рогушина Ю. В. Використання онтологічних знань для побудови рекомендацій для проактивної підтримки семантичному пошуку [Електронний ресурс] / Ю. В. Рогушина // Інформатика, математика, автоматика : матеріали науково-технічної конференції ІМА-2015. — Суми, 2015. — С. 29. — Режим доступу : http://elitconf.sumdu.edu.ua.
135. Рогушина Ю. В. Використання онтологічного аналізу предметних галузей у системах дистанційної освіти / Рогушина Ю. В., Антоненко В. М, Гладун А. Я. // Економіка і управління. — 2007. — № 1. — С. 86—93.
136. Рогушина Ю. В. Використання технологій відкритих джерел для інтелектуалізації навчального процесу дистанційної освіти / Ю. В. Рогушина // Нові інформаційні технології в освіті для всіх ІТЕА-2007: збірник праць Другої Міжнародної конференції. — Київ, 2007. — С. 88—95.

137. Рогушина Ю. В. Внедрение современных Интернет-технологий в образовательный процесс / Ю. В. Рогушина // Educational Technology & Society. – 2008. — № 11(3). — С. 375—381.
138. Рогушина Ю. В., Гришанова І. Ю. Засоби інтелектуалізації пошуку мультимедійних даних в Інтернеті / Ю. В. Рогушина, І. Ю. Гришанова // Розробка систем програмного забезпечення: виклики часу та роль в інформаційному суспільстві: матеріали Міжнародної науково-практичної конференції. — К., 2005. – С. 98––101.
139. Рогушина Ю. В. Знание-ориентированные средства поддержки семантического поиска в Web / Ю. В. Рогушина // Открытые семантические технологии проектирования интеллектуальных систем OSTIS-2014 : материалы IV международной научно-технической конференции. — Минск : БГУИР, 2014. – С. 339—352.
140. Рогушина Ю. В. Знание-ориентированные средства поддержки семантического поиска в Web / Ю. В. Рогушина. — LAP LAMBERT Academic Publishing. — 214 с.
141. Рогушина Ю. В. Интеграция методов искусственного интеллекта с технологиями Semantic Web в сфере образования [Электронный ресурс] / Рогушина Ю. В., Гладун А. Я., Абдель-Бадех М. Салем // Інформаційні технології в освіті та науці: збірник наукових праць. ––Мелітополь, 2015. — С. 164—170. — Режим доступа : http://ito.mdpu.org.ua/index.php/partneri.
142. Рогушина Ю. В. Использование критериев оценки удобочитаемости текста для поиска информации, соответствующей реальным потребностям пользователя / Ю. В. Рогушина // Проблеми програмування. — 2007. — № 3. — С. 76—87.
143. Рогушина Ю. В. Использование метода индуктивного вывода для усовершенствования онтологии предметной области поиска / Ю. В. Рогушина, И. Ю. Гришанова // Системні дослідження та інформаційні технології. — 2007. — № 1. — С. 62—70.
144. Рогушина Ю. В. Использование онтологических знаний в рекомендующих системах / Ю. В. Рогушина // Проблеми програмування. — 2013. — № 2. — С. 71—86.
145. Рогушина Ю. В. Использование онтологического описания предметной области для повышения релевантности информационного поиска / Ю. В. Рогушина // Проблемы программирования. — 2003. — № 4. — С. 54—64.
146. Рогушина Ю. В. Использование современных интернет-технологий для повышения эффективности дистанционного образования / Ю. В. Рогушина // Сибирский учитель. — 2011. — № 5. — С. 62—66.

147. Рогушина Ю. В. Літературний твір наукового характеру «Модель мультиагентної інформаційно-пошукової системи «МАІПС» («Модель МАІПС») / Ю. В. Рогушина, І. Ю. Гришанова. — Свідоцтво про реєстрацію авторського права на твір № 32068.
148. Рогушина Ю. В. Менеджмент знаний в рекомендующих системах на основе онтологий / Ю. В. Рогушина // Интеллектуальный анализ информации ИАИ-2013: сборник научных трудов XIII международной конференции им. Т. А. Таран. —Киев, 2013. — С. 14—20.
149. Рогушина Ю. В. Мереологические аспекты онтологического анализа интеллектуальных Web-сервисов / Ю. В. Рогушина, А. Я. Гладун // Интеллектуальный анализ информации ИАИ-2007: тезисы VI Международной конференции. — Киев, 2007. — С. 312––321.
150. Рогушина Ю. В. Моделі подання знань про сталі інформаційні потреби користувачів в інформаційно-пошукових системах / Ю. В. Рогушина // Вісник НАУ. — 2004. — № 3(21). — С. 90—93.
151. Рогушина Ю. В. Модель мультиагентной информационно-поисковой системы, которая базируется на интенсиональных отношениях / Ю. В. Рогушина, Г. В. Снигирь // УсиМ. — 2003. — № 3. — С. 64—72.
152. Рогушина Ю. В. Онтологический анализ как основа семантического поиска в науке и образовании [Електронний ресурс] / Ю. В. Рогушина, Н. М. Богуш // Інформаційні технології в освіті та науці: збірник наукових праць. – Мелітополь, 2015. — С. 155—163. — Режим доступу : http://ito.mdpu.org.ua/ index. php/partneri.
153. Рогушина Ю. В. Онтологическая модель интеллектуализации сервис-ориентированных вычислений в распределенной среде Интернет [Электронный ресурс] / Ю. В. Рогушина, А. Я. Гладун // Проблеми програмування. — 2006. — № 2-3. — С. 526—536. — Режим доступа : http://dspace.nbuv. gov.ua/ handle/ 123456789/1595.
154. Рогушина Ю. В. Онтологический подход к мультилингвистическому анализу информационных ресурсов в сети Интернет / Ю. В. Рогушина, А. Я. Гладун // Интеллектуальный анализ информации ИАИ-2006: сборник трудов VI международн. конф. им. Т. А. Таран. — К. : Просвіта, 2006. — С. 237—246.
155. Рогушина Ю. В. Показатели индивидуальной легкости чтения текста как критерий поиска информационных ресурсов в сети Интернет / Ю. В. Рогушина // УсиМ. — 2007. — № 3. — С. 76—84.
156. Рогушина Ю. В. Применение методов индуктивного вывода для создания прикладных баз знаний / Ю. В. Рогушина // Матеріали

I Міжнародної науково-практичної конференції з програмування УкрПрог’98. — Київ, 1998. — С. 604.
157. Рогушина Ю. В. Программные агенты: определения, таксономии, модели / Ю. В. Рогушина // Управляющие системы и машины. — 2001. — № 5. — С. 39—45.
158. Рогушина Ю. В. Разработка знание-ориентированных средств поддержки семантического поиска в web / Ю. В. Рогушина // Інформаційне суспільство в Україні: тези доповідей Міжнародного наукового конгресу (29 жовтня 2013 р.) / Державне агентство з питань науки, інновацій та інформатизації України. — К. : Український дім, 2013. — С. 101—102.
159. Рогушина Ю. В. Разработка методов использования онтологической модели персонифицированного семантического поиска в Web при решении прикладных задач / Ю. В. Рогушина // Интеллектуальный анализ информации ИАИ-2015: сборник трудов XV международн. конф. им. Т. А. Таран. — К. : Просвіта, 2015. — С. 180—180.
160. Рогушина Ю. В. Разработка средств интеллектуализации поиска информации в Интернет / Ю. В. Рогушина // Проблемы программирования. — 2002. — № 1-2. — С. 378—385.
161. Рогушина Ю. В. Разработка формализованной онтологической модели интеллектуальной задачи как составной части научных исследований / Ю. В. Рогушина // Интеллектуальный анализ информации ИАИ-2011: сборник трудов XI международн. конф. им. Т. А. Таран. — К. : Просвіта, 2011. — С. 46—54.
162. Рогушина Ю. В. Семантическая Википедия как источник онтологий для интеллектуальных поисковых систем / Ю. В. Рогушина, А. Я. Гладун // В кн.: Advanced Research in Artificial Intelligence. International Book Series «Information Science and Computing». — ITHEA, Sofia, 2008. — P. 172—178.
163. Рогушина Ю. В. Семантический поиск как составляющая управления знаниями в Semantic Web / Ю. В. Рогушина // Материалы международной научно-технической конференции OSTIS-2012. — Минск : БГУИР, 2012. — С. 239—244.
164. Рогушина Ю. В. Средства персонализации интеллектуального поиска в Интернет / Ю. В. Рогушина // Труды Всероссийской конференции с международным участием «Знания-Онтологии-Теории». — 2007. — Т. 2. — С. 170—176.
165. Рогушина Ю. В. Управление знаниями на основе онтологий в дистанционном обучении / Ю. В. Рогушина. — LAP LAMBERT Academic Publishing, 2013. — 92 с.
166. Рогушина Ю. В. Управление онтологическими знаниями

