Міністерство освіти і науки України
Національний університет “Львівська політехніка”

На правах рукопису

Вовнянка Роман Володимирович

УДК 004.832.2 : 004.853

МЕТОДИ ТА ЗАСОБИ ПЛАНУВАННЯ ДІЙ СПЕЦІАЛІЗОВАНИХ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ

Спеціальність 01.05.03 – математичне та програмне забезпечення обчислювальних машин і систем

Дисертація на здобуття вченого ступеня кандидата технічних наук

Науковий керівник:
д.т.н., професор кафедри ІСМ
Литвин Василь Володимирович

Львів - 2017

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

АП – автоматизоване планування,

БЗ – база знань,

ІА – інтелектуальний агент,

ПІП – підсистема інформаційного пошуку,

ПТД – природномовний текстовий документ,

ПО – предметна область,

СДО – суб’єкт-дія-об’єкт,

DL – описова логіка,

MDP – Марківські процеси прийняття рішень,

POMDP – Марківські процеси прийняття рішень в частково спостережуваному середовищі,

PEAS – опис проблемного середовища,

CAPO – система автоматизованої розбудови онтології,

O – онтологія,

$\hat{O}$ – адаптивна онтологія,

C – множина понять онтології,

R – множина відношень онтології,

F – множина функцій інтерпретації елементів онтології,

W – вага поняття адаптивної онтології,

L – вага відношення адаптивної онтології,

S – множина станів,

A – множина дій,

p – ймовірність,

π – стратегія,

π^* – оптимальна стратегія,

V – оцінка корисності стану,

X – множина ознак.

ВСТУП

Актуальність теми. Наукові дослідження в області розроблення і функціонування інтелектуальних агентів (ІА) полягають у створенні математичних моделей та методів побудови інформаційних систем, які орієнтовані на ті сфери діяльності людини, що вимагають логічного міркування, певної майстерності та досвіду, тобто базуються на знаннях. Клас таких прикладних задач включає планування та моніторинг діяльності; прогнозування та класифікація явищ тощо.

Основною компонентою інтелектуальних агентів є база знань, що формується відповідно до предметної області на яку зорієнтоване функціонування цієї системи. Як стандарт інженерії знань, розробники інтелектуальних агентів, використовують онтологічний інжиніринг. Результатом такого інжинірингу є онтологія бази знань. Онтологія – це детальна формалізація деякої області знань подана за допомогою концептуальної схеми. Така схема складається з ієрархічної структури понять, зв’язків між ними, теорем та обмежень, які є прийняті у певній предметній області.

Наукові дослідження в напрямі використання онтологій під час розроблення та функціонування ІА, почалися в кінці минулого століття та інтенсивно розвиваються. Основні теоретичні засади формальних математичних моделей онтологій розроблено у роботах Т.Грубера, Дж.Солтона, А.Гомес-Переса, які запропонували онтологію розглядати як тривимірний кортеж; використання онтологій під час функціонування прикладних інформаційних систем описано в роботах Р.Кнаппе, К.Джонса, Е.Кауфмана, Е.Мена, М.Бориса, А.Каллі, І.П.Норенкова, М.Ю.Уварова, Ю.В.Рогушина; проблему побудови інтелектуальних систем на основі онтологій розглянуто у роботах Т. Андреасена, Т.Бернерса-Лі, Д.Хендлера, О.Лазсіла, О.В.Палагіна, А.В.Анісімова, А.Я.Гладуна.

Особливу увагу серед ІА заслуговують агенти планування діяльності, оскільки багато прикладних задач зводяться до задачі планування (оптимальний розподіл ресурсів, раціональна поведінка, економічний розвиток об’єкту в часі, підвищення експлуатації окремих засобів тощо). Аналіз основних підходів, методів та засобів побудови інтелектуальних агентів планування діяльності показує, що у складі таких систем використовуються не всі можливості онтологій, особливо під час моделювання функціональності таких систем. Поведінка таких систем зводиться до пошуку оптимального шляху в просторі станів, однак не очевидно як такий пошук здійснювати. Пошук оптимального шляху повинен ґрунтуватись на правилах (законах), які задаються в межах певної предметної області. Для формалізації таких правил пропонується використовувати онтології.

Виникає завдання розроблення та запровадження уніфікованих методів побудови інтелектуальних агентів планування діяльності з використанням онтологічного підходу з метою підвищення ефективності процесів функціонування таких систем. У дисертаційній роботі подано вирішення цього завдання у вигляді теоретично обґрунтованих моделей функціонування та методів побудови ІА планування діяльності на основі онтологій, суть яких полягає в адаптації баз знань цих систем до специфіки задач відповідної предметної області, а також методів автоматизованої розбудови онтологій.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційну роботу виконано в межах наукового напряму «Нові комп’ютерні засоби та технології інформатизації суспільства» визначеного пріоритетним у переліку актуальних проблем Міністерством освіти і науки України, концепції програми інформатизації НАН України, визначеної пріоритетним напрямом, згідно розпорядження № 146 від 27.02.2004 р. та за тематикою наукових досліджень кафедри інформаційних систем і мереж Національного університету «Львівська політехніка», зокрема за темою: «Розроблення інтелектуальних розподілених систем на основі онтологічного підходу з метою

інтеграції інформаційних ресурсів» № держреєстрації 0115U004228 (автор розробив метод побудови інтелектуальних агентів планування діяльності на основі онтологій, що дало змогу підвищити ефективність функціонування інтелектуальних систем бізнес-аналітики).

Окрім того, отримані результати використано під час виконання науково-дослідної роботи у Фізико-механічному інституті ім. Г .В. Карпенка НАН України «Розроблення інформаційних технологій автоматизованого синтезу онтології матеріалознавства» (2011–2013рр.), номер державного реєстру 0111U002382 (автор спроектував та реалізував визначення окремих елементів онтології фізико-хімічної механіки матеріалів за допомогою дескриптивної логіки).

Мета і задачі дослідження. Метою роботи є розроблення математичного та програмного забезпечення функціонування спеціалізованих інтелектуальних агентів планування дій з використанням онтологічного підходу.

Метою дисертаційної роботи визначено необхідність виконання таких задач:

- провести аналіз специфіки функціонування спеціалізованих інтелектуальних агентів та методів побудови планування їх дій;
- розробити математичне забезпечення планування дій спеціалізованих інтелектуальних агентів з використанням онтологічного підходу (побудова простору станів та пошук у ньому шляху переходу);
- розробити метод автоматизованого наповнення онтологій та на його основі реалізувати відповідне програмне забезпечення;
- розробити архітектуру програмного комплексу планування дій спеціалізованих інтелектуальних агентів;
- провести апробацію отриманих результатів шляхом розроблення та впровадження прикладних спеціалізованих інтелектуальних агентів планування дій.

Об'єктом дослідження є процес побудови інтелектуальних агентів планування діяльності.

Предметом дослідження є методи та засоби побудови спеціалізованих інтелектуальних агентів планування діяльності на основі онтологічного підходу.

Методи дослідження. Для досягнення поставленої мети використано: теорію множин, теорію графів та методи подання знань для моделювання структури онтології та розроблення процедур її автоматизованої розбудови; теорію формальних систем та функціонального аналізу для побудови моделей функціонування інтелектуальних агентів планування діяльності; методи системного аналізу, методи об'єктно-орієнтованого аналізу і проектування – для розроблення прикладних інтелектуальних агентів планування діяльності; теорію реляційних баз даних, методи штучного інтелекту, об'єктно-орієнтоване програмування – для програмної реалізації розроблених моделей, методів та алгоритмів функціонування прикладних інтелектуальних агентів планування дій.

Наукова новизна одержаних результатів. Наукова новизна одержаних результатів полягає в науковому обґрунтуванні та вирішенні завдання підвищення ефективності функціонування інтелектуальних агентів планування дій шляхом використання онтологій у цих системах і методів стимулюючого навчання. Отримано такі нові наукові результати:

- удосконалено математичне забезпечення функціонування спеціалізованих інтелектуальних агентів планування дій шляхом використання онтології предметної області, в межах якої функціонує інтелектуальний агент, яке, на відміну від існуючих, задає простір станів та переходів між ними на основі онтологічних знань, що дало змогу звести задачу функціонування інтелектуальних агентів планування діяльності до задачі динамічного програмування;

- вперше побудовано критерій раціональної поведінки інтелектуального агента на основі методів стимулюючого навчання, що, на відміну від інших підходів, дає змогу формалізувати процес функціонування інтелектуальних агентів планування дій, ядром бази знань яких є онтології;
- одержав подальший розвиток метод автоматизованої розбудови онтологій предметної області для задач планування дій шляхом врахування міри довіри до джерела інформації, яке використовується для розбудови онтології, що дало змогу будувати простір станів, релевантний до бази знань спеціалізованої предметної області.

Наукове значення результатів, отриманих у дисертаційній роботі, полягає в розробленні методу функціонування інтелектуальних агентів планування дій з використанням онтологічного підходу. Автор використав методи стимулюючого навчання для задання критерію оптимальної поведінки такого агента. Розроблені методи та засоби дали можливість будувати ефективні інтелектуальні агенти планування дій у тих предметних областях, в яких знання чітко формалізуються за рахунок онтологій.

Практичне значення одержаних результатів. Практичну цінність отриманих наукових результатів дисертаційної роботи підтверджує те, що завдяки використанню розроблених методів підвищується ефективність функціонування ІА планування дій. Зокрема, практично цінними є такі результати:

- врахування міри довіри до джерела інформації дає змогу зменшити простір станів, у якому здійснюється пошук шляху розв’язку задачі планування;
- застосування процедур автоматизованої розбудови онтологій природно-мовними текстами суттєво розширює сферу використання таких онтологій та зменшує затрати на їх реалізацію.

Результати дисертаційної роботи упроваджено під час розроблення віртуального автоматизованого робочого місця наукового працівника у Фізико-механічному інституті ім. Г. В. Карпенка НАН України (м. Львів), а також

теоретичні та практичні результати дослідження використовуються у навчальному процесі кафедр «Інформатики і математичного моделювання» та «Комп'ютерних наук» факультету комп'ютерно-інформаційних систем та програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя. Окремі результати дисертаційного дослідження використовуються у Відокремленому структурному підрозділі Золочівський коледж Національного університету «Львівська політехніка» при викладанні дисциплін «Організація баз даних та знань», «Об'єктно-орієнтоване програмування», що підтверджено відповідними актами.

Особистий внесок здобувача. Усі наукові результати, подані в дисертації, одержані здобувачем особисто. У друкованих працях, опублікованих у співавторстві, особистий внесок здобувача такий: [1, 2] – розробка методів та засобів для побудови онтології; [3, 4, 13] – побудова петлі Бойда, центральною компонентою якої є онтологія; [5] – дослідження методів і засобів для моделювання поведінки раціонального агента на основі стимулюючого навчання; [6, 7] – розроблено методи та засоби для автоматизованої розбудови онтології; [8] – розроблено математичне забезпечення функціонування інтелектуальних агентів, центральною компонентою яких є онтологія; [9, 10, 11] – моделювання поведінки петлі OODA при взаємодії з онтологією; [12, 14, 15] – дослідження методів для розробки онтології.

Апробація результатів дисертації. Основні результати дисертаційної роботи доповідалися на міжнародних, українських та міжвузівських конференціях та семінарах, зокрема на: II та IV міжнародних наукових конференціях „Інформація, комунікація, суспільство" (ІКС-2013 та ІКС-2015), Львів-Славське, 16-19 травня 2013 р. та 20-23 травня 2015 р.; Міжнародних науково-практичних конференціях „Інформаційні технології. Освіта", Луцьк-Світязь, 3-4 червня 2013 р. та 6-8 червня 2014 р.; Міжнародній науковій конференції „Інтелектуальні системи прийняття рішень і проблеми

обчислювального інтелекту", Залізний Порт – Херсон, 2014 р.; XIIIth International Conference „The Experience of Designing and Application of CAD Systems in Microelectronics", Polyana-Svalyava, 24-27 лютого 2015 р.; Десятій міжнародній науково-практичній конференції „Математичне та імітаційне моделювання систем" (МОДС 2015), Чернігів, 22-26 червня 2015 р.

Результати дисертаційного дослідження регулярно доповідалися на наукових семінарах кафедри «Інформаційні системи та мережі» Національного університету «Львівська політехніка» (2012-2015 рр.).

Публікації. Основні результати роботи відображені у 15 опублікованих працях, у тому числі 6 статей у наукових фахових виданнях України та 2 статті в наукових періодичних виданнях інших держав, що включені до наукометричних баз даних, 7 публікацій у збірниках тез конференцій.

РОЗДІЛ 1 . АНАЛІЗ МЕТОДІВ ПЛАНУВАННЯ ДІЙ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ

У першому розділі проведено аналіз методів функціонування інтелектуальних агентів, які здійснюють планування своїх дій. Виділено задачі, які не розв’язані, зокрема не розроблено ефективних методів та алгоритмів планування дій інтелектуальних агентів, які б давали змогу раціонально використовувати знання агента для максимізації його виграшу в процесі своїх дій.

1.1 Аналіз моделей планування дій інтелектуального агента

1.1.1 Основні поняття, терміни та означення

На сьогоднішній день практично в усіх галузях розширюється застосування складних автоматизованих систем, для виконання найрізноманітніших задач, зокрема задач планування.

Автоматизоване планування (англ. Automated planning, AP) – проблемна область штучного інтелекту, що стосується формування та виконання стратегії або послідовності дій необхідних для отримання максимального прибутку. На відміну від класичних проблем управління і функціонування, вирішення такого типу задач носить комплексний характер і повинно розроблятися та оптимізуватися з урахуванням різних аспектів. Однією із задач, де використовується автоматизоване планування є проектування методів опрацювання інформації, яке дозволяє ефективно використовувати ресурси. Ці задачі є предметом досліджень в галузях : штучного інтелекту, машинного навчання, теорії керування, теорії прийняття рішень, робототехніки тощо. На сьогоднішній день існує велика кількість підходів до побудови систем автоматичного планування, що пояснюється різноманіттям сфер їхнього застосування та вимог до характеристик конкретної системи.

Під час побудови інтелектуальних агентів (ІА) гостро постає проблема реалізації функції автоматичного планування [16]. В ролі ІА можуть виступати роботи, зонди, програмні системи керування, програми пошуку інформації і т.ін.

Загальну модель системи планування можна подати як процес взаємодії трьох компонентів (див. рис. 1.1):

1) планувальника;
2) агента;
3) середовища.

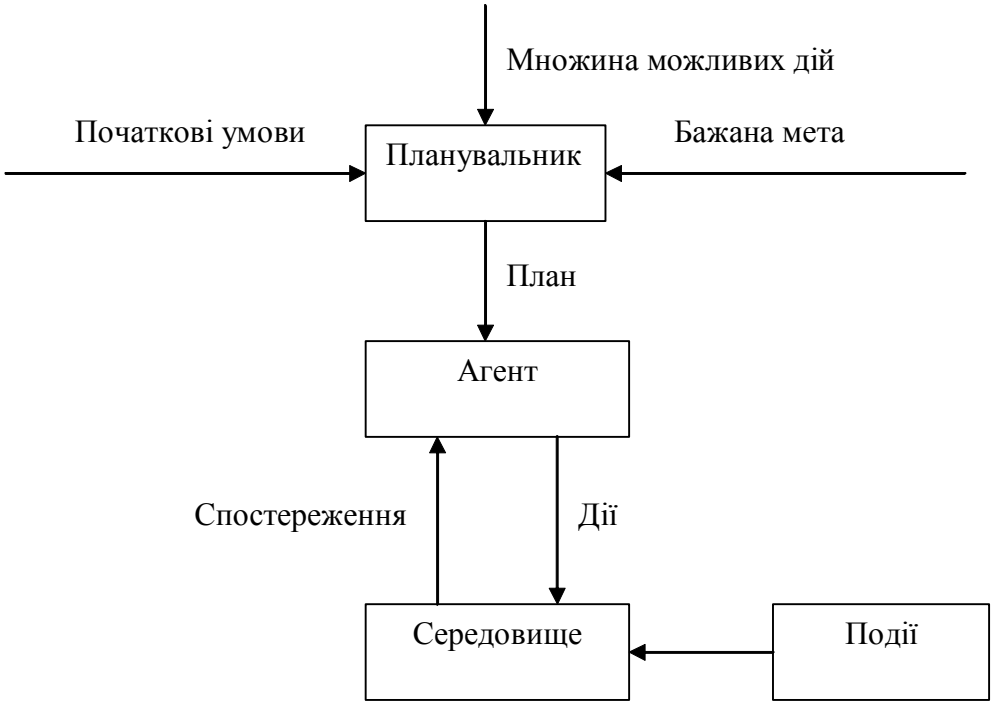

Рис. 1.1. Загальна модель системи планування

Планувальник встановлює послідовність дій, які переводять систему з початкового стану у стан, в якому досягається поставлена мета.

У типового планувальника три основні входи, а саме « початкові умови», «Множина можливих дій» (заданих формальною мовою) і «Бажана мета».

Вхід « Початкові умови» призначений для вводу в систему даних про стан та структуру середовища та об’єктів, які перебувають в ньому. На основі цих даних формується база знань відомостей про навколишній світ. Також до

цієї бази входять: перелік відомих та суттєвих факторів, які діють в даному середовищі, їх характеристики, сфера та сила впливу на об’єкти та дані про самого агента (потрібно наголосити, що агент теж є частиною середовища, а отже він повинен розглядати себе, як невід’ємну частину навколишнього світу).

На вхід «Множини можливих дій» подаються дані необхідні для проведення певного роду маніпуляцій агентом, які він може здійснювати у заданому середовищі. Кожна дія повинна бути належним чином описана і класифікована згідно визначених правил функціонування. Важливим фактором належного функціонування агента є правильне визначення пріоритету дій.

Вхід «Бажана мета» призначений для чіткого визначення даних про стан в який повинен перейти агент для досягнення поставленої мети. Також сюди необхідно вносити визначення прибутку, тобто ті параметри станів, які агент використовуватиме для побудови стратегії такого переходу.

Результатом роботи «Планувальника» є план – це набір директив поданих у формі зрозумілій агенту, виконавши які, той перейде в стан у якому відбудеться досягнення поставленої мети.

Агент здійснює безпосередню взаємодію з середовищем. Така взаємодія полягає у виконанні певних дій згідно отриманого від «Планувальника» плану та проведення дій в середовищі (див. рис. 1.1).

Середовище – це динамічна система, що змінюється під дією певних незалежних подій, які виникають як за певним законом, так і спонтанно, а також під впливом дій агента. На основі спостережень за станом середовища агент формує уявлення про стан у якому він зараз перебуває і виконує ті дії, які передбачені планом для цього стану, щоб перейти у наступний стан найближчий до стану мети.

Однак динамічні зміни середовища можуть бути настільки суттєвими, що попередньо розрахований план дій не дозволить досягнути мети. Виходом з цієї ситуації є динамічне планування. Його суть полягає у тому, щоб агент постійно передавав «Планувальнику» інформацію про стан системи, а той у свою чергу

створював новий план базуючись на оновленій інформації про середовище (див. рис. 1.2).

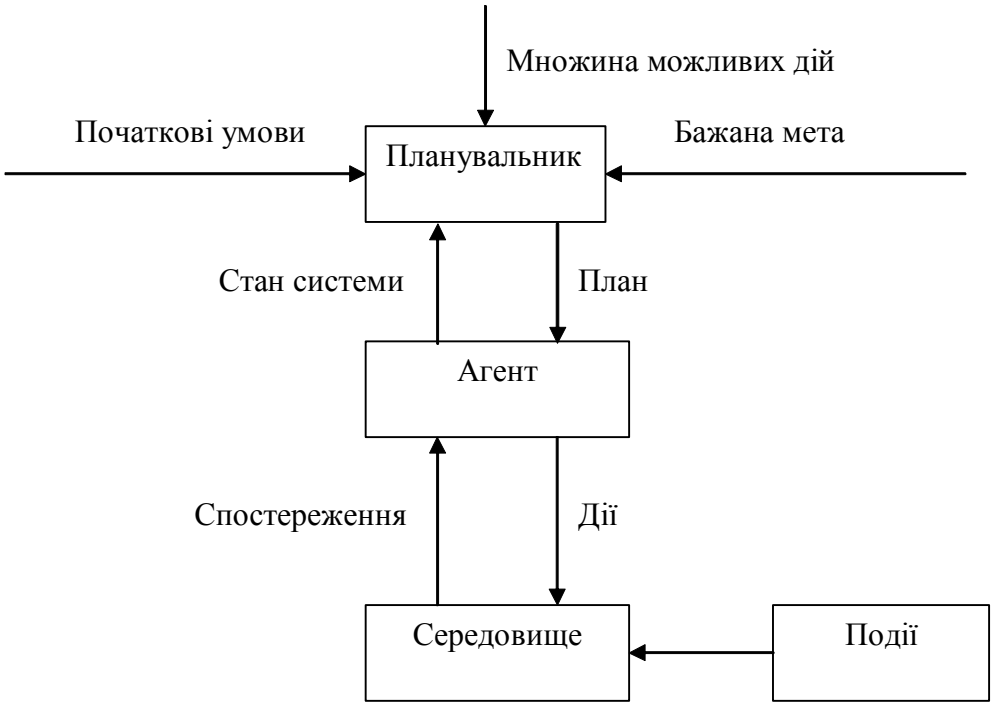

Рис. 1.2. Загальна модель системи динамічного планування

1.1.2 Планування в умовах невизначеності

Такий тип планування, який зазвичай називають класичним [17] має ряд обмежень, а саме:

1. Детермінізм. Тобто виконання будь-якої дії обов'язково переведе агента у новий стан середовища.
2. Повне спостереження середовища. Агент має повну інформацію про поточний стан усієї системи.
3. Досяжність мети. Мета – це множина станів, тобто план полягає в досягненні одного із станів мети.

1.1.2.1 Недетермінізм

Детермінізм – це спрощений погляд на світ, що базується на припущені, що його розвиток відувається за одним передбаченим шляхом. Тобто уся

динаміка середовища є прогнозованою та фіксованою під час діяльності агента. У деяких випадках детермінована модель системи може бути корисною абстракцією для дослідження основних типів поведінки. Також у детермінованій системі необхідно передбачити усі можливі події та фактори середовища, а усі не передбачені вважаються не суттєвими та не враховуються «Планувальником» для складання плану.

Недетермінована модель є адекватнішою, оскільки дозволяє планувальнику аналізувати декілька різних результатів дії, іноді серед яких зовсім нетипові, а також результати появи яких можна передбачити лише з певною імовірністю. Основною проблемою недетермінованих систем є те, що досягнення стану мети можна здійснити різними шляхами. Таким алгоритмам планування потрібні ефективні способи аналізу всіх можливих результатів дій та генерування планів за якими дії залежать від умов середовища.

1.1.2.2 Частково спостережуване середовище

У ряді ситуацій середовище є лиш частково спостережуваним під час виконання плану і як наслідок агент може розрізнити лише деякі стани. Найчастіше виникають такі ситуації:

- частину параметрів та факторів середовища агент ніколи не зможе спостерігати;
- частину параметрів та факторів агент може спостерігати лише в деяких станах або тільки після виконання певних дій.

Планування при частковому спостереженні середовища – це доволі складна в теоретичному, так і практичному плані задача,. Основний наслідок часткового спостереження полягає в тому, що результатами спостережень спочатку стає множина станів, а не певний конкретний стан. Кількість елементів отриманої множини не може перевищувати кількість станів середовища. У випадку планування з імовірностями, спостереження повертають розподіл ймовірностей переходу у стан.

1.1.2.3 Розширення стану мети

У недетермінованому середовищі опис мети потребує визначення різних параметрів для оцінки невизначеності після виконання дій. Також можливий як простий випадок розширення мети, а саме коли агент робить все, щоб величини певних параметрів досягли потрібного значення так і дещо складніший випадок, коли у продовж деякого періоду величини певних параметрів повинні обов'язково підтримуватися на певному рівні.

Розширення мети може бути представлене по-різному. Один з підходів – це подання мети функціями корисності. У цьому випадку планування складається з пошуку стратегії, яка залежно від ситуацій максимізує чи мінімізує функцію корисності. Інший підхід полягає в поданні стану мети за допомогою формул описової логіки (DL) [18].

1.1.3 MDP планування

Марківські процеси прийняття рішень (MDP) призначені для побудови плану діяльності агента у стохастичному середовищі з марківською моделлю переходів та множиною цілей. Їх поділяють на два типи: MDP Марківські процеси прийняття рішень у повністю спостережуваному середовищі та POMDP Марківські процеси прийняття рішень в частково спостережуваному середовищі, частковим випадком яких є MDP. Їх основна ідея полягає в поданні задачі планування як задачі оптимізації [16, 17].

У Марківських процесах прийняття рішень виділяють такі складові.

1. Середовище моделюється як стохастична система, тобто це система з недетермінованими переходами між станами, з розподілом функції імовірності на кожен перехід. Отже, дії (переходи) моделюються на основі розподілу функції імовірності.
2. Цілі визначаються значеннями функції корисності.
3. Плани подані у вигляді стратегій, які визначають ті дії, які потрібно виконувати в тому чи іншому стані.

4. Проблема планування розглядається як задача оптимізації, в якій алгоритми планування будують план, що максимізує функцію корисності.
5. Частково спостережуване середовище моделюється на основі імовірного стану. Імовірний стан – це розподіл імовірності перебування в деякому стані на усі можливі стани [19]. Проблема планування при частковому спостережуваному середовищі розв'язується як задача планування при повністю спостережуваному середовищі в просторі імовірнісного стану і створенні стратегій, які зіставляють імовірні стани з діями.

Середовище описують за допомогою трьох компонент: $S = \{s_1, s_2, ..., s_k\}$ – множина станів, $A = \{a_1, a_2, ..., a_n\}$ – множина дій, $P_s(s' \mid a)$ – імовірності переходу P з стану s в стан s' при виконанні дії a.

Результатом роботи планувальника є стратегія. Стратегія визначає ті дії, які потрібно виконувати в тому чи іншому стані. Стратегію π подають як функцію відображення множини станів на множину дій:

$$\pi : S \to A. \tag{1.1}$$

Якщо агент має повний опис стратегії, то йому завжди відомо, що робити далі незалежно від результату попередньої дії. При неодноразовому здійснені стратегії, починаючи з початкового стану, стохастичний характер середовища призводить до формування кожного разу іншої історії перебування агента в середовищі. Тому доцільно вважати за визначення ефективності стратегії очікувану корисність від можливих перебувань у середовищі. Оптимальна стратегія π^* – стратегія, яка забезпечує максимальну очікувану корисність [20].

Корисність стану s визначають як різницю прибутку $R(s)$ від перебування в стані s та затрат $C(s,a)$ на виконання дії a:

$$V(s,a) = R(s) - C(s,a). \tag{1.2}$$

Корисність стратегії $V(\pi)$ – це сума корисності станів історії побудованої згідно із стратегією π, де для кожного стану *s* виконується дія відповідно до стратегії $\pi(s)$

$$V(\pi) = \sum_{i \geq 0} V(s_i, \pi(s_i)). \quad (1.3)$$

Підставивши формулу (1.2) отримаємо

$$V(\pi) = \sum_{i \geq 0} (R(s_i) - C(s_i, \pi(s_i))). \quad (1.4)$$

Одна з проблем такого визначення у тому, що така функція корисності при нескінченній множині станів постійно наростатиме, що не дасть можливості порівняти різні історії стратегії. Поширений спосіб уникнення такої ситуації – це введення коефіцієнта знецінення γ, що описує перевагу поточних винагород над майбутніми і дає змогу навіть у нескінченній множині станів визначити підмножину, на основі якої можна чітко обчислити функцію корисності. Тоді формула (1.4) набуде такого вигляду:

$$V(\pi) = \sum_{i \geq 0} \gamma^i (R(s) - C(s_i, \pi(s_i))). \quad (1.5)$$

Отже, цінність будь-якої стратегії – це очікувана сума отриманих знецінених винагород. Звідси випливає, що оптимальна стратегія π^* – це стратегія з максимальною цінністю:

$$\pi^* = \arg\max_{\pi} V\left[\sum_{i \geq 0} \gamma^i (R(s) - C(s_i, \pi(s_i))) \mid \pi \right]. \quad (1.6)$$

1.1.4 Алгоритми планування

1.1.4.1 Корисність станів

Виконання стратегії вибудовує певну історію перебування в середовищі, а та, відповідно, є послідовністю станів, які відвідує агент, отже, цінність стану

залежить не лише від власної корисності, а й від корисності станів, у які з них можна потрапити [17]. У такому випадку можна вважати, що цінність деякого стану *s* дорівнює сумі безпосередньої винагороди за перебування в цьому стані та очікуваної знеціненої корисності наступного стану s', за умови, що агент вибирає оптимальну дію *a*.

Отже, цінність $V(s)$ стану *s* набуде такого вигляду:

$$V(s) = R(s) + \gamma \max_a \sum_{s'} (P_s(s' \mid a)V(s') - C(s,a)). \quad (1.7)$$

Рівняння (1.7) називають рівняння Беллмана [16]. Отже, на основі рівнянь Беллмана корисність деякого стану визначають з послідовності наступних станів.

1.1.4.2 Алгоритми ітерації

В основі алгоритму ітерації за значеннями (див. Алгоритм 1) лежить розв'язок рівнянь Беллмана – по одному для кожного стану: якщо в середовищі є *n* станів, то кількість рівнянь Беллмана також *n*. На першому кроці алгоритму потрібно випадковим чином обрати цінності $R_0(s)$ для кожного стану $s \in S$ середовища. Після цього відбувається багаторазове уточнення значення корисності. Для усіх станів *s* на кожному кроці *k* розраховують значення очікуваної корисності $V(s)_k$, базуючись на значеннях $V(s)_{k-1}$, які були розраховані на попередньому кроці. Алгоритм знаходить такі дії *a*, при яких $V(s)$ набуває максимальних значень і записує їх у стратегію. Після проведення достатньої кількості ітерацій можна стверджувати, що похибка точності визначення корисності станів та відповідно побудованої стратегії набуває деякого значення *j*, яке відповідає поставленим вимогам [21]:

$$\max_{s \in S}(V(s)_k - V(s)_{k-1}) < j. \quad (1.8)$$

Алгоритм 1. Ітерація за значеннями

Value_Iteration (S,A,γ)

for each $s \in S$

$R_0(s)=random()$

$i=1$

for $\max(V(s)_i - V(s)_{i-1}) < j$

for each $s \in S$

for each $a \in A$

$$V(s,a)=R(s)+\gamma \max_a \sum_{s'} (P_s(s'|a)V(s')-C(s,a))$$

$R_i(s)=max_{a \in A}V(s,a)$

$\pi(s)=argmax_{a \in A}V(s,a)$

$i=i+1$

return(π)

end

Алгоритм ітерації за стратегією

Основна ідея алгоритму полягає в тому, щоб поступово ітеративно покращувати початково довільно обрану стратегію π (див. Алгоритм 2). Алгоритм можна поділити на 2 основні етапи: (1) етап визначення вартості стратегії, на цьому етапі визначають цінність поточної стратегії шляхом розв'язання рівняння Беллмана, (2) етап покращення стратегії, на якому стратегія покращується до нової стратегії з вищою корисністю шляхом порівняння корисності стану s при виконанні дій згідно із стратегією $V(s,\pi(s))$ з корисністю станів при виконанні альтернативних дій a у цьому стані $V(s,a)$. При вищій ефективності нової дії a вона записується в стратегію для цього стану. Алгоритм зупиняється, коли немає альтернативних дій, які можуть покращити стратегію.

Алгоритм 2. Ітерація за стратегією

Policy_Iteration (S,A,γ)

$\pi = 0$

setect any $\pi' \neq 0$

for $\pi \neq \pi'$

$\pi = \pi'$

for each $s \in S$

$$V(s,\pi(s))=R(s)+\gamma\sum_{s'}(P_s(s'|\pi(s))V(s')-C(s,\pi(s)))$$

for each $s \in S$

$$V(s,a)=R(s)+\gamma\sum_{s'}(P_s(s'|a)V(s')-C(s,a))$$

if $\exists a \in A$ *exist* $V(s,\pi(s))<V(s,a)$

then $\pi' \leftarrow a$

else $\pi'(s) \leftarrow \pi(s)$

return(π)

end

1.1.5 MDP для частково спостережуваного середовища (POMDP)

На відміну від MDP у POMDP агент не може точно визначити, в якому стані він знаходиться, тому вводиться поняття імовірний стан b для стану s $b(s)$. Імовірний стан b – це множина з n елементів, де n – кількість станів і кожен елемент відповідає своєму стану s, в якій записаний розподіл імовірностей перебування агента у всіх можливих станах s множини станів S [17]. Сума усіх імовірностей перебування дорівнює 1: $\sum_{s\in S} b(s) = 1$.

У POMDP середовище описується за допомогою чотирьох компонент [19]:

- S – множина станів,
- A – множина дій,
- $P_s(s' \mid a)$ – ймовірності P переходу з стану s в s' при виконанні дії a,
- O – множина спостережень з імовірністю $P_a(o \mid s)$. $P_a(o \mid s)$ – імовірність отримати спостереження o після виконання дії a та потрапляння в стан s. $\sum_{o\in O} P_a(o \mid s) = 1$.

Множина спостережень O вводиться для моделювання частково спостережуваного середовища, тому що немає інших способів для отримання інформації про стани окрім як через спостереження. Отже отримуючи спостереження агент уточнює своє місце перебування і так змінює свій імовірний стан $b(s)$.

У задачах POMDP оптимальна дія залежить тільки від поточного імовірного стану агента. Це означає, що оптимальна стратегія π^* визначається, як функція відображення множини імовірних станів на множину дій [22]:

$$\pi : B \to A . \tag{1.9}$$

За таких умов агенту немає потреби знати свій фактичний стан, щоб приймати оптимальні рішення. Тому розв'язок задач POMDP полягає в застосуванні алгоритмів MDP для множини імовірних станів.

Але після виконання дій, імовірний стан $b(s)$ повністю оновлюється, тому часто алгоритми, які повертають наближено оптимальні стратегії виявляються ефективнішими ніж алгоритми точного планування.

Найчастіше для пошуку оптимальної стратегії в задачах POMDP, використовується так званий «пошук вперед» [23]. Суть алгоритму полягає в тому, щоб на кожному кроці вибирати дію $a = \pi^*(b)$ з урахуванням поточного імовірного стану b, а далі вираховувати новий імовірний стан виходячи з попереднього та отриманих результатів спостережень o.

Таким чином імовірність $b_a(s')$ потрапляння в стан s' з стану s після виконання дії a у імовірному стані b:

$$b_a(s') = \sum_{s \in S} P_s(s' \mid a) b(s) . \tag{1.10}$$

У свою чергу імовірність спостереження o після виконання дії a в стані s $b_a(o)$:

$$b_a(o) = \sum_{s \in S} P_s(o \mid a) b(s) . \tag{1.11}$$

Імовірність $b_a^o(s')$ перебування в стані s', після виконання дії a в довіреному стані b і отриманому спостереженні o:

$$b_a^o(s') = \frac{P_s(o \mid a) b_a(s')}{b_a(o)}. \quad (1.12)$$

У POMDP корисність станів визначається як сума корисностей усіх можливих станів перебування агента помножена на імовірність перебування в них агента:

$$R(b) = \sum_{s \in S} R(s) b(s). \quad (1.13))$$

За таким же принципом визначаються витрати C на виконання дії a у імовірному стані b:

$$C(b,a) = \sum_{s \in S} C(s,a) b(s). \quad (1.14)$$

Враховуючи вищенаведені формули, рівняння Беллмана подають як:

$$V(b) = R(b) + \gamma \max_a \sum_{o \in O} (b_a(o) V(b_a^o) + C(b,a)). \quad (1.15)$$

Проте на відміну від MDP у POMDP функція корисності – визначається множиною векторів дій або як їх ще називають α-вектори [19], α-вектор – це вектор розміру $|S|$ елементами якого є корисності стратегії за умови виконання дії a в усіх станах s.

Максимальне значення функції корисності можуть забезпечувати різні дії в залежності від імовірного стану агента (див. рис. 1.3). Тому в стратегії для одного стану вноситься весь набір найефективніших дій разом з діапазонами значень імовірного стану, перебуваючи у яких потрібно виконувати ту чи іншу дію.