при семантическом поиске / Ю. В. Рогушина // Тези доповідей Міжнародної наукової конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку». — Киев, 2013. — С. 218—220.
167. Рогушина Ю. В. Тезаурус предметной области – инструмент представления знаний при проблемно-ориентированном поиске в Web / Ю. В. Рогушина, А. Я. Гладун // Штучний інтелект. Інтелектуальні системи: збірник праць Міжнародної науково-технічної конференції. — К., 2010. — Т. 2. — С. 78—90.
168. Рогушина Ю. В. Технологии Semantic Web и их использование при разработке интеллектуальных приложений / Ю. В. Рогушина, А. Я. Гладун // Проблеми програмування. — 2008. — № 2-3. — С. 385—394.
169. Рогушина Ю. В. Формирование тезауруса предметной области как средства моделирования информационных потребностей пользователя при поиске в Интернете / Ю. В. Рогушина, А. Я. Гладун // Вестник компьютерных и информационных технологий. – 2007. – № 1.
170. Россеева О. И. Организация эффективного поиска на основе онтологий [Электронный ресурс] / О. И. Россеева, Ю. А. Загорулько. – Режим доступа : http://www.dialog-21.ru/archive_article.asp?param=7029.
171. Руководство по Protege 4. 2. [Электронный ресурс] / – Режим доступа : https://docs.google.com/document/d/1Fg9u9pf5RXBu8bklLh48MkZ01-DLrd8hn3ZJbTh6xg4/edit#.
172. Сегаран Т. Программируем коллективный разум / Т. Сегаран. – СПб. : Символ-Плюс, 2008. – 368 с.
173. Сейдаметова З. С. Розробка навчальних планів у галузі комп'ютінга: історія і принципи / З. С. Сейдаметова // Комп'ютер у школі та сім'ї. – 2007. – № 2. – 2007. – С. 6–10. .
174. Сидорова Е. А. Семантический подход к анализу документов на основе онтологии предметной области [Электронный ресурс] / Сидорова Е. А., Загорулько Ю. А., Кононенко И. С. // Труды международной конференции «Диалог 2006», 2006. – С. 468–473. – Режим доступа : http://www.dialog-21.ru/digests/dialog2006/materials/pdf/SidorovaE.pdf.
175. Ситник В. Ф. Інтелектуальний аналіз даних (дейтамайнінг): навч. посібник / В. Ф. Ситник, М. Т. Краснюк. – К. : КНЕУ, 2007. – 376 с.
176. Словник Wordnet [Електронний ресурс]. – Режим доступу : http://wordnet.princeton.edu/perl/webwn.
177. Смирнов С. В. Онтологии как смысловые модели / С. В. Смирнов // Онтология проектирования. – 2013. – № 2(8). – С. 11–19.

178. Соловьев В. Д. Онтологии и тезаурусы. Учебное пособие / Соловьев В. Д., Добров Б. В., Иванов В. В., Лукашевич Н. В. – Казань; Москва, 2006. – 173 с..
179. Спенсер С. Компетенции на работе / С. Спенсер, Л. Спенсер; пер. с англ. — М. : HIPPO, 2005. — 384 с.
180. Средства Microsoft SQL Server Data Tools [Электронный ресурс]. – Режим доступа : https://msdn.microsoft.com/ru-ru/data/tools.aspx.
181. Сухарніков Ю. Концептуальні підстави розробки і впровадження національної рамки (академічних) кваліфікацій України / Ю. Сухарніков // Вища школа. – 2012. – № 3. – С. 16–38.
182. Сухов С. В. Онтология управления организациями [Электронный ресурс] / С. В. Сухов. – Режим доступа : http://www.dis.ru/ manag/arhiv/2003/5/6. htm.
183. Сухомлин В. Подготовка бакалавров и магистров в области ИТ [Электронный ресурс] / В. Сухомлин // Открытые системы. – 2002. – № 3. – Режим доступа : http://www.osp.ru/os/2002/03/181287/.
184. Сухомлин В. А. Концепция нового образовательного направления [Электронный ресурс] / В. А. Сухомлин, В. В. Сухомлин // Открытые системы. – 2003. – № 2. – Режим доступа : http://www.osp.ru/os/2003/02/182628/.
185. Тарасов В. Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика / В. Б. Тарасов. – М. : Эдиториал УРСС, 2002. – 352 с.
186. Терехов А. Перспективы развития ИТ-образования [Электронный ресурс] / А. Терехов, В. Павлов // Открытые системы. – 2003. – № 2. – Режим доступа : http://www.osp.ru/os/2003/02/182637.
187. Терехов А. А. Software Engineering и российское образование [Электронный ресурс] / А. А. Терехов, А. Н. Терехов // Открытые системы. – 2006. – № 8. – Режим доступа : http://www.osp.ru/os/2006/08/3282281.
188. Трофимов И. В. Эволюция выразительных способностей языка OWL [Электронный ресурс] / И. В. Трофимов // Программные системы: теория и приложения. – 2011. – № 4(8). – С. 85–94. – Режим доступа : http:// 95.129.137.165/rented/psta/www/ read/psta2011_4_85-94.pdf.
189. Тузовский А. Ф. Работа с онтологической моделью организации на основе дескриптивной логики [Электронный ресурс] / А. Ф. Тузовский // Известия Томского политехнического университета. – 2006. – Т. 309. – № 7. – Режим доступа : http://www.duskyrobin.com/tpu/2006-07-00030.pdf.
190. Тузовский А. Ф. Системы управления знаниями (методы и технологии) / Тузовский А. Ф., Чириков С. В., Ямпольский В. З. –

Томск : Издательство НТЛ, 2005. – 260 с.
191. Филиппов В. А. Интеллектуальный анализ данных: методы и средства / В. А. Филиппов. – М. : Эдиториал УРСС, 2001.
192. Флегонтов А. Образовательные программы в области информационных технологий и систем (краткий обзор по CC2005) [Электронный ресурс] / А. Флегонтов, О. Флегонтов. – Режим доступа : http://educit.spb.ru/siop2006/ publication.php?id= 55e5b7cc797b30fabdded8ca6b387295&chapter=general.dat.
193. Черняк О. І. Інтелектуальний аналіз даних / О. І. Черняк, П. В. Захарченко. – К. : Знання, 2014.
194. Чубукова И. А. Data Mining: Учебное пособие / И. А. Чубукова. – М. : Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2006. – 112 с.
195. Шаров С. В. Бази даних та інформаційні системи. Навчальний посібник / С. В. Шаров, В. В. Осадчий. – Мелітополь : Вид-во МДПУ ім. Б. Хмельницького, 2014. – 352 с.
196. Шелевицький І. Освіта у сфері інформаційних технологій, або реформа чи мімікрія (Частина 1) [Електронний ресурс] / І. Шелевицький. – Режим доступу : http://osvita.org.ua/articles/ ?article_id=15.
197. Щербак С. О формальных основах OWL [Электронный ресурс] / С. Щербак. – Режим доступа : http://shcherbak.net/2009/03/o-formalnyx-osnovax-owl/.
198. Яловец А. Л. Представление и обработка знаний с точки зрения математического моделирования / А. Л. Яловец. – К. : Наукова думка, 2011. – 360 с.
199. About Adult Services [Електронний ресурс] – Режим доступу : http://www.connexionslive.com/Adults/Default.aspx
200. About Cognitum [Електронний ресурс]. — Режим доступу : http://www.cognitum.eu/company/Default.aspx.
201. A Direct Mapping of Relational Data to RDF. W3C Working Draft 24 March 2011 [Електронний ресурс]. — Режим доступу : http://www.w3.org/TR/rdb-direct-mapping/.
202. A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools. Edition 1.2., 2009 [Електронний ресурс]. — Режим доступу : http://phd.jabenitez.com/wp-content/uploads/2014/03/A-Practical-Guide-To-Building-OWL-Ontologies-Using-Protege-4.pdf.
203. Acciarri A. QUONTO: Querying ONTOlogies / Acciarri A., Calvanese D., De Giacomo G., Lembo D., Lenzerini M., Palmieri M., Rosati R. // Proc. of the 20th National Conf. on Artificial Intelligence. — V. 4. — Pittsburgh, Pennsylvania, 2005. — P. 1670—1671.
204. Alberts L. K. YMIR: an Ontology for Engineering Design /

L. K. Alberts. — University of Twente, 1993.
205. Amerland D. Google Semantic Search: Search Engine Optimization (SEO) Techniques That Gets Your Company More Traffic, Increases Brand Impact and Amplifies Your Online Presence / D. Amerland. — Que Publishing, 2013. — 230 p.
206. Andon P. Approach to Automatic Creation of Ontology from Documents for Improving Existent Information Retrieval / P. Andon, V. Deretsky // Proc.of 2-nd Balkan Conference in Informatics (BCI'2005) November 17-19, 2005. — P. 236—241.
207. Antoniou G. Web ontology language: Owl / G. Antoniou, Van Harmelen F. // In Handbook on ontologies. — Springer Berlin Heidelberg, 2004. — P. 67—92.
208. Ashpole B., Ehrig M., Euzenat J., Stuckenschmidt H. // Proc. of the work-shop on Integrating Ontologies at K-CAP, 2005.
209. Baker T. A grammar of Dublin Core [Електронний ресурс] / T. Baker // D-lib magazine, 6(10), 2000. — P. 47—60. — Режим доступу : http://www.agiimc.de/internet.nsf/0/4858ed9b8029d0c1c125699f002d6197/$FILE/A%20Grammar%20of%20Dublin%20Core%20@%20.pdf.
210. Bell D. A framework for deriving semantic web services / Bell D., de Cesare S., Iacovelli N., Lycett M. and Merico A. // Information Systems Frontiers. – 2007. — № 1. – Volume 9. — P. 69—84.
211. Benjamin P. C. IDEF5 Method Report / Benjamin P. C., Menzel C. P., Mayer R. J., Fillion F., Futrell M. T., de Witte P. S., Lingineni M. // Information Integration for Concurrent Engineering (IICE). Contract: F33615-C-90-0012. Knowledge Based Systems, Inc. 1994. — 187 p.
212. Berry M. W. Survey of text mining / M. W. Berry // Computing Reviews 45.9, 2004. — 244 p.
213. Best C. Open source intelligence. Mining massive data sets for security: advances in data mining, search, social networks and text mining and their applications for security[Електронний ресурс] / C. Best // IOS Press, Amsterdam, 2008. — P. 331—344. — Режим доступу : https://www.google.com.ua/books?hl=uk&lr=&id=U2yUAgAAQBAJ&oi=fnd&pg=PA129&dq=Open+Source+Intelligence&ots=1x9xI1EJVL&sig=8DNbcZu8X0lKhBVCT2lDRWVqU8I&redir_esc=y#v=onepage&q=Open%20Source%20Intelligence&f=false..
214. Bouchiha D. Towards re-engineering Web Applications into semantic Web services / D. Bouchiha, M. Malki // The first International IEEE Conference on Machine and Web Intelligence (ICMWI'2010). — Algeria, Algiers, 2010.
215. Bouchiha D. An Empirical Approach for Annotating Web Services / Bouchiha D., Malki M., Alghamdi A., Alnafjan K. // Proc. of The 24th

International Conference on Computer Applications in Industry and Engineering. Hawaii, USA, 2011.
216. Brachman R. An Overview of the KL-ONE Knowledge Representation System / R. Brachman, J. Schmolze // Cognitive Science. — 1985. — №2. — V. 9. — P. 171—216.
217. Bray T. Extensible markup language (XML) [Електронний ресурс] / Bray T., Paoli J., Sperberg-McQueen C. M., Maler E., Yergeau F. — World Wide Web Consortium Recommendation REC-xml-19980210, 1998. — Режим доступу : http://www.w3. org/TR/1998/REC-xml-19980210, 16.
218. Broekstra J. Enabling knowledge representation on the web by extending RDF schema [Електронний ресурс] / Broekstra J., Klein M., Decker S., Fensel D., Van Harmelen F., Horrocks I. // Computer networks, 39(5), 2002. — P.609—634. — Режим доступу : http://www.sciencedirect.com/science/article/ pii/ S138912860200217 77.
219. Bruijn J. On the Relationship between Description Logic-based and F-Logic-based Ontologies [Електронний ресурс] / J. Bruijn, S. Heymans // Fundam. Inform, № 82(3), 2008. — P. 213—236. — Режим доступу : http://portal.acm.org/ citation. cfm?id=1377800.
220. Brusilovsky P. Adaptive and Intelligent Web-based Educational Systems [Електронний ресурс] / P. Brusilovsky, C. Peylo // International Journal of Artificial Intelligence in Education, № 13, 2003. — P. 156—169. — Режим доступу до журналу : http://www.sis.pitt.edu/~peterb/papers/AIWBES.pdf.
221. Calle F. J. Cognos: a pragmatic annotation toolkit for the acquisition of natural interaction knowledge [Електронний ресурс] / Calle F. J., Albacete E., Olaziregi G., Sánchez E., del Valle D., Rivero J., Cuadra D // Procesamiento del lenguaje natural, 47, 2011. — P. 269—276. — Режим доступу : http:// journal. sepln.org/ index. php/pln/article/view/972.
222. Calvanese D. Tractable reasoning and efficient query answering in description logics: The DL-Lite family / Calvanese D., De Giacomo G., Lembo D., Lenzerini M., Rosati R. // JAR, 39(3), 2007. — P. 385—429.
223. Calvaneze D. Ontologies and Databases. Tutorial. Reasoning Web Summer School, 2009 [Електронний ресурс] / D. Calvaneze. — Режим доступу : http://www.inf.unibz.it/ calvanese/teaching/2009-09-ReasoningWeb-school-ontologies-dbs/ReasoningWeb-2009-ontologies-dbs.pdf.
224. Chakrabarti S. Data mining for hypertext: A tutorial survey [Електронний ресурс] / S. Chakrabarti // ACM SIGKDD Explorations Newsletter ACM SIGKDD Explorations Newsletter Homepage archive, V. 1, Issue 2, 2000. — P. 1—11.

225. Chimaera [Електронний ресурс]. — Режим доступу : http://www. ksl. stanford.edu/software/chimaera/
226. Cimiano P. Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Ontology Learning and Population from Text: Algorithms, Evaluation and Applications / P. Cimiano. — Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2006. — 347 p.
227. Cognos Business Intelligence [Електронний ресурс]. — Режим доступу : http://www-03.ibm.com/software/products/ru/business-intelligence.
228. Computing Curricula 2005. The Overview Report, ACM, AIS,IEEE-CS, 30 September 2005 [Електронний ресурс]. — Режим доступу : http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf.
229. Cooley R. Web mining: Information and pattern discovery on the world wide web / R. Cooley, M. Bamshad, S. Jaideep // Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on IEEE, 1997.
230. Core Software Ontology. Core Ontology of Software Components. Core Ontology of Services [Електронний ресурс]. — Режим доступу : http://cos.ontoware.org/.
231. Cowles P. Web Services and the Semantic Web [Електронний ресурс] / P. Cowles. — Режим доступу : http://ezolin.pisem.net/logic/ws_and_sw_rus.html.
232. Cycorp. Home of Smarter Solutions [Електронний ресурс]. — Режим доступу : http://www.cyc.com.
233. Cyganiak R. RDF 1. 1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014 [Електронний ресурс] / Cyganiak R., Wood D., Lanthaler M. — Режим доступу : http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.
234. DAML+OIL. Joint US/EU ad hoc Agent Markup Language Committee. 2001 [Електронний ресурс]. — Режим доступу : http://www. daml. org/2001/ 03/ daml+oil-index.
235. Davies J. Towards the Semantic Web: Ontology-driven knowledge management / Davies J., Fensel D., van Harmelen F. — John Wiley & Sons Ltd, England, 2002. — 288 p.
236. DCMI Home: Dublin Core Metadata Initiative (DCMI) [Електронний ресурс]. — Режим доступу : http://dublincore.org.
237. De Campos L. M. Using personalization to improve XML retrieval [Електронний ресурс] / De Campos L. M., Fernández-Luna J. M., Huete J. F., Vicente-Lopez E. // Knowledge and Data Engineering, IEEE Transactions on, 26(5), 2014. — P. 1280—1292. — Режим доступу : http://www.google.com.ua/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0CDEQFjAD&url=http%3A%2F%2Fdecs

ai.ugr.es%2F~lci%2Fjournal-papers df%2Ftkde14preprint. pdf&ei= dXcdVY 4HoTYPK ywgWA&usg= AFQjCNGRlbSkxGg-XL3d_rSewgBtiZB-Fg&sig2= oIQ_ slT9kNyfB4iq8j1QaQ.
238. Debruyne C., De Leenheer P. Business Semantics as an Interface between Enterprise Information Management and the Web of Data: A Case Study in the Flemish Public Administration // M. A. Aufaure and E. Zimányi, editors, eBISS 2012 — V. 138. – P. 208—233.
239. Deng X. Measuring inconsistencies in ontologies / Deng X., Haarslev V., Shiri N. // The Semantic Web: Research and Applications, 2007. — P. 326—340.
240. Dennett D. C. The Intensional Stance / D. C. Dennett. — The MIT Press : Cambridge, MA, 1987. — 282 p.
241. Devedžić V. Web Intelligence and Artificial Intelligence in Education [Електронний ресурс] / V. Devedžić. — 2004. — № 7(4). — P. 29—39. — Режим доступу : http://www.ifets.info/journals/7_4/6.pdf.
242. Di Noia T. Semantic Matchmaking as Non-Monotonic Reasoning: A Description Logic Approach / Di Noia T., Di Sciascio E., Donini F. M. // Journal of Artificial Intelligence Research 29, 2007. — P. 269—307.
243. DIG 1. 1 — OBDA [Електронний ресурс]. — Режим доступу : http://obda.inf.unibz.it/dig-ll-obda/
244. DL Implementation Group [Електронний ресурс]. — Режим доступу : http://dl.kr.org/dig/
245. Do H. COMA – a system for flexible combination of schema matching approaches / Do H., Rahm E. // Proceedings of VLDB, 2002.
246. Do H. Matching large schemas: Approaches and evaluation / Do H., Rahm E. // Information Systems, 2007.
247. Domshlak C. Rank aggregation for automatic schema matching / Domshlak C., Gal A., Roitman H. // IEEE Transactions on Knowledge and Data Engineering, 2007.
248. Dublin Core [Електронний ресурс]. — Режим доступу : http://semanticweb.org/wiki/Dublin_Core.
249. ELK reasoner [Електронний ресурс]. — Режим доступу : http://www.cs.ox.ac.uk/isg/tools/ELK/.
250. Euzenat J. Alignment infrastructure for ontology mediation and other applications / J. Euzenat // Proc.of the workshop on Mediation in Semantic Web Services, 2005.
251. Euzenat J. Semantic precision and recall for ontology alignment evaluation / J. Euzenat // Proc. of IJCAI, 2007.
252. Euzenat J. Ontology alignments: an ontology management perspective / Euzenat J., Mocan A., Scharffe F. // Ontology management: semantic web, semantic web services, and business applications. Springer, 2008.