Головною проблемою POMDP є експонентне зростання числа лінійних компонентів у значенні функції корисності, оскільки кожне оновлення вводить

додаткові лінійні компоненти у функції корисності *V*, а кожне вимірювання підносить до квадрату число лінійних компонентів. Головним чином саме тому у класичному вигляді алгоритми POMDP виявляються непрактичними для більшості реальних задач [24].

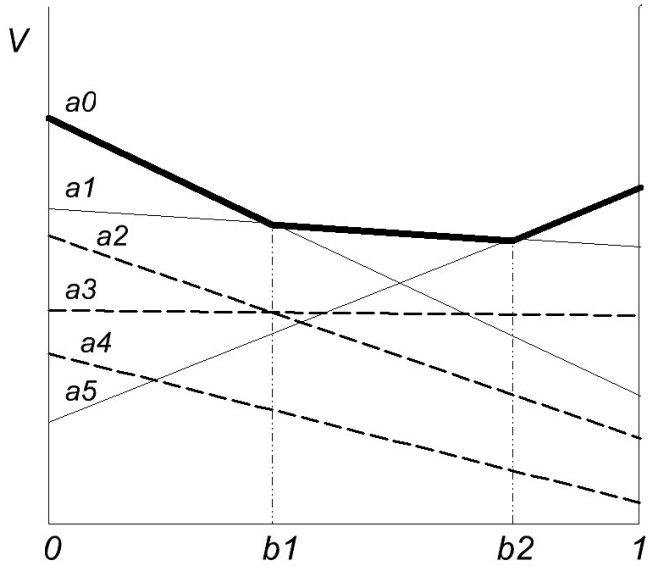

Рис. 1.3. Функція корисності

1.1.6 Усунення векторів

Для покращення роботи алгоритмів POMDP, а також для розширення сфери їх застосування використовують метод усунення векторів.

Метод базується на тому, що деякі α-вектори домінують над іншими, а це означає, що вони забезпечують вищі значення функції корисності. Важливим фактом є те, що кожен домінуючий вектор забезпечує максимальне значення функції корисності лише в одному унікальному діапазоні. Отже для отримання оптимального значення функції корисності можна розглядати лише множину домінуючих векторів, усуваючи усі інші вектори [22].

1.1.7 Алгоритм ітерації за значеннями POMDP

POMDP алгоритм ітерації за значеннями – це MDP алгоритм ітерації за значеннями застосований для імовірного стану [17].

На першому кроці алгоритму випадковим чином обираються цінності $R_0(b)$ для кожного імовірного стану $b \in B$ середовища. Далі уточнюються значення корисності імовірних станів поки похибка точності визначення не буде менша деякого значення *j*, а саме:

$$\max_{b \in B}(V(b)_k - V(b)_{k-1}) < j. \qquad (1.16)$$

Отримані в результаті роботи алгоритму оптимальні значення функції корисності, представляються у вигляді множини α-векторів. Отже фактично завданням алгоритму ітерації за значеннями POMDP є пошук множини α-векторів, що представляє значення функції корисності побудованої на базі отриманої на попередньому кроці множини α-векторів. Тому для покращення роботи алгоритму після кожної ітерації рекомендується проводити усунення α-векторів.

1.2 Види агентів. Інтелектуальні агенти

Детальні відомості щодо видів агентів наведено у роботі [16]. Подамо основні означення та введемо поняття, які використовуватимуться у цій роботі надалі.

Аґентом є все, що сприймає своє середовище за допомогою давачів (сенсорів) і впливає на це середовище за допомогою виконавчих механізмів (рис. 1.4). Людина, що розглядається в ролі аґента, наділенама органами чуття (очі, вуха і т.ін.), а виконавчими механізмами при цьому слугують руки, ноги, рот та інші частини тіла. Для робота, що виконує функції аґента, давачами є відеокамери, інфрачервоні далекоміри і т.ін., а виконавчими механізмами – двигуни. Програмне забезпечення, що виступає аґентом, вхідними сенсорними даними отримує коди натиснення клавіш, вміст файлів і мережеві пакети, а його дія на середовище виражається в тому, що програмне забезпечення виводить дані на екран, записує файли і передає мережеві пакети.

Використовуватимемо термін «сприйняття» для позначення отриманих аґентом сенсорних даних у будь-який конкретний момент часу. Послідовністю актів сприйняття аґента називається повна історія всього, що було коли-небудь ним сприйняте.

Загалом, вибір аґентом дії в будь-який конкретний момент часу може залежати від всієї послідовності актів сприйняття, що спостерігалися до цього моменту часу. Якщо існує можливість визначити, яка дія буде вибрана аґентом у відповідь на будь-яку можливу послідовність актів сприйняття, то можливим стане подання більш-менш точного визначення аґента. Це рівносильне твердженню, що поведінка деякого аґента може бути описана за допомогою функції аґента, яка відображає будь-яку конкретну послідовність актів сприйняття на деяку дію.

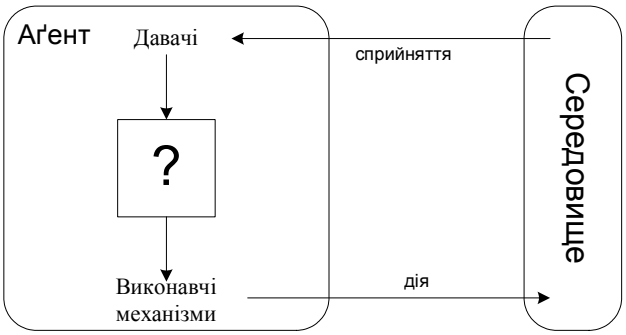

Рис. 1.4. Аґент взаємодіє із середовищем за допомогою давачів і виконавчих механізмів

1.2.1 Інтелектуальні агенти

Інтелектуальний агент (ІА) (англ. Intelligent agent) – система, котра спостерігає за навколишнім середовищем, взаємодіє з ним, і до того ж її поведінка раціональна в тому сенсі, що аґент розуміє суть власних дій і ці дії скеровані на досягнення певної мети [21]. Такий агент може бути як фізичним роботом, так і програмною системою. Про інтелектуальність агента можна говорити у випадку, якщо він взаємодіє з навколишнім середовищем приблизно так, як би діяла на його місці людина. Останнє твердження викликає

у фахових колах суперечки, оскільки не має вичерпних підстав стверджувати, що людський спосіб мислення є єдино правильним і допустимим. Багато з систем, які називають “інтелектуальними аґентами”, є доволі простими програмними продуктами, інтелектуальність яких зосереджується у декількох операторах IF-THEN, які присутні у програмному коді. Ця, дещо дивна ситуація, пояснюється тим, що відомо два визначення інтелектуального аґента: наведене вище, що є ближчим до сфери штучного інтелекту, та інше дещо загальніше, яке відносять до сфери комп’ютерних наук, загалом. Згідно з останнім визначенням, інтелектуальний агент (ІА) – це програма, яка самостійно виконує завдання продовж тривалих проміжків часу. Прикладом роботи такого аґента може бути постійний пошук і збір інформації в Інтернеті, який виконують браузери пошукових систем. До цих систем також зараховують різноманітні боти, сервіси, демони і навіть комп’ютерні віруси. Під “інтелектуальністю” таких систем розуміють їхню можливість пристосовуватися та навчатися.

Фахівці виділяють п’ять основних видів ІА [21]:

- агенти з простою поведінкою: діють лише на основі поточних знань, агентську функцію можна описати продукційним правилом “ЯКЩО умова ТО дія”; такі аґенти можуть бути успішними лише тоді, коли вони мають повну інформацію про навколишнє середовище, а така ситуація спостерігається доволі рідко;
- агенти з поведінкою, заснованою на моделі: такі агенти містять відображення тої частини світу, яка не піддається спостереженню, а, отже, гнучкіші, ніж попередні; та, на жаль, таке подання світу займає великий об’єм, а тому за незначне збільшення ефективності доводиться платити пам’яттю та часом, який витрачається на опрацювання цих даних;
- цілеспрямовані агенти доповнюють попередній тип множиною ситуацій, які вважаються несприятливими і які не можна допускати, та цільовою

ситуацією, до якої треба прямувати; розрізняють лише два стани – мету досягнуто та мету не досягнуто;

- практичні агенти: агенти, що мають “функцію корисності”, яка проектує множину станів на множину мір корисності станів; метою такого аґента є максимізація функції корисності;
- аґенти, що навчаються: власне, “справжні” інтелектуальні системи, які вміють змінювати поведінку в реальному часі на основі інформації, отриманої з навколишнього світу; вміють аналізувати свої дії, оцінювати їхню ефективність, пристосовуватися до нових умов, пропонувати нові способи вирішення невідомих раніше проблем.

Інтелектуальним аґентом є такий аґент, який виконує правильні дії. Очевидно, що виконання правильних дій є кращим,а ніж здійснення неправильних. Водночас залишається актуальним одержання відповіді на запитання «що розуміється під висловлюванням виконання правильних дій»? У першому наближенні можна стверджувати, що правильною дією є така дія, яка забезпечує найбільш успішне функціонування аґента. А тому слід володіти певним способом вимірювання успіху. Критерії успіху, разом з описом середовища, а також давачів і виконавчих механізмів аґента, подають вичерпну специфікацію завдання, з яким стикається аґент. Маючи ці компоненти, можна точніше визначити, що саме розуміється під терміном «інтелектуальний».

Показники продуктивності втілюють у собі критерії оцінки успішної поведінки аґента. Після занурення в середовище аґент виробляє послідовність дій, відповідних отриманим ним сприйняттям. Ця послідовність дій змушує середовище пройти через послідовність станів. Якщо така послідовність відповідає очікуваному результату, то аґент функціонує добре. Безумовно, що не може бути одного постійного показника, відповідного для всіх аґентів. Можна було б дізнатися у аґента його суб’єктивну думку про те, наскільки він задоволений своєю власною продуктивністю, при цьому слід зважати на те,що деякі аґенти не здатні відповісти на це запитання, а інші можуть бути

схильними до самообману. Тому необхідно добиватися застосування об'єктивних показників продуктивності, і, як правило, проектувальник, конструюючи аґента, передбачає використання саме таких показників.

У довільний момент часу оцінка інтелектуальності дій аґента залежить від чотирьох перерахованих нижче чинників:

- показників продуктивності, які визначають критерії успіху;
- знань аґента про середовище, набутих раніше;
- дій, які можуть бути виконані аґентом;
- послідовності актів сприйняття аґента, які відбулися раніше.

З урахуванням цих чинників сформулюємо наступне означення інтелектуального агента: для кожної можливої послідовності актів сприйняття інтелектуальний аґент повинен вибрати дію, яка, як очікується, максимізувала б його показники продуктивності, з урахуванням фактів, наданих певною послідовністю актів сприйняття і всіх знань, якими володіє аґент.

В поданому нами означенні вимагається, щоб інтелектуальний аґент не тільки збирав інформацію, але також максимально можливо навчався на тих даних, які він сприймає. Початкова конфіґурація аґента може відображати деякі попередні знання про середовище, але, у міру придбання аґентом досвіду, ці знання можуть модифікуватися і поповнюватися. Існують крайні випадки, в яких середовище попередньо повністю відоме. У цих випадках аґентові не потрібно сприймати інформацію або навчатися; він зразу ж діє правильно.

В аґентах, що успішно діють, завдання обчислення функції аґента розбивається на три окремі етапи: при проектуванні аґента деякі обчислення здійснюються його проектувальниками; додаткові обчислення аґент здійснює, вибираючи одну зі своїх чергових дій; а у міру того, як аґент вчиться на підставі досвіду, він здійснює інші допоміжні обчислення для ухвалення рішення про те, як модифікувати свою поведінку.

Якщо ступінь, в якому аґент покладається на апріорні знання свого проектувальника, а не на свої сприйняття, дуже високий, то такий аґент

трактуєється як такий, що має недостатню автономність. Інтелектуальний аґент повинен бути автономним – він зобов'язаний навчатися всьому, що може засвоїти, для компенсації неповних або неправильних апріорних знань. На практиці аґентові рідко висувається вимога, щодо його повної автономії із самого початку: якщо аґент є малодосвідченим, то це вимушує його діяти випадковим чином, якщо проектувальник не надав йому певної допомоги. Після набуття достатнього досвіду перебування у своєму середовищі поведінка інтелектуального аґента може за суттю стати незалежною від його апріорних знань. Тому включення у проект здібності до навчання дозволяє проектувати інтелектуальних аґентів, які можуть успішно діяти у різноманітних варіантах середовища.

1.2.2 Аґенти, засновані на корисності

Цілі дозволяють здійснити лише жорстку бінарну відмінність між станами «досягнення мети» і «недосягнення мети», тоді як загальніші показники продуктивності повинні забезпечувати порівняння різних станів світу в точній відповідності з тим, наскільки задоволеним стане аґент, якщо їх вдасться досягти. Оскільки поняття «задоволеності» є не зовсім науковим, частіше застосовується термінологія, за якою стан світу, переважніший в порівнянні з іншим, розглядається, якщо має вищу корисність для аґента.

Функція корисності відображає стан (або послідовність станів) як дійсне число, яке позначає відповідний ступінь задоволеності аґента. Повна специфікація функції корисності забезпечує можливість ухвалювати інтелектуальні рішення в описаних нижче двох випадках, коли цього не дозволяють зробити цілі. По-перше, якщо є конфліктні цілі, тобто можуть бути досягнуті тільки деякі з них (наприклад, або швидкість, або безпека), то функція корисності дозволяє знайти прийнятний компроміс. По-друге, якщо є декілька цілей, до яких може прагнути аґент, але жодна з них не може бути досягнута з усією певністю, то функція корисності надає зручний спосіб зваженої оцінки вірогідності успіху з урахуванням важливості цілей.

Будь-який інтелектуальний аґент повинен поводитися так, ніби він володів функцією корисності, очікуване значення якої він намагається максимізувати. Тому аґент, що володіє явно заданою функцією корисності, має можливість ухвалювати інтелектуальні рішення і здатний робити це за допомогою алгоритму загального призначення, не залежного від конкретної максимізованої функції корисності. Завдяки цьому «ґлобальне» визначення інтелектуальності (згідно з яким інтелектуальними вважаються функції аґента, що мають найвищу продуктивність) перетвориться в «локальне» обмеження на проекти інтелектуальних аґентів, яке може бути виражене у вигляді простої програми.

Структура аґента, що діє з урахуванням корисності, показана на рис. 1.5.

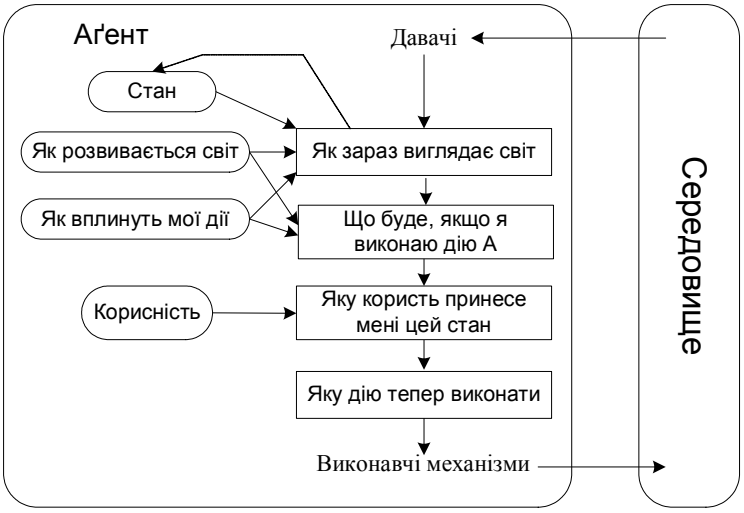

Рис. 1.5. Аґент, заснований на моделі та корисності

У ньому модель світу використовується разом із функцією корисності, яка вимірює переваги аґента стосовно станів світу. Потім аґент обирає дію, яка веде до якнайкращої очікуваної корисності. Для обчислення очікуваної корисності виконується усереднення за всіма можливими результатними станами з урахуванням коефіцієнта, що визначає вірогідність кожного результату.

1.2.3 Аґенти, що навчаються

Вище були описані програми аґентів, в яких застосовуються різні методи вибору дій. Але досі ще не були подані відомості про те, як створюються програми аґентів. Тюринґ запропонував ідею про те, як фактично має здійснюватися програмування запропонованих ним інтелектуальних машин вручну. Він оцінив об'єм роботи, який для цього буде потрібний, і прийшов до такого висновку: «Бажано було б мати якийсь продуктивніший метод». Запропонований ним метод полягав у тому, що необхідно створювати машини, що навчаються, а потім здійснювати їх навчання. Тепер цей метод став домінантним методом створення найсучасніших систем у багатьох областях штучного інтелекту. Як наголошувалося вище, навчання має ще одну перевагу: воно дозволяє аґентові функціонувати у заздалегідь невідомих йому варіантах середовища і ставати компетентнішим у порівнянні з тим, що могли б дозволити тільки його початкові знання.

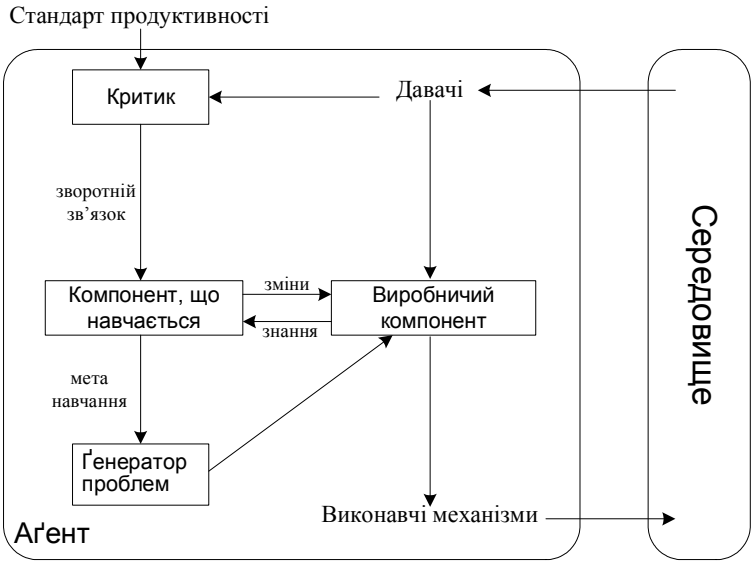

Рис. 1.6. Загальна модель аґентів, що навчаються

Структура аґента, що навчається (див. рис. 1.6), поділяється на чотири концептуальні компоненти. Найважливіша відмінність спостерігається між повчальним компонентом, який відповідає за внесення вдосконалень, і продуктивним компонентом, який забезпечує вибір зовнішніх дій. Продуктивним

компонентом є те, що досі розглядалося, як аґент отримує інформацію й ухвалює рішення про виконання дій. Компонент, що навчається використовує інформацію зворотного зв'язку від критика з оцінкою того, як діє аґент, і визначає, яким чином має бути модифікований продуктивний компонент, щоб він успішніше діяв у майбутньому.

Проект повчального компоненту багато в чому залежить від проекту продуктивного компоненту. Здійснюючи спробу спроектувати аґента, який навчається розвивати певні здібності, треба перш за все прагнути знайти відповідь на запитання: «Якого роду продуктивний компонент буде потрібний моєму аґентові після того, як він буде навчений тому, як виконувати свої функції?», а не на запитання: «Як приступити до розв'язування задачі навчання його для виконання цих функцій?». Після того, як спроектований сам аґент, можна приступати до конструювання повчальних механізмів, що дозволяють удосконалити будь-яку частину цього аґента.

Критик повідомляє повчальний компонент, наскільки добре діє аґент з урахуванням постійного стандарту продуктивності. Критик необхідний, оскільки самі результати сприйняття не дають ніяких вказівок на те, чи успішно діє аґент. Наприклад, шахова програма може отримати результати сприйняття, які вказують на те, що вона поставила мат своєму супротивникові, але їй потрібний стандарт продуктивності, який дозволив би визначити, що це – добрий результат; самі дані сприйняття нічого про це не говорять. Важливо, щоб стандарт продуктивності був постійним. У принципі, цей стандарт треба розглядати як повністю зовнішній по відношенню до аґента, оскільки аґент не повинен мати можливості його модифікувати так, щоб він більшою мірою відповідав його власній поведінці.

Останнім компонентом аґента, що навчається, є ґенератор проблем. Його завдання полягає в тому, щоб пропонувати дії, які повинні привести до отримання нового й інформативного досвіду. Річ у тому, що якщо продуктивний компонент наданий самому собі, то продовжує виконувати дії,

які є якнайкращими з погляду того, що він знає. Але якщо аґент готовий до того, щоб дещо проекспериментувати і в короткочасній перспективі виконувати дії, які, можливо, виявляться не зовсім оптимальними, то він може виявити набагато кращі дії з погляду довготривалої перспективи. Ґенератор проблем призначений саме для того, щоб пропонувати такі дослідницькі дії. Саме цим займаються вчені, здійснюючі експерименти.

Щоб перевести весь цей проект на конкретний ґрунт, повернемося до прикладу автоматизованого таксі. Продуктивний компонент складається з тієї колекції знань і процедур, яка застосовується водієм таксі при виборі ним дій з водіння. Водій таксі за допомогою цього продуктивного компоненту виїжджає на дорогу і керує своєю машиною. Критик спостерігає за світом і в ході цього передає відповідну інформацію повчальному компоненту. Наприклад, після того, як таксі швидко виконує поворот наліво, перетинаючи три смуги руху, критик помічає, які висловлювання використовують інші водії. На підставі цього досвіду повчальний компонент здатний сформулювати правило, яке свідчить, що це – неприпустима дія, а продуктивний компонент модифікується шляхом встановлення нового правила. Ґенератор проблем може визначити деякі області поведінки, що вимагають удосконалення, і запропонувати експерименти, такі як перевірка гальм на різних дорожніх покриттях і за різних умов.

Навчальний компонент може вносити зміни до будь-якого з компонентів «знань». У простих випадках навчання здійснюватиметься безпосередньо на підставі послідовності актів сприйняття. Спостереження за парами послідовних станів середовища дозволяє аґентові освоїти інформацію про те, «як змінюється світ», а спостереження за результатами своїх дій може дати аґентові можливість дізнатися, «який вплив мають мої дії». Наприклад, після того, як водій таксі прикладе певний гальмівний тиск під час їзди мокрою дорогою, він незабаром дізнається, яке зниження швидкості фактично було досягнуте. Очевидно, що ці

два завдання навчання стають складнішими, якщо середовище спостережуване лише частково.

Ті форми навчання, які були описані в попередньому абзаці, не вимагають доступу до зовнішнього стандарту продуктивності, вірніше, в них застосовується універсальний стандарт, згідно з яким зроблені прогнози мають бути узгоджені з експериментом. Ситуація стає дещо складнішою, коли йдеться про аґента, заснованого на корисності, який прагне освоїти у процесі навчання інформацію про корисність. Наприклад, припустимо, що аґент, який займається водінням таксі, перестає отримувати чайові від пасажирів, які під час втомливої поїздки відчули себе повністю розбитими. Зовнішній стандарт продуктивності повинен інформувати аґента, що відсутність чайових – це негативний внесок в його загальну продуктивність; у такому разі аґент дістає можливість засвоїти в результаті навчання, що грубі маневри, втома пасажирів не дозволяють підвищити оцінку його власної функції корисності. У цьому сенсі стандарт продуктивності дозволяє виділити визначену частину вхідних результатів сприйняття як винагороду (або штраф), які безпосередньо надходять від даних зворотного зв’язку і впливають на якість поведінки аґента. Саме з цієї точки зору можуть розглядатися жорстко закріплені стандарти продуктивності, такі як біль або голод, якими характеризується життя тварин.

Підводячи підсумок, відзначимо, що аґенти мають різноманітні компоненти, а самі ці компоненти можуть бути подані в програмі аґента багатьма способами, тому створюється враження, що різноманітність методів навчання надзвичайно велика. Проте всі ці методи мають єдиний аспект, що їх об’єднує. Процес навчання, здійснюваний в інтелектуальних аґентах, можна загалом охарактеризувати як процес модифікації кожного компоненту аґента для забезпечення точнішої відповідності цих компонентів доступній інформації зворотного зв’язку і тим самим поліпшення загальної продуктивності аґента.

ІА використовує під час свого функціонування інформацію, отриману з навколишнього середовища, аналізує її, зіставляючи з уже відомими йому

фактами і, на основі результатів аналізу, приймає рішення про подальші дії. Підсистему ІА, яка займається зберіганням, впорядкуванням та керуванням інформацією про навколишній світ, називають БЗ (англ. Knowledge base) [40].

База знань – це система, яка розроблена для керування знаннями, тобто їхнім збиранням, збереженням, пошуком і видаванням. Найважливіший параметр БЗ – якість та повнота знань про предметну область (ПО), яку вона задає. Тут треба чітко розмежувати поняття середовище та ПО, які в деякій мірі є синонімами. ІА діє у середовищі, а ПО відображає це середовище за допомогою моделі знань.

Якість БЗ залежить від структури та формату знань, способу їх подання. Залежно від рівня складності та спеціалізованості систем, де використовуються БЗ, виділяють декілька основних їх типів: БЗ всесвітнього масштабу; національні БЗ; галузеві БЗ; БЗ експертних систем; БЗ організацій; БЗ спеціалістів.

Розрізняють два основних види БЗ – придатні для читання людиною (англ. Human-readable knowledge bases) та придатні для машинного читання (англ. Machine-readable knowledge bases) [25]. Придатні для читання людиною БЗ – це зазвичай енциклопедії, довідники, статті, графічні схеми та інші інформаційні документи, за допомогою яких людина може отримати знання про ПО, яка її цікавить. Придатні для машинного читання БЗ – це зазвичай файли в комп'ютерній пам'яті, записані у форматі, передбаченому системою-користувачем. Такі БЗ часто містять логічні вирази, які використовуються під час логічного виведення, та інші дані, що важко сприймаються людиною [26].

Для широкого впровадження будь-якої технології чи методики необхідний чіткий і аргументований стандарт. Як вже зазначалось вище, у галузі розроблення БЗ таким стандартом стали онтології [27]. Онтологія – це спроба всеохопної і детальної формалізації деякої галузі знань за допомогою концептуальної схеми [28]. Така схема, зазвичай, складається з ієрархічної структури даних, що містить всі релевантні класи об'єктів, їхні зв'язки, теореми

й обмеження, прийняті у певній ПО. Важливою перевагою онтології, як способу подання знань, є те, що її однаково легко сприймає як людина, у вигляді, наприклад, графу, так і машина [29].

Вважаємо, що БЗ є ширшим поняттям, ніж онтологія. А саме –онтологія є ядром БЗ. Тобто ядро БЗ містить знання, які вписуються у концептуальну схему. Однак, окрім онтології, БЗ містить й інші знання, які не вписуються у концептуальну схему. Такими знаннями є специфічні знання експерта з певної ПО або знання, отримані на основі інтелектуального аналізу даних, наприклад, на основі дерева рішень чи асоціативних правил тощо.

1.3 Основні результати та висновки до розділу

У цьому розділі відображено такі результати:

1. Проаналізовано сучасні підходи до розроблення інтелектуальних систем підтримки прийняття рішень та напрями їх розвитку. Визначено недоліки сучасних підходів побудови інтелектуальних систем підтримки прийняття рішень. Обґрунтовано важливість проектування та наповнення баз знань таких систем для ефективного їх функціонування. Одним з підходів до побудов баз знань є використання онтологій.
2. Розглянуто поняття онтології з різних ракурсів. З погляду проектування інтелектуальних систем підтримки прийняття рішень онтології розуміють як бази знань спеціального типу, які зберігають ієрархію понять предметної області, відношення між ними та їхню інтерпретацію. Обґрунтовано вибір моделей подання знань для моделювання структури та функцій онтології.

.

РОЗДІЛ 2 . МОДЕЛЮВАННЯ ДІЙ СПЕЦІАЛІЗОВАНИХ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ НА ОСНОВІ ОНТОЛОГІЙ

2.1 Моделювання дій інтелектуального агента

Задача моделювання плану поведінки ІА, ціль якого полягає у досягненні кінцевого стану, є актуальною задачею сьогодення, оскільки саме від правильно побудованих моделей залежать кількісні показники, отримані під час досягнення цілі (затрачені ресурси, витрачений час для досягнення цілі, оцінка досягнутого стану тощо). Відомо, що для моделювання такого плану знаходження оптимального рішення використовуються стохастичні або детерміновані мережі з вершинами типу І/АБО і методи пошуку вглиб, вшир та різноманітними евристичними функціями [42]. Однак ці моделі можна використовувати для задач, в яких стани чітко задаються множиною фактів, які реалізуються однією з формальних логік або набором продукційних правил та не враховують витрату ресурсів. Очевидно, що у складних прикладних галузях описування станів та вибір альтернатив для відповідних переходів між станами вимагає зовсім інших підходів, а отже, і моделей, які ґрунтуються на онтологіях задач та онтології ПО.

З огляду на постановку задачі – досягнення цільового стану ІА – для розв’язування цієї задачі запропоновано використовувати орієнтовані графи (зокрема мережі Петрі [41]) для моделювання шляхів (процесів) досягнення цільового стану, байєсівські мережі для моделювання імовірнісних оцінок переходів між станами, онтологію ПО для описування та оцінювання станів, а також для оцінювання витрат ресурсів для переходів між станами.

Отже, необхідно розробити модель функціонування ІА, поведінка якого полягає у досягненні деякого цільового стану як правило з додатковою вимогою, яка полягає в мінімізації витрат ресурсів ІА під час такого переходу.

2.1.1 Основні припущення та поняття

Для досягнення цільового стану ІА насамперед повинен побудувати план досягнення цього стану із всіма можливими альтернативами. Планування ґрунтується на декомпозиції. Задача планування ZP містить три складові: множину станів St, множину дій A, множину цільових станів *Goal* (станів мети); тобто

$$ZP = \langle St, A, Goal \rangle .$$

Надалі вважатимемо, що стан мети єдиний. Якщо станів мети декілька, то мету можна записати як диз'юнкцію цих станів. Тоді досягнення такого стану є розв'язок деякої підзадачі, тому припущення про єдиність стану мети є нормальним.

Своєю чергою дія A складається із чотирьох частин: ім'я дії, список параметрів, передумова та результат. А сам план визначається як кортеж із чотирьох елементів – <Множина дій, Множина обмежень впорядкування, Множина причинних зв'язків, Множина відкритих передумов> [16]. Для врахування декомпозиції, і/або залежностей між станами та переходами, відображення альтернатив досягнення цільових станів ***пропонуємо*** використовувати графову модель діаграми станів UML (Unified Modeling Language) [43]. Приклад такого орієнтованого графа з цільовим станом *Goal* наведено на рис. 2.1.

На початку фішка мережі перебуває у стані $St(0)$. Тоді очевидно, що суть функціонування ІА полягає у переміщенні фішки в кінцевий стан *Goal* з мінімальною витратою ресурсів (ресурси можна задавати у формі грошового еквівалента, людино-годин, часу тощо). Щоб почати розв'язувати цю задачу, спочатку треба довести, що орієнтований граф побудований таким чином, що кінцевий стан *Goal* досягнути насправді можна. Відомо, що для такого доведення використовуються планувальники – програми, які шукають розв'язок або доводять неіснування розв'язку. Ця робота оминає розгляд такого

доведення. Апостеорно вважаємо, що такий перехід існує і не один, бо інакше задача не має змісту.

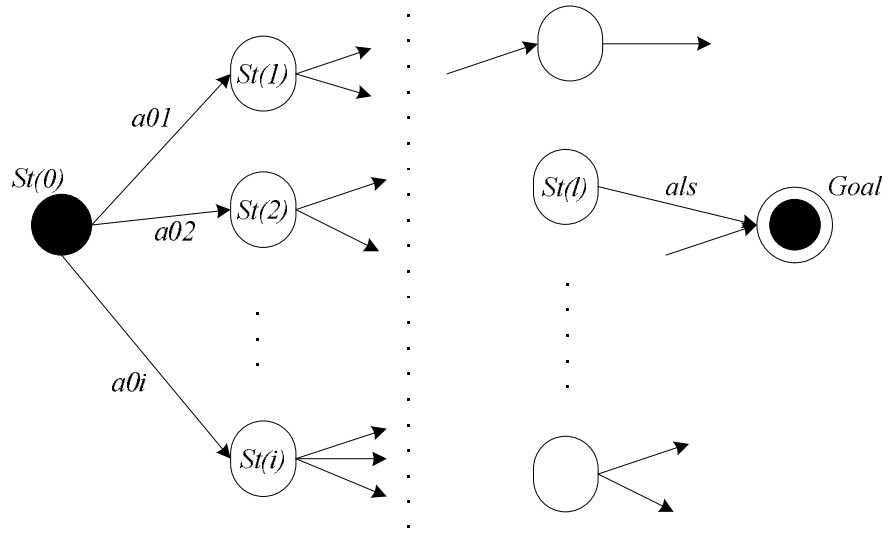

Рис. 2.1. Приклад діаграми станів UML моделювання поведінки інтелектуального агента

Стан $St(i)$ характеризується базою даних (БД) та БЗ, поданих у вигляді множини фактів з відповідними ймовірнісними оцінками. Дія a_{ij} подається у вигляді відображення зі стану $St(i)$ в стан $St(j)$ з відповідною ймовірністю p_{ij}, тобто: $St(i) \to St(j)$ з ймовірністю p_{ij}. Стохастичні процеси у цій роботі не розглядаються, а лише керовані процеси. Рекомендуємо в якості моделювання стохастичних процесів використати теорію MDP (марківські процеси прийняття рішень), які розглядались вище (у першому розділі).

2.1.2 Онтологічний підхід до подання знань

Основною компонентою інтелектуальних агентів є база знань, що формується відповідно до предметної області на яку зорієнтоване функціонування цього агента. Традиційні методи інженерії знань (отримання знань від експерта, інтелектуальний аналіз даних, машинне навчання тощо) не ґрунтуються на системі вивірених та загальноприйнятих стандартів, тому

побудовані на їхній основі бази знань з часом втрачають свою функціональність через низьку ефективність їх функціонування. Як стандарт інженерії знань використовують онтологічний інжиніринг, у результаті застосування якого отримують онтологію бази знань. Онтологія – це детальна формалізація деякої області знань подана за допомогою концептуальної схеми. Така схема складається з ієрархічної структури понять, зв'язків між ними, теорем та обмежень, які є прийняті у певній предметній області.

Використання онтологій у складі баз знань інтелектуальних агентів допомагає вирішити низку проблем методологічного та технологічного характеру, які виникають під час розроблення таких систем. Зокрема для України характерні проблеми полягають у відсутності концептуальної цілісності й узгодженості окремих прийомів та методів інженерії знань; нестачі кваліфікованих фахівців у цій галузі; жорсткості розроблених програмних засобів та їх низькій адаптивній здатності; складності впровадження інтелектуальних систем підтримки прийняття рішень, що зумовлено психологічними аспектами. Все це свідчить та підтверджує актуальність проблематики досліджень використання онтологій у процесі побудови інтелектуальних агентів.

Онтологія (від грец. онтос – суще, логос – навчання, поняття) – термін, що визначає вчення про буття, про сутність, на відміну від гносеології – вчення про пізнання. Вже у Х. Вольфа (1679–1754), автора терміну "онтологія", вчення про буття було відокремлене від вчення про пізнання. До цього онтологія була частиною метафізики, наукою самостійною, незалежною і не пов'язаною з логікою, з "практичною філософією", з науками про природу. Її предмет становить вивчення абстрактних і загальних філософських категорій, таких, як буття, субстанція, причина, дія, явище тощо, а сама онтологія як наука домагалася повного пояснення причин усіх явищ [30].

З погляду, ближчого до понять, пов'язаних зі штучним інтелектом, онтологія – це знання, формально відображені на базі концептуалізації.

Концептуалізація – опис множини об’єктів і понять, знань про них і зв’язків між ними. Онтологією називається експліцитна специфікація концептуалізації [31]. Формально онтологія складається з термінів (понять, концептів), організованих в таксономію, їхніх визначень і атрибутів, а також пов’язаних з ними аксіом і правил виведення.

Ця система понять зв’язується як універсальними залежностями типу “загальне–частинне”, “частина–ціле”, “причина–наслідок” тощо, так і специфічними, залежно від моделі предметної області (ПО). Онтологія – це модель ПО, яка використовує всі доступні засоби подання знань, релевантних ПО [32].