253. Euzenat J. Results of the ontology alignment evaluation initiative 2006 / Euzenat J., Mochol M., Shvaiko P., Stuckenschmidt H., Svab O., Svatek V., Hage W., Yatskevich M. // Proc. of the workshop on Ontology Matching at ISWC, 2006.
254. Euzenat J., Shvaiko P. Ontology matching / J. Euzenat, P. Shvaiko. — Springer-Verlag Berlin Heidelberg, 2007. — 332 p.
255. Extensible Markup Language (XML) 1.0, W3C Recommendation 1998 [Електронний ресурс]. — Режим доступу : http://www.w3.org/TR/1998/REC-xml-19980210.
256. Falconer S. A cognitive support framework for ontology mapping / S. Falconer, M. Storey // Proc. of ISWC/ASWC, 2007.
257. Falcons Semantic Web Search Engine [Електронний ресурс]. — Режим доступу : http://iws.seu.edu.cn/services/falcons/ conceptsearch/index.jsp?query
258. Fensel D. OIL: An Ontology Infrastructure for the Semantic Web [Електронний ресурс] // Fensel D., Harmelen F., Horrocks I., McGuinness D., Patel-Schneider P. — Режим доступу : http://www.cs.man.ac.uk/%7Ehorrocks/ Publications/download/ 2001/IEEE-IS01.pdf.
259. Fernández-López M. METHONTOLOGY: From Ontological Art Towards Ontological Engineering / Fernández-López M, Gómez-Pérez A, Juristo N. // Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, 1997. — P. 33—40.
260. Fernández-López M. Building a Chemical Ontology Using Methontology and the Ontology Design Environment / Fernández-López M, Gómez-Pérez A, Pazos A, Pazos J. // IEEE Intelligent Systems & their applications 4(1), 1999. — P. 37—46.
261. Finin O. KQML – a language and protocol for knowledge and information exchange. In Proceedings of the 11th Intl / O. Finin, R. Fritzson // Distributed Artificial Intelligence Workshop, page» 127–136, Seattle, WA, USA, 1994. — P. 127—136.
262. FIPA – Federation of Intelligent Physical Agents. Home Page [Електронний ресурс]. — Режим доступу : http://www.cselt.stet.it/fipa/fipa_rationale.htm.
263. Flesch Reading Ease Readability Formula [Електронний ресурс]. — Режим доступу : http://oleandersolutions.com/fleschreadingease.html.
264. FP4, Project INCO-COPERNICUS, Grant № 960114 – EXPERNET "A distributed Expert System for the Management of the National Network of Ukraine" [Електронний ресурс] / Skurihin V. I., Gladun A. J., Lastovchenko M. M., Doroshenko A. E. (Розподілена експертна система керування Національною телекомунікаційною мережею України), МННЦІТС НАНУ і МОНУ. — Режим доступу :

http://lpis.csd.auth.gr/projects/ExperNet.html.
265. Gal A. A framework for modeling and evaluating automatic semantic reconciliation / Gal A., Anaby-Tavor A., Trombetta A., Montesi D. // The VLDB Journal, 2005.
266. gCUBE / Framework [Електронний ресурс]. — Режим доступу : www.gcube-system.org
267. gCUBE Metadata Management [Електронний ресурс]. — Режим доступу : https://gcube.wiki.gcube-system.org/gcube/index.php / Metadata_Management# Metadata_Management
268. Genesereth M. R. Knowledge Interchange Format (version 3.0) Reference Manual. Interlingua Working Group of the DARPA Knowledge Sharing Effort / M. R. Genesereth, R. E. Fikes et al. — Computer Science Department, Stanford University. Report Logic-92-1.
269. Genesereth M. R., Nilsson N. J. Logical Foundation of Artificial Intelligence / M. R. Genesereth, N. J. Nilsson. — Morgan Kaufmann, Los Altos, California, 1987.
270. Geroimenko V. Dictionary of XML Technologies and the Semantic Web / V. Geroimenko. — Springer, 2004. — 248 p.
271. Ghosh S. Cognos: crowdsourcing search for topic experts in microblogs [Електронний ресурс] / Ghosh S., Sharma N., Benevenut F., Ganguly N., Gummadi K. // Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, ACM, 2012. — P. 575—590. — Режим доступу : http://dl.acm.org/citation.cfm?id=2348361.
272. Gil Y. A Comparison of (Semantic) Markup Languages [Електронний ресурс] / Y. Gil., V. Ratnakar // FLAIRS Conference, 2002. — Режим доступу : http://www.aaai.org/Papers/FLAIRS/2002/FLAIRS02-081.pdf.
273. Gilbert N., Bankes S. Platforms and methods for agent-based modeling / N. Gilbert., S. Bankes // Proceedings of the National Academy of Sciences, 99(suppl 3), 2002. — P. 7197—7198.
274. Ginsberg M. L. Knowledge Interchange Format / M. L. Ginsberg // The KIF of DeathAI Magazine. — 1991. — V. 12. — № 3. — P. 57—63.
275. Giunchiglia F. Encoding classifications into lightweight ontologies / Giunchiglia F., Marchese M., Zaihrayeu I. // Journal of Data Semantics, 2007.
276. Giunchiglia F. Discovering missing background knowledge in ontology matching / Giunchiglia F., Shvaiko P., Yatskevich M. // Proc. of ECAI, 2006.
277. Giunchiglia F. Semantic matching / Giunchiglia F., Shvaiko P., Yatskevich M. // Encyclopedia of Database Systems, 2009.
278. Giunchiglia F. Semantic schema matching / Giunchiglia F., Shvaiko P.,

Yatskevich M. // Proc. of CoopIS, 2005.
279. Giunchiglia F. A large scale dataset for the evaluation of ontology matching systems / Giunchiglia F., Yatskevich M., Avesani P., Shvaiko P. // The Knowledge Engineering Review, 2008.
280. Giunchiglia F. Semantic matching: Algorithms and imple-mentation / Giunchiglia F., Shvaiko P., Yatskevich M. // Journal on Data Semantics, 2007.
281. Gladun A. An application of intelligent techniques and Semanic Web technologies in e-learning environments / Gladun A., Rogushina J., Garcia-Sanchez F., Martinez-Bejar R., Fernandez-Breis J. T. // Expert Systems with Applications, An International Journal. — 2009. — V. 36. — P. 1922—1931.
282. Gladun A. Distant control of student skills by formal model of domain knowledge / A. Gladun, J. Rogushina // International Journal of Innovation and Learning (IJIL), InderScience Publishers. — 2010. — Vol. 7. — № 4. — P. 394—411.
283. Gladun A. Domain Ontology an Instrument of Semantic Web Knowledge Management in e-Learning / Gladun A., Rogushina J., Schreurs J. // International Journal of Advanced Corporate Learning (iJAC). — 2012. — V. 5, Issue 4. — P. 21—–31.
284. Gladun A. Integration of Financial Domain Knowledge on Base of Semantic Web Technologies / Gladun A., Rogushina J., Martínez-Béjar R., García-Sanchez F., Valencia-García R. // Information Models of Knowledge. Edited by K. Markov, V. Velychko, O. Voloshin. — IT H E A, Kiev-Sofia, 2010. — P. 106—112.
285. Gladun A. Formalization of Search Context on Base of Ontologies and Multilinguistic Thesauruses / A. Gladun, J. Rogushina // Computing. — 2007. — V 6, issue 3. — P. 16—22.
286. Gladun A. Ontologies as a Perspective Direction of Intellectualization of Informational Retrieval in Multiagent Systems of E-commerce / A. Gladun, J. Rogushina // The Proceedings of XI-th Intern. Conf. «Knowledge-Dialogue-Solution», KDS'2005, Varna, Bulgaria. — P. 112—120.
287. Gladun A. Ontology-based competency analyses in new research domains / A. Gladun, J. Rogushina // Journal of Computing and Information Technology. — 2012. – V. 20. — № 4. — P. 277—293.
288. Gladun A. Ontology-based knowledge recognition in service-oriented virtual research environments Case: application in e-learning [Електронний ресурс] / Gladun A., Rogushina J., Schreurs J., Salem Abdel-Badeeh // Proc. of The 7th International Conference on Information Technology ICIT-2015, Al Zaytoonah University of Jordan, Amman, Jordan, 2015. — P. 148—155. — Режим доступу :

http://icit.zuj.edu.jo/icit15/DOI/Artificial_Intelligence/0022.pdf.
289. Gladun A. Semantics-driven modelling of user preferences for information retrieval in the biomedical domain / Gladun A., Rogushina J., Valencia-García R., Martínez-Béjar R. // Informatics for health and social care. — 2013. — V. 38. — № 2. — P. 150—170.
290. Gladun A. Use of Semantic Web Technologies and Multilinguistic Thesauri for Knowledge-Based Access to Biomedical Resources [Електронний ресурс] / A. Gladun, J. Rogushina // International Journal of Intelligent Systems and Applications. — 2012. — № 1. — P. 11—20. — Режим доступу : http://www.mecs-press.org/ijisa/ijisa-v4-n1/IJISA-V4-N1-2.pdf
291. Gladun A. Use of Semantic Web technologies in design of informational retrieval systems / A. Gladun, J. Rogushina // in Book «Building and Environment». — Nova Scientific Publishing, New-York, USA, 2009. – – P. 89—103.
292. Gladun A. Use of the ontological approach to semantic search in the envitonment of the Internet of Things / Gladun A., Rogushina J., Andrushevich A., Kurbatski A. // Материалы IV международной научно-технической конференции «Открытые семантические технологии проектирвоания интеллектуальных систем» OSTIS-2014. — Минск : БГУИР, 2014. — С. 353—356.
293. Gladun A. User-Oriented Recognition of Intelligent Information Objects in Distributed Dynamic Informational Web-Space / Gladun A., Rogushina J., Andrushevich A., Kurbatski A. // Proc. of the 12th International Conference on Pattern Recognition and Information Processing (PRIP-2014) Minsk, Belarus, 2014. — P. 70—75.
294. Gligorov R. Using google distance to weight approximate ontology matches / Gligorov R., Aleksovski Z., Kate W., Harmelen F. // Proc. of WWW, 2007.
295. GML [Електронний ресурс]. — Режим доступу : http://en.wikipedia.org/ wiki/ IBM_Generalized_Markup_Language.
296. Grau B. C. OWL 2: The Next Step for OWL [Електронний ресурс] / Grau B. C., Horrocks I., Motik B., Parsia B., Patel-Schneider P., Sattler U. // Journal of Web Semantics: Science, Services and Agents on the World Wide Web, № 6(4), 2008. – P. 309—322. — Режим доступу : http://www.w.websemanticsjournal.org/ index.php/ps/article/download/156/154.
297. Gruber T. R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing / T. R. Gruber // International Journal of Human-Computer Studies. – 1995. — V. 43, Issues 5-6. — P. 907—928.
298. Gruber T. R. Ontolingua: A mechanism to Support Portable Ontologies. Knowledge Systems Laboratory / T. R. Gruber. — Stanford University,

1990.
299. Gruber T. R. What is an Ontology? [Електронний ресурс] / T. R. Gruber. – Режим доступу : http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.
300. Gruber T. R. A Translation Approach to Portable Ontology Specifications / T. R. Gruber // In Knowledge Acquisition. — 1993. — № 5.— P. 199—220.
301. Grüninger M. Methodology for the design and evaluation of ontologies / M. Grüninger, M. Fox // Skuce D (ed) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, 1995. — P. 6. 1— 6. 10.
302. Gruninger M. An Ontology Framework / M. Gruninger, L. Obrst // Ontology Summit. — NIST, 2007.
303. Guarino N. Formal Ontology in Information Systems / N. Guarino // Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, June 6-8, 1998 – Amsterdam : IOS Press, 1998. — P. 3—15.
304. Guarino N. Understanding, Building, and Using Ontologies / N. Guarino // IJHCS. — 1996. — № 46(2–3).
305. Haarslev V. The RacerPro Knowledge Representation and Reasoning Systems [Електронний ресурс] / Haarslev V., Hidde K., Moller R., Wessel M. // Semantic Web Journal, IOS Press, 2011. — Режим доступу : http://www.franz.com/agraph/cresources/white_papers.
306. Harth A. A semantic matchmaker service on the grid / Harth A., He Y., Tangmunarunkit H., Decker S., Kesselman C. // WWW 2004, 2004. — P. 326—327.
307. Hartung M. Management of evolving semantic grid metadata within a collaborative platform / Hartung M., Loebe F., Herre H., Rahm E. // Information Sciences 180, 2010. – P. 1837—1849.
308. Horrocks I. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 2004 [Електронний ресурс] / Horrocks I., Patel-Schneider P. F., Boley H., Tabet S., Grosof B., Dean M. — Режим доступу : http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.
309. Horrocks I. A tableaux decision procedure for SHOIQ / I. Horrocks, U. Sattler // Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), 2005. — P. 448—453.
310. IDEF5 Method Report, Information Integration for Concurrent Engineering (IICE). — Knowledge Based Systems, Inc., 1994.
311. Intelligent Systems Division. USC/Information Sciences Institute (ISI) [Електронний ресурс]. — Режим доступу : http://ai.isi.edu/.
312. Introduction to Semantic Web [Електронний ресурс]. — Режим доступу : http://www.mphasis.com/knowledge-center/white- papers-

all.asp.
313. ISO 25964-1:2011, Thesauri and interoperability with other vocabularies. Part 1: Thesauri for information retrieval / Geneva: International Organization for Standards, 2011.
314. Jansen B. J. Real life, real users, and real needs: a study and analysis of user queries on the Web [Електронний ресурс] / Jansen B. J., Spink A., Saracevic T. — Режим доступу : http://citeseer.nj.nec.com/jansen00real.html.
315. Jeffery K. G. Next Generation Grids for Environmental Science / K. G. Jeffery // Environmental Modelling & Software. — 2007. — V. 22. — № 3. — P. 281—287.
316. Kalfoglou1 Y. Ontology mapping: the state of the art [Електронний ресурс] / Y. Kalfoglou1, M. Schorlemmer // The knowledge engineering review. — 2003. — № 18(01). — P. 1—31. — Режим доступу : http://eprints. soton.ac. uk/ 260519/ 1/ker02-ontomap.pdf.
317. KAON2 – Ontology Management for the Semantic Web [Електронний ресурс]. — Режим доступу : http://kaon2.semanticweb.org/.
318. Kay A. Computer Software / A. Kay // Scientific American. — 1984. — V. 251. — № 2. — P. 53—59.
319. Khribi M. Automatic Recommendations for E-Learning Personalization Based on Web Usage Mining Techniques and Information Retrieval [Електронний ресурс] / Khribi M., Jemni M., Nasraoui O. — 2009. — № 12(4). — P. 30—42. — Режим доступу : http://www.ifets.info/journals/12_4/4.pdf.
320. Kiryakov A. Reasoning in the Semantic Repositories, Handbook of Semantic Web Technologies / A. Kiryakov, M. Damova. — Springer, 2011. — P. 245—258.
321. Krafzig D. Enterprise SOA: service-oriented architecture best practices / Krafzig D., Banke K., Slama K. — Prentice Hall Professional, 2005. — 378 p.
322. Kutz O. Carnap, Goguen and the Hyperontologies: Logical Pluralism and Heterogeneous Structuring in OntologyDesign / Kutz O, Mossakowski T., Lucke D. // Logica Universalis 4.2: Special Issue on 'Is Logic Universal', 2010. — P. 255—333.
323. Laera L. Argumentation over ontology correspondences in MAS / Laera L., Blacoe I., Tamma V., Payne T., Euzenat J., Bench-Capon T. // Proc. of AAMAS, 2007.
324. Lambrix P., Tan H. A tool for evaluating ontology alignment strategies / P. Lambrix, H. Tan // Journal on Data Semantics, 2007.
325. Larson B. Delivering Business Intelligence with Microsoft SQL Server 2005. Osborne/McGraw-Hill, 2006 [Електронний ресурс] / B. Larson. – Режим доступу : http://dl.acm.org/citation.cfm?id=1204321.

326. Lassila O. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation [Електронний ресурс] / O. Lassila, R. Swick. — Режим доступу : http://www.w3.org/TR/REC-rdf-syntax.
327. Lawley M. J. Fast Classification in Protégé: Snorocket as an OWL 2 EL Reasoner / M. J. Lawley, C. Bousquet // Australasian Ontology Workshop 2010 (AOW 2010): Advances in Ontologies. — 2010. — V. 122 of CRPIT. — P. 45—50.
328. Lawrence S. Context in the Web Search [Електронний ресурс] / S. Lawrence. — Режим доступу : http://citeser.nj.nec.com/ lawrence00context.html.
329. Lee Y. eTuner: tuning schema matching software using synthetic scenarios / Lee Y., Sayyadian M., Doan A., Rosenthal A. // The VLDB Journal, 2007.
330. Lenat D. B. Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project / D. B. Lenat, R. V. Guha. — Addison-Wesley, Boston, Massachusetts, 1990.
331. Leuf B. The Wiki way: collaboration and sharing on the Internet, 2001 [Електронний ресурс] / B. Leuf, W. Cunningham. — Режим доступу : http://www.citeulike.org/group/13847/article/7659081.
332. Long A. Calculating Reading Level. Tameri Guide for Writers [Електронний ресурс] / A. Long. — Режим доступу : www. tameri. com/edit/ levels. html.
333. Loom Project Home Page. Artificial Intelligence research group University of Southern California's Information Sciences Institute [Електронний ресурс]. — Режим доступу : http://www.isi.edu/isd/LOOM/
334. Lundqvist K. O. An ontological approach to competency management [Електронний ресурс] / Lundqvist K. O., Baker K. D., Williams S. A. – – Режим доступу : http://www.eife-l.org/publications/proceedings/ ilf07/ Contribution110. doc. pdf.
335. Macdonald C. Voting for candidates: adapting data fusion techniques for an expert search task / C. Macdonald, I. Ounis // CIKM 2006: Proc. of the 15th ACM International Conference on Information and Knowledge Management. ACM, New York, 2006. — P. 387—396.
336. Madhavan J. Corpus-based schema matching / Madhavan J., Bernstein P., Doan A., Halevy A. // Proc. of ICDE, 2005.
337. Madhavan J. Generic schema matching with Cupid / Madhavan J., Bernstein P., Rahm E. // Proc. of VLDB, 2001.
338. Magnini B. Merging Global and Specialized Linguistic Ontologies / B. Magnini, M. Speranza // Proceedings of OntoLex, 2002. — P. 43—48.