Забезпечення можливості використання знань ПО стало однією з рушійних сил недавнього сплеску у вивченні онтологій. Наприклад, для моделей багатьох різних ПО необхідно сформулювати поняття часу. Цей термін містить поняття тимчасових інтервалів, моментів часу тощо. Якщо одна група вчених детально розробить таку онтологію, то інші можуть просто повторно використовувати її у своїх ПО. Крім того, якщо нам потрібно створити велику онтологію, ми можемо інтеґрувати дещо з існуючих онтологій, які описують частини великої ПО. Ми також можемо повторно використовувати основну онтологію і розширити її для описування ПО, яка нас цікавить.

Створення явних допущень у ПО, що лежать в основі реалізації, дає можливість легко змінити ці припущення у разі зміни наших знань про ПО. Жорстке кодування припущень про світ мовою програмування призводить до того, що ці припущення не тільки складно знайти і зрозуміти, але і також складно змінити, особливо непрограмісту. Крім того, явні специфікації знань ПО корисні для нових користувачів, які мають розуміти значення термінів ПО.

Відокремлення знань ПО від оперативних знань – це ще один варіант загального застосування онтологій. Ми можемо описати задачу конфігурації продукту з його компонентів відповідно до необхідної специфікації і впровадити програму, яка робить цю конфігурацію незалежною від продукту і самих

компонентів. Після цього ми можемо розробити онтологію компонентів і характеристик комп'ютерних комплексів і застосувати цей алгоритм для конфігурації нестандартних комп'ютерних комплексів. Ми також можемо використовувати такий самий алгоритм для конфігурації ліфтів, якщо надамо йому онтологію компонентів ліфта.

Аналіз знань у ПО можливий тоді, коли є декларативна специфікація термінів. Формальний аналіз термінів надзвичайно цінний як за спроби повторного використовування наявних онтологій, так і у разі їх розширення [33].

Під формальною моделлю онтології O розуміють трійку такого вигляду [34]:

$$O = \langle C, R, F \rangle, \tag{2.1}$$

де C – скінченна множина понять (концептів, термінів) ПО, яку задає онтологія O; $R : C \to C$ – скінченна множина відношень між концептами (поняттями, термінами) заданої ПО; F – скінченна множина функцій інтерпретації (аксіоматизація, обмеження), заданих на концептах чи відношеннях онтології O [35].

Онтологію можна подати у вигляді графу (такий граф називають концептуальним), де вершини графу – концепти ПО, дуги вказують напрями відношень між концептами. Приклад такого графу наведено на рис. 2.2. Вершини можуть бути як інтерпретовані (визначені аксіоми понять), так й не інтерпретованими. Інтерпретовані вершини позначені темним кольором. Своєю чергою, відношення діляться на вертикальні (суцільні лінії) та горизонтальні (штрих-пунктирні лінії).

Для того, щоб можна було на онтологіях будувати метрику, нами запропоновано розширити модель (2.1) за рахунок введення двох скалярних величин – ваги важливості понять БЗ та семантичних зв'язків між ними [36]. Ці ваги дозволяють адаптувати онтологію БЗ до особливостей ПО, визначають закладені в її структуру елементи та механізми її оптимізації (точніше,

адаптації) за допомогою самонавчання під час експлуатації. Ваги важливості понять та зв'язків мають відповідати таким основним вимогам [37]:

- відображати семантичну важливість понять ПО;
- формуватися під час наповнення БЗ та коректуватися відповідно до визначених правил;
- забезпечувати контроль цілісності БЗ;
- відповідати вимогам метрики під час їх використання для порівняння семантичної та ознакової близькості понять.

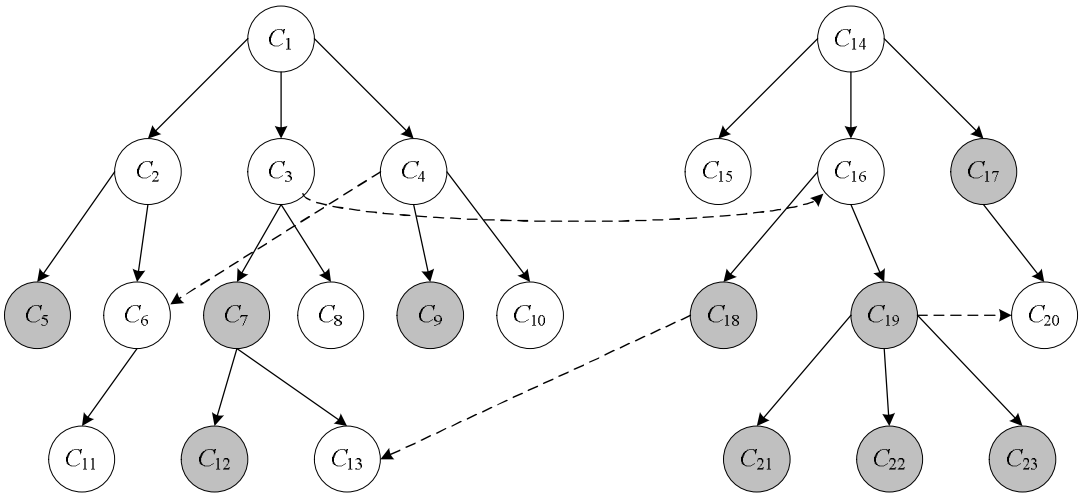

Рис. 2.2. Приклад графу онтології

Є завдання сформулювати відповідний набір правил присвоєння ваг важливості поняттям та твердженням у моделі БЗ, що забезпечить оцінку актуальної цінності її інформаційного наповнення та досліджуваних поточних ситуацій.

Покажемо можливість виконання сформульованого завдання, вводячи деякі спрощення і припущення. Подамо БЗ у вигляді зваженого концептуального графа, числові семантичні характеристики вершин і ребер якого визначаються за певними правилами. Він є орієнтованим зваженим мультиграфом з такими властивостями:

1) на кожний елемент (вершину) може бути довільна кількість посилань;

2) кожний елемент може мати зв’язок з будь-якою кількістю інших елементів;
3) кожному зв’язку (ребру) у моделі відповідає певний напрям і вага важливості зв’язку відповідного твердження, кожному поняттю (вершині) – вага важливості поняття.

Вага важливості поняття (зв’язку) – це чисельна міра, котра характеризує значущість певного поняття (зв’язку) у конкретній ПО і динамічно змінюється за певними правилами під час експлуатації системи [34].

Підхід до подання знань у формі зважених концептуальних графів полягає у тому, що будь-яке можливе узагальнення, тобто комплексне, складене поняття завжди явно артикульоване, назване і як окремий концепт фігурує в БЗ [38, 39]. Тому, якщо деяке узагальнення має спільні властивості чи способи функціонування, вони фізично можуть бути реалізовані через властивості та опрацювання подій відповідного узагальнювального концепту.

Отже, ми розширимо поняття онтології, ввівши в її формальний опис ваги важливості понять та відношень. Тоді така онтологія визначається як:

$$\hat{O} = \left\langle \hat{C}, \hat{R}, F \right\rangle,$$

де $\hat{C} = \langle C, W \rangle$, $\hat{R} = \langle R, L \rangle$, де у свою чергу W – важливість понять C, L – важливість відношень R. Взагалі кажучи, W – вектор розмірності кількості різних класів, якщо онтологія використовується для інтелектуальної системи класифікації або розмірності кількості задач, які розв’язуються ІА планування діяльності.

Визначену таким способом онтологію називатимемо адаптивною, тобто такою, що адаптується до ПО за рахунок модифікації понять та ваг важливості цих понять і зв’язків між ними [36].

2.1.3 Оцінювання станів задач планування дій

Для вибору необхідних дій ІА повинен вміти оцінювати стани. Легше це здійснити зі станами, в яких він вже перебував. Важче оцінити майбутні стани. Для оцінювання використовуються евристичні функції або метазнання. Тому спочатку розглянемо оцінку пройдених станів, потім дій, і насамкінець їхню комбінацію, що веде до нового (майбутнього) стану.

Нехай $v\left(St\left(i\right)\right)$ – оцінка стану $St\left(i\right)$. Для оцінювання станів, у яких вже перебував ІА, використовуватимемо адаптивну онтологію ПО $\hat{O}$.

Стан мети $Goal$ визначається необхідністю деякій множині ознак X досягнути певних значень $z\left(x,Goal\right)\ \forall x\in X$.

Будь-який стан $St\left(i\right)$ задається своєю множиною ознак Y_i, які набувають значень $z\left(y,St\left(i\right)\right)\ \forall y\in Y_i$.

Для оцінювання стану $St\left(i\right)$ необхідно здійснити відображення ψ множини ознак та їх значень стану $St\left(i\right)$ в множину ознак та значень стану $Goal$. Очевидно, що таке відображення повинне використати БЗ, а саме додатковий модуль онтологій Semantic Web Rule Language (SWRL).

$$\psi : Y_i \xrightarrow{\hat{O}} X\ . \tag{2.2}$$

Тоді оцінка стану $v\left(St\left(i\right)\right)$ обчислюється

$$v\left(St\left(i\right)\right)=d\left(St\left(i\right),Goal\right)=\sum_{x\in X_W}\varphi\left(z\left(\psi\left(y\right),St\left(i\right)\right),z\left(x,Goal\right)\right), \tag{2.3}$$

де X_W – множина ознак із найбільшими вагами W в адаптивній онтології.

Очевидно, що чим оцінка стану менша, тим стан кращий.

Розглянемо функцію $\varphi\left(x,y\right)$. Очевидно, що x може бути діапазоном, тобто нечіткою множиною, де D – універсальна множина $x\subseteq D$; числовим

значенням або нечисловим значенням. Залежно від цього $\varphi(x,y)$ визначається по-своєму:

$$\varphi(x,y)=\begin{cases} 1-\mu_x(y),\ x-\text{нечітка множина},\\ \lambda\cdot|x-y|,\ x,y-\text{числові значення},\\ 1-\mu(x,y),\ x,y-\text{нечислові значення}, \end{cases} \quad (2.4)$$

де $\mu_x(y)$ – коефіцієнт впевненості того, що y належить нечіткій множині x, λ – числова величина, яка залежить від ПО, щоб $\lambda\cdot|x-y|\in[0,1]$ (розмірність величини λ обернено-пропорційна до розмірності величин ξ та η), $\mu(x,y)\in[0,1]$ – нечітка величина подібності значень x та y. Наприклад, $\mu(x,y)=1$, якщо $x=y$, $\mu(x,y)=0,8$, якщо $x\approx y$, $\mu(x,y)=0$, якщо $x\neq y$.

В якості функції $\varphi(x,y)$ також можна вибрати одну із відомих метрик [**Ошибка! Источник ссылки не найден.**] (Евклідова, Манхеттенська метрика тощо), в залежності від типу ознак, які використовуються в задачі. Однак в межах окремої задачі можна використовувати лише одну з наведених метрик.

2.1.4 Оцінювання дій задач планування діяльності

У наших дослідженнях для вибору дій ІА ми ґрунтуватимемось на інтелектуальності агента як прагнення мінімізувати витрати ресурсів для досягнення кінцевого стану. Тому вважатимемо, що кожна дія a_{ij} однозначно визначається витратами ресурсів g_{ij}^k (ціна переходу зі стану в стан), де $k=1,2,...,n_i$. n_i – кількість альтернатив α_k для здійснення переходу a_{ij}. Тому надалі дію позначатимемо трьома індексами a_{ij}^k: перехід із стану $St(i)$ в стан $St(j)$, використовуючи альтернативу α_k. Наприклад, для зняття захисного покриття з поверхні трубопроводу можна використати три альтернативи: механічне, хімічне та термічне зняття.

Кожна з альтернатив характеризується витратами ресурсів та терміном експлуатування. Інформація про альтернативи та витрати ресурсів повинна зберігатися в онтології. Інформація про значення ознак та виграш від переходу в стан (терміни експлуатування тощо) зберігається в БД. Очевидно, що можуть появлятися нові альтернативи, тому ІА необхідно постійно відстежувати нові текстові документи з метою пошуку нових знань, та заносити їх в онтологію. Задачу автоматизованого наповнення онтології на основі аналізу текстових документів описано в наступному розділі.

Оскільки чим оцінка менша, тим краще, то оцінка дії прямо пропорційна витраті ресурсів, тобто:

$$v\left(a_{ij}^{k}\right) = E \cdot g_{ij}^{k}, \tag{2.5}$$

де E – скалярна величина, яка зводить вимірювання оцінки дії до одного вимірювання з оцінкою станів.

Загалом рішення стосовно вибору дії на основі альтернативи здійснюємо згідно деякого відношення між станом та дією

$$o_i\left(a_{ij}^{k}\right) = \delta\left(v\left(a_{ij}^{k}\right), v\left(St\left(j\right)\right)\right). \tag{2.6}$$

Зокрема таке відношення може бути лінійним:

$$o\left(a_{ij}^{k}\right) = \omega v\left(a_{ij}^{k}\right) + \left(1-\omega\right)v\left(St\left(j\right)\right), \tag{2.7}$$

де $\omega \in \left[0,1\right]$ – частка альтернативі дії, яку ІА віддає під час прийняття рішення, інша частка належить стану, в який він перейде.

Після оцінювання дій та станів, задача вибору шляху зводиться до задачі динамічного програмування [44]. Справді ми отримаємо таку модель переходів між станами:

$$St\left(j\right) = a\left(St\left(i\right), o_i\right) \tag{2.8}$$

із критерієм оптимізації:

$$\Theta\left(St(0),\vec{o}\right)\Rightarrow \min(\max). \qquad (2.9)$$

Задача (2.8)-(2.9) є задачею динамічного програмування [47]. Використовуючи методи, придатні для розв'язування таких задач, знаходимо розв'язок у вигляді шляху переходу з початкового у кінцевий стан.

2.1.5 Звуження простору станів

Суть запропонованого методу звуження простору пошуку *Path* полягає в наступному: окремим поняттям $\tilde{C}=\left\{\tilde{C}_1,\tilde{C}_2,...,\tilde{C}_m\right\}$ онтології, які задають альтернативи переходу між станами надаємо вагу W_i, $i=1,2,...,m$. З часом ця вага буде змінюватись, а саме зростатиме в залежності від міри довіри σ до джерела на основі якого ця альтернатива була додана в онтологію. Ті поняття із множини $\tilde{C}$ вилучатимемо, збільшення ваг яких за певний термін не перевищуватиме деякий поріг λ. Детальніше цей метод описано в 3-му розділі, де буде розглянуто питання розбудови онтології. Відзначимо, що існують альтернативи $C'=\left\{C'_1,C'_2,...,C'_m\right\}$ переходів між станами, які вилучати із онтології не можна, тобто $\tilde{C}\cap C'=\varnothing$. Такі елементи онтології визначаються експертами ПО. Крім того експерти відзначають, які концепти онтології переходять із множини $\tilde{C}$ в C' і навпаки.

Після того як простір пошуку *Path* звужено, розглянемо задачу вибору шляху переходу між двома сусідніми станами. Така задача є двохкритеріальною. Спочатку розглянемо кожний критерій окремо, а потім зведемо двохкритеріальну задачу до одного критерію.

2.2 Моделювання діяльності інтелектуальних агентів у конкурентному середовищі

Відповідно з ідеями Джона Бойда та його послідовників будь-яка діяльність у конкурентному середовищі з певним ступенем наближення може бути представлена у вигляді кібернетичної моделі OODA [46]. Зазначена

модель передбачає багаторазове повторення петлі дій, складеної з чотирьох послідовних взаємодіючих процесів (рис. 2.3): спостереження (observation); орієнтація (orientation); прийняття рішення (decision); дія (action).

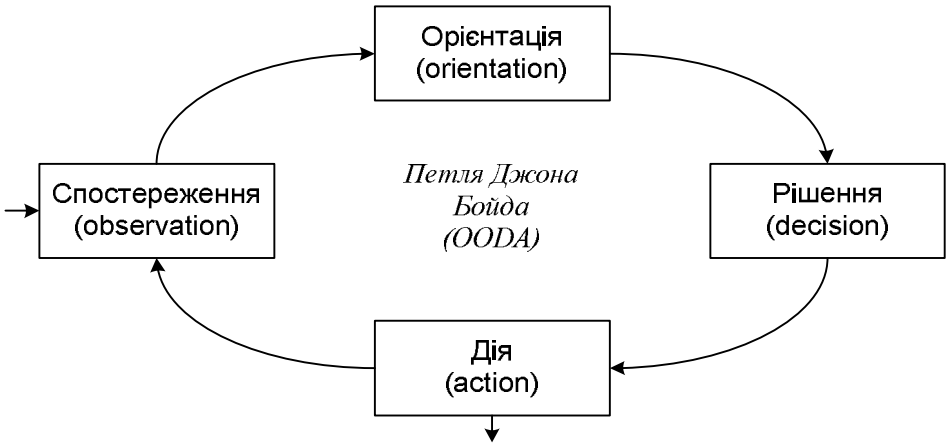

Рис. 2.3. Процеси петлі OODA

Така модель з успіхом почала застосовуватися для моделювання діяльності та прийняття рішень у бізнесі, політиці, соціології тощо, тобто у тих сферах, де наявна конкуруюча сторона.

Згідно теорії Бойда кожна людина або організація при вирішенні поставлених перед ними завдань має свою петлю прийняття рішень і діяльності.

Згідно до літературних джерел з теорії Д.Бойда, підсумок яких здійснено у [46], мету етапів петлі та їх функціональне призначення є такими.

Спостереження (observation) – це процес збору інформації, необхідної для прийняття рішення у деякому конкретному випадку. Необхідна інформація може бути отримана як від зовнішніх, так і від внутрішніх джерел. Під внутрішніми джерелами інформації розуміються елементи зворотного зв'язку петлі. В якості зовнішніх використовуються датчики, а також інші канали отримання інформації.

Щоб спостереження набуло наукового характеру, потрібно, аби воно:

- було планомірним, а не випадковим;

- здійснювалося послідовно й систематично;
- було забезпечене достатньо широкою інформацією про явище, яке є предметом спостереження (слід оперувати якомога більшою кількістю фактів);
- передбачало точну фіксацію результатів спостереження.

Збір даних може здійснюватися:

1) механічним способом; механічна реєстрація даних полягає в тому, що джерело інформації, тобто «подія» або «явище», виявляється у вигляді зміни деякого фізичного стану, і цей новий стан реєструється механічним способом;

2) експертом; спостереження, здійснюване експертом, із наступним відновленням результатів по пам'яті, які називають «записом»;

3) шляхом експериментального дослідження, особливість якого полягає в тому, що явище (предмет дослідження) вивчається за різних умов та обставин; застосування цього методу дослідження сприяє глибокому і дуже точному вивченню певної психологічної закономірності.

Орієнтація (orientation) – найвідповідальніший і найскладніший з когнітивної точки зору етап у всьому циклі OODA. Етап орієнтації складається з двох підетапів: руйнування (destruction) і творення (creation). Руйнування передбачає розбиття ситуації на дрібні елементарні частини, які більш легкі для розуміння. Людина або організація, які приймають рішення, будуть прагнути здійснити декомпозицію завдання до такого рівня, поки новоутворені складові завдання стають близькими до стандартних, або типових ситуацій, для яких об’єкт прийняття рішень (ОПР) має план дій. Ознайомлення з цими елементарними типовими підзадачами досягається шляхом навчання, тренування, накопичення досвіду та інструктажу. ОПР ідентифікує поточну ситуацію до тих, з якими він знайомий, і застосовує заздалегідь заготовлений план дій для цієї підзадачі. Потім ці складові елементарні підплани об'єднуються в загальний план дій, що і відповідає підетапу «творення». Якщо немає планів, з числа яких може бути обрано рішення, то процес залишається

на етапі орієнтації і здійснюється подальша декомпозиція задачі. Якщо не вдається розробити план з реальними шансами на успіх, то подальше подрібнення може призвести до останнього циклу.

Для орієнтації використовуються методи аналізу і синтезу які тісним чином пов'язані між собою. Вони призначені для обробки інформації, отриманої в результаті застосування дослідницьких методів.

Аналіз являє собою вивчення якостей, властивостей і характеристик досліджуваного об'єкта за допомогою його умовного поділу на окремі складові частини.

У свою чергу синтез полягає в узагальненні інформації про окремих складових і формуванні сукупності інформаційних даних про об'єкт дослідження в цілому.

Результати, отримані в процесі аналізу та синтезу, служать основою для складання різного роду прогнозів на найближчу і далеку перспективу. Прогнозування може здійснюватися методами розрахунку і екстраполяції.

На етапі декомпозиції системи здійснюється:

- визначення та декомпозиція загальної мети дослідження та головної функції системи як обмеження траєкторії в просторі станів системи або в області допустимих ситуацій. Найчастіше декомпозицію виконують побудовою дерева цілей та дерева функцій;
- виділення системи із середовища (поділ на «систему» та «несистему»);
- опис впливових факторів;
- опис тенденцій розвитку;
- опис системи як «чорного ящика»;
- функціональна (за функціями), компонентна (за типом елементів), структурна (за типом відношень між елементами) декомпозиція системи.

Прийняття рішення (decision) – третій етап циклу OODA. Якщо до цього етапу ОПР змогла сформувати тільки один реальний план, то приймається рішення – виконувати цей план чи ні. Якщо ж сформовані кілька

альтернативних варіантів дій, то ОПР на даному етапі здійснює вибір найкращого з них для подальшої реалізації. Вибір найкращого плану може здійснюватися за критерієм ефективність-вартість. В умовах ліміту часу кращим вважається той план, що відповідає вимогам швидкої надійності.

Для прийняття рішення використовуються такі методи:

- метод ефективність-вартість враховує три етапи: побудова моделі ефективності, побудова моделі вартості, синтез вартості й ефективності;
- методи теорії і практики надійності базуються на застосуванні апарата теорії ймовірностей і випадкових процесів, математичної статистики та моделювання.

Дія (action) – заключний етап циклу, що передбачає практичну реалізацію обраного курсу дій або плану.

Існують два основних способи досягнення конкурентних переваг при здійсненні різних видів професійної діяльності. Перший шлях – зробити в кількісному вимірі свої цикли дій більш швидкими. Це дозволить діяти першим номером і змусить конкурентів реагувати на наші дії. Другий шлях – покращити якість прийнятих рішень, тобто приймати рішення, які у більшій ступені відповідають ситуації, яка склалася, ніж рішення конкурентів.

Підвищення якості власних рішень може бути досягнуто різними способами, до числа яких відносяться застосування сучасних формальних математичних методів, застосування автоматизованих систем керування, систем підтримки прийняття рішень, експертних систем. Якщо використовувати останні, то сучасний підхід до їх побудови використовує в якості ядра баз знань онтології [34]. Тому виникає задача з розроблення методів використання онтологій у петлі OODA.

Відмітна риса циклу OODA від інших циклічних моделей полягає в тому, що в будь-якій ситуації завжди передбачається наявність конкурентної сторони. На рис. 2.4 наведено три об’єкти управління, які знаходяться в деяких своїх початкових станах й мають власні стани мети. Те, що ці об’єкти функціонують

в конкурентному середовищі, змушує під час проходження петлі OODA аналізувати стани в яких знаходяться конкуренти та їх дії, що відображено на цьому рисунку відповідними стрілками.

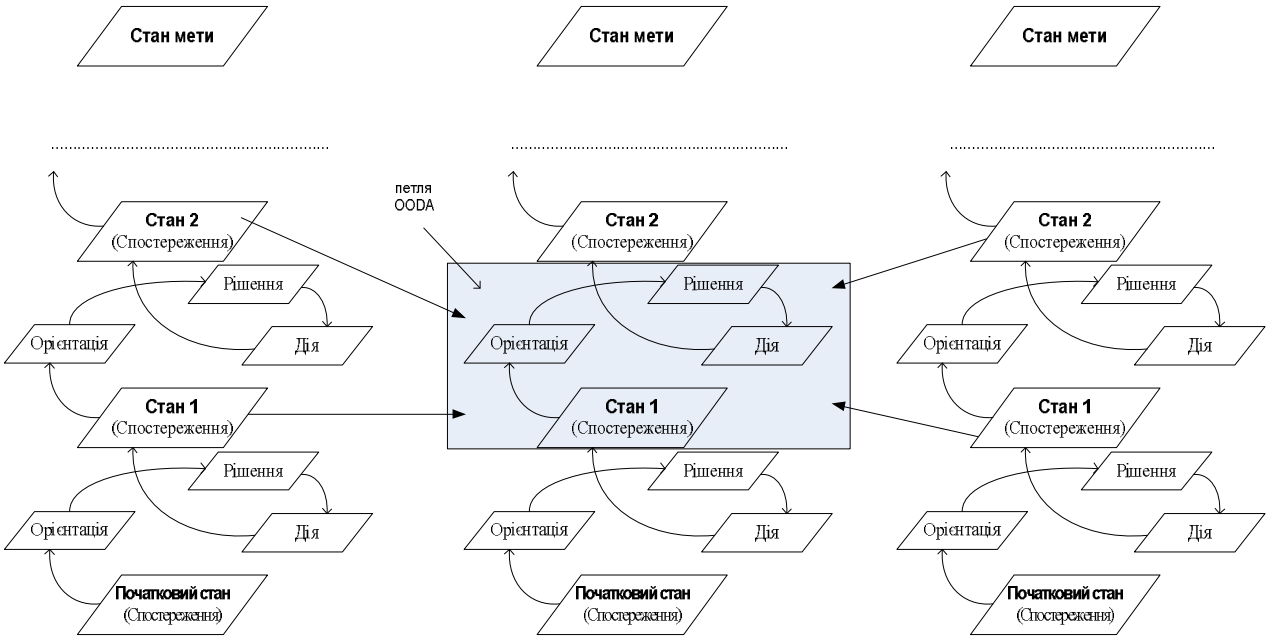

Рис. 2.4. Функціонування об’єктів у конкурентному середовищі

Зміст онтології напряму впливає на 2-й і 3-й етапи циклу, а сама структура та наповнення онтології залежить від 1-го та 2-го етапів (див. рис. 2.5).

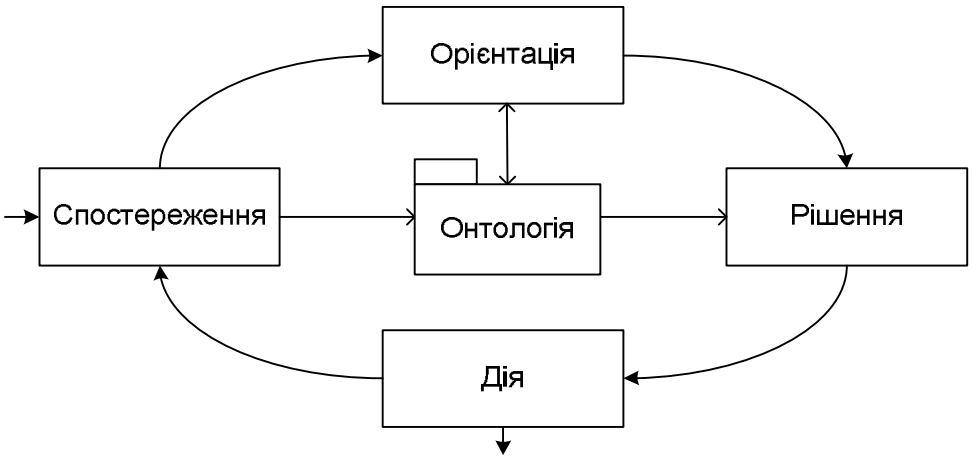

Рис. 2.5. Використання онтологій в петлі OODA

Процес наповнення онтології буде розглянуто нижче. Вплив онтології на процес рішення моделюється за допомогою математичної моделі (2.9), якщо онтологія є адаптивною.

Однак зміна онтології, тобто її навчання, призводить до зміни ваг понять онтології. Тому для моделювання поведінки інтелектуальних агентів необхідно використати методи навчання. Пропонується використати стимулююче навчання, оскільки наші ІА безпосередньо взаємодіють з навколишнім середовищем.

2.3 Моделювання поведінки інтелектуальних агентів на основі стимулюючого навчання

2.3.1 Модель кінцевого горизонту

Одним із способів вимірювання успіху діяльності ІА є використання методів, які ґрунтуються на стимулюючому навчанні. Під час стимулюючого навчання аґент, що навчається, взаємодіє з навколишнім середовищем, виконуючи якісь дії (вибираючи їх, звичайно, з фіксованого набору). Навколишнє середовище його якимсь чином заохочує за ці дії, а аґент продовжує їх робити. Єдиний спосіб для аґента зрозуміти, що він чинить правильно, – стежити за заохоченнями від навколишнього середовища.

Формально задача має такий вигляд. Нехай на кожному кроці аґент перебуває у стані s з деякої множини станів S. На кожному кроці він вибирає з наявного набору дій A деяку дію a. У відповідь на це навколишнє середовище повідомляє аґенту, яку винагороду він отримав і в якому стані.

У загальному випадку – аґент повинен досліджувати навколишнє середовище і вибирати оптимальну поведінку. Виникає задача як порівнювати і оцінювати алгоритми стимульованого навчання під час діяльності раціональних агентів.

Проаналізуємо алгоритми, що навчаються отримувати винагороду від навколишнього середовища. Такі алгоритми «мають добру поведінку», тобто

отримують велику винагороду. Але що таке «добре»? На це запитання є декілька відповідей, вибір між якими залежить від того, на який термін життя аґента вони розраховані і на що необхідно звернути увагу у своїх оцінках.

Перша, проста модель – це так звана модель кінцевого горизонту (finite horizon model). У ній якість поведінки аґента вимірюється тільки по відношенню до наступних h кроків, і здатністю максимізувати потрібно величину $E\left[\sum_{t=0}^{h} r_t\right]$ [48].

Ця модель відповідає ситуації, коли в аґента обмежений термін життя. Наприклад: ми знаємо, що у нас лише десять спроб, і тому нас не цікавить, чи успішно ми навчимося виконувати одинадцяту.

Природно хотілося б врахувати всі можливі кроки в майбутньому, тому що час життя аґента може бути не визначений. Але при цьому в більшості реальних задач чим раніше ми отримаємо нагороду від навколишнього середовища, тим краще. Наприклад, гривня сьогодні – це набагато краще, ніж гривня через рік, адже її за цей час можна розумно вкласти і примножити.

Як це врахувати в математичній моделі? Для цього досить надати швидшому прибутку більші ваги, а віддаленішому в часі – менші. За рахунок уведеного коефіцієнта γ та ще й при сумуванні за нескінченною довжиною життя аґента ряд почне збігатися (вважають, що виплати r_t обмежені зверху – врешті-решт, кількість різних станів і різних виплат скінченна). Отже математичне очікування суми ряду має вигляд:

$$E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right],$$

де γ – деяка константа (звана в англомовних текстах discount factor).

Така модель називається моделлю нескінченного горизонту (infinite horizon model). У нашому подальшому викладі, користуватимемося саме її [16].

Третя модель, яка все рідше зустрічається, – модель середньої винагороди (average-reward model). У ній максимізації підлягає границя очікування середньої винагороди:

$$\lim_{h\to\infty} E\left[\frac{1}{h}\sum_{t=0}^{h} r_t\right].$$

Ця модель не дає швидкого виграшу, тому на практиці нечасто є кращою за модель нескінченного горизонту [48].

Всі вищеописані моделі різні, і нескладно побудувати граф станів, на якому всі три моделі приводитимуть до різних оптимальних стратегій. Окрім поведінки вже готового алгоритму, треба також навчитися оцінювати і якість його навчання – те, наскільки якісно він прагне до своєї границі.

Щонайперша (і зазвичай не надто складно доказова) властивість алгоритмів стимульованого навчання – збіжність до оптимального алгоритму (для конкретної моделі, звичайно). Це корисна властивість, але, на жаль, вона нічого не говорить ні про швидкість збіжності, ні про те, з якими втратами ми підійдемо до оптимального алгоритму.

Щодо оцінки швидкості збіжності, то тут можливі два альтернативні підходи: оцінка швидкості збіжності до якоїсь фіксованої частини оптимальності (власне оптимального алгоритму аґент, що навчається, зазвичай досягає лише на нескінченності) і оцінка якості роботи алгоритму через деякий фіксований час. На жаль, в обох випадках виникають конкретні запитання: яка потрібна частина оптимальності? Через який час треба порівнювати якість алгоритмів? Однозначну відповідь на ці запитання знайти важко.

Третій можливий підхід – мінімізувати ціну (regret) вибору тієї чи іншої стратегії навчання. Будь-яка стратегія не може бути кращою за оптимальну стратегію із самого початку. Іншими словами, задача оптимального навчання – це задача мінімізації різниці між загальною сумою виграшу при навчанні у порівнянні із застосуванням оптимальної стратегії із самого початку. Якщо вдається підрахувати цю різницю, то можна дуже просто порівнювати

алгоритми навчання – у якого різниця менша, той і переміг. Ця міра, власне, й відображає справжню кінцеву мету того, що відбувається. Але, на жаль, вона погано піддається аналізу: розумні результати про цю міру отримати дуже складно.

Кожний алгоритм стимульованого навчання повинен вивчати навколишнє середовище і користуватися своїми знаннями, щоб максимізувати виграш. Виникає запитання – як досягти оптимального співвідношення?

2.3.2 Формалізація функціонування інтелектуальних агентів

Рухатимемося від простого до складного. Розглянемо поведінку ІА, що мають лише один стан ($|S|=1$). У такого аґента є фіксований набір дій A і можливість вибору дії з цього набору.

У штучному інтелекті загальноприйнята витончена і доступна для розуміння модель поведінки такого аґента. Нехай аґент перебуває в кімнаті з декількома ігровими автоматами. У кожного автомата є своє, невідоме аґентові очікування виграшу. Аґент повинен за обмежену кількість спроб отримати максимальний виграш.

Відзначимо, що така (здавалося б, сильно спрощена) модель насправді реалізує всі можливі ситуації з одним станом. Дійсно, за кожну виконану дію середовище дає винагороду (можливо, випадково) і повертає аґента в початковий стан (результат наступного підходу до автомата не залежить від попередніх).

Розглянемо загальні принципи, якими повинен керуватися аґент в такій ситуації. Зрозуміло, хочеться застосувати в якомусь сенсі «жадібну» стратегію, тобто завжди вибирати стратегію, яка максимізує виграш [49]. Проте абсолютно очевидно, що аґент в кімнаті не знає із самого початку, що йому робити, і не може вибрати найкращу стратегію. Понад це, навіть якщо він вже смикнув по одному разу за ручку кожного автомата і лише один раз переміг, це зовсім не означає, що треба тепер весь час вибирати автомат, на якому він переміг.

Для цього в літературі використовується евристика – оптимізм при невизначеності [50]. Інакше кажучи, вибирати стратегію поведінки треба жадібно, але при цьому бути вельми оптимістичним щодо очікуваного виграшу. Мають бути отримані серйозні негативні наслідки, перш ніж та чи інша стратегія буде відхиленою.

Розглянемо методи, які ґарантовано знаходять оптимальний розв'язок або збігаються до нього. Одним із таких методів є динамічне програмування. Динамічне програмування широко використовується для різних задач. Не стало винятком і стимульоване навчання. Динамічне програмування тут можна застосувати, якщо заздалегідь відомий термін життя аґента: припустимо, що аґент діє впродовж *h* кроків.

Тоді для визначення оптимальної стратегії можна використати нескладний байєсівський підхід [51]. Треба обчислити відображення зі всіх можливих станів досвіду (belief states) аґента в множину дій, таким чином задавши йому стратегію поведінки.

Стан досвіду аґента виражається як $S = \{n_1, w_1, ..., n_k, w_k\}$, що означає, що кожний автомат *i* запустили n_i разів, отримавши при цьому виграш w_i разів (тут і далі ми вважаємо, що результат бінарний – або виграв, або програв).

Позначимо через $V^*(S)$ очікуваний виграш. Базою для рекурентних співвідношень служитиме випадок, коли $\sum_{i=1}^{k} n_i = h$. У цьому випадку аґентові більше нічого робити, і $V^*(S) = 0$. Якщо ж ми знаємо V^* для всіх станів, коли у аґента ще залишилося *t* спроб, ми зможемо перерахувати V^* і для станів з *t*+1 спробою, що залишилася:

$$V^* = \{n_1, w_1, ..., n_k, w_k\} = \max_i \left(p_i \left(1 + V^*(...., n_i + 1, w_i + 1, ...)\right) + (1 - p_i) V^*(...., n_i + 1, w_i + 1, ...)\right),$$

де p_i – апостеріорна ймовірність того, що дія *i* виправдається при даних досвіду (n_i, w_i). У випадку з випробуваннями Бернуллі:

$$p_i = \frac{w_i + 1}{n_i + 2}.$$

Отриманий алгоритм наведений на рис. 2.6.