339. Marmanis H. Algorithms of the intelligent web [Електронний ресурс] / Marmanis H., Babenko D. — Greenwich: Manning, 2009. — 345 p. — Режим доступу : https://sisis.rz.htw-berlin.de/inh2010/12378040.pdf.
340. Matuszek C. An Introduction to the Syntax and Content of Cyc [Електронний ресурс] / Matuszek C., Cabral J., Witbrock M., DeOliveira J. // AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, 2006. — P. 44—49. — Режим доступу : http://citeseerx.ist.psu.edu/ viewdoc/ download? doi= 10.1.1.68.1357&rep=rep1&type=pdf.
341. Mayer R. IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications / Mayer R., Painter M., deWitte P. — Knowledge Based Systems, Inc., 1992.
342. McArthur D. The Roles of Artificial Intelligence in Education: Current Progress and Future Prospects, 1993 [Електронний ресурс] / McArthur D., Lewis M., Bishay M. — Режим доступу : http://www.rand.org/hot/mcarthur/ Papers/role.html.
343. McCann R. Doan and A. Matching schemas in online communities: A web 2.0 approach / R. McCann, W. Shen // Proc. of ICDE, 2008.
344. McCarthy J. Ascribing mental qualities to machines / J. McCarthy. — Techn.report, Stanford University Al Lab., Stanford, 1978.
345. McGrath R. E. Semantic infrastructure for a ubiqutious computing environment, 2005 [Електронний ресурс] / McGrath R. E. — Режим доступу : http://www.cs.uiuc.edu/research/ techreports.php?report=UIUCDCS-R-2005-2587& downioad=pdf.
346. McLaughlin H. SMOG grading a new readability formula [Електронний ресурс] / H. McLaughlin // Journal of Reading. — 1969. — № 22. — P. 639—646.
347. Meacham E. D. Distance Education: Selecting Textbooks and Writing Study Guides [Електронний ресурс] / E. D. Meacham. — Режим доступу : http:// www.cito.ru/gdenet/technology/print/textbooks/1.
348. Microformats [Електронний ресурс]. — Режим доступу : http:// microformats.org/.
349. Middleton S. Ontology-Based Recommender Systems / Middleton S., De Roure D., Shadbolt N. // in Handbook on Ontologies, Edt. by S. Staab, R. Studer, Springer, 2009. — P. 779—796.
350. Miles T. H. The fog index: a practical readability scale / T. H. Miles // In Critical Thinking and Writing for Science and Technology. Harcourt Brace Jovanovich, 1990. — P. 280—284.
351. Minsky M. Progress Report on Artificial Intelligence [Електронний ресурс] / M. Minsky // Seymour Papert ec 11, 1971. — Режим доступу : http:// web.media.mit.edu/~minsky/papers/PR1971.html.

352. Mizoguchi R. A Step Towards Ontological Engineering [Електронний ресурс] / R. Mizoguchi. — Режим доступу : http://www.ei.sanken.osaka-u.ac.jp/english/step-onteng. html.
353. Mochol M. Applying an analytic method for matching approach selection / Mochol M., Jentzsch A., Euzenat J. // Proc. of the workshop on Ontology Matching, 2006.
354. Møller A. An introduction to XML and Web Technologies / A. Møller, M. Schwartzbach. — Pearson Education, 2006.
355. Moss L. T. Business intelligence roadmap: the complete project lifecycle for decision-support applications [Електронний ресурс] / L. T. Moss, S. Atre // Addison-Wesley Professional, 2003. — 545 p. — Режим доступу: https:// www.google.com.ua/ books?hl=uk&lr=&id=HSeE7rOXKsUC&oi=fnd&pg=PR17&dq=Business+Intelligence&ots=twC1FqTly&sig=f0yv4fQRuepcFQe7siMmTpeYtOY&redir_esc=y#v=onepage&q=Business%20Intelligence&f=false.
356. Microsoft SQL Server [Електронний ресурс]. — Режим доступу : **Error! Hyperlink reference not valid.**.
357. Negroponte N. Agents: from direct manipulation to delegation / N. Negroponte. — Software Agents, 1997.
358. Nenov Y. Modal Logics for Mereotopological Relations [Електронний ресурс] / Y. Nenov, D. Vakarelov. — Режим доступу : http:// aiml08. loria.fr/ talks/ nenov.pdf.
359. Next Generation Grids 2: Requirements and Options for European Grids Research 2005–2010 and Beyond, Expert Group Report, 2004 [Електронний ресурс]. — Режим доступу : http://www.semanticgrid.org/docs/ngg2_eg_final.pdf.
360. Nigro H. O. ed. Data Mining with Ontologies: Implementations, Findings, and Frameworks: Implementations, Findings and Frameworks / H. O. Nigro ed. — IGI Global, 2007. — 289 p.
361. Niles I., Pease A. Towards a Standard Upper Ontology [Електронний ресурс] / I. Niles, A. Pease // Proc. of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), 2001. — P. 2—9. — Режим доступу : http://www.researchgate.net/profile/ Adam_Pease/publication/22123 4966_Towards_a_standard_upper_ ontology/ links/0fcfd511f0fd5f092e000000.pdf.
362. Nobles R. The Future Of Search Engine Optimizing: Theme Engines. the next generation of search engines has arrived [Електронний ресурс] / R. Nobles. — Режим доступу : http://www.searchengineworkshops.com/ articles/se-optimization-future.html.
363. Noy N. Collecting community-based mappings in an ontology repository / Noy N., Griffith N., Musen M. // Proc. of ISWC, 2008.

364. Noy N. SMART: Automated Support for Ontology Merging and Alignment [Електронний ресурс] / N. Noy, M. Musen // Proc. of the 12th Workshop on Knowledge Acquisition, Modelling, and Management (KAW'99), Banf, Canada, 1999. — Режим доступу : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.3346&rep=rep 1&type=pdf.
365. Noy N. The PROMPT Suite: Interactive Tools For Ontology Merging and Mapping Stanford Medical Informatics, 2003 / N. Noy, M. Musen. – – SMI-2003-0973.pdf.
366. Noy N. F. Ontology Development 101: A Guide to Creating Your First Ontology [Електронний ресурс] / N. F. Noy, D. L. McGuinness. — Режим доступу : http://protege.stanford.edu/publications/ontology_development/ontology101.pdf.
367. Noy N. F. The PROMPT suite: interactive tools for ontology merging and mapping [Електронний ресурс] / N. Noy, M. Musen // International Journal of Human-Computer Studies, 59.6, 2003. — P. 983—1024. — Режим доступу : http://ce.sungshin.ac.kr/~jkjang/seminar/science.pdf.
368. Nwana H. Coordination in software agent systems / Nwana H., Lee J., Jennings N. // British Telecommunication Technology Journal. — 1996. — 14(4). — P. 79—88.
369. Obrst L. The evaluation of ontologies [Електронний ресурс] / Obrst L., Ceusters W., Mani I., Ray S., Smith B. // Semantic Web, Springer US, 2007. — P. 139–158. — Режим доступу : http://philpapers.org/archive/OBRTEO-6.pdf.
370. Official Google Blog: Square your search results with Google Squared [Електронний ресурс]. — Режим доступу : http://googleblog.blogspot.com/ 2009/06/ square-your-search-results-with-google.html.
371. Ontolingua Reference Manual Ontolingua. Laboratory for Applied Ontology of the Institute of Cognitive Science and Technology Italian National Research Council [Електронний ресурс]. — Режим доступу : http://www.loa-cnr.it/medicine/reference-manual/index.html.
372. Ontolingua [Електронний ресурс]. — Режим доступу : http://www.ksl. stanford.edu/software/ontolingua/.
373. Ontolingua. Artificial Intelligence Laboratory of the Department of Computer Science at Stanford University [Електронний ресурс]. — Режим доступу : http://www.ksl.stanford.edu/software/ontolingua/.
374. Ontology Summit 2013 Communique. Towards Ontology Evaluation across the Life Cycle. Version vl.0.4, 2013.05.31 [Електронний ресурс]. . – Режим доступу : http://ontolog.cim3.net/cgi-bin/wiki.pl7OntologySummit2013_Communique.
375. Open Source Initiative [Електронний ресурс]. — Режим доступу :

http://opensource.org.
376. Operational Conceptual Modelling Language [Електронний ресурс]. — Режим доступу : http://technologies.kmi.open.ac.uk/ocml/.
377. OSINT (Open Source Intelligence) – Online Strategies [Електронний ресурс]. — Режим доступу : http://www.onstrat.com/osint/.
378. OWL 2 Web Ontology Language Document Overview. W3C. 2009 [Електронний ресурс]. — Режим доступу : http://www.w3.org/TR/owl2-overview/.
379. OWL 2 Web Ontology Language Profiles (Second Edition). W3C Recommendation, 2012 [Електронний ресурс]. — Режим доступу : http:// www. w3. org/TR/owl2-profiles/.
380. OWL 2 Web Ontology Language. Document Overview. W3C Recommendation, 2009 [Електронний ресурс]. — Режим доступу : http:// www. w3. org/TR/owl2-overview/.
381. OWL Web Ontology Language Semantics and Abstract Syntax. Section 2. Abstract Syntax [Електронний ресурс]. — Режим доступу : http://www.w3. org/ TR/owl-semantics/syntax.html.
382. OWL Web Ontology Language. Overview. W3C Recommendation: W3C, 2009 [Електронний ресурс]. — Режим доступу : http://www.w3.org/TR/owl-features/.
383. OWL-S: Semantic Markup for Web Services [Електронний ресурс]. — Режим доступу : http://www.w3.org/Submission/OWL-S/.
384. Pak A. Twitter as a Corpus for Sentiment Analysis and Opinion Mining / A. Pak., P. Paroubek // LREC. — 2010. — Vol. 10. — P. 1320—1326.
385. Pang Bo and Lillian Lee. Opinion mining and sentiment analysis / Pang Bo and Lillian Lee // Foundations and trends in information retrieval, 2. 1-2, 2008. — P. 1—135.
386. Parsia B. Pellet: An owl dl reasoner [Електронний ресурс] / B. Parsia, E. Sirin // Third International Semantic Web Conference-Poster. — 2004. — V. 18. — Режим доступу : http://iwayan.info/Research/ Ontology/ Papers_Research/ Reasoner/ Parsia_PelletOWLDLReasoner.pdf.
387. Pedersen T. WordNet: Similarity – measuring the relatedness of concepts / Pedersen T., Patwardhan S., Michelizzi J. // Proc. of the Nineteenth National Conference on Artificial Intelligence (AAAI-04), 2004. — P. 1024—1025.
388. Powell A. Guidelines for implementing Dublin Core in XML. 2002 [Електронний ресурс] / A. Powell, P. Johnston. — Режим доступу : http://dublincore.org/documents/2002/09/09/dc-xml-guidelines/.
389. Protege Plugin Library [Електронний ресурс]. — Режим доступу : http:// protegewiki.stanford.edu/wiki/Protege_Plugin_Library. \
390. Protégé [Електронний ресурс]. — Режим доступу : http:// protege.

stanford.edu/.
391. Protégé. W3C Semantic Web [Електронний ресурс]. — Режим доступу : http://www.w3.org/2001/sw/wiki/Protégé.
392. Quinlan J. R. Discovery rules from large collections of examples: a case study / J. R. Quinlan // Expert Systems in the Microelectronic Age. — Edinburg, 1979. – P. 87—102.
393. QUOnto Querying ONTOlogies [Електронний ресурс]. — Режим доступу : http://www.dis.uniromal. it /quonto/q=node/26.
394. Raman M. Designing knowledge management systems for teaching and learning with wiki technology [Електронний ресурс] / Raman M., Ryan T., Olfman L. // Journal of Information Systems Education. — 2005. — Vol. 16(3). — P. 311—320. — Режим доступу : http://fantasticfour.pbworks.com/f/designing_km_teaching_wiki.pdf.
395. Raygor Readability Estimate [Електронний ресурс]. — Режим доступу : http://en.wikipedia.org/wiki/Raygor_Estimate_Graph.
396. RDF Vocabulary Description Language 1.0: RDF Schema. RDF Vocabulary Description Language 1.0. World Wide Web Consortium, 2010 [Електронний ресурс]. – Режим доступу : http://www.w3.org/Metadata/Activity.html.
397. Resource Description Framework (RDF) Model and Syntax Specification. W3C Proposed Recommendation. January 1999 [Електронний ресурс]. — Режим доступу : http://www.w3.org/TR/PR-rdf-syntax.
398. Ricci F. Recommender Systems Handbook / Ricci F., Rokach L., Shapira B., Kantor P. — Springer, 2011. — 842 p.
399. RIF Overview. W3C Editor's Draft, 2009 [Електронний ресурс]. — Режим доступу : http://www.w3.org/2005/rules/wg/draft/ED-rif-overview-20090929/all.pdf.
400. RIF Use Cases and Requirements [Електронний ресурс]. — Режим доступу : http://www.w3.org/TR/2006/WD-rif-ucr-20060710/.
401. Rodriguez-Muro M. Realizing Ontology Based Data Access: A Plug-in for Protege [Електронний ресурс] / Rodriguez-Muro M., Lubyte L., Calvanese L. — Режим доступу : http://www.inf.unibz.it/lubyte/pub/iimas08.pdf.
402. Rogushina J. Ontological Approach to Domain Knowledge Representation for Informational Retrieval in Multiagent Systems / J. Rogushina, A. Gladun // International Jornal «Information Theories & Applications». — 2006. — Vol. 13. — № 4. — P. 354—362.
403. Rogushina J. Ontology-based competency analyses in new research domains / J. Rogushina, A. Gladun // Journal of Computing and Information Technology. — 2012. – V. 20. — № 4. — P. 277—293.
404. Ronchetti M. & Sant J. Curriculum Management and Review: an

ontology-based solution [Електронний ресурс] // In T. Bastiaens & S. Carliner (Eds.), Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education. — Chesapeake, VA: AACE, 2007. — P. 6476– 6482. — Режим доступу : http://eprints.biblio.unitn.it/1195/1/dtr-07-021.pdf.
405. Sabou M. Exploring the semantic web as background knowledge for ontology matching / Sabou M., d'Aquin M., Motta E. // Journal on Data Semantics, 2008.
406. Sabou M. Learning domain ontologies for semantic Web service descriptions / Sabou M., Wroe C., Goble C., Stuckenschmidt H. // Journal of Web Semantics. — 2005. — V. 3. — № 4. — P. 340—365.
407. Sarma A. Bootstrapping pay-as-you-go data integration systems / Sarma A., Dong X., Halevy A. // Proc.of SIGMOD, 2008.
408. Satnam A. Collective intelligence in action [Електронний ресурс] / A. Satnam. — New York : Manning, 2009. — 397 p. — Режим доступу : http:// www.manning.com.p.hostingprod.com/ alag/ch2_alag.pdf.
409. Schreurs J. Use of mereological approach for ontological constructing in education tasks [Електронний ресурс] / Schreurs J., Gladun A., Rogushina J. // Інформаційні технології в освіті та науці: збірник наукових праць. — Мелітополь, 2015. — P. 237—242. — Режим доступу : http://ito.mdpu.org.ua/index.php/partneri.
410. Seidman C. Data Mining with Microsoft SQL Server 2000 Technical Reference [Електронний ресурс] / C. Seidman. — Microsoft Press, 2001. — Режим доступу : http://dl.acm.org/citation.cfm?id=516986.
411. Semantic Knowledge Management Framework [Електронний ресурс]. — Режим доступу : http://www.cognitum.eu/semantics/Ontorion/.
412. Semanticwebsearch, Semantic Web Search Engine [Електронний ресурс]. — Режим доступу : http://www.semanticwebsearch.com.
413. Shearer R. HermiT: A Highly-Efficient OWL Reasoner [Електронний ресурс] / Shearer R., Motik B., Horrocks I. // OWLED. — 2008. — V. 432. — Режим доступу : http://owl1-1.googlecode.com/svn/!svn/bc/608/trunk/ www. webont. org/ owled/2008/papers/owled2008eu_submission_12.pdf.
414. Shoham Y. Agent-oriented programming / Y. Shoham // Artificial Intelligence. — 1993. — № 60. — P. 51—92.
415. Shvaiko P. Ten challenges for ontology matching / P. Shvaiko, J. Euzenat // Proc. of The 7th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE), 2008.
416. Siadaty M. Self-regulated Workplace Learning: A Pedagogical Framework and Semantic Web-based Environment / Siadaty M.,

Gašević D., Jovanović J., Pata K., Milikić N., Holocher-Ertl T., Jeremić Z., Ali L., Giljanović A. & Hatala M. // Educational Technology & Society. — 2012. — Vol. 15(4). — P. 75—88.
417. Simperl E. Ontology Engineering: A Reality Check / E. Simperl, C. Tempich // Proc.of the 5th International Conference on Ontologies, Databases and Applications of Semantics ODBASE-2006, 2006.
418. Simperl E. A Methodology for Ontology Learning in Frontiers in Artificial Intelligence and Applications / Simperl E., Tempich C., Vrandečić D. // Proc. of the 2008 Conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge, 2008. — P. 225—249.
419. Sindice, Semantic Web Search Engine [Електронний ресурс]. — Режим доступу : http://sindice.com/.
420. Sirin E. Pellet: A practical owl-dl reasoner / Sirin E., Parsia B., Grau B. C., Kalyanpur A., Katz Y. // Web Semantics: science, services and agents on the World Wide Web, 2007, 5(2). — P. 51—53.
421. SOAP (Simple Object Access Protocol) [Електронний ресурс]. — Режим доступу : http://searchsoa.techtarget.com/definition/SOAP.
422. SOAP Version 1.2, 2007 [Електронний ресурс]. — Режим доступу : http://www.w3.org/TR/soap/
423. Soubbotin D. Interview of SenseBot, Summarization, the Answer to Web Search [Електронний ресурс] / D. Soubbotin // Search Engine Journal, 2007. — Режим доступу : http://www.searchenginejournal.com/ summarization-the-answer-to-web-search-interview-with-dmitri-soubbotin-of-sensebot/6094/.
424. Soumen C. Mining the Web: Discovering knowledge from hypertext data. Morgan Kaufmann, 2003. — 345 p.
425. Souzis A. Building a semantic wiki. Intelligent Systems [Електронний ресурс] / A. Souzis // IEEE, 2005, 20(5). — P. 87—91. — Режим доступу : http:// ieeexplore.ieee.org/xpl/ login.jsp?tp=&arnumber=1512004&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1512004.
426. Sowa J. Mathematical Background (Predicate Logic)[Електронний ресурс] / J. Sowa. — Режим доступу : http://www.jfsowa.com/logic/math.htm.
427. SPARQL current status [Електронний ресурс]. — Режим доступу : http://www.w3.org/standards/techs/sparql#w3c_all.
428. SPARQL Query Language for RDF. W3C Recommendation, 2008 – http://www.w3.org/TR/rdf-sparql- query/.
429. Staab S. Knowledge Processes and Ontologies / Staab S., Schnurr H., Studer R, Sure Y. // IEEE Intelligent Systems 2001, 16(1). — P. 26—34.
430. Stanisław Leśniewski [Електронний ресурс]. — Режим доступу :