1. Ініціалізувати таблицю $\left\{V^* = \left(n_1, w_1, ..., n_k, w_k\right)\right\}_{n_i, w_i = 1}^{h}$.

2. Для всіх елементів, для яких $\sum_{i=1}^{n} n_i \geq h$, присвоїти $V^* = \left\{n_1, w_1, ..., n_k, w_k\right\} := 0$.

3. Для всіх t від h до 0:

для всіх елементів, для яких $\sum_{i=1}^{n} n_i = t$, присвоїти

$$V^* = \left\{n_1, w_1, ..., n_k, w_k\right\} :=$$

$$= \max_i \left\{ \frac{w_i + 1}{n_i + 2}\left(1 + V^*\left(...., n_i + 1, w_i + 1, ...\right)\right) + \left(1 - \frac{w_i + 1}{n_i + 2}\right) V^*\left(...., n_i + 1, w_i, ...\right) \right\}.$$

4. На кожному кроці із заданим фіксованим $\left\{n_1, w_1, ..., n_k, w_k\right\}$, вибираємо дію i, для якої

$$\max_i \left\{ \frac{w_i + 1}{n_i + 2}\left(1 + V^*\left(...., n_i + 1, w_i + 1, ...\right)\right) + \left(1 - \frac{w_i + 1}{n_i + 2}\right) V^*\left(...., n_i + 1, w_i, ...\right) \right\}.$$

Рис. 2.6. Динамічне програмування для стимульованого навчання

Цей алгоритм вийшов не дуже ефективним: ціна побудови такої таблиці лінійна добутку кількості станів досвіду на кількість дій $|A|$, що експоненціально залежить від h. Інакше кажучи, динамічне програмування можна використовувати тільки у випадках, коли потрібно планувати не дуже далеко. Зате такий підхід ґарантовано приводить до оптимальної стратегії.

Очікуваний виграш V^* абсолютно не залежить від конкретної ситуації, а від конкретних очікувань виграшу автоматів. Метод динамічного програмування побудує шлях по цій таблиці, який вибере оптимальна стратегія.

Нехай ми вже *n* разів зіграли з автоматом і отримали *w* одиниць виграшу. Існують таблиці індексів розподілу Гіттінса (Gittins allocation indices) *I(n,w)*, які враховують як очікуваний виграш, так і кількість нової інформації, яку ми отримаємо, якщо зробимо цю дію ще раз. Отже, оптимальна стратегія проста: вибирати автомат, для якого значення *I(n,w)* максимальне.

Індекси розподілу Гіттінса працюють відмінно, але, на жаль, ніяк не узагальнюються – їх аналоги для декількох станів аґента невідомі [52].

Третій прийом серед тих, аналіз яких провести досить просто, пов'язаний з «тренуванням» оптимальної стратегії. Дійсно, в моделі з кімнатою з автоматами будь-який алгоритм можна подати як набір імовірності, з якою він вибирає ту чи іншу дію (все решта не важливо для кінцевого результату). І цю ймовірність можна тренувати приблизно як нейронну мережу. Алгоритм лінійної винагороди-бездіяльності (linear reward-inaction algorithm, рис. 2.7) лінійно збільшує ймовірність дії a_i, якщо вона привела до успіху:

$$p_i := p_i + \alpha(1 - p_i),$$
$$p_j := p_j - \alpha p_j, j \neq i.$$

Якщо вона безуспішна, то ймовірність зберігається. У цих формулах α – константа, задає швидкість навчання.

Алгоритм лінійної винагороди-бездіяльності з імовірністю 1 збігається до вектора з однієї одиниці і решти нулів. Він не завжди збігається до оптимальної стратегії; цілком ймовірно, що не дуже характерні перші декілька запусків «навчать» алгоритм настільки, що вона вже ніколи не повернеться до оптимального варіанту, а приведе до одиниці ймовірність субоптимального вибору. Однак ймовірність помилитися можна зробити скільки завгодно малою, зменшуючи .

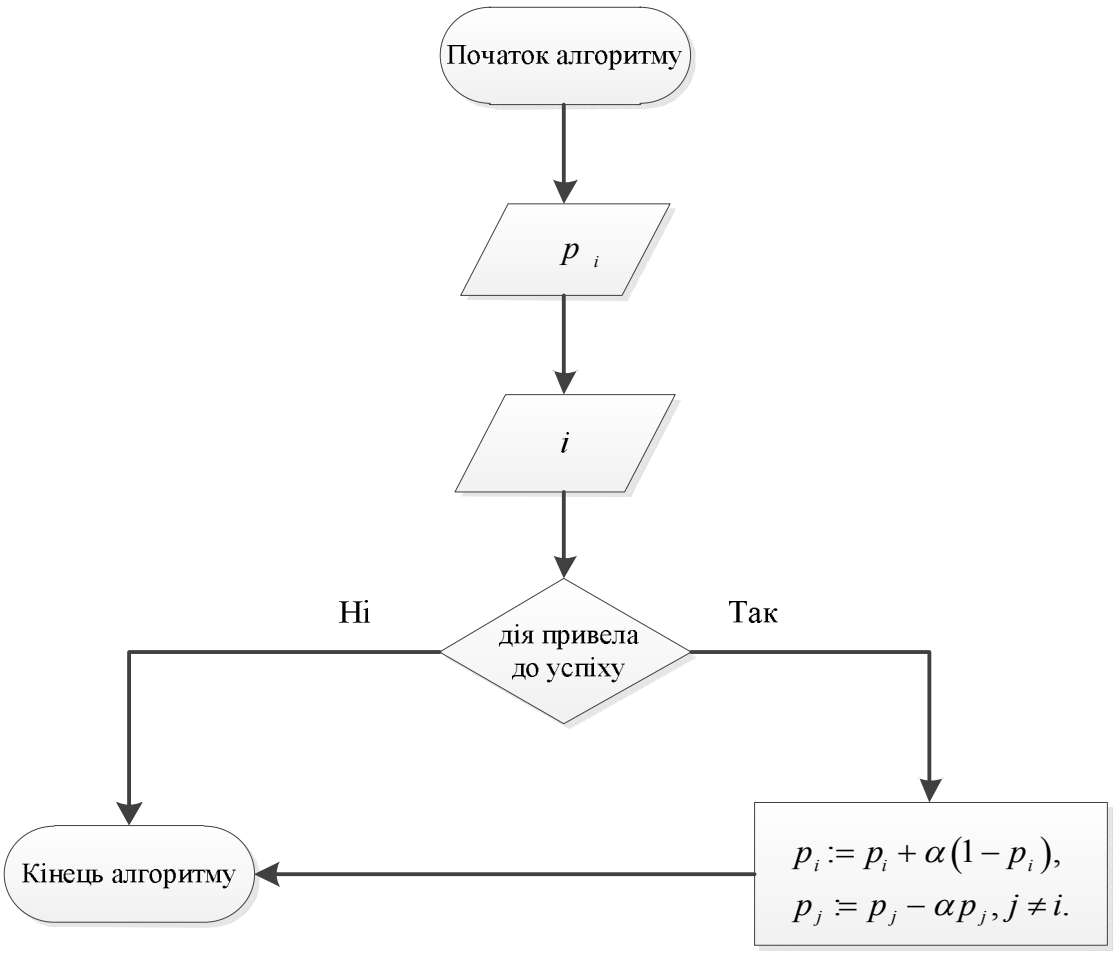

Рис. 2.7. Алгоритм лінійної винагороди-бездіяльності

Крім алгоритмів, які доказово збігаються до оптимального розв'язку, бувають також випадки, в яких оптимальність абсолютно не ґарантована, однак на практиці виявляється, що метод працює добре.

Розглянемо детальніше такі випадки. Перший з них – випадкові стратегії. Проста випадкова стратегія виглядає так: вибрати дію з якнайкращим очікуваним виграшем з імовірністю p, а з імовірністю $1 - p$ вибрати випадкову дію. Щоб імовірність не збіглася до субоптимальної дії, зазвичай починають з маленьких значень p, а потім їх збільшують.

Основний недолік цього алгоритму очевидний: він виділяє лідера, але не відрізняє хорошу йому альтернативу від поганої. Щоб взяти до уваги відмінності між всіма стратегіями, треба «розмазати» ймовірність по всіх діях, беручи до уваги їх очікуваний виграш, але при цьому не забуваючи про

додаткові дослідження. Одним з відповідних методів є дослідження за Больцманом (Boltzmann exploration):

$$p(a) = \frac{e^{ER(a)/T}}{\sum_{a'} e^{ER(a')/T}},$$

де *ER* – очікуваний виграш, *T* – фіксована константа. Ця константа визначає, наскільки велика різниця між ймовірністю вибору хороших і поганих стратегій; очевидно, що чим вища константа, тим ближче ймовірності одна до одної, а чим нижча, тим більша ймовірність вибору оптимальної і близької до неї дії. Зазвичай така константа починається з достатньо високого значення, а потім з часом знижується.

Друга евристична стратегія відноситься до класу «оптимістично жадібних» алгоритмів. Припустимо, що нам потрібно вибрати стратегії досить оптимістично, але при цьому все ж таки врахувати наявний негативний досвід. Теорія ймовірності тут пропонує вельми розумний вихід – довірчі інтервали!

Для кожної дії треба зберігати статистику *n* і *w*, і перед ухваленням рішення обчислювати довірчий інтервал для ймовірності успіху (з деякою наперед заданою межею $1-\alpha$), а для вибору дії, що реалізовується, максимізувати верхню межу цього інтервалу.

Алгоритм стимульованого навчання методом довірчих інтервалів наведено на рис. 2.8.

1. Ініціалізація довірчих інтервалів $I_k = (I_{1k}, I_{2k}) := (0,1)$.

2. Поки агент продовжує роботу:

а) вибрати дію $j = \arg\max_{i=1..k} I_k^2$.

б) Перерахувати довірчий інтервал I_j, залежно від результатів цієї дії.

Рис. 2.8. Алгоритм стимульованого навчання методом довірчих інтервалів

2.3.3 Модель агентів з декількома станами

Вище розглянуто ситуацію – коли у аґента рівно один стан, тобто впродовж всієї роботи алгоритму множина дій і їх результатів не змінюється. Проте в реальному житті алгоритмам, що навчаються, зазвичай треба переходити з одного стану в інший, перш ніж буде досягнутий якийсь певний результат. Отже, потрібно побудувати адекватну модель, що описує переходи між цими станами і таку, що задає максимізацію отриманого виграшу. Для цього пропонується використати марковські процеси (детально розглянуто вище у 1-му розділі).

Марковський процес прийняття рішень (Markov decision process) складається з:

- множини станів S;
- множини дій A;
- функції заохочення $R : S \times A \to R$;
- функції переходу між станами $T : S \times A \to \Pi(S)$, де $\Pi(S)$ – множина розподілів ймовірностей над S.

Отже, ймовірність потрапити зі стану s у стан s' після здійснення дії a позначається через $T(s, a, s')$.

Відзначимо, що в нашій моделі переходи між станами ймовірнісні: після тієї чи іншої дії новий стан настає не з абсолютною необхідністю, а з деяким (заданим) розподілом ймовірності. А ось функцію винагороди для простоти вважаємо постійною.

Переходи між станами не залежать від історії попередніх переходів, тобто перебуваючи у певному стані, аґент за однакові дії отримуватиме однакову винагороду, незалежно від того, яким чином аґент прийшов у цей стан.

Задача полягає в максимізації виграшу. Зрозуміло, що в реальній ситуації на початку процесу агент перебуваємо в абсолютному незнанні – не відома реакція системи на жодні дії, у тому числі й переходи між станами. Однак, будемо вважати, що модель задачі відома. Саме розв’язок цієї задачі

необхідний для розв'язування загальнішої. Тому спочатку розглянемо алгоритми, які знаходять оптимальну стратегію для відомої моделі, а потім перейдемо до складнішої задачі навчання.

Введемо таке поняття як оптимальне значення стану. Оптимальне значення стану – очікуваний сумарний виграш, який отримає аґент, якщо почне з цього стану і слідуватиме оптимальній стратегії:

$$V^*(s) = \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right].$$

Іншими словами, оптимальне значення стану – це та нагорода, яку отримуємо, якщо гратимемо найкращим чином. Це значення можна визначити як розв'язок рівнянь:

$$V^*(s) = \max_{a \in A}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s')\right). \qquad (2.10)$$

Якщо його знати, то вибір оптимальної стратегії здійснюємо згідно формули:

$$\pi^*(s) = \arg\max_{a}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s')\right).$$

На рис. 2.9 та рис. 2.10 наведено алгоритми, які розв'язують цю задачу ітераційним способом. Перший з них здійснює ітерації за значеннями, тобто намагається на кожному кроці підрахувати функцію $Q(s,a)$ – поточну версію очікування виграшу у випадку вибору дії a у стані s, а потім, коли значення цієї функції нас вже задовольняють, вибирає ту дію, яка максимізує це очікування. Алгоритм зупиняється тоді, коли значення $Q(s,a)$ перестають змінюватися (тобто різниця між послідовними значеннями за модулем перестає перевищувати деяку заздалегідь фіксовану межу ε).

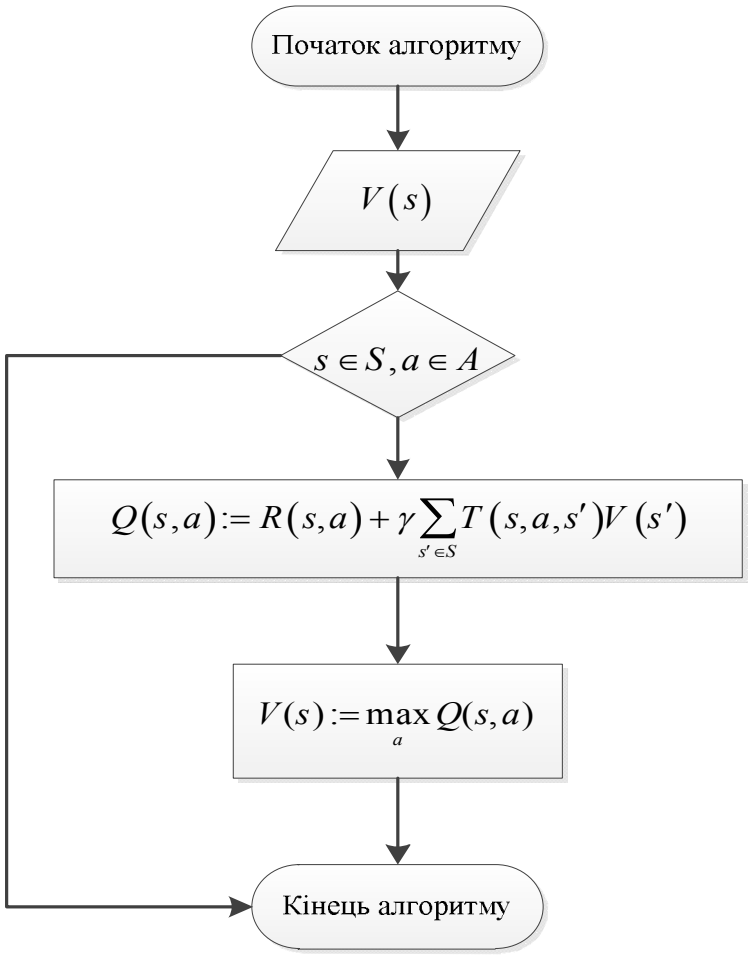

Рис. 2.9. Алгоритм пошуку оптимального значення стану ітераціями за значеннями

Відзначимо, що перерахунок в цьому алгоритмі використовує інформацію від всіх станів-попередників. Але можна використати й інший, «стохастичний» варіант:

$$Q(s,a) := Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right).$$

Доведено, що він працює, якщо кожна пара (s,a) зустрічається нескінченну кількість разів, *s*' вибирають з розподілу $T(s,a,s')$, а *r* вибирають із середнім $R(s,a)$ і обмеженою варіацією.

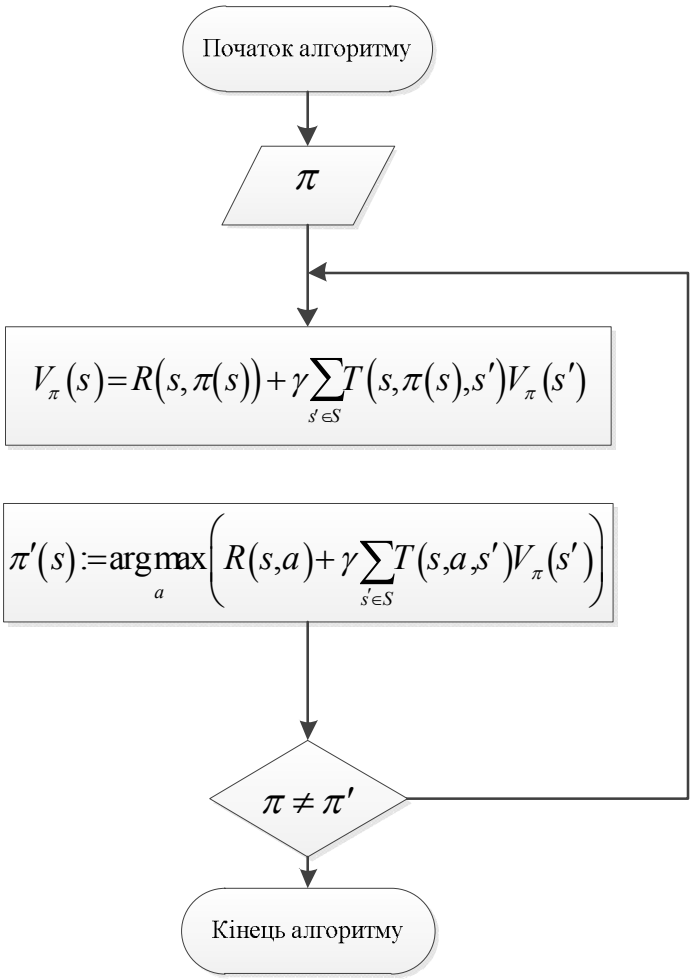

Рис. 2.10. Алгоритм пошуку оптимального значення стану ітераціями за стратегіями

Алгоритм ітерації за стратегіями (рис. 2.10) використовує ту саму ідею, але ітерації відбуваються не за очікуваними значеннями, а власне за стратегіями π, якими може слідувати агент. Алгоритм має більшу складність, ніж алгоритм ітерації за значеннями, оскільки потрібно на кожному кроці розв'язувати системи лінійних рівнянь. Зате для ітерацій за стратегіями очевидна збіжність, оскільки на кожному кроці цільова функція строго покращується, а всього існує скінченна кількість $\left(|A|^{|S|}\right)$ стратегій. Правда, це міркування дає значну оцінку на час роботи алгоритму; на практиці він працює куди швидше, але теоретично не відомо, чи він поліномний.

Беручи до уваги (2.9) та (2.10) отримуємо двохкритеріальну задачу. З математичної точки зору не існує ідеального методу або способу розв'язання таких задач. Кожний з них має свої певні переваги та недоліки і область застосування. Проаналізувавши відомі методи, обрано метод головної компоненти, якщо цільові (2.9) та (2.10) можна оцінити відповідно знизу або зверху; якщо їх оцінити неможливо, то використовуємо метод комплексного критерію. Тим самим отримуємо одну із 3-ьох задач:

$$\min V^*,\ Q^* \geq Q, \tag{2.11}$$

$$\max Q^*,\ V^* \leq V, \tag{2.12}$$

$$\min f = \frac{V^*}{Q^*} \tag{2.13}$$

.

Задачі (2.11)-(2.13) є багатокроковими оптимізаційними задачами, тобто задачами динамічного програмування. Використовуючи метод розв'язку таких задач (наприклад функціональних рівнянь) знаходимо шлях переходу із початкового стану в стан мети. Для її розв'язування обрано метод головної компоненти, якщо цільові функції можна оцінити або метод комплексного критерію, якщо ці функції оцінити неможливо.

2.4 Основні результати та висновки до розділу

У цьому розділі роботи отримано такі результати:

1. Розроблено математичне забезпечення функціонування інтелектуальних агентів планування діяльності на основі онтологій, що дало змогу формалізувати поведінку таких агентів у просторі станів. Використання онтологій дає змогу звужувати простір пошуку шляху із початкового стану в стан мети, відкидаючи нерелевантні альтернативи.

2. Задача планування діяльності інтелектуального агента зводиться до задачі динамічного програмування, де функцією мети є композиція двох функцій, які задають конкурентні критерії. Тобто у результаті отримуємо двохкритеріальну задачу. Для її розв’язування обрано метод головної компоненти, якщо цільові функції можна оцінити або метод комплексного критерію, якщо ці функції оцінити неможливо.

РОЗДІЛ 3 . ОНТОЛОГІЧНИЙ МЕТОД ПОДАННЯ ЗНАНЬ СПЕЦІАЛІЗОВАНИХ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ В ЗАДАЧАХ ПЛАНУВАННЯ ДІЯЛЬНОСТІ

Для побудови онтології, яка адекватно описує семантичну модель ПО, необхідно, насамперед, розв’язати задачі одержання знань із різних джерел для виявлення множини концептів і встановлення ієрархії на цій множині. Оскільки значна частина інформації міститься в природномовних текстах (ПМТ), перспективним є одержання знань із текстової інформації, а також інтелектуальне опрацювання спеціально підібраних колекцій ПМТ.

3.1 Автоматизоване навчання онтології природомовними текстами

Одним з найефективніших підходів до наповнення онтології є її автоматизоване навчання природомовними текстами. Автоматизоване наповнення можна реалізувати за допомогою аналізу текстових документів, застосувавши процесор знань (рис. 3.1). Детальніше такий підхід розглянуто у монографії [19].

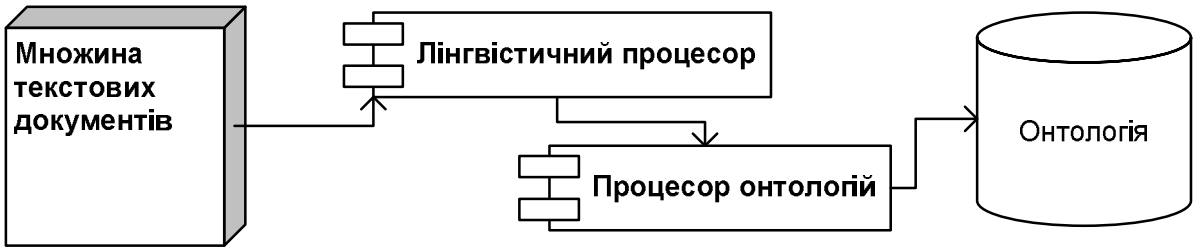

Рис. 3.1. Структурно-функціональна схема процесора знань

У поданій схемі завдання лінґвістичного процесора – виконати його лексичний, лексико-граматичний, синтаксичний та семантичний аналіз. У результаті цього онтологія поповнюється: поняттями, СДО-трійками (суб’єкт – дія – об’єкт) і причинно-наслідковими зв’язками між СДО-трійками. Іншу

частину важливих зв’язків між поняттями та їхніми властивостями встановлює процесор онтологій, котрий будує онтологічну структуру для кожного концепту C, отриманого після аналізу тексту. Робота процесора онтологій підтримується відповідною БЗ, основними компонентами якої є: по-перше, множина правил, по-друге, універсальна логічна БД MModWN [40] типу WordNet [54]. Процесор знань застосовують у системі автоматизованого одержання знань із текстових документів, котра, своєю чергою, застосовується для розв’язання задачі семантичного пошуку в повнотекстових БД. Серед систем, розроблених в Україні, треба відзначити розробку колективу кафедри математичної інформатики Київського національного університету імені Тараса Шевченка – систему опрацювання текстів природною мовою [55]. Система створена для розв’язування таких задач, як аналіз та синтез текстів природною мовою, автоматизоване ґенерування реферату тексту, автоматизована індексація (визначення тематики) тексту.

Найвагомішим технічним рішенням у системі є можливість «зважувати» вершини семантичної мережі тексту. Найважливішими вершинами мережі вважаються вершини, котрі мають найбільшу кількість зв’язків з іншими. Цю процедуру можна застосовувати під час побудови образу реферату зважуванням вершин і відкиданням найлегших – «марґінальних». Очевидні і недоліки: по-перше, процедура зважування не враховує важливість зв’язків-відношень між вершинами-поняттями, по-друге, відсутні критерії, на основі яких в системі будується оптимізований граф тексту, де вершини і зв’язки мають свою тимчасову оцінку.

Основна перевага нашого підходу полягає у побудові ІА, який визначає цінність повідомлень, які пропонуються додавати в онтологію в залежності від вибраного плану управління.

Особливості функціонування спеціалізованого ІА визначаються його інтересом – вектором оцінок бажаності можливих станів агента. Для опису інтересу агента, за допомогою якого він розрізняє стани довколишнього світу

та позиціонує себе у ньому, застосовується функція корисності, котра є числовою оцінкою його бажаності для агента. Корисності об'єднуються з імовірностями дій для визначення очікуваної корисності кожної дії.

Нехай *U(S)* – корисність стану *S* з точки зору агента, що приймає рішення щодо вчинення деякої дії *A*. Довільна недетермінована дія може спричинити результуючий стан $Result_i(A)$, де індекс *i* пробігає по усіх можливих результатах. Перш ніж вчинити дію *A*, агент оцінює імовірність $P(Result_i(A)|Do(A),E)$ кожного з можливих результатів, де *E* – сукупність доступних агенту параметрів його стану, а *Do(A)* − висловлювання, згідно з яким в біжучому стані виконується дія *A*. Таким чином, можна обчислити умовну корисність дії $EU(A|E)$ з врахуванням відомих параметрів стану:

$$EU(A \mid E) = \sum P(Result_i(A) \mid Do(A), E) \cdot U(Result_i(A)).$$

Якщо ІА керується принципом максимальної очікуваної корисності (Maximum Expected Utility − MEU), він змушений вибирати дію, яка максимізує очікувану корисність для агента. Так функціонує механізм мотивації поведінки раціонального інтелектуального агента незалежно від сфери його застосування.

У випадку інформаційно-пошукового агента його інтерес може бути заданий через оцінку новизни отриманих повідомлень, яка потребує застосування методів інтелектуального аналізу природомовних текстів. Вважаємо, що текст побудований як повідомлення. Структура повідомлення орієнтована на сприйняття іншим агентом, тому складається з двох частин (рис. 3.2): констатуючої частини, за якою адресат оцінює релевантність повідомлення (1) та визначає його контекст (2), та конструктивної частини – потенційно нових для читача знань у даному контексті (3).

Якщо нове знання є не повним алгоритмом, а лише окремим фактом чи правилом, що все ж вносять уточнення до вже відомих агенту алгоритмів, зміна їх функції корисності служить оцінкою новизни даного факту (правила) та їх важливості для агента. Детальніше такий підхід описаний у монографії [50].

Представлення знань у формі онтології передбачає, що будь-яке можливе узагальнення, тобто комплексне, складене поняття завжди явним чином артикульоване, назване і як окремий концепт фігурує в базі знань.

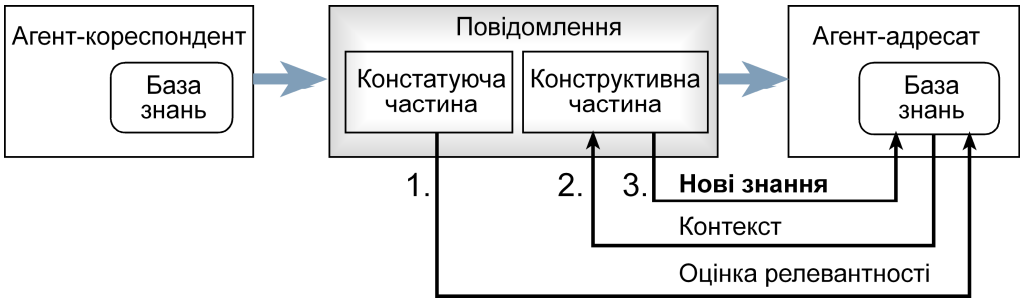

Рис. 3.2. Розпізнавання нових знань у повідомленні з метою наповнення ними бази знань

3.2 Метод видобування знань з тексту

3.2.1 Огляд досліджень у галузі видобування знань та навчання онтології

Світова наукова спільнота вже усвідомила революційне значення для забезпечення прискореного розвитку науки формування галузевих онтологій та засобів їх навчання. Відповідно значними є і капіталовкладення у розроблення онтологій. Американська National Human Genome Research Institute заснувала у 2001 році консоціум генної онтології (GO Consortium) [57]. У 2009 році фінансування консорціуму склало 4.4 мільйона американських доларів. У 2005 році Національний центр біомедичної онтології (National Center for Biomedical Ontology – NCBO) отримав фінансування загалом на п’ятирічний період розвитку в сумі 18.8 мільйонів американських доларів [58]. Зусилля по створенню онтології інфекційних хвороб отримали фінансову підтримку в сумі 1 мільйон 250 тисяч доларів на найближчі 4 роки [59]. Також у 2009 році американський фонд фінансування наукових досліджень National Science Foundation виділив понад 900 тис. доларів на два роки в онтологію

перепончатокрилих [60]. Американська Національна медична бібліотека починаючи з 2007 року платила приблизно 6 мільйонів щорічно за розбудову SNOMED-CT [61], після початкового капіталовкладення у 32,4 мільйона доларів у 2003 році [62].

Подібні системи сьогодні широко розробляються не лише в медицині і не лише у Сполучених Штатах [63] і незабаром стануть обов'язковим елементом онтолого-керованих інтелектуальних систем.

Наведемо перелік відомих розробок у порівнянні їх характерних рис [64] (див. табл. 3.1).

Таблиця 3.1. Розробки в галузі навчання і наповнення онтологій

Назва	Метод виділення	Аналіз	Генерування	Апробація	Розвиток
Генерування онтології з бізнес-моделі	Людиною	–	C – Без об'єднання. Пряме перетворення з використанням XSLT	Людиною, знизу вгору	–
XML2OWL	B – статична таблиця відповідностей	–	C – Без об'єднання. Пряме перетворення з використанням XSLT	Людиною, знизу вгору	–
UML2OWL	B	–	C – Без об'єднання. Пряме перетворення з використанням XSLT	Людиною, знизу вгору	–
Напівавтоматична побудова онтології зі структур DTD	C - автоматичне виділення з джерел у форматі DTD	B – аналіз структури	C – Нестандартне представлення онтології	Людиною	–
Learning OWL ontologies from free texts	C – текстові джерела, методи NLP. WordNet онтологія/словник	–	C – формат OWL	–	–
Побудова онтології для вибору інформації	C	–	C	–	–
TERMINAE	C – текстові джерела, методи NLP.	B – аналіз взаємозв'язків між поняттями	C – нестандартне представлення онтології	Людиною	–
SALT	D – текстові джерела, методи NLP. За кількома джерелами	C – аналіз подібності між поняттями	B – Нестандартне представлення онтології	B – обмежене втручання людини	–

Назва	Метод виділення	Аналіз	Генерування	Апробація	Розвиток
Новий метод злиття онтологій за концептами з використанням WordNet	–	B	C – автоматичне поєднання. Нестандартне представлення онтології	–	–
Розроблення ститеми автоматичної побудови онтології для заданої ПО	B – головні поняття визначаються експертом у цій ПО.	–	C	–	–
Наповнення надвеликих онтологій з використанням WWW	C – розширення існуючої онтології	–	C	–	–
Видобування знань заданої ПО та їх впорядкування (класифікація з використанням WordNet	C – головні поняття визначаються експертом у цій ПО	B – граматичний аналіз тексту	C	Людиною	–
A метод напівавтоматичного наповнення онтології з Intranet	C – методи NLP. Багатократне опрацювання джерел	B – аналіз значення понять	B	B – в конфліктних випадках потрібна участь користувача	B – циклічне застосування може забезпечити еволюцію
SymOntoX	–	C – аналіз співпадінь	B – Забезпечення деяких наперед заданих базових понять	Людиною	B – керування версіями, проте за участю людини
Protégé (з використанням відповідних плагінів-додатків)	B – виділення з реляційної бази даних та даних у XML фоматі	D – аналіз співпадіння і відповідності	B – Кероване зв'язування. Експорт у різні формати онтології	Людиною	C – розпізнавання еволюції онтології
LOGS	C – аналіз текстових джерел. Механізм NLP. Морфологічний і семантичний аналіз. Машинне навчання правил.	C – подібність базується на аналізі понять і зв'язків	C – Різні формати. Внутрішня структура онтології матрична	B – перевірка в кінці кожного модуля	–
Навчання онтології	D – виділення з різних форматів (XML, UML, OWL, RDF, text…). NLP, Семантичнй і лексичний аналіз. Багато-входовий аналіз джерел	C – бібліотеки для кластеризації, формального аналізу понять та асоціативних правил	C-OWL та RDF/S	B – за підтримки людини	–

Для лаконічності викладу в таблиці використано наступні умовні позначення:

- A означає, що рішення цієї проблеми в даному проекті ще немає;
- B – для напівавтоматичного методу, реалізованого у проекті;
- C – для реалізацій, в яких участь людини необов’язкова;
- D – для апріорі автоматичних методів, що не передбачають участі людини.

Застосовується кілька основних підходів до опрацювання тексту з цією метою – символьний, статистичний та змішаний. Серед найпоширеніших символьних підходів – застосування лексико-семантичних паттернів (lexico-semantic pattern – LSP) [65]. У такому підході опрацювання тексту виконується шляхом виявлення певних наперед відомих або встановлених шляхом машинного навчання реляційних маркерів, які існують у природній мові і дозволяють розпізнати семантичні ролі синтаксичних конструкцій, а у поєднанні з ідентифікацією онтологічних сутностей, які у даному тексті представляють ці синтаксичні конструкції, виконувати проекцію тексту на онтологію, отримуючи таким чином розпізнаний зміст, а за ним – оцінювати новизну, достовірність і корисність отриманих за цим змістом знань. Методи, що базуються лише на статистичних лінгвістичних моделях здатні лише поверхово розпізнавати дискурс, але не в стані виявляти зміст тексту, тобто відображену там логіку семантичного взаємозв’язку між поняттями даної проблемної області. Перелік методів наведено у табл. 3.2.

У 2010 році науковій громадськості був представлений проект NELL - система, здатна наповнювати БЗ з Інтернету: [106]. Її БЗ складається з масиву фактів у форматі "ключ-значення". Для застосування машини виводу в проекті використані горнівські правила. На момент представлення NELL містила (розпізнавала) 280 видів семантичних зв'язків типу “Plays for” («Грає за»), 390 тис. фактів типу “San Francisco is a city”, робила логічний вивід з існуючих фактів. За словами розробників система здатна вдосконалювати процес

самонавчання. Подібний проект TextRunner виконується у Вашингтонському університеті під керівництвом Oren Etzioni [107].

Таблиця 3.2. Задачі навчання онтології та відповідні методи

Мета	Базовий метод	Додаткові методи	Автори
Виявлення понять та синонімів	символічний	Compound noun information	Hamon [66]
		Symbolic Lexico-syntactic patterns (LSP)	Downey [67]
		LSP + compound noun information	Moldovan, Girju [66], Church [67], Smadia [68]
	статистичний	Clustering	Grefenstette [69], Hindel [70], Geffet, Dagan [71], Agirre [72], Faatz, Steimetz [73], Collier [74]
		Hidden Markov Model (HMM)	Bikel [75], Morgan [76], Shen [77]
		Support Vector Model (SVM)	Kazama [78], Yamamoto [79]
		Conditional Random Fields (CRFs)	Chanlekha [80]
Виявлення таксономічних зв’язків	символічний		Hearst [81], Caraballo [82]
		LSP	Cederberg, Widdow [83], Fiszman [84], Snow [85], Riloff [86]
		Compound noun information	Velardi [87], Cimiano [88], Rinaldi [89], Morin [90], Bodenreider [91], Ryu [92]
	статистичний	Clustering	Alfonseca, Manandler [93]
		Machine learning	Witschel [94], Berland [95]
Виявлення нетаксономічних зв’язків	символічний	LSP	Sundblad [96], Girju [97], Nenadicr , Ananiadou [98]
	статистичний	Co-occurring information	Kavalec [99], Gulla [100]
		Association rule mining	Chefi [101], Bodenreider [102]
Побудова онтології (комбінація всіх задач)	статистичний	Dependency triples	Lin [103]
		Nearest neighbor clustering	Blaschke, Valencia [104]

Цитата: "Інструменти NELL включають в себе програми, які витягують і класифікують текст з Інтернету, програми, які виявляють паттерни та на основі кореляцій, і програми, що формують правила". Система враховує вживання слів у різних контекстах, і виключає неоднозначність шляхом розгортання ієрархії відповідних правил. Перші шість місяців NELL накопичувала знання без участі

людини, але в результаті четвертина отриманих фактів мала дуже низьку достовірність.

3.2.2 Підхід до розпізнавання змісту природомовного текстового документу

Поняття знання відноситься до галузі наукових досліджень методів і засобів прийняття оптимальних рішень. В процесі набуття знань через навчання суб'єкт прийняття рішень використовує доступну йому інформацію для побудови оптимальної стратегії прийняття рішень. Інформація у нашому розумінні набуває статусу знань рівно тією мірою, якою вона допомагає носію цієї інформації вирішити його завдання і може бути чисельно оцінена як виграш від її використання при прийнятті рішень в процесі досягнення відповідних цілей. Належним чином організована і впорядкована сукупність знань інтелектуального агента носить назву бази знань. Система впорядкування знань у такій базі знань формально-логічно сформульована у її онтології [56], тобто явне формальне означення понять і допустимих семантичних зв'язків між ними.

Суть методу видобування знань з природомовного текстового документа, іншими словами – розпізнавання змісту текстового документа, полягає у побудові плану (стратегії) діяльності інтелектуального агента – інформаційної моделі суб'єкта розпізнавання, або уточнення такого плану на підставі даних, виділених у текстовому документі, що розпізнається. Тут вважаємо план конкретною реалізацією оптимальної стратегії вирішення деякої задачі, що стоїть перед інтелектуальним агентом в рамках заданої проблемної області.

План будується тою формальною мовою подання знань, якою було розроблено інформаційну модель − базу знань інтелектуального агента. Враховуючи, що така база знань вже становить собою певний загальний план функціонування інтелектуального агента, план, збудований на основі розпізнавання змісту природомовного тексту, є субпланом, тобто, уточненням

(виправленням) і/або деталізацією цього загального плану і базується на ньому. Цінність інформації, отриманої внаслідок розпізнавання змісту текстового документа, визначається за приростом очікуваної корисності від реалізації уточненого таким чином плану функціонування інтелектуального агента.

Переважна частина доступної, сформульованої у певній логічній послідовності і тому зручної для опрацювання інформації зберігається у текстових документах, зокрема, на електронних носіях. Достатньо велика частина таких документів доступна on-line, до того ж безоплатно. Серед них є можливість вибрати такі, що написані за достатньо жорстко встановленими правилами побудови і вимогами до змісту таким чином, що з одного боку вони залишаються природомовними текстами, а з другого максимально формалізовані для їх машинного опрацювання і виділення релевантної інформації, яка може інтерпретуватися інтелектуальною системою розпізнавання змісту як корисні знання.

До такого специфічного класу природомовних текстів можна віднести анотації наукових статей. Їх можна знайти через мережу Інтернет, вони, як правило, знаходяться у відкритому доступі, не містять графічного матеріалу, побудовані за строго встановленими правилами, написані окрім інших також англійською мовою, не містять модальних зворотів, а лише логічно зв’язану послідовність стверджувальних речень. Необхідний для заданої ПО корпус таких текстів можна вибрати за допомогою інформаційного пошуку за ключовими словами з використанням цілого ряду як спеціалізованих пошукових серверів наукових видавництв, так і пошукових серверів загального призначення.

Задачу вибору потрібного корпусу текстів було вирішено шляхом реалізації в рамках цієї роботи підсистеми інформаційного пошуку програмного пакету [7]. Програмний код підсистеми наведено у додатку А. На вході підсистеми – множина ключових слів, на виході – множина англомовних анотацій, розміщених в базі даних СУБД MySQL.

Процес видобування знань передбачає здатність як до розпізнавання окремих понять, згаданих у документі, так і до логічної інтерпретації сутності і характеру зв’язків між цими поняттями. Ці дані служать лише первинною інформацією для ієрархічної, багатоетапної процедури розпізнавання змісту природомовного текстового документу (ПТД). На відміну від традиційних статистичних методів опрацювання ПТД, у яких текст розглядається як множина окремих термінів (слів та словосполучень) без врахування семантичного взаємозв’язку як між термінами, так і між цілими твердженнями, вираженими закінченими реченнями, запропонована і розроблена у даній роботі процедура базується на розпізнаванні логічних тверджень і тому складається з трьох основних етапів: лінгвістичного, статистично-логічного та планувального. На першому, лінгвістичному етапі засобами морфологічно-синтаксичного аналізу мови, на якій даний текст написано, будується послідовність триплетів «суб’єкт зв’язку – семантичний зв’язок – об’єкт зв’язку», кожен елемент яких знаходиться або по ходу аналізу додається до онтології інтелектуального агента. На другому етапі методами машинного навчання на основі отриманої послідовності триплетів розпізнаються твердження у логіці предикатів першого порядку, їх семантичний зміст у термінах онтології інтелектуального агента та логічний взаємозв’язок між ними. На третьому, заключному етапі на базі прототипу плану або діючого загального плану функціонування інтелектуального агента з отриманої послідовності предикатів будується (доповнюється, коригується) ієрархічна система цілей (задач) і засобів їх досягнення (вирішення).

По суті маємо ієрархію розпізнавання: окремі слова, далі – словосполучення, далі зв'язки, далі – твердження, які вже являють собою базовий елемент, цеглини моделі світу інтелектуального агента. Далі можна говорити про розуміння агентом відмінностей між різними моделями світу: своєї і чужої, автора повідомлення, що аналізується даним агентом.

Загальна схема реалізації методу видобування знань з тексту включає наступні кроки:

1. Вибираємо прототип онтології як OWL-модель контексту ПО.
2. Перетворюємо аналізований текст на множину речень. Якщо джерелом тексту є анотація наукової публікації у друкованому виданні, першим реченням множини додаємо назву публікації. Останнім – назву друкованого видання.
3. В циклі розбираємо послідовно усі речення множини і будуємо з кожного з них окрему множину пар слів, з'єднаних метасемантичним зв'язком, яка служитиме вектором ознак для розпізнавання виду семантичного зв'язку.
4. Окремо з речення виділяємо групу іменника – суб'єкт розпізнаного на попередньому кроці семантичного зв'язку та групу іменника – об'єкт цього зв'язку.
5. До створеного на 1-му кроці шаблону онтології додаємо поняття, які вдається розпізнати в групах іменників, отриманих на попередньому кроці. Поняття додаються як екземпляри відповідних класів.
6. Якщо онтологія містить і об'єкт і суб'єкт зв'язку, тоді між ними встановлюється виявлений зв'язок. Одночасно до бази знань додається предикат, що відповідає даному зв'язку.
7. Для визначеної у п.2 множини речень та відповідної їй множини предикатів розпізнаємо логічні залежності між предикатами. Виявлені залежності вносимо до бази знань у формі SWRL-правил.
8. При внесенні нового правила перевіряємо базу правил на наявність суперечностей. Конфлікти вирішуємо з врахуванням достовірності джерел інформації за якими були внесені предикати, які конфліктують, а також логічної залежності з іншими предикатами бази знань.
9. Отриману систему понять і зв'язків, збудовану на їх основі систему предикатів та функцій, а також збудовану на їх основі систему аксіом і правил використовуємо для побудови плану інтелектуального агента.

10. Задаємо, уточнюємо або визначаємо за виявленими предикатами винагороди за досягнення проміжних цілей плану, імовірність їх досягнення при вчиненні допустимих дій, а також затрати на виконання цих дій. Розраховуємо оптимальний план, його очікувану корисність.
11. Процес навчання онтології полягає у послідовному (або паралельному) повторенні цієї процедури для всього корпусу навчальних текстів.

Отриманий план інтелектуального агента служить інформаційною моделлю публікації з точки зору цілей і задач її потенційного читача.

Детально складові елементи методу розглядаються далі.

3.2.3 Попереднє опрацювання природомовного тексту

Корпус текстів для заданої ПО складається з окремих текстових документів (далі – текстів), кожен з яких містить від одного до 10-20 речень. Ці речення знаходяться у послідовному логічному зв’язку, тому не можуть розглядатися відокремлено без втрати змісту. Саме тому текст розбивається на впорядковану множину речень, над якими надалі послідовно будуть виконуватись основні процедури розпізнавання. Складні речення розбиваються на прості засобами синтаксичного аналізу. В процесі розділення виконується підстановка займенників іменниками з першої частини речення, на які ці займенники посилаються. Підготовка речень виконується таким чином, щоб надалі однозначно ідентифікувати усі поняття, задіяні в сформульованих у реченнях твердженнях, а також відрізняти узагальнені поняття, які будуть розпізнані як класи, та конкретні поняття (зокрема, власні назви), що є унікальними, а тому будуть розпізнані як екземпляри відповідних класів. Кожне з таких декомпонованих і формалізованих речень слугуватиме матеріалом, вхідними даними для побудови на його основі концептуального графа [108, 109] і/або предиката, який відображає логічний взаємозв’язок між поняттями даної ПО. Таким чином, попереднє опрацювання тексту складається з таких кроків:

1. Розбиття тексту на окремі речення шляхом розпізнавання розділювачів «крапка», «кінець абзацу», «кінець тексту».

2. Розбиття складних речень на прості шляхом застосування синтаксичного аналізу з одночасною заміною займенників наступних речень відповідними їм іменниками з попередніх.

3. Переведення букв тексту в нижній регістр з маркуванням слів всередині речення, відсутніх у словнику загальних назв понять, як власні назви. Вилучення зайвих розділових знаків. Стеммінг.

4. Занесення отриманих текстів до масиву речень даного документа.

Результатом роботи процедури попереднього опрацювання тексту є масив послідовно логічно пов’язаних формалізованих та очищених речень, готових для опрацювання семантичним парсером з метою виділення ознак семантичних зв’язків у цих реченнях.

3.2.4 Виділення формальних ознак семантичних зв’язків між поняттями у реченні

Як слідує з загальної схеми реалізації методу видобування знань з тексту, поданої раніше у цьому розділі на наступному етапі опрацювання природомовний текст необхідно перетворити на послідовність предикатів логіки першого порядку. Ця задача може бути розкладена на кілька підзадач:

- розпізнавання іменникових груп;
- визначення їх ролі суб’єкта чи об’єкта семантичного зв’язку та відповідного цій ролі місця у предикаті;
- розпізнавання дієслівної групи кожного речення як типу семантичного зв’язку і відповідного цьому типу зв’язку предикату.

Розпізнавання іменникових груп виконується шляхом виділення базового іменника і послідовного уточнення (звуження) його значення внаслідок додавання до нього описових іменників та прикметників з виділеної іменникової групи. При цьому відповідність даного іменника до поняття в

онтології визначається за написанням цього слова після його стеммінгу (нормалізації). Послідовне уточнення значення іменникової групи здійснюється від останнього іменника (усталеного словосполучення іменникового типу) в групі до першого до тих пір, доки в онтології знаходиться відповідне такому поєднанню слів поняття. Усі наступні поєднання іменників і прикметників у іменниковій групі відображаються в онтології як підкласи понять, які були впізнані у іменниковому словосполученні на попередньому кроці – до додавання до цього словосполучення нового уточнюючого іменника або прикметника. Наприклад, якщо в онтології є поняття garden, але немає apple tree garden, хоча є або поняття apple tree, або лише окремі іменники tree та apple, система вносить до онтології уточнюючим підкласом класу garden одразу поняття apple tree garden, або лише tree garden відповідно у кожному з двох випадків. При цьому у другому випадку вже на наступному кроці до онтології буде внесено поняття apple tree garden як підклас класу tree garden.

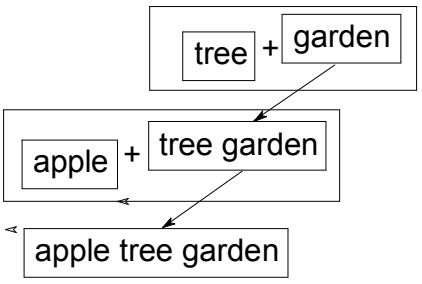

Рис. 3.3. Уточнення понять за складовими іменникової групи

Визначення ролі розпізнаного поняття у семантичному зв’язку у найпростішому випадку здійснюється на підставі місця, яке займає відповідна іменникова група відносно групи дієслова у реченні та у залежності від виду семантичного зв’язку, який представлений цією дієслівною групою. Якщо іменникова група стоїть перед дієслівною, основне дієслово якої закінчується на “-ed + by”, ця іменникова група представляє об’єкт семантичного зв’язку. В інших випадках – суб’єкт цього зв’язку.

Для формального подання природомовного речення у термінах онтології і описової (дескриптивної) логіки предикатів 1-го порядку необхідно визначити тип предиката. Розпізнавання типу можна виконати за дієслівною групою цього речення та службовими словами, які до дієслівної групи можуть не входити. Для цього необхідно застосувати метод машинного навчання системи розпізнавання, вхідними даними для якої служитимуть результати розбору природомовного речення спеціальним синтаксично-семантичним парсером. Такий парсер розбиває речення на пари слів, пов’язані деяким метасемантичним зв’язком. В результаті кожне речення парсер представляє множиною триплетів, що складаються з суб’єкта такого зв’язку, об’єкта зв’язку і самого метасемантичного зв’язку певного виду. Ці триплети можуть бути використані як ознаки присутності в реченні того чи іншого семантичного зв’язку, на основі якого має бути збудований предикат як логічне формальне представлення цього речення.

У роботі застосовано Link Grammar Parser (далі – LGP) [110]. Цей програмний засіб є ‘open source’-продуктом, має відкриту ліцензію типу GPL, добре документований, а тому доцільність його застосування для цієї задачі не викликає сумнівів. Приклад вікна з довідковою інформацією та ілюстрацією результатів розбору простого речення, при роботі з програмою з командного рядка наведено на рис. 3.4:

Передумовою виявлення семантичного зв'язку з застосуванням LGP є наяність (і виявленя) дієслівної групи. Її присутність визначається відповідними дієслівній групі великими і малими буквами з достатньо великого переліку. Наприклад, усі зв'язки, розташовані вправо від зв'язку 'S*' вказують на дієслівну групу.

Слова, що супроводжують (на які вказують) ці букви-символи зв'язку визначають вид семантичного зв'язку (ім'я предиката). Ці слова є, як правило, дієсловами: "знає", "має", "належить", "відноситься", або дієслівними словосполученнями " належить до", "складається з" і т.п. Прикметники

інтерпретуються як властивості і також можуть бути розпізнані у реченнях через семантичний зв'язок "має властивість".

Щоб розпізнати семантичний зв'язок, необхідно виконати наступні дії:

3.7. COMMANDS AND VARIABLES. It is possible to modify the running of the parser in various ways, while running it, by typing in certain commands. The basic commands can be seen by typing "!help". Others are listed under "!variables". Many of these are self-explanatory. For example, "!width" changes the width of the parser display. Other commands relate to speed and robstness features; see section 7.

A few commands deserve special mention. One useful command is "!![word]". This queries the parser for information about a particular word. The parser will output list any entries of the word, with their word subscripts, the word-files in which they appear, if any, and the number of disjuncts on each word. (A disjunct is a combination of connectors which constitutes a legal use of the word.) Multiple entries of a word will be listed with their word subscripts.

The "!verbosity" command controls the amount of information that is displayed. With "!verbosity=1" (the default), information such as the following is shown:

```
linkparser> the quick brown fox jumped over the lazy dog
++++Time                                                0.04 seconds
(0.04 total)
Found 2 linkages (2 had no P.P. violations)
  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=18)


 +---------Ds---------+              +-------Js------+
 |     +-------A------+              |   +-----Ds----+
 |     |       +---A--+---Ss--+--MVp-+   |     +--A--+
 |     |       |      |       |      |   |     |     |
the quick.a brown.a fox.n jumped.v over the lazy.a dog.n

Press RETURN for the next linkage.
linkparser>
```

With "verbosity=0", no information is shown except for the graphic linkage display. With verbosity set at 2 or 3, information is shown about the individual stages of parsing the sentence. (Infomation is also shown about the constituent derivation process, if this is being done.) If one wants to suppress the graphic display as well, this can be done with the command "!graphics". (This can be useful it one wants to have only the constituent bracketing as output; in that case, type "!verbosity=0", "!graphics", and "!constituents=1 (or 2)".

Рис. 3.4. Довідка Link Grammar Parser з прикладом результатів розбору простого речення

- розібрати речення за допомогою LGP;

- знайти дієслівну групу через символи зв'язку справа від "Ss";
- знайти дієслово, на які вказують ці символи зв'язку;
- знайти суб'єкт дії (підмет у реченні), на який вказує символ "Ss";
- знайти об'єкт дії (очевидно, означення у реченні), тобто, предмет, на який спрямована дія;
- перевірити в онтології наявність цього виду семантичного зв'язку і у разі відсутності, створити його;
- перевірити наявність в онтології сутностей, що означають об'єкт та суб'єкт дії; тут можливі різні варіанти (див. табл. 3.3).

У випадку (4), очевидно, речення ігнорується. У випадку (3) також нічого пов'язувати: так, наприклад, якщо використовується відомий онтології зв'язок IS-A, який пов'язує два невідомі онтології терміни: X та Y, таке твердження також доведеться ігнорувати. У випадках (2) і (5) з'являється можливість внести поняття до онтології через відомий зв'язок, наприклад, у реченні "Іван курить X" X вноситься до онтології як сутність, яку можна курити. При цьому слід розрізняти онтологію, як множину допустимих семантичних зв'язків між поняттями і базу знань, як множину фактів про дану модель дійсності. Онтологія описує зв'язки між класами, а база знань – зв'язки між екземплярами цих класів.

Таблиця 3.3. Можливі дії системи

№	Є зв'язок	Є суб'єкт	Є об'єкт	Можлива дія
	+	+	+	Занести до бази знань
	+	+	–	Додати невідоме поняття до онтології
	+	–	–	Ігнорувати
	–	–	–	Ігнорувати
	+	–	+	Додати невідоме поняття до онтології

Зв'язки можуть бути безумовними та умовними. Умовні зв'язки записуються як правила. Безумовні зв'язки є частковим випадком умовних і записуються, як факти у вигляді предикатів.

Таким чином, для навчання системи навикам розпізнавання нових типів семантичних зв'язків у реченнях потрібний модуль індуктивного навчання за семантичними ознаками. Речення-приклад дає послідовність семантичних зв'язків між словами. Кілька таких однотипних речень підряд з вказанням назви зв'язку дає системі можливість виявити підмножину спільних ознак і створити ознакову функцію:

```
{Vj}=>Linkx ,
де Vj – j-та ознака у вигляді:
organ->S->is
is->O->part
a->D->part
part->M->of
of->J->organism
an->D->organism
```

Вхідними даними для модуля індуктивного навчання служать змінні – необмежена множина слів і константи – обмежену множину символів граматичних зв'язків {S, D, O, J, M, ...}. Маємо також результат роботи LGP - пари слів, поєднані метасемантичними зв'язками у певній послідовності, маємо множину дієслів, кожне з якої може стати початком координат в реченні в разі виявлення.

Суть алгоритму виглядає таким чином: шукаємо перше відоме дієслово (LGP дає відповідну розмітку) і найближчий іменник, що передує цьому дієслову. За англійською граматикою у переважній більшості випадків (виключення згадані вище у цьому параграфі) цей іменник (іменникова група) і є підметом – суб'єктом дії. Якщо це займенник, шукаємо у попередньому реченні підмет за аналогічною процедурою і підставляємо у реченні, що

аналізується на місце займенника. Дія і є семантичним зв'язком (видом предиката), проте таким зв'язком може виступати не саме дієслово а ціла дієслівна група в реченні, тому її належить виявити і зафіксувати в онтології одним терміном на зразок "is-a" чи "is-a-part-of".

Після такого навчання сигнатура (формула) семантичного зв'язку (див. рис. 3.5) вноситься до онтології для подальшого використання та можливого уточнення. Для цього в лінгвістичній частині онтології зберігаються екземпляри класів 1) метасемантичних зв’язків, 2) об’єктів та 3) суб’єктів цих зв’язків, які додаються в міру машинного навчання онтології кожному з семантичних зв’язків як деякі паттерни і враховуються при розрахунку імовірності виявлення котрогось із зв’язків у тексті, що аналізується.

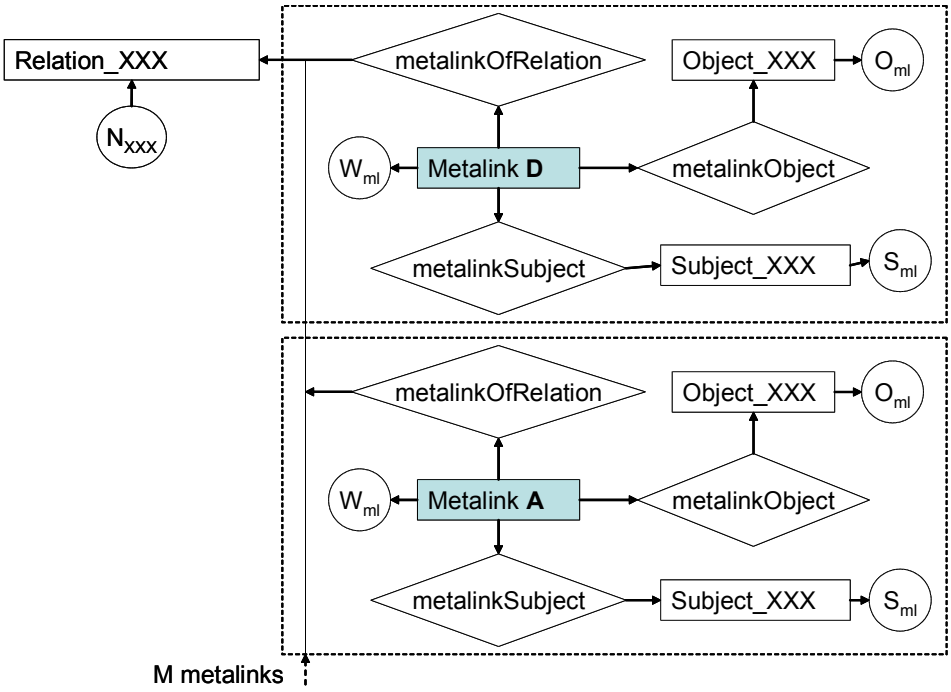

Рис. 3.5. Схема понять, зв'язків і властивостей для занесення до бази знань паттернів видів зв'язків

Виявлення семантичних зв’язків в лінгвістичній підсистемі побудовано на застосуванні Баєсівського розпізнавання множини ознак збережених в онтології патернів відомих семантичних зв’язків. Вивчення *d* ознак *j*-го семантичного зв’язку:

$$p(X \mid C_j) \propto \prod_{k=1}^{d} p\left(X_k \mid C_j\right). \quad (3.1)$$

Розпізнавання *j*-го семантичного зв'язку за *d* виявленими ознаками:

$$p(C_j \mid X) \propto p(C_j) \prod_{k=1}^{d} p\left(X_k \mid C_j\right). \quad (3.2)$$

В якості ознак (дескрипторів) використано результат розбору речення природомовного тексту на пари слів, пов'язаних синтаксично-метасемантичними зв'язками за допомогою LGP. Для простого тестового речення:

```
[(a)(test.n)(is.v)(an)(example.n)]
```

результат розбору:

```
[[0 1 0 (Ds)][1 2 0 (Ss)][2 4 0 (Ost)][3 4 0 (Ds)]]
```

результат розпізнавання типу семантичного зв'язку за (2):

```
1) cause: 1.0882684165532656E-4;
2) caused-by: 0.013810506200916856;
3) is-a: 0.024124901979118252;
4) is-about: 0.0;
5) part-of: 0.0022765542079946285;
6) same-as: 0.0;
7) similar-to: 1.0261341731138478E-6;
```

Таким чином тестування розроблених програмних засобів, що реалізують описаний вище алгоритм, підтверджує коректність його роботи [6].

3.2.5 Оцінка довіри до джерела інформації наповнення онтології

Метод розбудови онтології має сенс лише у складі деякої інтелектуальної системи. Оптимальним на нашу думку є рішення, у якому такою інтелектуальною системою є система інформаційного пошуку, для якої адаптивна онтологія з одного боку є інструментом для інформаційного пошуку, аналізу і класифікації, а з другого – сама використовує засоби пошуку для

постачання нових даних для свого наповнення, синтезу нових предикатів і правил, навчання нових понять та семантичних зв'язків між ними. Таким рішенням стала інтелектуальна система інформаційного пошуку на базі адаптивної онтології, бази знань у галузі матеріалознавства та бази даних наукових публікацій у цій галузі. Крім того, ми не лише розбудовуємо онтологію, а й вилучаємо з неї елементи, які, на нашу думку, стали не релевантними з точки зору задач, які розв'язує ІА.

Нехай в деякий момент часу *t* вага концепту $\tilde{C}_k$ онтології *O* рівна W_k^t. Для зростання ваг концептів використовуються електронні природномовні документи *T* (статті, тези конференцій, анотації статті, якщо сама стаття недоступна, монографії тощо). Кожний такий документ *T* належить до певного джерела інформації *U* (науковий журнал, сайт тощо). Тобто існує множина джерел $U = \{U_1, U_2, ..., U_K\}$, а кожне таке джерело містить множину текстових документів $U_i = \{T_{i1}, T_{i2}, ..., T_{ik_i}\}$. Міру довіри до джерела U_i позначатимемо $\sigma_i \in [0,1]$. $\sigma_i = 0$ – повна недовіра до джерела U_i, $\sigma_i = 1$ – максимальна довіра до джерела U_i. Насамперед вважаємо, що міра довіра усіх джерел рівна 0,5. Під час наповнення онтології концептами із текстових документів міра довіри до джерела змінюється. Запропоновано такий метод зміни міри довіри: нова міра довіри σ_H до джерела рівна $\sigma_H = 2 \cdot \sigma_C - \sigma_C^2$, якщо експерт включив текстовий документ із цього джерела для наповнення онтології, де σ_C – стара міра довіри і $\sigma_H = \sigma_C - \sigma_C^2$, якщо експерт не включив жодного текстового документу із цього джерела для наповнення онтології. Якщо $\sigma_H < 1$, то таке джерело інформації далі не розглядається. Тобто, якщо з певного джерела 6 раз підряд не взято жодного документу для наповнення онтології, то його $\sigma_H \approx 0,099$ й таке джерело виключається з подальших переглядів.

Тоді ваги концептів $\tilde{C}_k$ змінюються за такою формулою:

$$W_k^{t+1} = W_k^t + \sum_{T_{ij} \otimes U_i} \sigma_i ,$$

де запис $T_{ij} \otimes U_i$ означає, що текст T_{ij} був використаний для наповнення онтології.

Із онтології виключаються ті концепти для яких $W_k^{t+1} - W_k^t < \Delta^t$. Такий метод дає змогу зменшити простір пошуку *Path*.

Отримаємо таку діаграму діяльності системи автоматизованої розбудови онтології (САРО) (див. рис. 3.6).

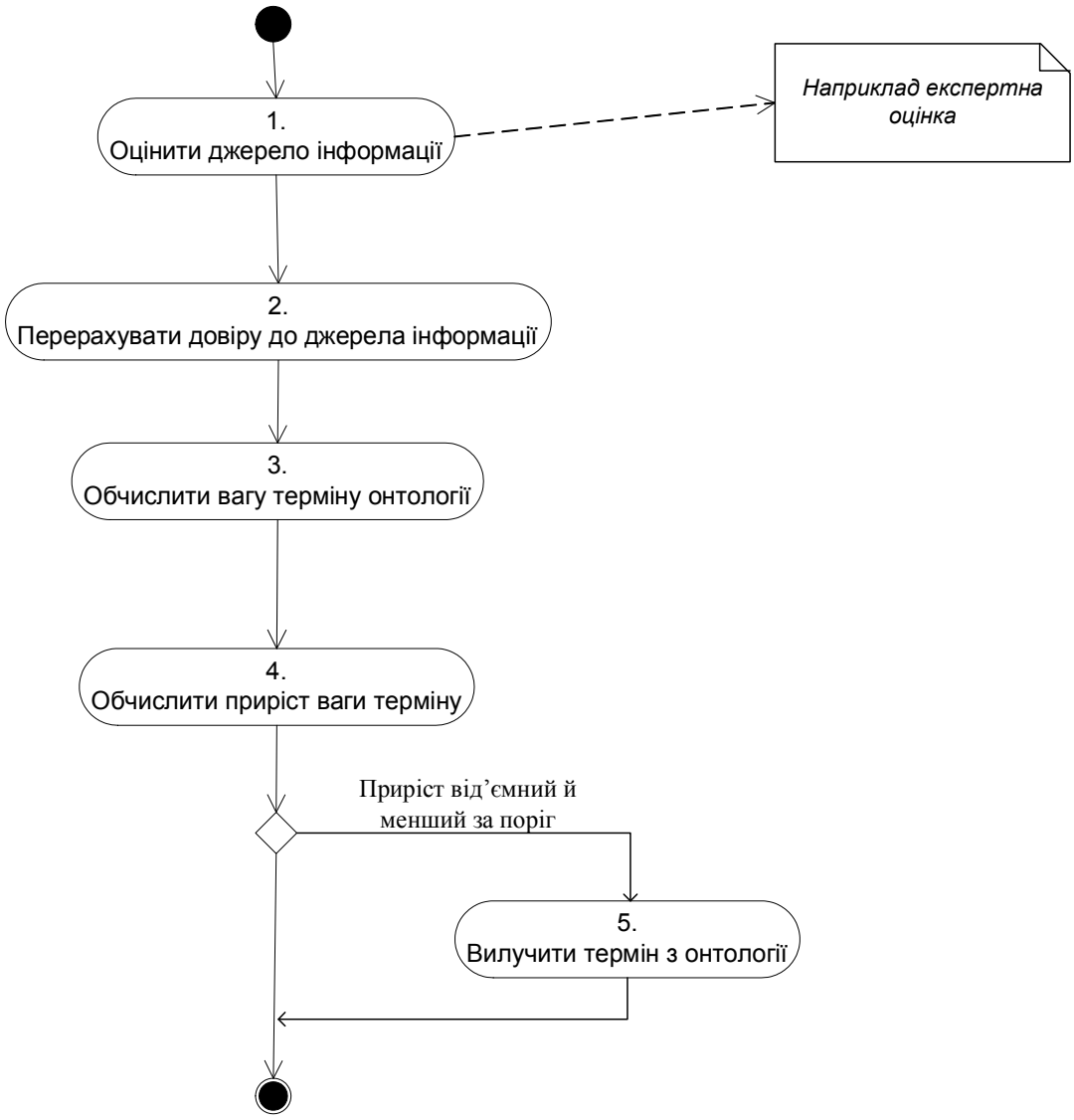

Рис. 3.6. Діаграма діяльності САРО

3.3 Програмно-технічні рішення побудови інтелектуальних агентів

Аналіз стану досліджень та розробок в галузі інтелектуальних інформаційних систем та Інтернет-послуг [111] дає підстави вважати, що виправданими є наступні програмно-технічні рішення:

- реалізація системи синтезу онтології як підсистеми портальної служби Інтернет-пошуку;
- застосування **OWL** як мови подання знань в онтології;
- застосування **HTN** та **OWL-S** як структури та мови автоматичного планування бази знань;
- **Java API для Protege-OWL** – як програмного середовища та бібліотеки класів опрацювання, зокрема, машинного навчання (навчання з підкріпленням) OWL-онтології та БЗ;
- **Link Grammar Parcer** – як засобу граматично-семантичного аналізу англомовних текстових документів в електронному форматі;
- **Apache-PHP-MySQL** – як програмних засобів для побудови інтерфейсу з користувачем за архітектурою веб-порталу;
- **Wget** – як веб-служби для автоматизованого доступу до пошукових серверів за запитом, сформованим з ключових слів;
- **SWRL** – як мову правил логічного виводу нових знань дедуктивним та індуктивним методами.

Онтологія на мові OWL містить понятійний апарат верхнього рівня та предметної області матеріалознавства, описаної раніше. Онтологія верхнього рівня забезпечує:

- логічне виведення нових знань, доповнення отриманих повідомлень контекстом;
- верифікацію істинності отриманих тверджень;
- оцінку вірогідності джерел повідомлень;

- забезпечення логічної цілісності БЗ.

Машинне навчання реалізується засобами Java API Protege-OWL. Ці засоби містять бібліотеки класів, в яких реалізовано методи роботи з OWL-структурами: їх читання, доповнення. Таким чином, засоби машинного навчання (ЗМН) функціонують у взаємодії з OWL-онтологією, беручи з неї шаблони граматично-семантичних структур для розпізнавання тверджень (предикатів логіки 1-го порядку) у досліджуваних і/або навчальних текстах та додаючи до неї нові елементи в результаті такого розпізнавання. Для цього застосовується Link Grammar Parser, який розбиває стверджувальне речення, написане граматично правильною англійською мовою, на семантично-пов'язані між собою пари слів (понять). LGP містить у своєму складі таблицю відповідності між граматичними конструкціями англійської мови та типами синтаксично-семантичних зв'язків між словами (поняттями). API LGP дозволяє пов'язати цю таблицю з OWL-онтологією, завдяки чому таблиця може динамічно адаптуватися в процесі навчання до заданої ПО.

ЗМН на базі Java-бібліотек Protege-OWL містить узагальнений опис семантичного зв'язку, який служить шаблоном для генерування в процесі навчання нових типів семантичних зв'язків та формування для їх ідентифікації в тексті відповідних векторів ознак цих зв'язків. При цьому до ОВР додаються відповідні класи зв'язків та їх властивості. Екземпляри цих класів служать для опису існуючих та нових класів онтології шляхом їх використання як предикатів логіки 1-го порядку.

Як було обгрунтовано у попередньому розділі, ЗМН онтології мають сенс лише у складі деякої інтелектуальної системи. Оптимальним на нашу думку є рішення, у якому такою інтелектуальною системою є система інформаційного пошуку, для якої адаптивна онтологія з одного боку є інструментом для інформаційного пошуку, аналізу і класифікації, а з другого – сама використовує засоби пошуку для постачання нових даних для свого наповнення, синтезу нових предикатів і правил, навчання нових понять та семантичних зв’язків між

ними. Таким рішенням стала інтелектуальна система інформаційного пошуку на базі адаптивної онтології, бази знань у галузі матеріалознавства та бази даних наукових публікацій у цій галузі.

Розроблена архітектура системи синтезу онтології матеріалознавства була реалізована з використанням вибраних і описаних раніше засобів і програмно-технічних рішень як модуль програмного забезпечення САРО.

3.3.1 Основні модулі системи САРО

Загальна концептуальна схема системи САРО представлена на рис. 3.7. Підсистема навчання онтології використовує навчальні тексти анотацій наукових публікацій бази даних статей. Для наповнення бази даних система формує множину ключових слів, за якою вибирає з зовнішнього джерела публікацій в мережі Інтернет (ScienceDirect, CiteSeer, Wiley Online Library, Springer) основні метадані про публікації в заданій ПО, зокрема їх анотації, які стають основою для аналізу та навчання онтології. За результатами такого аналізу анотацій наукові публікації ранжуються за їх релевантністю до інформаційних потреб користувача, тобто за відповідністю до онтології, яка ці потреби відображає. З цією метою здійснюється аналіз кожної анотації як природомовного тексту, будується її образ в термінах онтології у вигляді множини предикатів та правил, ці предикати та правила додаються до бази знань системи і повторно обчислюється очікувана корисність оптимального плану інтелектуального агента. При такому ранжуванні ближче до початку списку система розташовує ті публікації, внесення даних яких призводить до більшої зміни корисності.