http://en.wikipedia.org/wiki/Stanisław_Leśniewski.
431. Stanislaw Lesniewski [Електронний ресурс]. — Режим доступу : http://www-history.mcs.st-andrews.ac.uk/Biographies/Lesniewski.html.
432. Studer R. Knowledge Engineering: Principles and Methods / Studer R., Benjamins V. R., Fensel D. // In Data & Knowledge Engineering, 1998, 25. — P. 161—197.
433. Suggested Upper Merged Ontology (SUMO) [Електронний ресурс]. — Режим доступу : http://www.ontologyportal.org/index.html.
434. Swoogle, Semantic Web Search Engine [Електронний ресурс]. — Режим доступу : http://swoogle.umbc.edu/.
435. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. [Електронний ресурс]. — Режим доступу : http://www.w3.org/Submission/SWRL.
436. SWRL-IQ (Semantic Web Rule Language Inference and Query tool) [Електронний ресурс]. — Режим доступу : https://www. onistt. org/ display/ SWRLIQ/SWRL-IQ.
437. SWSE Semantic Web Search Engine [Електронний ресурс]. — Режим доступу : http://swse.deri.org/.
438. Sycara K. Multiagent systems [Електронний ресурс]. — Режим доступу : http://www.aaai.org/Pathfinder/html/multiagent_systems.htm.
439. Talia D. How Distributed Data Mining Tasks can Thrive as Knowledge Services / D. Talia, P. Trunfio // Communications of the ACM. — 2010. — V. 53, № 7. – P. 132—137.
440. Tarski A. Logic, Semantics, Metamathematics / A. Tarski. — Oxford University Press, 1956. — 258 p.
441. The Upper Cyc Ontology [Електронний ресурс]. — Режим доступу : http://www.cyc.com/cyc-2-1/index.html.
442. Thomas E., Pan J. Z., Ren Y. TrOWL: Tractable OWL 2 reasoning infrastructure / Thomas E., Pan J. Z., Ren Y. // The Semantic Web: Research and Applications, 2010. — P. 431—435.
443. TopBraid Composer [Електронний ресурс]. — Режим доступу : http://semanticweb.org/wiki/TopBraid_Composer.
444. Tudorache T. Collaborative Protege: Enabling Community-based Authoring of Ontologies [Електронний ресурс] / Tudorache T., Noy N. F., Musen M. A. // International Semantic Web Conference, 2008. — Режим доступу : http://ceur-ws.org/Vol-401/iswc2008pd_submission_60.pdf.
445. Turhan A.-Y. DIG 2.0 – Towards a Flexible Interface for Description Logic Reasoners [Електронний ресурс] / Turhan A.-Y., Bechhofer S., Kaplunova A., Liebig T., Luther M., Moller R., Noppens O., Patel-Schneider P., Suntisrivaraporn B., Weitho-ner T. — Режим доступу : http://www.owllink.org/publications/dig20-OWLED06.pdf.

446. UDDI (Universal Description, Discovery, and Integration) [Електронний ресурс]. — Режим доступу : http://searchsoa.techtarget.com/definition/UDDI.
447. UDDI Tutorial [Електронний ресурс]. — Режим доступу : http:// www. tutorialspoint.com/uddi/.
448. Uschold M. Knowledge Level Modeling: Concepts and Terminology / M. Uschold // In The Knowledge Engineering Review. — 1998. — Vol. 13:1. — P. 5– 29.
449. Uschold M. Ontologies: Principles, Methods and Applications / M. Uschold, M. Grüninger // Knowledge Engineering Review. — 1996. — № 11(2). — P. 93—155.
450. Using Dublin Core [Електронний ресурс]. — Режим доступу : http:// dublincore.org/documents/usageguide/.
451. Van Heijst G. Using Explicit Ontologies in KBS Development / Van Heijst G., Schreiber A. T., Wielinga B. J. // In International Journal of Human and Computer Studies. — 1996. — № 46(2-3). — P. 183—292.
452. Vargas-Vera M. MnM: Ontology driven tool for semantic markup [Електронний ресурс] / Vargas-Vera M., Motta E., Domingue J., Lanzoni M., Stutt A., Ciravegna F. // Proc. Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002), 2002. — P. 43—47. — Режим доступу : http://citeseerx.ist. psu. edu/ viewdoc/summary?doi=10.1.1.20.3590
453. Veale T. A context-sensitive framework for lexical ontologies / T. Veale, Y. Hao // Knowledge Engineering Review. — 2007. — Vol. 23(1). — P. 101—115.
454. Vega-Gorgojo G. A semantic approach to discovering learning services in grid-based collaborative systems, Future Generation / Vega-Gorgojo G., Bote-Lorenzo M. L., Gomez-Sanchez E., Dimitriadis Y. A., Asensio-Perez J. I. // Computer Systems 2006, 22. — P. 709—719.
455. W3C Semantic Web Activity [Електронний ресурс]. — Режим доступу : http://www.w3.org/2001/sw/Activity/.
456. Wagner C. Wiki: A technology for conversational knowledge management and group collaboration [Електронний ресурс] / C. Wagner // The Communications of the Association for Information Systems. — 2004. — V. 13(1). — P. 264—289. — Режим доступу : http://aisel.aisnet.org/cgi/viewcontent.cgi? article= 3238& context= cais.
457. Walker D. W. The Grid, Virtual Organizations, and Problem-Solving Environments, Third IEEE International Conference on Cluster Computing [Електронний ресурс] / D. W. Walker. — Режим доступу : http://www. computer. org/portal/web/csdl/doi/10.1109/CLUSTR.2001.960011.
458. Warren P. Knowledge Management and the Semantic Web: From

Scenario to Technology / P. Warren // IEEE Intelligent Systems. — 2006. — V. 21. — № 1. — P. 53—59.
459. Watson H. J., Wixom B. H. The current state of business intelligence / H. J. Watson, B. H. Wixom // Computer. — 2007. — 40(9). — P. 96—99.
460. WatsOn, Semantic Web Search Engine [Електронний ресурс]. — Режим доступу : http://watson.kmi.open.ac.uk/WatsonWUI/.
461. Web Services Activity [Електронний ресурс]. — Режим доступу : http:// www.w3.org/2002/ws/.
462. Web Services Architecture Working Group [Електронний ресурс]. — Режим доступу : http://www.w3.org/2002/ws/arch/.
463. Web Services Description Language (WSDL) 1.1. [Електронний ресурс]. – Режим доступу : http://www.w3.org/TR/wsdl.
464. Weber S. The success of open source / S. Weber // Cambridge, MA: Harvard University Press. — 2004. — Vol. 897. — P. 332—344.
465. What is Schema.RDFS.org? [Електронний ресурс]. — Режим доступу : http://schema.rdfs.org.
466. Wielinga B. Framework and Formalism for Expressing Ontologies (Version 1) / Wielinga B., Schreiber A. T., Jansweijer W., Anjewierden A. van Harmelen F. // ESPRIT Project 8145 KACTUS, Free University of Amsterdam Deliverable, DO1b.1, 1994.
467. Wielinga B. J. & Schreiber A. T. Reusable and Sharable Knowledge Bases: a European Perspective / B. J. Wielinga & A. T. Schreiber // In Proceedings International Conference on Building and Sharing of Very Large-Scaled Knowledge Bases, Tokyo, Japan Information Processing Development Center, 1993. — P. 103—115.
468. WikipediA [Електронний ресурс]. — Режим доступу : https:// www. wikipedia.org.
469. Wolfram S. Wolfram Alpha computational knowledge engine, 2009 [Електронний ресурс]. — Режим доступу : http:// basetechnology. blogspot. com/ 2009/03/wolfram-alpha-computational-knowledge.html.
470. WSDL Tutorial [Електронний ресурс]. — Режим доступу : http:// www. tutorialspoint.com/wsdl/.
471. XML Schema Part 0: Primer, W3C Recommendation, 2.05.2001 [Електронний ресурс]. — Режим доступу : http://www.w3.org/TR/xmlschema-0/.
472. XML Schema Requirements, W3C Note 15.02.1999 [Електронний ресурс]. – Режим доступу : http://www.w3.org/TR/NOTE-xml-schema-req.
473. XML Tutorial [Електронний ресурс]. — Режим доступу : http:// www. w3schools.com/xml/
474. Zhang S. Experience in aligning anatomical ontologies / Zhang S. and

Bodenreider O. // International Journal on Semantic Web and Information Systems, 2007.
475. Zhdanova A. Community-driven ontology matching / A. Zhdanova, P. Shvaiko // Proc. of ESWC, 2006.
476. Zhu J. Integrating multiple document features in language models for expert finding / Zhu J., Huang X., Song D., Rüger S. // Knowledge and Information Systems, 2010, 23(1). — P. 29—54.
477. Zhu X. Incorporating quality metrics in centralized/distributed information retrieval on the Worls Wide Web, 2000 [Електронний ресурс] / X. Zhu, S. Gauch. — Режим доступу : http://citeseer.nj.nec.com/zhu00incorporating.html.
478. Zhuge H. China's e-Science Knowledge Grid Environment [Електронний ресурс] / H. Zhuge // IEEE Intelligent Systems. — 2004. — V. 19. — Режим доступу : http://www.worldscibooks.com/compsci/5738.html.
479. Zhuge H. Semantic-based query routing and heterogeneous data integration in peer-to-peer semantic link networks / Zhuge H., Jie Liu, Liang Feng, Chao He // ICSNW, 2004. — P. 91—107.

Наукове видання

Рогушина Ю.В., Гладун А.Я., Осадчий В.В., Прийма С.М.

# ОНТОЛОГІЧНИЙ АНАЛІЗ У WEB

Монографія

Підписано до друку 27.11.2015 р., Формат 60*84/16
Папір офсетний. Гарнітура Times New Roman.
Друк цифровий. Ум. друк. арк. 23,83
Наклад 300 примірників. Замовлення № 1513

Видавець
Мелітопольський державний педагогічний університет
імені Богдана Хмельницького
Адреса: 72312, м. Мелітополь, вул. Леніна, 20
Тел. (0619) 44 04 64
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виробників і розповсюджувачів
видавничої продукції від 16.05.2012 р. серія ДК № 4324

Надруковано ФО-П Однорог Т.В.
72313, м. Мелітополь, вул. Героїв Сталінграду, 3а
Тел. (067) 61 20 700
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виробників і розповсюджувачів
видавничої продукції від 29.01.2013 р. серія ДК № 4477