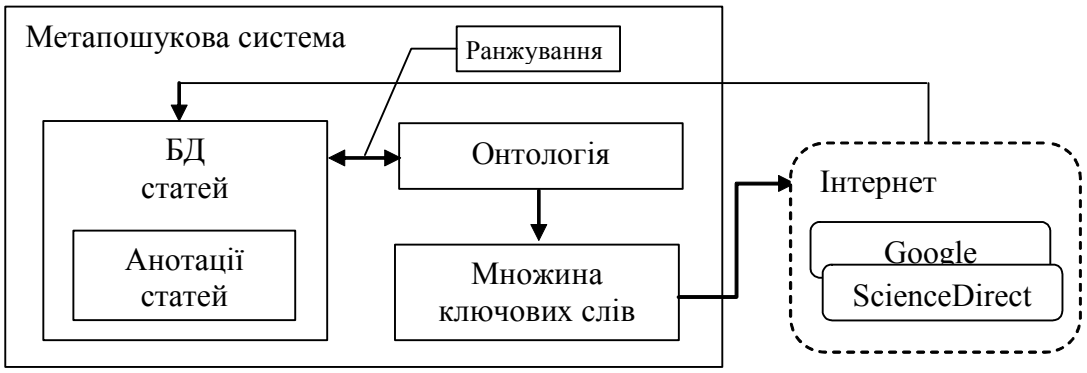

Рис. 3.7. Концептуальна схема системи CAPO

Архітектура системи CAPO наведена на рис. 3.8. Через користувацький інтерфейс клієнт має можливість керувати пріоритетами у ранжуванні документів, тобто коригувати порядок їх розташування у переліку найбільш важливих (релевантних до інформаційних потреб клієнта) документів і/або класифікувати їх. При цьому найважливіші документи використовуються для навчання онтології та побудови ефективних наборів ключових слів, а нові, отримані з Інтернету статті (їх метадані, включно з анотацією) заносяться до бази даних публікацій у взаємозв’язку з преференціями користувача та іншими передумовами отримання документа, передусім – джерела його отримання.

Опрацювання анотації відбувається після попередньої обробки, зміст якої описано раніше, перетворення на множину предикатів за результатами граматично-синтаксичного розбору модулем Link Grammar Parser та доповнення сформованої таким чином моделі анотації семантично близькими предикатами з онтології – контексту цієї анотації. Доповнені моделі порівнюються між собою для обчислення семантичної відстані між їх центрами семантичної ваги і таким чином обираються найбільш близькі за змістом документи з їх подальшим ранжуванням та класифікацією.

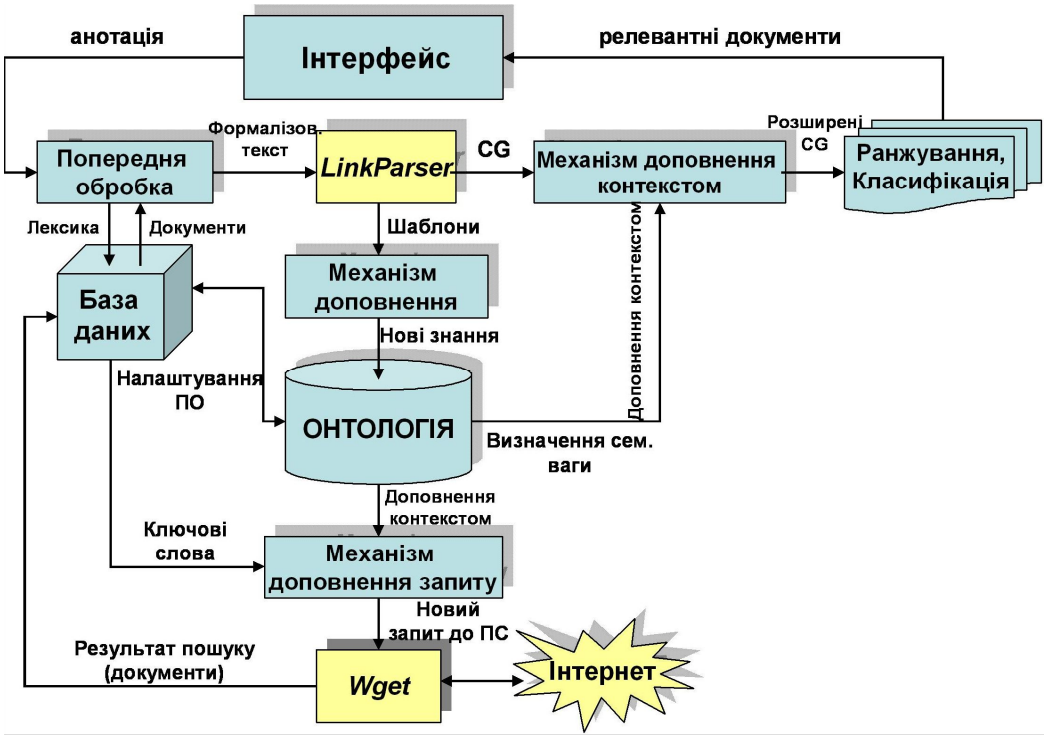

Рис. 3.8. Архітектура системи CAPO

Система CAPO має дві основні функції:

1) інтерактивна автоматизована побудова онтології заданої проблемної області;

2) пошук, збереження і класифікація (ранжування) наукових публікацій як у інтерактивному напівавтоматичному, так і в автоматичному режимі.

Кожна з цих функцій реалізована своїм базовим набором функціональних модулів, але частина з цих програмних модулів має подвійне призначення. Система CAPO реалізована мовою програмування Java за об'єктно-орієнтованою парадигмою як ієрархія класів програмного коду, екземпляри яких викликають одне одного з визначеними на момент виклику параметрами, і/або взаємодіють через події та їх обробники. Більшість модулів системи мають графічний інтерфейс Swing та AWT бібліотек. Всі підключені бібліотеки мають статус відкритих і безкоштовно розповсюджуються. Завдяки їх застосуванню проект є повнофункціональним і має всі необхідні засоби для послідовного розвитку.

3.3.2 Функціональне призначення модулів системи CAPO

Функціональне призначення основних модулів системи наведено у таблиці табл. 3.4.

Таблиця 3.4. Функціональне призначення основних модулів системи CAPO

Модуль	**Призначення**
Attribute.java	Підпрограма виявлення атрибутивних (прикметникових) зв'язків в реченнях. Точніше, дієслівних зв'язків. Кожний прикметниковий зв'язок насправді є формою дієслівного: «Помідор зелений» ≡ скорочення від «помідор **має колір** 'зелений'». При цьому онтологія має містити бінарний предикат <**має колір**>(<домен, тт. область визначення = «помідор»>, <діапазон допустимих значень = «зелений колір»>)
BinaryLink.java	Визначення ваги горизонтального бінарного зв'язку
CGrammarObject.java	Розбирає речення на слова-токени, які після крапки супроводжує буква, що означає тип слова як частина мови (n-noun, v-verb etc.), таким чином, що за допомогою методів **getName**() чи **getType**() можна отримати слово та його відповідний частині мови тип, визначений LinkGrammarParser'ом
CGrammarObjectArr.java	Конструктор екземплярів цього класу формує масив граматичних об'єктів з текстового рядка 'sentence'
CGrammarObjectType.java	Клас типів елементів масиву CGrammarObjectArr
CLinkType.java	Клас методів розбирання стрічки типу зв'язку між парами слів у реченні на основний вид зв'язку **getType**() та його підвиди **getSpecification**()
ControlGUI.java	Головне графічне вікно управління програмою
CSOLink.java	Розбирає результати роботи Link Parser'a – визначає відповідність вказаних парсером типів зв'язків словам, розміченим дужками у попередньому рядку.
CSOLinksArr.java	Складає екземпляри типу CSOLink у динамічний масив
Descriptor.java	Формує множину паттернів в один масив

Модуль	Призначення
DParsedData.java	Формуються текстові рядки з результатами роботи LGP
FunctionGUI.java	Шаблон графічного інтерфейсу
GSL.java	Формує паттерн виду семантичного зв'язку для розпізнавання семантичні зв’язки за їх паттернами
Is_a.java	Процедура додавання підкласів до існуючих класів
MainProc.java	Основна процедура[1] у «нерафічному варіанті» - Більше не підтримується 02.05.2011 12:38. 1) Створюються константні параметри для запуску зовнішньої для цього пакету програми Link Grammar Parses 2) Відкриваються три потоки – для входу Parser'a, виводу його даних та його помилок. 3) вказується URL-адреса онтології, розташованої десь on-line; 4) з рядка запуску усієї програми зчитується ім'я файла тексту, з якого буде доповнюватися онтологія (1-й параметр в рядку); 5) зчитується онтологія з заданої раніше адреси; 6) виводиться її вміст у стандартний канал виводу; 7) за допомогою процедури DumpOWLModel виводиться вміст онтології у файл resulting.owl; 8) виконується пошук Is-a зв'язків; 9) за допомогою процедури DumpOWLModel виводиться вміст онтології у файл resulting.owl; 10) виконується пошук Consist-of зв'язків; 11) виконується процедура Attribute.
MetaObject.java	Створює структуру для опису семантичного об'єкта – суб'єкта дії, об'єкта дії чи дії як виду зв'язку між суб'єктом і об'єктом
OntologyMapClass.java	Додає класи на певний рівень онтології **(addClassesToLevel)**
OWLEvaluation.java	Найчастіше застосовувана процедура цього класу (його об'єктів) Розрахунок ваги понять та зв’язків онтології DumpOWLModel(String file_name) виконує вивід онотології у <файл.owl> ShowAll – виконує процедуру візуалізації онтології в стандарному каналі виводу

Модуль	Призначення
Parser.java	Запуск Link Grammar Parser на виконання, налаштування його режимів, подання на вхід текстового файлу з реченнями, збереження у заданому файлі результатів роботи парсера.
Part_of.java	Розпізнає семантичні зв’язки типу Part-of у англомовних реченнях (після їх обробки Link Grammar Parser).
Pattern.java	Конструктор класу Pattern, у якому створюється екземпляр паттерну семантичного зв'язку **для його розпізнавання в реченнях статистичним методом**. Для цього збираються DIM найважливіших елемента триплета "суб'єкт -> метазв'язок -> об'єкт", а саме "суб'єкт -> метазв'язок" та "метазв'язок -> об'єкт", для яких вказано, що вони виникли під час навчання цьому семантичному зв'язку, та відіграють в ньому найважливіше значення (бо найчастіше зустрічаються). В онтології статистика зберігається у вигляді екземплярів даного класу семантичних зв'язків у форматі: default_<ім'я-сем.-зв'язку>_<metalink:{D,S,O,...}>_<Subject|Object>_<ім'я-суб'єкта/об'єкта>.
SemLinkDescriptArr.java	З досліджуваного речення вибирає послідовність дескрипторів типу SemLinkDescriptor і створює з них динамічний масив
SemLinkDescriptor.java	Дескриптор семантичного зв’язку у вигляді триплета: `metaSubject (string) -> link-type (string) -> metaObject (string)` (З таких триплетів має складатися вектор ознак семантичного зв’язку. Кожен триплет для кожного типу семантичного зв’язку має свій ваговий коефіцієнт. Розмірність такого вектора вибрана рівною 10.
ViewResults.java	Виведення результатів операції в окреме вікно

Базовим елементом управління системою CAPO є модуль ControlGUI (графічний користувацький інтерфейс управління). Цей модуль має графічний інтерфейс, через який користувач може виконувати процедури, передбачені функціональністю системи. Для цього в модулі передбачене основне меню, а найважливіші його функції виведені на панель інструментів. Контрольний вивід здійснюється на відповідну текстову панель, а для вказання типу семантичного зв’язку в процесі навчання онтології на навчальних реченнях в

нижній частині головного вікна передбачене відповідне текстове поле вводу (див. рис. 3.9).

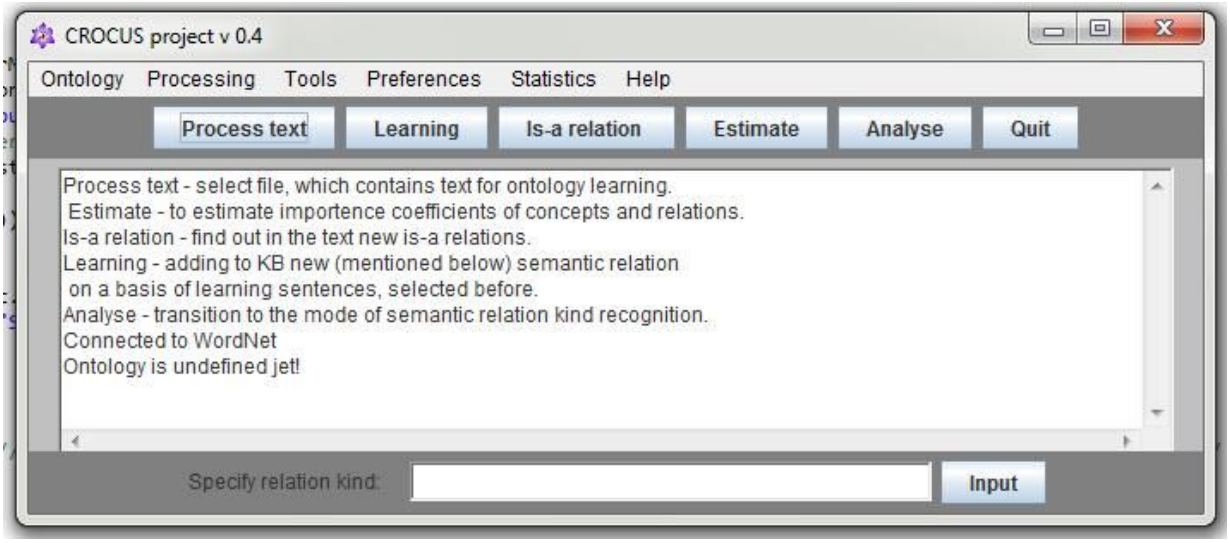

Рис. 3.9. Головне вікно користувацького інтерфейсу CAPO

Зважаючи на важливість ефективного діалогу система-користувач, як діалогу виду людина-комп'ютер, значну увагу при розробці було приділено графічному дизайну інтерфейсу, його інтуїтивності і лаконічності у поєднанні з функціональною повнотою реалізації завдань проекту, а також його придатності до масштабованості та можливості розвитку функціональності. Досвід подібних зарубіжних проектів підтверджує успішність такого підходу (Наприклад, проект Protege з опрацювання OWL-онтологій, проект GATE та інші).

У системі передбачена інтернаціоналізація усіх текстових діалогів. Користувач може вибрати зручну йому мову інтерфейсу з чотирьох можливих на даний момент. Доповнення переліку мов іншою мовою не становить проблеми і полягає у перекладі на цю мову файла діалогів MessagesBundle_xx_XX.properties, де XX – код мови (RU – для російської, uk-UA – для української і т.п.).

Для вибору мови діалогу необхідно в пункті головного меню 'Preferences' вибрати підпункт 'language'.

3.3.3 Обгрунтування вибору програмних засобів

Використання стандартних бібліотек програмних засобів дозволяє уникнути невиправданих перевитрат часових, фінансових та людських ресурсів на їх повторне розроблення. Тому в роботі було досліджено широкий перелік діючих аналогічних до розробленого проектів, переважна більшість яких спираються на концепцію відкритого програмного коду та розповсюдження програмних продуктів на умовах безоплатного ліцензування. Провідні групи розробників забезпечують свої проекти засобами API (Application Programming Interface), завдяки яким функціональність цих проектів може бути ефективно використана простим застосуванням каталогізованих і добре документованих процедур і функцій з відповідними параметрами.

Співтовариство розробників програмних засобів напрацювало засади використання пакетів прикладного програмного забезпечення під різними ліцензійними умовами, а також участі у підтримці і розвитку діючих проектів, завдяки чому кожен розробник має можливість отримати, встановити для особистого використання та розвивати на власний розсуд і в міру можливостей ці проекти, або бібліотеки програмних засобів, які входять до їх складу. Такі Інтернет-портали як SourceForge.net містять в собі всю необхідну палітру інструментальних засобів для розміщення там, документування і супроводу проектів довільного ступеня складності, стадії розроблення, рівня доступу і популярності серед користувачів. Розробники активно використовують також спеціалізовані сервери контролю версій розробки, які забезпечують колективне (хай і тисячі учасників) розроблення програмних засобів. Найпопулярнішим серед таких є сервер Git. Він може бути встановлений окремо як індивідуальний чи корпоративний сервер, а може використовуватись загальнодоступний глобальний Git-сервер GitHub.

Серед мов програмування, ак показали дослідження, левова частка розробок в галузі опрацювання природомовних текстових документів, майже всі розробки в галузі побудови та навчання онтологій припадають на мову Java.

Крім того, Java зберігає домінування серед мов проектів, розміщених на ресурсі SourceForge (див. рис. 3.10).

Вирішальним аргументом на користь вибору Java як мови програмування проекту стала наявність і доступність Java API у проекту Стенфордського університету (США) Protege-OWL, адже саме Стенфордський Центр досліджень з біомедичної інформатики (Stanford Center for Biomedical Informatics Research), став флагманом практичних розробок у галузі засобів розроблення, редагування та навчання баз знань та онтологій мовою подання знань OWL.

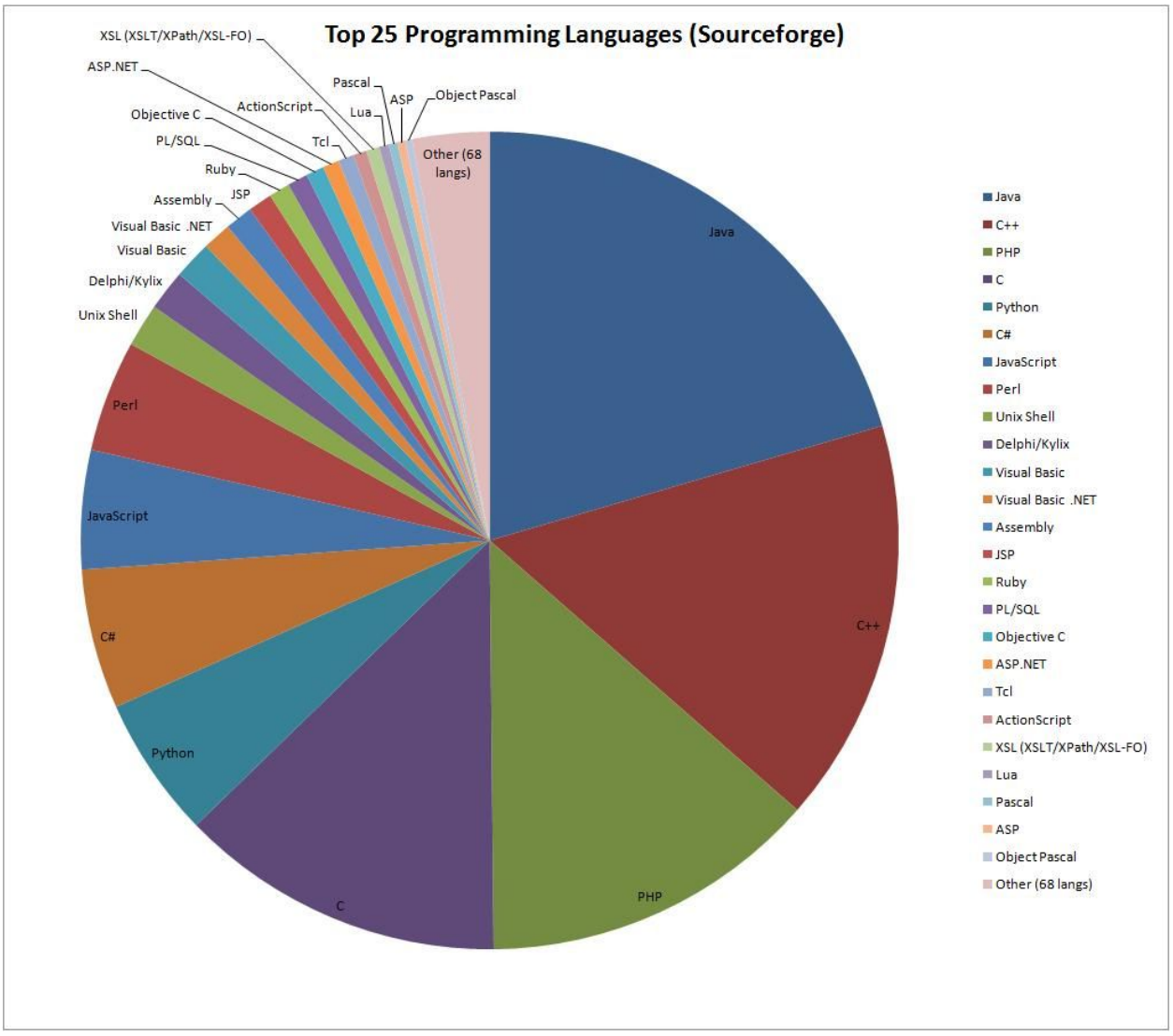

Рис. 3.10. Частки 25 найпопулярніших мов програмування серед розробників, що надають доступ до своїх проектів через портал SourceForge.net

Також мовою Java розроблено проекти:

- Gate [http://gate.ac.uk/] – множина засобів опрацювання текстових документів з метою виявлення нових знань.
- owlapi.sourceforge.net - ще один Java-проект, який являє собою бібліотеку Java-класів з широкою функціональністю з опрацювання OWL-документів.
- Pellet [http://clarkparsia.com/pellet/] – програмний засіб – машина логічного виводу на Java для реалізації міркувань (виведення нових знань) з бази знань на мові OWL 2.0

3.4 Принцип роботи та структура парсера веб-джерел інформації в системі автоматизованої розбудови онтології

3.4.1 Мета та технічне завдання щодо призначення парсера

Оскільки не існувало уніфікованої системи для пошуку статтей у інтернеті, які б несли наукову цінність та новизну, було поставлена мета – створити таку ситему. Тобто у створенні системи котра б за ключовими словами користувачів видавала результатом лише наукові статті для подальшого опрацювання. Цінність такої системи у економії праці науковця у пошуку необхідного матеріалу чи документу, що вочевидь збільшує його продуктивність праці.

Для досягнення мети були поставленні такі технічні завдання:

- створення уніфікованої системи пошуку інформації, яка містить наукову цінність;
- систему необхідно реалізувати у вигляді програми (незалежного модуля), котра повинна бути написана на сучасній мові програмування;
- програма повинна містити компонент для забезпечення зв’язку з веб-джерелом інформації (наукові веб-ресурси, бібліотеки, архіви, сховища даних тощо);

- програма повинна містити компонент, котрий забезпечує отримання вмісту веб-сторінки;
- програма повинна містити компонент, котрий забезпечує зчитування та обробку вмісту статті, а також супутніх відомостей з віддаленого веб-ресурсу;
- програма повинна бути побудована за таким шаблоном проектування, який дозволяє розширяти її можливості без суттєвої зміни існуючої кодової бази;
- програма повинна містити технічну документацію, яка дозволяє зрозуміти принцип роботи та спростити подальшу розробку;
- програма повинна відповідати основним критеріям ООП та загально-прийнятим стандартам розробки;
- програма повинна бути кросплатформенною та містити мінімум залежностей від сторонніх бібліотек обраної мови програмування.

3.4.2 Інструментарій необхідний для створення парсера

Для створення уніфікованої системи пошуку інформації, яка містить наукову цінність, а саме – комп’ютерної програми, яка реалізує саму систему було обрано мову програмування Java.

Однією з головних переваг мови Java – її незалежність від платформи , на якій виконуються програми . Таким чином, один і той же код можна запускати під управлінням операційних систем Windows , Linux , FreeBSD , Solaris , Apple Mac та ін.. Це стає дуже важливим, коли програми завантажуються за допомогою глобальної мережі інтернет і використовуються на різних платформах. Іншою, не менш важливою перевагою Java є велика схожість з мовою програмування C++. Тому тим програмістам, які знайомі з синтаксисом C і C++ буде просто освоїти Java. Крім того, Java – повністю об'єктно-орієнтована мова, навіть більшою мірою , ніж C++. По суті все в мові Java є об'єктами, за винятком небагатьох основних типів (primitive types), наприклад чисел. Важливо і те, що розробляти на Java програми , які не містять помилок

значно легше, ніж на C++ . Вся справа в тому, що розробниками мови Java з компанії Sun був проведений фундаментальний аналіз програм мовою C++. Аналізувалися "вузькі місця" вихідного коду, які й призводять до появи помилок . Тому було прийнято рішення проектувати мову Java з урахуванням можливості створювати програми, в яких були б приховані найбільш поширені помилки. Для цього було зроблено наступне:

- Розробники виключили можливість явного виділення і звільнення пам'яті. Наприклад , пам'ять в Java звільняється автоматично за допомогою механізму збору сміття. Виходить , що програміст застрахований від помилок , які виникають від неправильного використання пам'яті.
- Введення істинних масивів і заборона покажчиків. Тепер програмісти не можуть стерти дані з пам'яті з причини неправильного використання покажчиків. Була виключена можливість переплутати оператор присвоювання з оператором порівняння на рівність . Як правило , проблема зі знаком "=" дуже часто призводить в C і C++ до логічних помилок , які не так просто виявити. Особливо у великих програмах.
- Повністю виключено множинне спадкування. Воно було замінено інтерфейсом. Інтерфейс дає програмісту практично все, що той може отримати від множинного спадкоємства , уникаючи при цьому складнощів , які виникають при управлінні ієрархіями класів .

Окрім того у Java передбачені технічні засоби по створенню автодокументації (JavaDoc), а це - спрощує реалізацію однієї з умов поставлених технічним завданням.

Для написання коду програми було обрано середовище розробки Eclipse, що э світовим стандартом в області інтегрованих середовищ розробки.

Шаблоном проектування було обрано «Шаблонний метод» (template method), оскільки цей шаблон максимально чітко задовільняє умови поставлені технічними завданнями.

Бібліотекою парсера було обрано Jericho HTML Parser. Jericho HTML Parser – це бібліотека Java, яка є надбудовою інтегрованої бібліотеки парсера Java – DOM. Тобто, Jericho HTML Parser максимально реалізовує можливості DOM. Окрім того, одна з ключових особливостей Jericho HTML Parser – приведення вхідного HTML коду до валідного вигляду перед здійсненням операцій синтаксичного аналізу над ним, що виключає появу помилок в процесі опрацювання, а також у процесі компіляції програми.

3.4.3 Бібліотека парсера. Використані методи, поля та їх призначення

При створенні системи було здійснено аналіз механізмів синтаксичного аналізу вмісту веб-сторінок інтегрованих в Java. Таких механізмів, а точніше інтерфейсів є два:

- DOM – Document Object Model;
- SAX – Simple API for XML.

При використанні DOM документ, у якому необхідно зробити аналіз систематизується у вигляді DOM-дерева (дерева ієрархій). До елементів такої ієрархії зручно доступатись, обробляти. Окрім цього пошук необхідного елемента в DOM володіє високою швидкодією, проте сам механізм інтерфейсу потребую більших затрат пам’яті, ніж SAX.

Пошук необхідної інформації при механізмі SAX відбувається ітеративним методом, тобто методом перебору всіх елементів в документі від першого елемента до останнього. Тобто на кожній ітерації циклу віддається лише один елемент на опрацювання і звернутись до нього можна лише, коли ітерація дійде до останнього елемента документа та почне новий цикл. SAX механізм володіє набагато меншою швидкодією, ніж DOM, проте потребує менше затрат пам’яті.

Отже, для реалізації системи було обрано DOM механізм, оскільки звертатись до необхідних елементів документа значно швидше, якщо вони

систематизовані в DOM моделі. Реалізацій та надбудов для DOM існує надзвичайно багато. Основні з них:

- Jericho HTML Parser;
- Java XML Parser;
- JTidy;
- HTML Parser.

З цієї множини аналізаторів було обрано Jericho HTML Parser, оскільки основними його перевагами є:

- не валідний HTML документ не викликає помилок в процесі аналізу;
- HTML документ аналізується навіть, якщо у ньому присутні серверні теги;
- розбір параметрів відбувається за допомогою класу StreamedSource, що дозволяє пам'яті ефективно обробляти великі файли. Це надає додаткові функції, які недоступні в інших потокових аналізаторів;
- номер рядка і стовпчика кожної позиції у вихідному документі легко доступні.

Jericho HTML Parser є бібліотекою Java з відкритим кодом, що розповсюджується по двом ліцензіям: Eclipse Public License (EPL) і GNU Lesser General Public License (LGPL). Тому можна використовувати його в комерційних цілях у відповідності з умовами докладно описані в умовах ліцензій. Javadocs містять вичерпну інформацію про всьому API.

Використані класи, методи та поля для створення системи:

- Source;
- fullSequentialParse();
- Segment;
- getAllElements(StartTagType);
- getTextExtractor();
- HTMLElementName;
- getAttributeValue(String attributeName);
- getParentElement();

- getName();
- getChildElements();
- Source – класс, об’єкт якого містить HTML документ.
- fullSequentialParse() – метод класу Source, котрий аналізує всі теги вихідного документу.
- Segment – класс, об’экт якого розбирає на сегменти вміст об’єкту Source.
- getAllElements(StartTagType) – метод класу Segment, котрий створює DOM ієрархію вихідного документу відповідно до заданої умови.
- getTextExtractor() – метод, котрий повертає чистий текст та видаляє усі теги у заданому документі.
- HTMLElementName – клас, який містить статичні методи для вибору необхідних тегів.
- getAttributeValue(String attributeName) – метод, котрий повертає декодоване значення атрибута з вказаним ім’ям.
- getParentElement() – метод, котрий повертає батьківський елемент стосовно заданого в DOM ієрархії.
- getName() – метод, котрий повертає ім’я даного елементу.
- getChildElements() – повертає список безпосередніх нащадків цього елемента в ієрархії елементів документа.

3.4.4 Загальна та принципова схема структури парсера.

Система складається з трьох основних модулів (інтерфейсів), а також допоміжних інтерфейсів, які забезпечуєть ітерацію (див. рис. 3.11).

Цими модулями є: iWebInfoParser, iWebInfoSource, iConnectionProvider.

Допоміжні інтерфейси, котрі забезпечуєть ітерацію: Iterable<Publication>, Iterable<String>.

Модуль iConnectionProvider являє собою інтерфейс, котрий реалізується класами StraightConnection або ProxyConnection в залежності від обраного

параметру з'єднання (пряме включення або включення через сервер аутентифікації).

Клас StraightConnection включає два основних публічних метода, котрі реалізують інтерфейс iConnectionProvider: connect(URL), isConnectible(URL).

Метод connect(URL) – забезпечує з'єднання з веб-джерелом інформації.

Метод isConnectible(URL) – перевіряє наявність з'єднання.

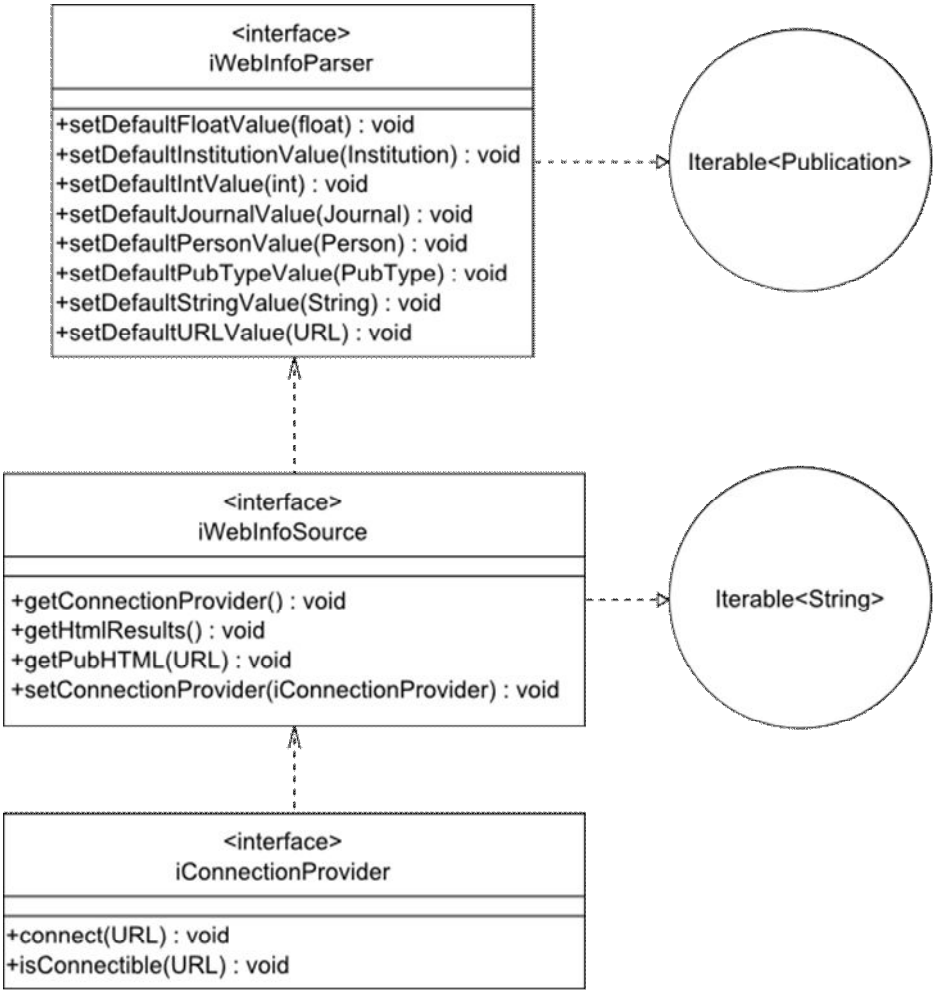

Рис. 3.11. Загальна схема структури парсера

Клас ProxyConnection включає в себе поля, які містять дані котрі необхідні для авторизації на сервері аутентифікації (див. рис. 3.12):

- proxyHost (назва серверу аутентифікації у виді доменної або IP адрес);
- proxyPort (порт серверу аутентифікації);
- proxyUserName(логін користувача на сервері аутентифікації);

- proxyUserPassword (пароль користувача на сервері аутентифікації).

Усі ці поля є закритими згідно концепції інкапсуляції. Доступ до них забезпечується за допомогою методів set та get.

Також у клас ProxyConnection включає два основних публічних метода, котрі реалізують інтерфейс iConnectionProvider: connect(URL), isConnectible(URL).

Метод connect(URL) – забезпечує з'єднання з веб-джерелом інформації.

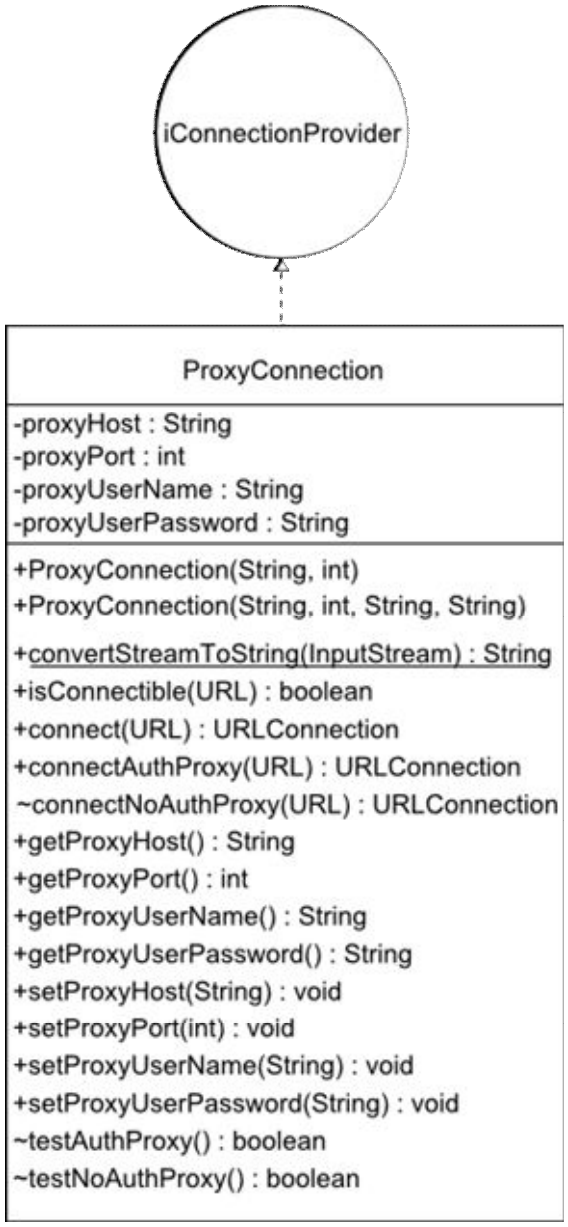

Рис. 3.12. UML-діаграма класу ProxyConnection

Метод isConnectible(URL) – перевіряє наявність з'єднання.

Також ProxyConnection включає ще такі ключеві методи:

- connectAuthProxy(URL),
- connectNoAuthProxy(URL),
- testAuthProxy(),
- testNoAuthProxy().

Компонент iWebInfoSource складається з класів (див. рис. 3.13):

- AbstractWebInfoSource,
- ScienceDirectWIS,
- CiteSeerXWIS,
- WileyOnlineLibraryWIS.

Клас AbstractWebInfoSource є абстрактиним классом та реалізовує інтерфейс iWebInfoSource. Класи ScienceDirectWIS, CiteSeerXWIS та WileyOnlineLibraryWIS реалізовують отримання «сирих» непроаналізованих даних з веб-джерел http://www.sciencedirect.com, http://citeseerx.ist.psu.edu та http://onlinelibrary.wiley.com відповідно.

Модуль iWebInfoParser складається з таких класів: AbstractParser, ScienceDirectParser, CiteSeerXParser, WileyOnlineLibraryParser.

Клас AbstractParser є абстрактиним классом та реалізовує інтерфейс iWebInfoParser.

Класи ScienceDirectParser, CiteSeerXParser та WileyOnlineLibraryParser здійснюють аналіз «сирих» непроаналізованих отриманих від компоненту iWebInfoSource.

Фрагмент програмної реалізації наведено у додатку А.

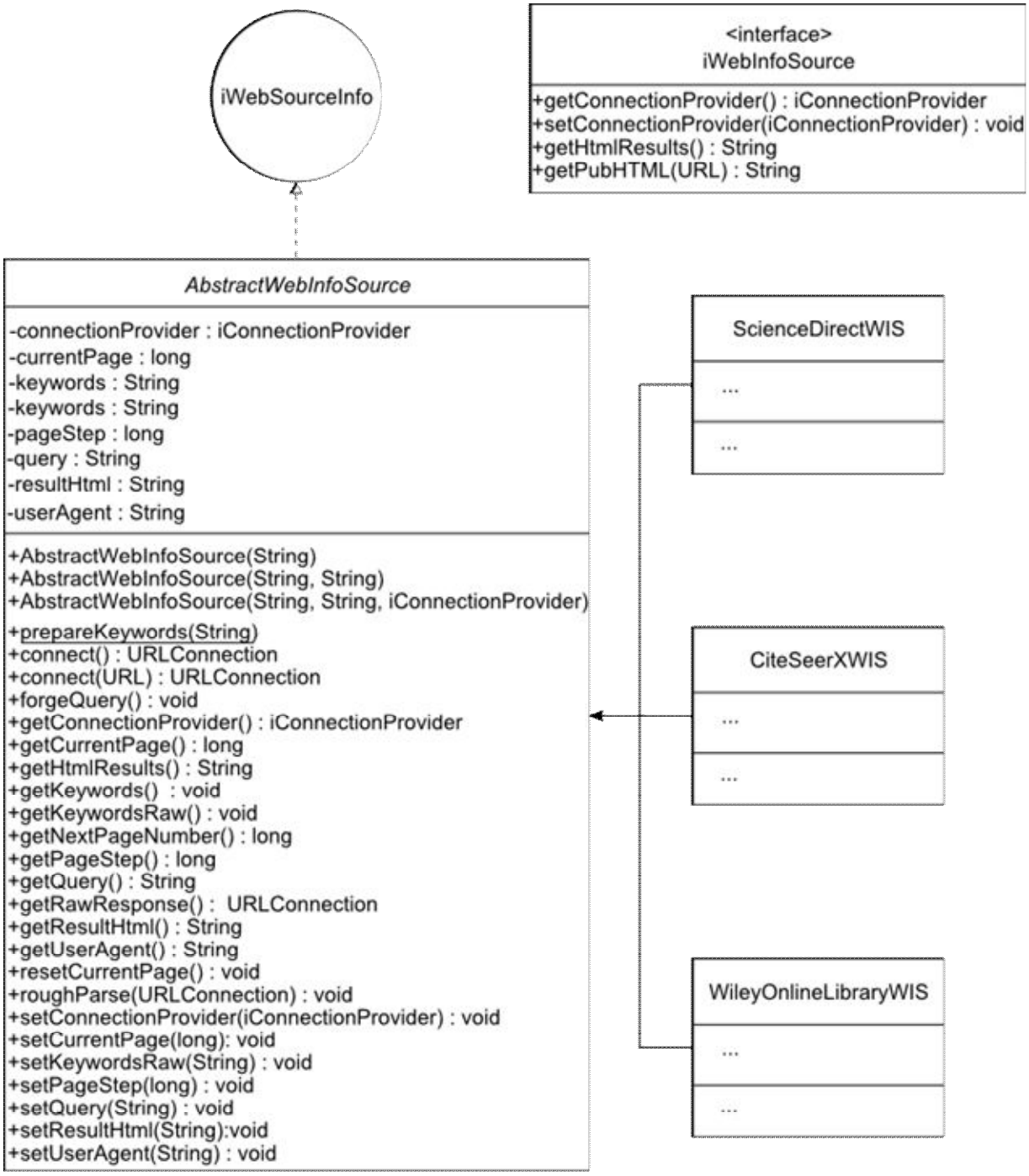

Рис. 3.13. Принципова схема компоненту iWebInfoSource

3.5 Основні результати та висновки до розділу

У цьому розділі роботи отримано такі результати:

1. Для забезпечення автоматизованої побудови онтології матеріалознавства як вузькоспеціалізованої онтології нижнього рівня її доповнено необхідним об’ємом базових понять та семантичних зв’язків цієї проблемної області.

2. Запропоновано новий метод оцінювання релевантності текстового документа до інформаційних потреб клієнта системи інформаційного пошуку, який базується на побудові моделі інформаційних потреб у формі оптимальної стратегії інтелектуального агента, оцінці її очікуваної корисності та зміні цієї корисності внаслідок уточнення плану шляхом доповнення інформацією з досліджуваного документа.
3. Розроблені інформаційні технології автоматизованого синтезу онтології були реалізовані як програмне забезпечення CAPO, яке може бути застосоване для реалізації запропонованого методу оцінювання релевантності у системах інформаційного пошуку та видобування знань з природомовних текстів.

РОЗДІЛ 4 . ПРОГРАМНИЙ КОМПЛЕКС ПЛАНУВАННЯ ДІЙ СПЕЦІАЛІЗОВАНОГО ІНТЕЛЕКТУАЛЬНОГО АГЕНТА

4.1 Архітектура програмної системи плануванні дій спеціалізованих інтелектуальних агентів

Нехай ІА перебуває в стані $St(0)$, і в нього наявний деякий матеріальний ресурс *G*. Перед ІА є завдання перейти у деякий цільовий стан *Goal*, використовуючи ресурс *G*. Якщо він досягнув цього стану, то в майбутньому він може ставити нові завдання переходу в новий цільовий стан, вважаючи стан *Goal* початковим. А наразі обмежимось завданням переходу зі стану $St(0)$ в стан *Goal*. Стан *Goal* може полягати в отриманні певного прибутку від своєї діяльності, модернізації устаткування. Очевидно, що такий перехід можна здійснити, використовуючи ресурс *G* та знання про ПО. Такі знання зберігатимемо в онтології *O*. Отже, формально нашу задачу *P* запишемо у вигляді

$$P:\ St(0) \xrightarrow{G,O} Goal\ . \qquad (4.1)$$

Розроблення системи почнемо з побудови діаграми варіантів використання. Відповідна діаграма наведена на рис. 4.1.

Беручи до уваги метод, запропонований намий описаний у попередніх 2-ох розділах 2 та 3, отримаємо діаграму діяльності функціонування системи плануванні дій спеціалізованих інтелектуальних агентів, яка наведена на рис. 4.2. Функціонування системи складається з п’яти етапів. Між етапами 2 та 3 знаходиться проміжний етап, який полягає у автоматизованій розбудові онтології. Діаграма діяльності цього етапу наведена на рис. 4.3.

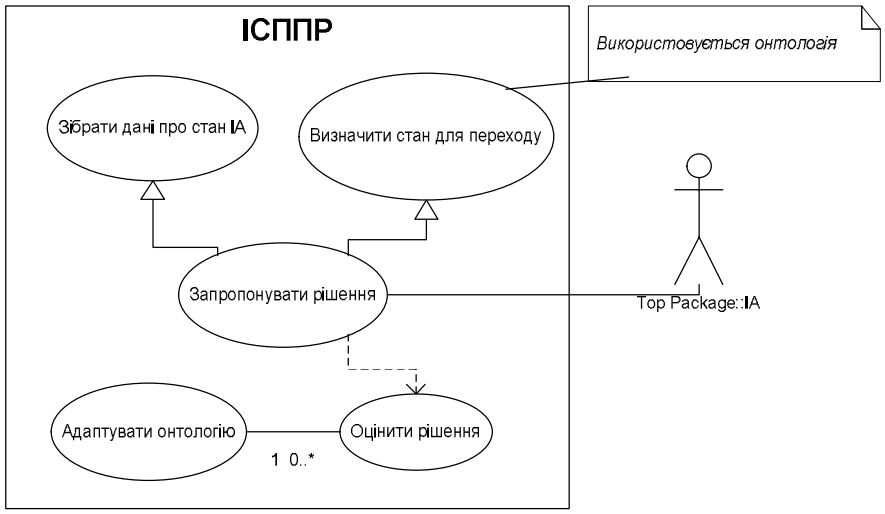

Рис. 4.1. Діаграма варіантів використання системи плануванн дій спеціалізованих інтелектуальних агентів

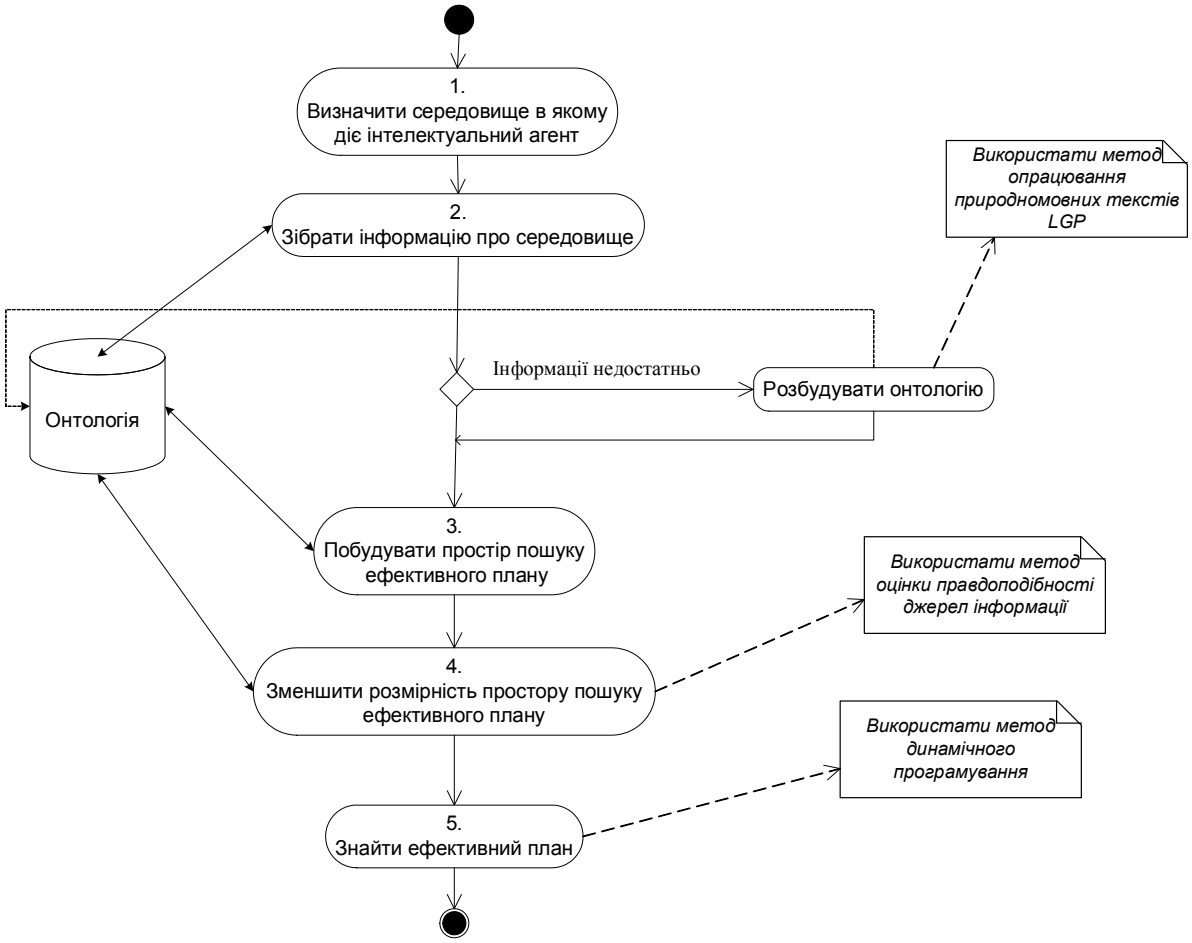

Рис. 4.2. Діаграма діяльності функціонування системи плануванн дій спеціалізованих інтелектуальних агентів

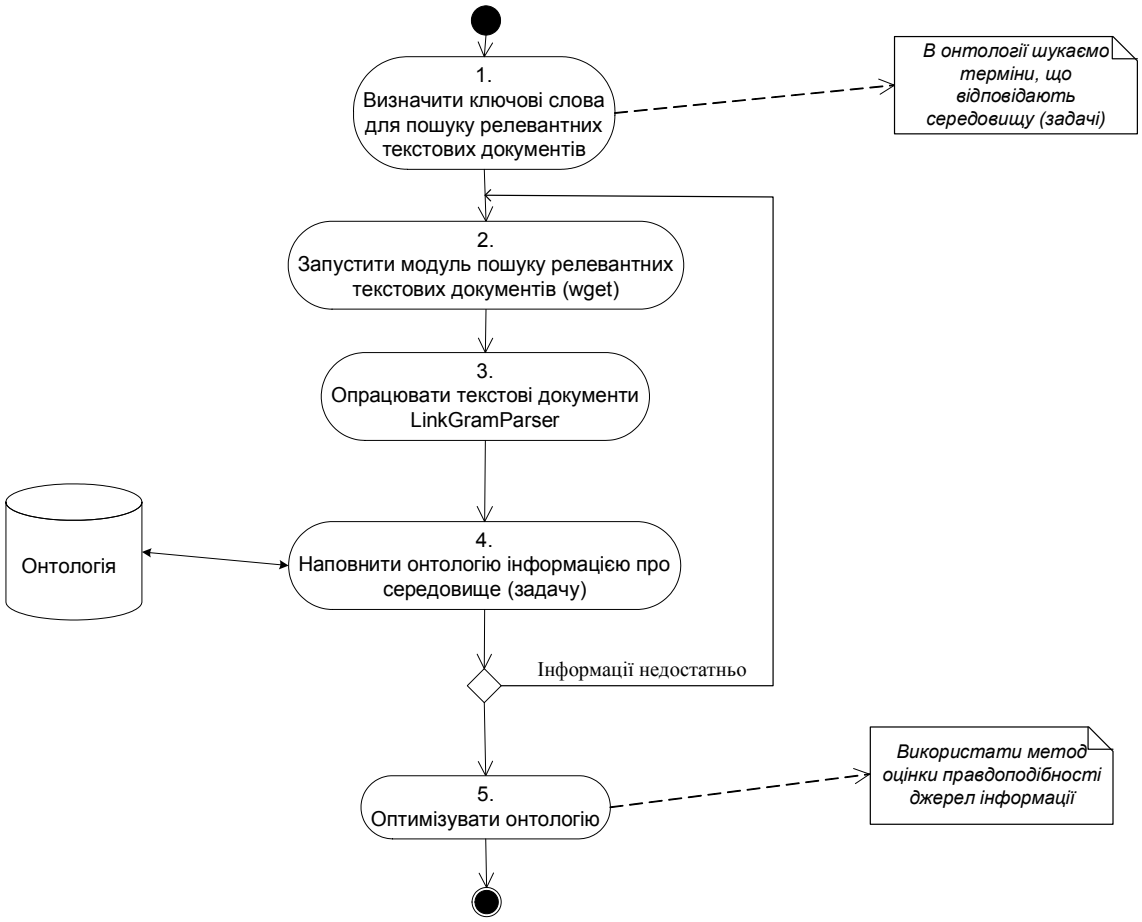

Рис. 4.3. Діаграма діяльності етапу автоматизованої розбудови онтології системи плануванн дій спеціалізованих інтелектуальних агентів

Беречи до уваги наведені вище діаграми варіантів використання та діяльностей, пропонуємо архітектуру системи плануванн дій спеціалізованих інтелектуальних агентів, яка наведена на рис. 4.4.

На основі онтології та бази даних будуємо простір станів в якому будемо шукати розв'язок задачі планування дій інтелектуального агента, тобто шлях переходу із початкового стну в стан мети. Для побудови такого простору станів використовуємо оптимізовану онтологію. Суть оптимізації полягає у використанні тих елементів онтології, які знаходяться у джерелах інформації з високою довірою. Після побудови простору станів та визначенні оцінки станів, запускаємо модуль, який розв’язує задачу динамічного програмування. На виході отримуємо пропоноване системою рішення.

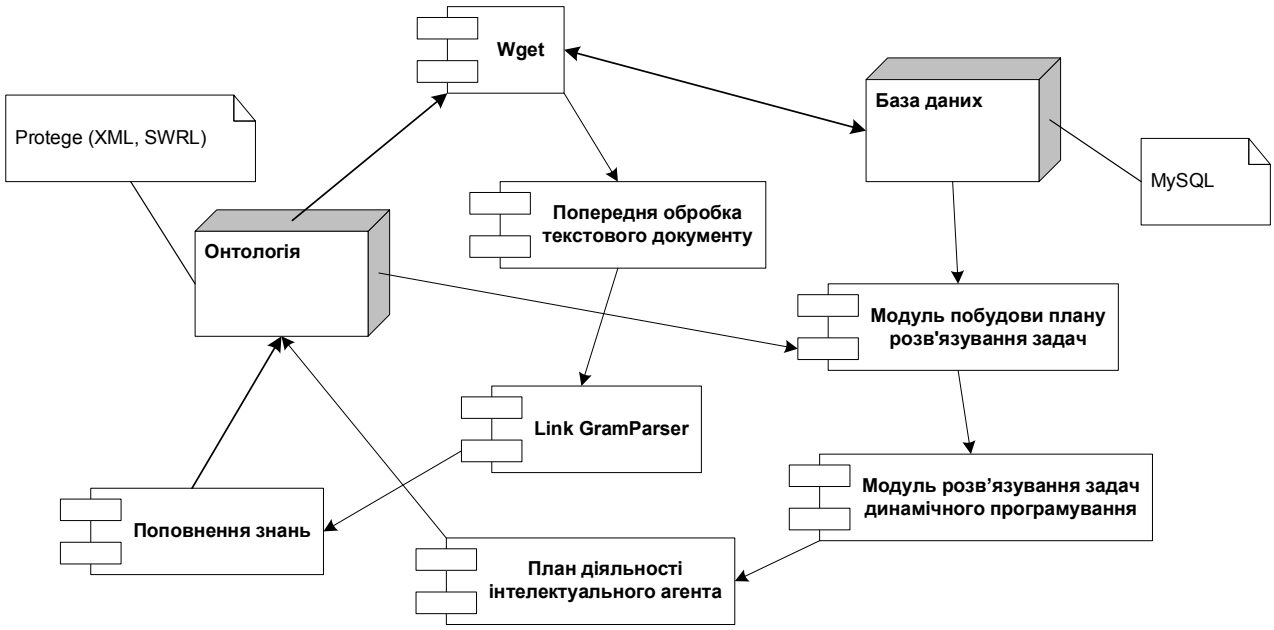

Рис. 4.4. Архітектура системи плануванні дій спеціалізованих інтелектуальних агентів

Архітектура системи складається із онтології, яка містить онтологію задач на розв'язання якої націлене функціонування спеціалізованого ІА та онтологію предметної області, яка задає альтернативи розв'язання окремих під задач. У базі даних зберігаються екземпляри окремих понять онтології, а також історія функціонування спеціалізованого ІА. Центральним модулем є модуль побудови плану розв'язування задач, який функціонує на основі онтології та екземплярів понять онтології (бази даних). Після побудови простору пошуку ефективного плану дії ІА, запускається модуль розв'язування задачі динамічного програмування методом функціональних рівнянь. У результаті отримуємо план діяльності ІА. Для розбудови та редагування онтології використовується модуль попередньої обробки текстових документів, програмний засіб Link Grammar Parser для пошуку понять предметної області у текстових документах, які релевантні предметній області, а також утиліта Wget для завантаження файлів за протоколами HTTP, HTTPS та FTP.

4.2 Побудова онтології предметної області

Знання набувають змісту лише в контексті певної проблемної області (ПО), заданої у даному випадку її онтологією. Набуті з текстового документа нові знання приймають форму змін у первинній онтології, яку слід попередньо сформувати вручну або шляхом застосування процедур навчання. Аналіз кожного наступного тексту базується на застосуванні онтології, доповненої в процесі аналізу попередніх текстів у тій частині, яка стосувалася заданої проблемної області. При розпізнаванні змісту текстів та доповненні онтології ПО ключовим є підхід, за якого першочергово необхідно виявити засоби досягнення мети, рекурсивно призначаючи їх підцілями і шукаючи як у тексті, так і у самій онтології (відповідній цій онтології базі знань) засоби досягнення цих підцілей. Тобто, читаючий агент будує на основі прочитаного тексту дерево цілей для задачі, рішення якої він шукає. У тексті засоби можуть бути формально ідентифіковані як іменникові групи, що слідують за дієсловом 'using', зворотом ‘by means’ або іншими подібними характерними зворотами, ідентифікувати які система розпізнавання змісту може навчитися засобами машинного навчання. У зв’язку з цим онтологія інтелектуального агента має містити у своїй основі як на верхньому рівні, так і на рівні прикладних проблем (задач) дерево цілей даної ПО та відповідне йому дерево рішень.

Для онтології матеріалознавства прототипом дерева цілей можуть служити діаграми, розроблені І.В.Федорович у її дисертаційній роботі. Ефективність прийняття рішення в будь якій проблемній області може бути визначена як відношення виграшу внаслідок рішення (послідовності рішень) до затрат чи втрат, пов'язаних з прийняттям (неприйняттям) цих рішень. Структура втрат для проблемної області «протикорозійний захист магістральних трубопроводів» подана на рис. 4.5.

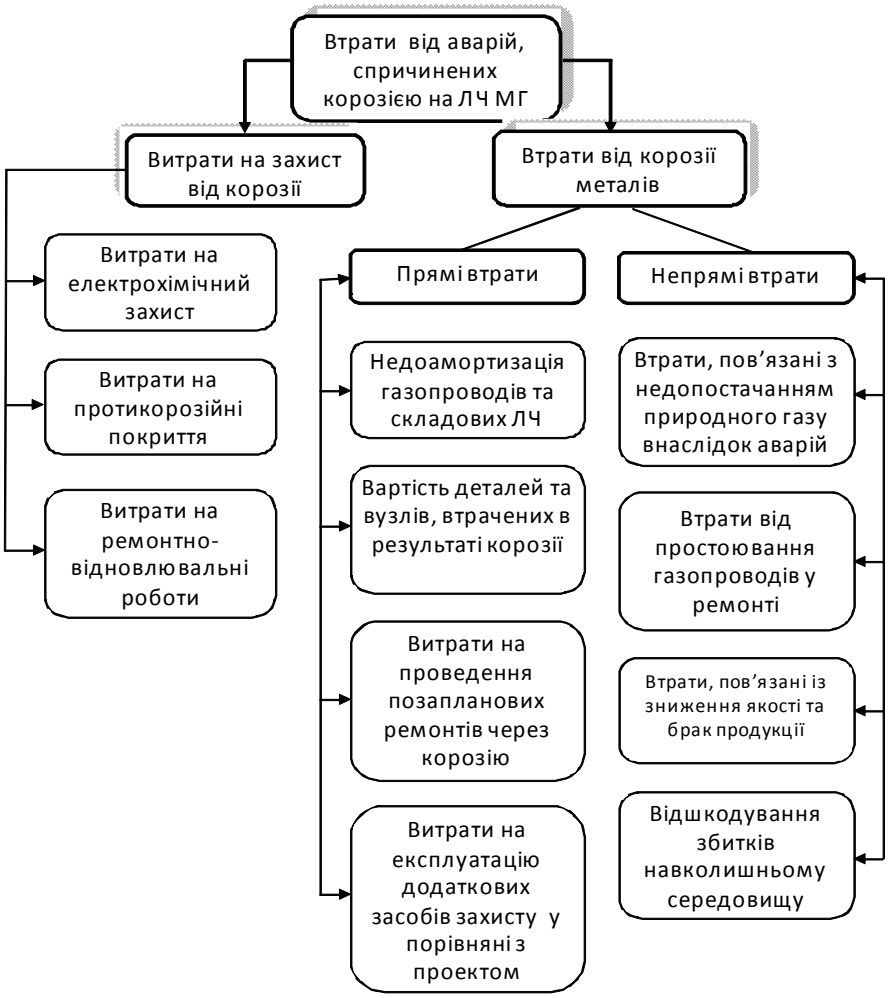

Рис. 4.5. Структура втрат для ПО «протикорозійний захист магістральних трубопроводів»

У цій роботі з метою визначення стратегії і тактики дій керівників газотранспортних підприємств у сфері відновлення основних засобів та сприяння пошуку ефективних методів забезпечення стабільності та ефективності внутрішнього механізму відтворення лінійної частини магістральних газопроводів було виділено основні чинники впливу. З врахуванням можливості контролю лише внутрішніх чинників, вони були розглянуті більш детально (див. рис. 4.6):

Рис. 4.6. Внутрішні чинники ефективного відтворення нафто-газотранспортної системи

Сукупність виділених чинників за результатами їх експертної оцінки формує структуру імовірностей досягнення успіху (його максимізації) внаслідок прийняття рішення стосовно відновлення та модернізації газотранспортної системи. Розподіл відмов та аварій на лінійній частині магістральних газопроводів, у залежності від причин їх виникнення показав, що основною причиною (54%) є корозія металу труби. За результатами аналітичного оцінювання було виявлено, що найбільшим впливом відзначаються такі чинники, як:

- якість виконання робіт з будівництва газопроводів;
- якість ремонтного обслуговування;
- рівень придатності ізоляційного покриття газопроводу;
- рівень корозійного руйнування газопроводу;
- рівень кваліфікації робітників-ремонтників та інженерно-технічних працівників;
- рівень досконалості прийняття управлінських рішень у процесі відновлення магістральних газопроводів.

Усі ці чинники мають знаходити своє представлення в онтології, а імовірність їх впливу відображена у дереві рішень цієї проблемної області. В результаті під час аналізу природомовного тексту пошук нових знань включає в себе не лише пошук підцілей як проміжних рішень головної задачі, а і уточнення впливу окремих чинників на імовірність досягнення відповідних підцілей і, як результат, головної цілі. Таким чином уточнюється загальна інформаційна модель проблемної області, що дозволяє ефективніше використовувати ресурси, приймаючи кращі рішення і таким чином оцінювати цінність нових знань через очікувану економію ресурсів.

Для проблеми модернізації трубопроводу загальна задача виглядає як зображено на рис. 4.7. Початковий стан: Необроблена. Кінцевий стан (стан мети): Оброблена.

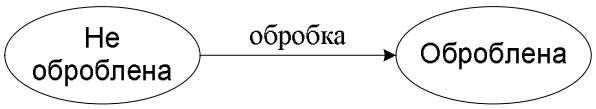

Рис. 4.7. Загальна задача модернізації трубопроводу

Задача ділиться на 3 підзадачі (підготовка, покриття, захист), перша з яких ділиться ще на 4 підзадачі (розкриття поверхні труби, зняття захисного покриття, знежирення, ґрунтування) як показано на рис. 4.8. Для розв’язування кожної підзадачі використовуються альтернативні рішення. Так для підзадачі зняття захисного покриття можна використати одну із трьох альтернатив:

механічне, хімічне, термічне. Вся ця інформація зберігається у онтології ПО модернізації нафто- та газопроводів

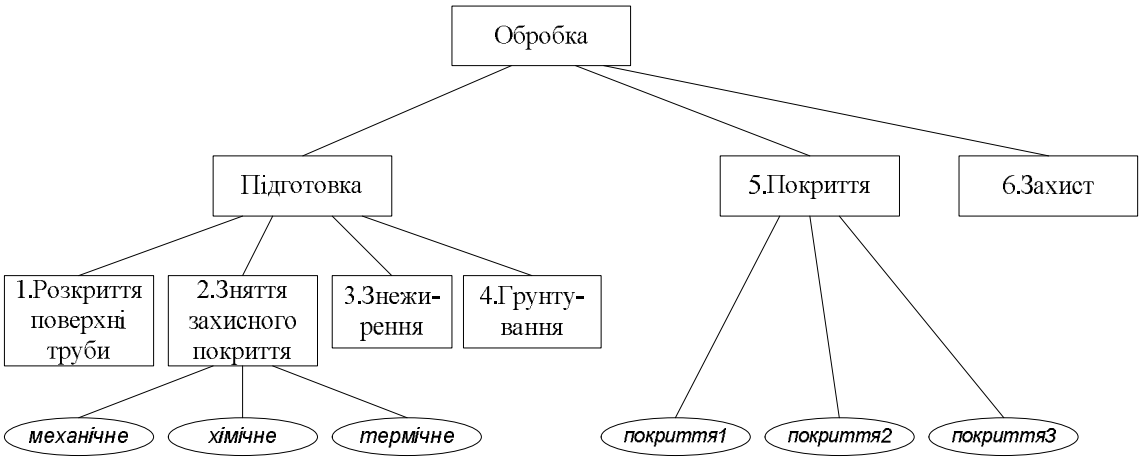

Рис. 4.8. Декомпозиція задачі «Обробка»

Отже загалом необхідно послідовно розв'язати 6 підзадач $P_1, P_2, ..., P_6$. Для кожної задачі необхідно вибрати метод розв'язку (альтернативу). Якщо G – наявний ресурс, r_e – бажаний термін експлуатації трубопроводу, то раціональність прийняття рішень полягатиме в:

$$\begin{cases} U = \sum_{i=0}^{N-1} U\left(a_{ij}^k\right) \to \max, \\ r \geq r_e, \\ \sum_{i=0}^{N-1} g_{ij}^k \leq G. \end{cases} \quad (4.1)$$

Нехай необхідно модернізувати трубу (газову чи для води). Необхідно цю трубу обробити, щоб якомога збільшити термін її експлуатації. Отже, початковий стан: необроблена. Кінцевий стан (стан мети): оброблена.

Оскільки складова одна, то розглядається одне завдання: оброблення (див. рис. 4.7). Завдання ділимо на три підзавдання (*підготовка, покриття, захист*), перше з яких ділимо ще на чотири підзавдання (*розкриття поверхні труби, зняття захисного покриття, знежирення, ґрунтування*), як показано на рис. 4.8. Отже, загалом необхідно послідовно виконати шість підзавдань $P_1, P_2, ..., P_6$. Для виконання кожного підзавдання використовують альтернативні

рішення. Для підзавдання *зняття захисного покриття* можна використати одну із трьох альтернатив: механічне, хімічне, термічне. Вся ця інформація зберігається у відповідній онтології.

Для визначення параметрів альтернатив (вартість модернізації та період експлуатації) використовуємо інтелектуальний пошук серед ПМТ. Тому вважаємо за доцільне запровадити в анотаціях до робіт, у яких відображені новий метод, методика чи технологія, явне описання необхідних затрат на їхню реалізацію та очікуваний ефект (виграш) від їх застосування.

4.3 Реалізація онтології матеріалознавства

4.3.1 Побудова базової онтології

Збір знань, релевантних ПО матеріалознавства, які містяться в онтології, здійснювався на основі книг Я. Середницького, Ю. Банахевича, А. Драгілєва "Сучасна протикорозійна ізоляція в трубопровідному транспорті" [75] та інформації, взятої із попередніх номерів журналів "Фізико-хімічна механіка матеріалів".

Мета функціонування ІА, який керуватиметься розроблюваною БЗ, визначає які знання мають бути подані у базі. Фактично, ідентифікація завдання відповідає розробленню PEAS, зокрема, розроблення оптимізаційного завдання САУ проблемного середовища, визначення цільової функції через визначення критеріїв оптимальності та обмежень. Отже, призначення цієї онтології полягає в реалізації ІА деякого плану на основі раціональної поведінки. Така раціональна поведінка складається з чотирьох компонент [21]:

а) множини дій, з яких складаються етапи плану;

б) множини обмежень впорядкування типу $A \prec B$ (A перед B);

в) множини причинних зв'язків (інтервалів захисту) типу $A \xrightarrow{P} B$ (A досягає P для B), коли за умовами ПО P, спричинене A не може змінитися доки не наступить стан B;

г) множини відкритих передумов для кожного з етапів плану.

Ієрархію понять ПО фізико-хімічної механіки матеріалів (ФХММ) ми почали із поняття “сутність” (essence), яка поділяється на абстрактні поняття (abstract) і фізичні (physical). Вигляд такої таксономії в редакторі Protégé наведено на рис. 4.9. Ці своєю чергою поділяються далі з різними критеріями, утворюючи таксономію понять.

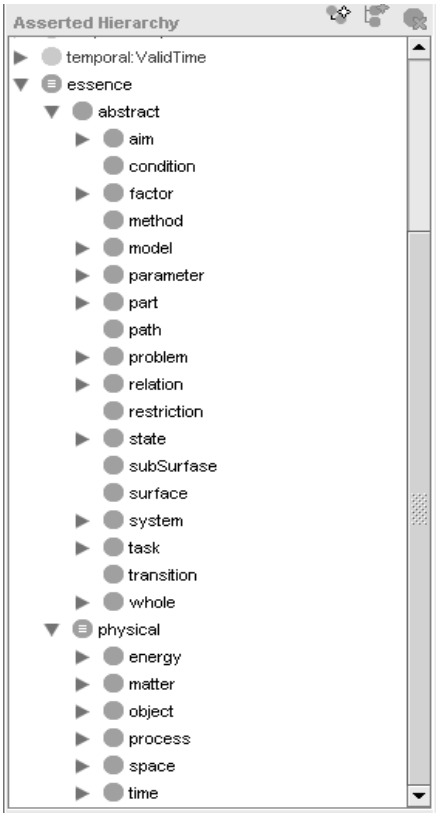

Рис. 4.9. Таксономія понять онтології матеріалознавства

Записано аксіоми термінів словника та атомарні висловлювання про екземпляри понять. Після цього здійснено налагодження БЗ. Неправильні аксіоми були виявлені на підставі того, що вони являють собою неправильні твердження про світ.

Допускається задавати класи понять, накладаючи обмеження на властивості їх екземплярів. Такі обмеження можна згрупувати у три основні категорії:

- кванторні обмеження (існування, загальності);
- обмеження кількості допустимих значень (мінімум ≤, якраз =, максимум ≥);

- обмеження типу “може приймати значення з множини”.

Обмеження існування описує клас екземплярів, які мають принаймні один зв’язок вказаного семантичного значення з екземпляром вказаного класу. У цьому разі квантор існування застосовується до множини зв’язків екземпляра (а не до множини екземплярів класу, як може видаватися). Отже, квантор існування свідчить, що цей клас містить лише ті екземпляри, множина зв’язків яких містить конкретний зв’язок: $\left\{x \middle| \exists r, r\left(x, y\right)\right\}$.

Квантор загальності свідчить, що цей клас містить лише ті екземпляри, множина всіх зв’язків яких містить вказаний тут явно винятковий перелік зв’язків:

$$\left\{x \middle| \forall r, R_i \in r, R_1\left(x, y_1\right) \wedge R_2\left(x, y_2\right) \wedge \ldots \wedge R_n\left(x, y_n\right)\right\}.$$

Як сама онтологія, так і БЗ ІА загалом мають бути зорієнтовані на певну ПО. У разі онтології матеріалознавства ПО необхідно звузити до цілком конкретної проблеми чи задачі, яку користувач ІСППР має вирішити. Розроблено базові елементи онтології методів та засобів протикорозійної ізоляції трубопровідного транспорту.

Проблема формулюється так: як при мінімальних затратах максимально продовжити ресурс трубопроводу, беручи до уваги, що:

- основним обмежувальним ресурсфактором слугує електрохімічна корозія труби;
- заданий орієнтовний економічний ефект, який отримує користувач ІСППР від експлуатації трубопроводу та можливі втрати від припинення експлуатації;
- затрати на протикорозійний захист відомі і визначаються технологією такого захисту;
- орієнтовні терміни безаварійної експлуатації трубопроводу за відомих (заданих) вжитих заходів з його протикорозійного захисту відомі з

експертних оцінок, нормативів, даних неруйнівного контролю та технічної діагностики.

Загальне правило заміни відновлення покриття формулюється так:

ЯКЩО ((Настав термін відновлення покриття) АБО (Настала подія пошкодження покриття) АБО (Вимірювані параметри перевищують встановлений раніше допустимий поріг)) І (Наявні ресурси для оновлення покриття) ТО (Виконати заміну покриття).

База знань деталізує це правило через систему уточнювальних продукційних правил, побудовану відповідно до Рете алгоритму [21].

Для ІА інформаційного пошуку цінною вважається інформація, яка дає змогу досягнути успіху у вирішенні цієї проблеми, тобто:

- інформація про нові види протикорозійного захисту, що дають подовжені терміни безаварійної експлуатації;
- інформація про уточнену оцінку ресурсу трубопроводу;
- інформація про ефективніші технології нанесення покриттів.

Для пошуку цінної інформації проаналізували анотації наукових статей журналу “Фізика-хімічна механіка матеріалів” за останні десять років. Окремі анотації записані в розробленій онтології за допомогою SWRL-правил. Наприклад розглянемо анотацію статті В.П. Силованюк, Р.Я. Юхим “Зародження втомних тріщин біля включень у пружно-пластичних матеріалах” (Т. 44, № 6, 2008, С. 12–17): “*Запропоновано підхід для обчислення періоду зародження тріщини в околі пружних включень в напругопластичних матеріалах. Встановлено фактори, від яких залежить інтенсивність утоми матеріалу в околиці включення. Такими є: амплітуда й асиметрія навантаження; геометрія поверхні включення, його відносна твердість; модуль Юнга, границі міцності матеріалу матриці; параметри циклічного зміцнення, його гранична деформація”.* Ця анотація подана у вигляді SWRL-правила:

$$matter(?x) \wedge depend(?x1, ?x) \wedge geometry\_inclusion\_surface(?x1) \rightarrow$$

$$intensity\_fatigue(?x),$$

де права частина правила означає: *matter(?x)* – змінна *?x* є матеріалом, *geometry_inclusion_surface(?x1)* – *?x1* – геометрія поверхні увімкнення; *depend(?x1, ?x)* – *?x* залежить від *?x1*, а ліва частина як висновок правила: *intensity_fatigue(?x)* – в результаті отримуємо інтенсивність втомлення матеріалу *?x*.

Для задання ваг важливості елементів онтології введено властивість has_weight як відображення сутностей (essence) в величину weigh_ontology (див. рис. 4.10). Своєю чергою weigh_ontology містить два підкласи: вага важливості поняття (weight_concept) та відношення (weight_relation).

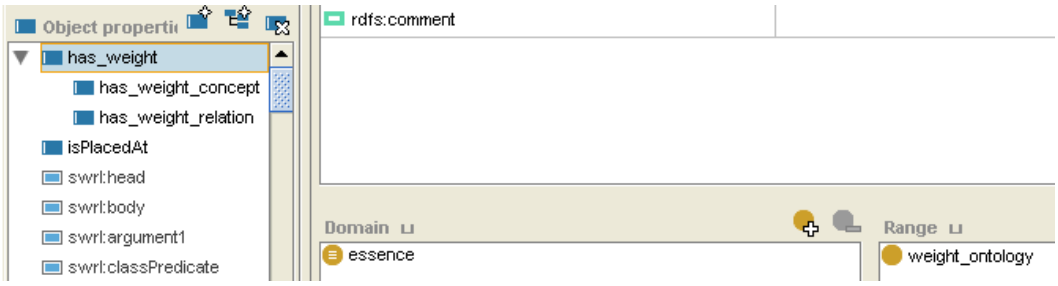

Рис. 4.10. Важливості понять та відношень в редакторі Protégé

Отже, для побудови онтології використано всі чотири методи подання знань. Так онтологія являє собою семантичну мережу фреймів (див. рис. 4.11). Для визначення поняття (значок) використовується логіка предикатів. Для побудови SWRL-правил використовуються продукційні правила.

Отже, в роботі запропоновано методику та розроблено базові елементи для побудови онтології ІА, який функціонує в галузі матеріалознавства. Даються формальні декларативні визначення базових класів та зв'язків, необхідних для побудови БЗ у цій ПО. Фрагмент програмного коду у форматі owl наведено у додатку Б.

Онтологія містить поза 5000 понять, 40% понять є визначеними. Запити до онтології показали, що показник її функціональної придатності складає порядка $\chi_1 = 90\%$.

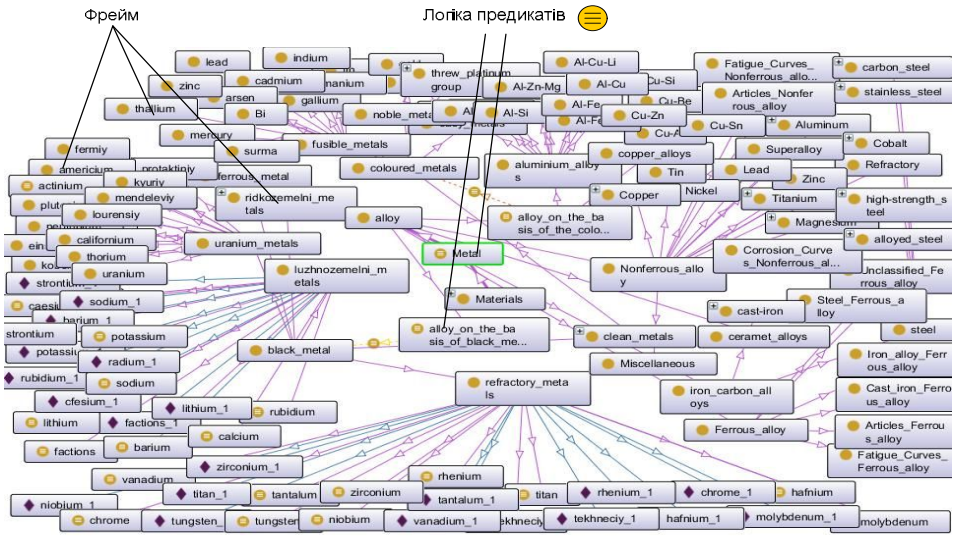

Рис. 4.11. Семантична мережа фреймів онтології матеріалознавства

Як зазначалось вище, для отримання показників періоду експлуатації та вартості робіт використовуємо мову запитів SPARQL до онтології. Приклад таких запитів наведено нижче.

Приклад 4.1. Запит:

```
PREFIX table: <http://www.owl-ontologies.com/Ontology1253189272>
SELECT *
FROM <http://www.owl-ontologies.com/Ontology1253189272.owl>
where {
{
$Cleaning rdfs:comment $value.
$Cleaning rdfs:subClassOf <#Cleaning>.
}
}
ORDER BY ASC(?value)
```

повертає методи очистки поверхонь трубопроводів:

Очистка ручним інструментом

Очистка електричним інструментом

Комерційна очистка

Очистка до майже чистого металу

Очистка до чистого металу

Приклад 4.2. Запит:

```
PREFIX table: <http://www.owl-ontologies.com/Ontology1253189272>
```

```
SELECT *
FROM <http://www.owl-ontologies.com/Ontology1253189272.owl>
where {
{
$Paints_varnishes rdfs:comment $value.
$Paints_varnishes rdfs:subClassOf <#Paints_varnishes>.
$Period > 20.
}
}
ORDER BY ASC(?value)
```

повертає лакофарбові матеріали, використання яких дозволяє використовувати металеву поверхню більше, ніж 20 років. Перелік таких матеріалів, які зберігаються в онтології, такий: *“Амберкоут – 2000”, “Амберкоут СЕЛ – 600”, “FC-210/ Амберкоут”, “Протегол УР – Коутінг 32-55”, “Десмодур/Десмофен СЖГ 17605 і 18045”, “Ромпур 804”; “ГІП”*.

4.3.2 Джерела автоматичного наповнення онтології

Базова онтологія ПО матеріалознавства будувалась на основі книги Я.Середницького, Ю. Банахевича, А. Драгілєва “Сучасна протикорозійна ізоляція в трубопровідному транспорті”. Автоматизована розбудова онтології здійснювалася на основі 7 джерел інформації, які наведені у табл. 4.1.

Таблиця 4.1. Джерела інформації для автоматизованої розбудови онтології матеріалознавства

	Джерело інформації	**Сторінка в інтернеті**
U_1	Open Access Materials Science Journal	http://www.mdpi.com/journal/materials
U_2	Journal Materials & Design	http://www.journals.elsevier.com/materials-and-design
U_3	Journal Nature Materials	http://www.nature.com/nmat/index.html
U_4	Materials science	http://www.nature.com/subjects/materials-science
U_5	Surface Technology / Functional Coating	http://na.henkel-adhesives.com/surface-treatment-industrial-solutions-18962.htm

U_6	Journal Surface and Coatings Technology	http://www.journals.elsevier.com/surface-and-coatings-technology/
U_7	Protective Coatings and Compounds	http://www.loctite.co.uk/protective-coatings-and-compounds-4486.htm

Було проведено 5 етапів наповнення онтології. Зміна міри довіри до джерела інформації наведена у табл. 4.2.

Таблиця 4.2. Зміна міри довіри до джерела інформації

	1	*2*	*3*	*4*	*5*
$U1$	0,5	0,75	0,938	0,9961	1
$U2$	0,5	0,75	0,75	0,75	0,938
$U3$	0,5	0,75	0,75	0,9375	0,934
$U4$	0,5	0,75	0,688	0,6875	0,688
$U5$	0,5	0,75	0,75	0,75	0,75
$U6$	0,5	0,75	0,75	0,6875	0,902
$U7$	0,5	0,75	0,75	0,75	0,75

Графік зміни міри довіри до цих 7 джерел інформації в залежності від етапу розбудови онтології наведено на рис. 4.12.

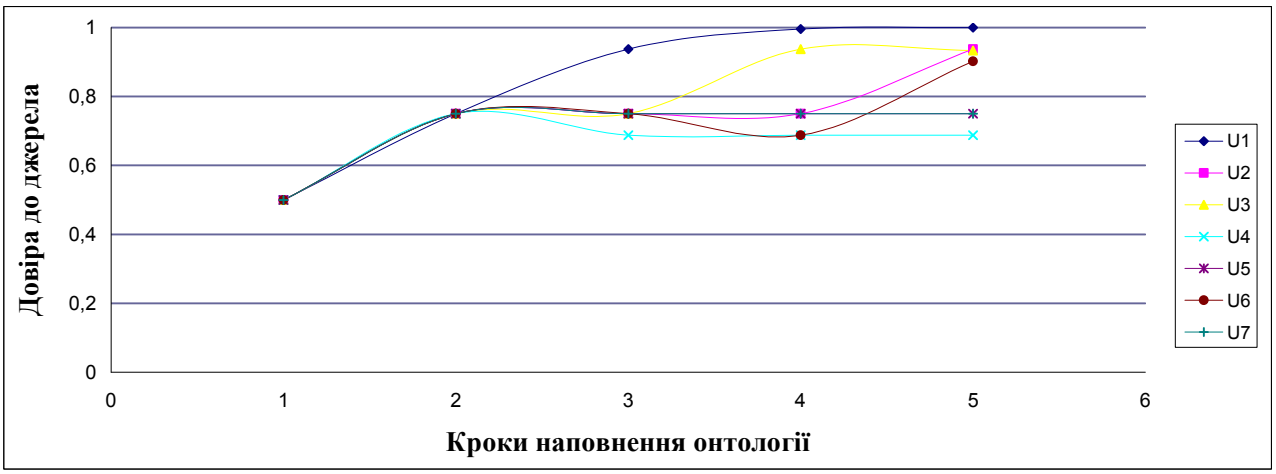

Рис. 4.12. Графік зміни міри довіри до джерел інформації під час автоматизованого наповнення онтології

4.4 Апробація функціонування системи плануваннй дій спеціалізованих інтелектуальних агентів

Кількість альтернатив для кожного етапу з використанням та без використання онтологій наведено у табл. 4.3.

Таблиця 4.3. Кількість альтернатив для кожного етапу з використанням та без використання онтологій

Етапи	Назва	К-сть альтернатив (стандарт)	К-сть альтернатив (після наповнення онтології)	К-сть альтернатив (після наповнення онтології з врахуванням оцінки довіри до джерела інформації)
1	Розкриття	3	15	12
2	Зняття покриття	4	26	16
3	Знежирення	5	32	22
4	Грунтування	4	45	31
5	Покриття	3	39	21
6	Захист	4	27	17
Min V	Вартість, у.о.	1200	920	950
Max Q	Час експлуатації, роки	20	42	42
minf=V/Q		60	21,9047619	22,61904762

Нехай ІА володіє ресурсом 6 одиниць. Приклад можливих витрат та доходів (виграшів) в залежності від номеру процесу та альтернативи наведені у табл. 4.4. Ці дані отримуються з відповідної онтології (див. наступний підрозділ), використовуючи мову запитів SPARQL до неї.

Таблиця 4.4. Таблиця витрат та доходів

№ альтр.	Процес 1		Процес 2		Процес 3		Процес 4		Процес 5		Процес 6	
	Витр.	Дох.	Витр.	Дох.	Витр.	Дох.	Витр.	Дох.	Витр.	Дох.	Вит.	Дох.
a_1	0	5	0	8	0	3	0	4	0	2	0	3
a_2	1	7	1	9	1	4	1	7	1	3	1	7
a_3	2	8	2	12	-	-	-	-	2	6	-	-

Використовуючи метод функціональних рівнянь, призначений для розв’язування задач динамічного програмування, отримаємо оптимальний шлях, який наведений на рис. 4.13.

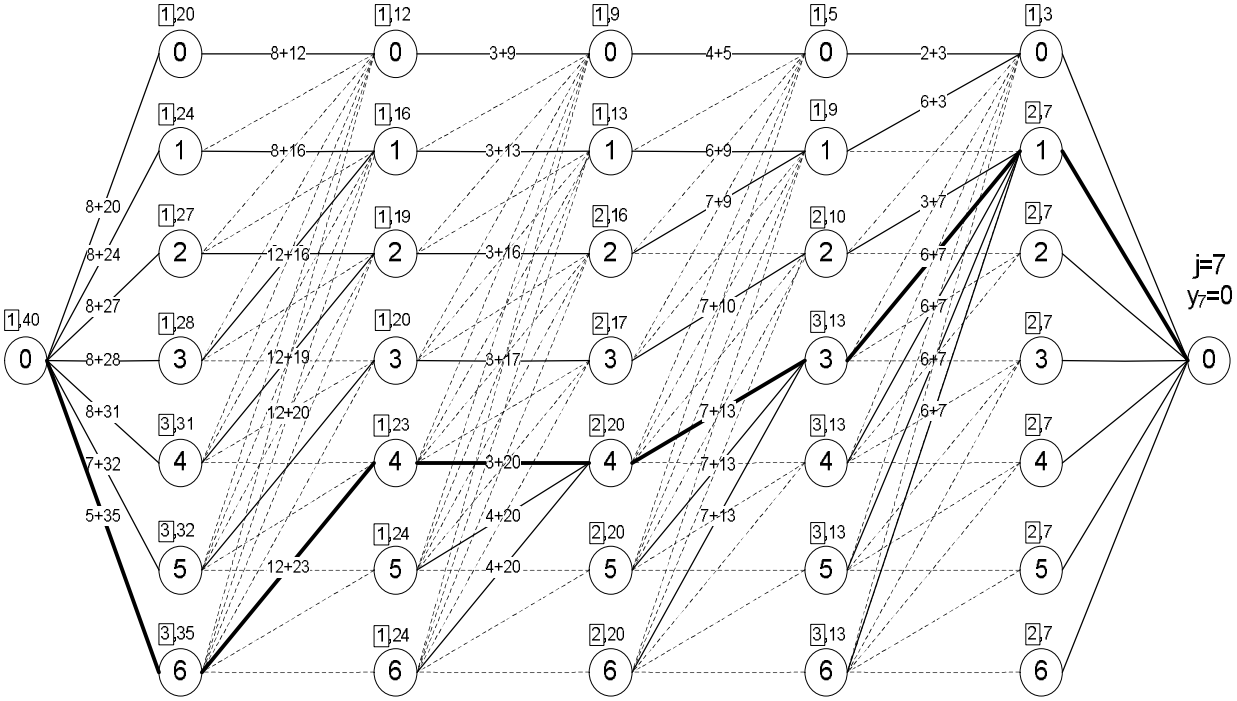

Рис. 4.13. Процес розв’язування задачі динамічного програмування

Отримуємо, що оптимальний план розподілення коштів на процеси обробки трубопроводу є такими:

Процес	**№ альтернативи**
Процес 1	1
Процес 2	3
Процес 3	1
Процес 4	2
Процес 5	3
Процес 6	2
Загальний дохід	40

4.5 Основні результати та висновки до розділу

1. Розроблено архітектуру системи плануванні дій спеціалізованих інтелектуальних агентів.

2. Побудовано базову онтологію матеріалознавства в складі системи плануванн дій спеціалізованих інтелектуальних агентів та процедури автоматизованого її наповнення.
3. Досліджено зміну довіри до джерел інформації на основі яких здійснюється автоматизована розбудова онтології.
4. Здійснено апробацію розроблених методів та алгоритмів функціонування системи плануванн дій спеціалізованих інтелектуальних агентів.

ВИСНОВКИ

У дисертаційній роботі вирішено важливе науково-прикладне завдання – побудови інтелектуальних агентів з використанням онтологічного підходу та підвищення ефективності таких систем, якого досягнуто завдяки застосуванню розробленого математичного та програмного забезпечення, що ґрунтується на використанні онтологій у цих системах, адаптацією онтологій до специфіки задач предметної області. Під час виконання роботи одержано такі результати.

1. Проаналізовано проблему функціонування інтелектуальних агентів планування діяльності. Обґрунтовано актуальність вирішення проблеми підвищення ефективності цих систем завдяки використанню онтологій, що дало змогу виділити не вирішені раніше проблеми з розроблення методів та засобів використання онтологій у складі інтелектуальних агентів планування діяльності.
2. Розроблено математичне забезпечення функціонування інтелектуальних агентів планування діяльності на основі онтологій, що дало змогу формалізувати поведінку таких агентів у просторі станів. Використання онтологій дає змогу звужувати простір пошуку шляху із початкового стану в стан мети, відкидаючи нерелевантні альтернативи. Сама задача планування діяльності інтелектуального агента зводиться до задачі динамічного програмування, де функцією мети є композиція двох функцій, які задають конкурентні критерії. Тобто у результаті отримуємо двохкритеріальну задачу. Для її розв’язування обрано метод головної компоненти, якщо цільові функції можна оцінити або метод комплексного критерію, якщо ці функції оцінити неможливо.
3. Отримав подальший розвиток процес автоматизованої розбудови адаптивної онтології на основі використання програмної системи Link Grammar Parser, яка розбиває стверджувальне речення, написане граматично правильною англійською мовою, на семантично пов'язані між собою пари слів.

Автоматична розбудова онтології реалізується засобами Java API Protege-OWL. Ці засоби містять бібліотеки класів, в яких реалізовано методи роботи з OWL-структурами: їх читання, доповнення. Таким чином, програмні засоби розбудови онтології функціонують у взаємодії з OWL-онтологією, беручи з неї шаблони граматично-семантичних структур для розпізнавання тверджень (предикатів дескриптивної логіки) у досліджуваних і/або навчальних текстах та, додаючи до неї нові елементи, в результаті такого розпізнавання.

4. Розроблено програмне забезпечення функціонування інтелектуальних агентів планування діяльності, яке ґрунтується на побудованих моделях, методах та алгоритмах, що дало можливість реалізувати окремі компоненти та функціональні модулі інтелектуальних агентів планування діяльності на основі онтологій, ядром баз знань яких є онтологія. Зокрема розроблено ІА в галузі діагностики та експлуатації виробів тривалої експлуатації, центральною компонентою якого є онтологія матеріалознавства. Діяльність такого агента надає задовільні розв’язки.

ЛІТЕРАТУРА

1. Vovnjanka R. Computer system for automated ontology building basic crocus / R. Vovnjanka, D. Maherovskyj, O. Oborska // Applied Computer Science : Instytut Technologicznych Systemów Informacyjnych. Politechnika Lubelska. – Poland, 2014. – Vol. 11, no 4. – P. 70-82.
2. Lytvyn V. Approach to decision support intelligent systems development based on ontologies / V. Lytvyn, O. Oborska, R. Vovnjanka // Econtechmod : Polish Academy of Sciences, Branch in Lublin. – Poland, 2015. – Vol. 4, no 4. – P. 29-35.
3. Литвин В. В. Метод використання онтологій в петлі OODA на прикладі функціонування вищих навчальних закладів / В. В. Литвин, Р. В. Вовнянка // Складні системи і процеси. – 2012. – № 2. – С. 38-43.
4. Литвин В. В. Метод моделювання процесу підтримки прийняття рішень у конкурентному середовищі / В. В. Литвин, О. В. Оборська, Р. В. Вовнянка // Математичні машини й системи : Науковий журнал Інституту проблем математичних машин і систем НАН України. – Київ, 2014. – №1. – С. 50-57.
5. Оборська О. В. Моделювання поведінки раціонального агента на основі стимулюючого навчання / О. В. Оборська, Р. В. Вовнянка // Інформаційні системи та мережі : Вісник Національного університету «Львівська політехніка"». – Львів, 2014. – № 805. – С. 61-69.
6. Вовнянка Р. В. Метод видобування знань з текстових документів / Р. В. Вовнянка, Д. Г. Досин, В. В. Ковалевич // Інформаційні системи та мережі : Вісник Національного університету «Львівська політехніка». – Львів, 2014. – № 783. – С. 302-312.
7. Литвин В. В. Комп'ютерна система автоматизованої розбудови базової онтології CROCUS / В. В. Литвин, Р. В. Вовнянка, Д. Г. Досин // Електротехнічні та комп'ютерні системи. – 2014. – № 13. – С. 135-143.

8. Метод побудови інтелектуальних агентів на основі адаптивних онтологій / В. В. Литвин, М. Я. Гопяк, О. В. Оборська, Р. В. Вовнянка // Інформаційні системи та мережі : Вісник Національного університету «Львівська політехніка». – Львів, 2015. – № 829. – С. 186-200.
9. Моделювання поведінки інтелектуального агента на основі петлі OODA / В. В. Литвин, М. Я. Гопяк, Р. В. Вовнянка, О. В. Оборська // Міжнародна науково-практична конференція «Інформаційні технології. Освіта». – Луцьк-Світязь, 2014. – С. 60-61.
10. Метод моделювання процесу підтримки прийняття рішень в конкурентному середовищі / В. В. Литвин, О. В. Оборська, Р. В. Вовнянка, М. Я. Гопяк // Матеріали 2-ї міжнародної наукової конференції «Інформація, комунікація, суспільство» (ICS-2013). – Львів-Славське, 2013. – С. 202-203.
11. Метод підтримки прийняття рішень у конкурентному середовищі на основі петлі OODA / В. В. Литвин, М. Я. Гопяк, Р. В. Вовнянка, О. В. Оборська // Матеріали міжнародної наукової конференції «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту». – Залізний Порт. – Херсон : ХНТУ, 2014. – С. 219-220.
12. Method development and qualiti evaluation of an ontology / Vasyl Lytvyn, Dmytro Dosyn, Roman Vovnjanka, Maria Hopyak, Oksana Oborska // XIIIth International Conference ‘The Experience of Designing and Application of CAD Systems in Microelectronics’. – Polyana-Svalyava (Zakarpattya), 2015. – P. 113-115.
13. Оборська О. В. Розробка модуля імітаційного моделювання бойових дій для етапу „орієнтація” циклу OODA / О. В. Оборська, Р. В. Вовнянка, М. Я. Гопяк // IV Міжнародна наукова конференція «Інформація, комунікація, суспільство» (ІКС-2015). – Львів-Славське, 2015. – С. 50-52.
14. Метод планування рішень у конкурентному середовищі на основі використання онтологій / В. В. Литвин, О. В. Оборська, М. Я. Гопяк, Р. В. Вовнянка // Десята міжнародна науково-практична конференція

«Математичне та імітаційне моделювання систем» (МОДС 2015). – Чернігів, 2015. – С. 460-465.

15. Литвин В. В. Моделювання процесу підтримки прийняття рішень в конкурентному середовищі на основі петлі OODA / В. В. Литвин, Р. В. Вовнянка, О. В. Оборська // Міжнародна науково-практична конференція «Інформаційні технології. Освіта». – Луцьк-Світязь, 2013. – С. 31-32.
16. Рассел С. Искусственный интеллект / С. Рассел, П. Норвиг. – М.; СПб.; К.: Вильямс, 2006. – 1408 с.
17. Malik G. Automated Planning Theory & Practice / G. Malik, N. Dana, P.Traverso. – San Francisco : Morgan Knaufman, 2004. – 635 p.
18. Представление и использование знания / под ред. Х. Уэно, М. Исидзука. – М.: Мир, 1989.
19. Braziunas D. Pomdp solution methods : technical report / D. Braziunas. – Toronto : University of Toronto, 2003. – 24 p.
20. Efficient solution algorithms for factored MDPs / C. Guestrin, D. Koller, R. Parr, S. Venkataraman // JAIR. – 2003. – №19. – P. 399–468.
21. Spaan M. Perseus: Randomized point-based value iteration for POMDPs / M. Spaan, N. Vlassis. // JAIR. – 2005. – №24. – P. 195–220.
22. Li H. Incremental Least Squares Policy Iteration for POMDPs / H. Li, X. Liao, L. Carin. // AAAI. – Palm Springs, 2006. — P. 1167-1172.
23. Poupart P. Value-directed compression of POMDPs / P.Poupart C.Boutilier. // NIPS. – 2003. – №15. – Режим доступу: https://papers.nips.cc/paper/2192.
24. Spaan M. Perseus: Randomized point-based value iteration for POMDPs / M. Spaan, N. Vlassis. // JAIR. – 2005. – №24. – P. 195–220.
25. Гаврилова Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В.Ф. Хорошевский. – СПб.: Питер, 2001. – 384 с.
26. Хейес-Рот Ф. Построение экспертных систем / Ф. Хейес-Рот, Д. Уотерман, Д. Ленат. – М.: Мир, 1987. – 430 с.

27. Gruber T. A translation approach to portable ontologies / T. Gruber // Knowledge Acquisition. – 1993. – № 5 (2). – P. 199–220.
28. Chandrasekaran B. What are ontologies, and why do we need them? / B. Chandrasekaran, J. Josephson, V. Benjamins // IEEE Intelligent Systems. – 1999. – P. 20–26.
29. Data exchange: Semantics and query answering / R. Fagin, P. Kolaitis, R. Miller, L. Popa // Theoretical Computer Science. – 2005. – Vol. 336, № 1. – P. 89–124.
30. Палагин А. В. Архитектура онтологоуправляемых компютерных систем / А. В. Палагин // Кибернетика и системный анализ. – 2006. – № 2. – С. 111–125.
31. Guarino N. Formal Ontology, Conceptual Analysis and Knowledge Representation / N.Guarino // International Journal of Human-Computer Studies. – 1995. – № 43(5-6). – P. 625–640.
32. Guarino N. Ontologies and knowledge bases: towards a terminological clafisication / N. Guarino, P. Giaretta // Knowledge Building Knowledge Sharing, ION Press. – 1995. – P. 25–32.
33. Kokar M. M. Ontology-based situation awareness / M. M.Kokar, C. J.Matheusb, K. Baclawski // International Journal of Information Fusion. – 2009. – № 10. – P. 83–98.
34. Литвин В. В. Бази знань інтелектуальних систем підтримки прийняття рішень / В. В.Литвин. – Львів: Видавництво Львівської політехніки, 2011. – 240 с.
35. Gladun A. Ontological Approach to Domain Knowledge Representation for Informational Retrieval in Multiagent Systems / A.Gladun, J.Rogushina // International Jornal "Information Theories & Applications". – Vol.13. – № 4. – 2006. – P.354–362.
36. Knappe R. Perspectives on Ontology-based Querying [Електронний ресурс] / R. Knappe, H. Bulskov, T. Andreasen // International Journal of Intelligent Systems, 2004. – Режим доступу: http://akira.ruc.dk/~knappe/publications/ijis

2004.pdf.

37. Necib C. B. Ontology based Query Processing in Database Management Systems / C.B.Necib, J.Freytag // Proceeding on the 6 th international on ODBASE. – 2003. – P. 37–99.
38. Shamsfard M. The state of the art in ontology learning: A framework for comparison / M. Shamsfard, A. Barforoush // The Knowledge Engineering Review. – 2003. – 18(4). – P. 293–316.
39. Graph-based Query Rewriting for Knowledge Sharing between Peer Ontologies / B.Qin, S.Wang, X.Du, Q.Chen, Q.Wang // Information Sciences. – 2008. – №178(18). – P. 3525–3542.
40. Искусственный интеллект. В 3-х кн. Кн. 2. Модели и методы: справочник / под ред. Д. А. Поспелова. – М.: Радио и связь, 1990. – 304 с.
41. Peterson J. Petri net theory and the modelling of systems / J. Peterson. – N.J.: Prentice-Hall INC, 1981.
42. Нильсон Н. Принципы икусственного интеллекта / Н. Нильсон. – М.: Радио и связь, 1985. – 373 с.
43. Фаулер М. UML в кратком изложени / М. Фаулер, К. Скотт. – М. : Мир. – 1999. – 340 с.
44. Свами М. Графы, сети и алгоритмы / М. Свами, К. Тхуласираман. – М.: Наука, 1984. – 256 с.
45. Литвин В.В. Методи та засоби побудови інтелектуальних систем підтримки прийняття рішень на основі адаптивних онтологій: дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 01.05.03 – математичне і програмне забезпечення обчислювальних машин і систем / Литвин В.В.; Національний університет «Львівська політехніка», Львів, 2012. – 430 с.
46. Ивлев А. А. Основы теории Бойда. Направления развития, применения и реализации / Ивлев А. А. – М., 2008.
47. Новиков Ф.А. Дискретная математика для программистов / Новиков Ф.А. – СПб.: Питер, 2004. – 364 с.

48. Литвин В. В. Методи та засоби інженерії даних та знань / В. В. Литвин – Львів : Магнолія-2006, 2012. – 241 с.
49. Глибовець М. М. Штучний інтелект / М. М. Глибовець, О. В. Олецький. – К.: КМ Академія, 2002. – 366 с.
50. Інтелектуальні системи, базовані на онтологіях / Д. Г. Досин, В. В. Литвин, Ю. В. Нікольський, В. В. Пасічник. – Львів: "Цивілізація", 2009. – 414 с.
51. Cooper G. A Bayesian method for the induction of probabilistic networks from data / G. Cooper, E. Herskovits // Machine Learning. – 1992. – № 9. – P. 309 – 347.
52. Wooldridge M. An Introduction to MultiAgent Systems / M. Wooldridge. – New York: John Wiley &Sons, 2009. – 412 p.
53. Постаногов Д. Ю. К вопросу многоязичности систем инженерии знаний и их приложений / Д. Ю. Постаногов, И. В. Совпель // Искуственный интелект. – 2006. – Вип. 3 – С. 474–479.
54. Miller G. A. WORDNET: A lexical database for English / G. A. Miller // Communications of ACM (11). – 1995. – P. 39–41.
55. Гладун А. Я. Формирование тезауруса предметной области как средства моделирования информационных потребностей пользователя при поиске в Интернете / А. Я. Гладун, Ю. В. Рогушина // Вестник компьютер. и информ. технологий. – 2007. – № 1. – С. 26–33.
56. Gruber T. R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing / T. R. Gruber // International Journal Human-Computer Studies. – 1995. – № 43(5-6). – P. 907–928.
57. National Institutes of Health. Research Portfolio Online Reporting Tools (RePORT), 2010. – Available from: http://projectreporter.nih.gov/ project_info_history.cfm?aid=7941562&icde=2611544.
58. BioInform. Stanford's Mark Musen on the New National Center for Biomedical Ontology, 2005. – Available from: http://www.genomeweb.com/ informatics/stanford-s-mark-musen-new-national-center-biomedical-ontology.

59. Du L. DUMC gets $1.25M for ontology, The Chronicle, 2009. – Available from: http://dukechronicle.com/node/148034.
60. National Science Foundation. The Hymenoptera Ontology: Part of a Transformation in Systematic and Genome Science, 2009. – Available from: http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0850223.
61. United States National Library of Medicine, FAQs: SNOMED CT_ in the UMLS_, 2003. – Available from: http://www.nlm.nih.gov/research/umls/ Snomed/snomed_faq.html.
62. United States National Library of Medicine. SNOMED Clinical Terms To Be Added To UMLS Metathesaurus, 2003. – Available from: http://www.nlm.nih.gov/research/umls/Snomed/snomed_announcement.html.
63. A Web-Based Semantic Focused Crawling Approach / Yongjian Liu, Deng Ma, Jianpeng Sun // IACSE, The 2013 International Conference on Cyber Science and Engineering CyberSE 2013, DEStech Publications. – 2013. – P. 287-293.
64. Bedini I. Automatic Ontology Generation: State of the Art. / Bedini I. , Benjamin Nguyen // Technical report / University of Versailles, 2007. – Available from: http://bivan.free.fr/Janus/Docs/Automatic_Ontology_Generatio n_State_of_Art.pdf.
65. Hearst MA. Automatic acquisition of hyponyms from large text corpora / Hearst MA. // Proceedings of the 12th Conference on Computational Linguistics, aug 23-28 1992. – Nantes,1992. – P.539-545.
66. Hamon T. Detection of synonymy links between terms: experiment and results / Hamon T, Nazarenko A. // Recent Advances in Computational Terminology / Bourigault D., Jacquemin C., L'Homme M-C, editors. – Amsterdam: John Benjamins Publishing Company; 2001. – P. 185–208.
67. Moldovan D.I. An Interactive tool for the rapid development of knowledge bases / Moldovan D.I., Girju R. // International Journal on Artificial Intelligence Tools. – 2001. – Vol.10,№1/2. – P.65-89.

68. Church K.W. Word association norms, mutual information, and lexicography / Church K. W, Hanks P. // Proceedings of 27th Annual Meeting of the ACL. – 1989. – P. 76–83.
69. Smadja F. Retrieving collocations from text: xtract / Smadja F. // Comput Linguist . – 1993. – Vol.19. – P.143–77.
70. Grefenstette G. Automatic thesaurus generation from raw text using knowledge-poor techniques / Grefenstette G. // Ninth Annual Conference of the UW Centre for the New OED and text Research – Making Sense of Words, 1993. –P. 86-101.
71. Hindle D. Noun classification from predicate-argument structures / Hindle D. // Proceedings of 28th ACL. – 1990. – P. 268–275.
72. Geffet M. The distributional inclusion hypotheses and lexical entailment / Geffet M., Dagan I. // Proceedings of the 43rd Annual Meeting of the ACL. –2005. – P. 107–114.
73. Enriching very large ontologies using the WWW / Agirre E., Ansa O., Hovy E., Martнnez D. // Proceedings of the Ontology Learning Workshop. – Berlin, 2000. – Режим доступу: https://www.cs.cmu.edu/~hovy/papers/00ECAI-ontol-enrich.pdf.
74. Faatz A. Ontology enrichment with texts from the WWW / Faatz A., Steinmetz R. // Semantic Web Mining Workshop. – Helsinki, 2002. – Режим доступу: http://www.ece.uc.edu/~mazlack/ECE.716.w08/Semantic.Web.Ontology.Papers /Faatz.ontology-enrichment.pdf.
75. Collier N. Extracting the names of genes and gene products with a Hidden Markov Model / Collier N., Nobata C., Jun-ichi Tsujii // Proceedings of COLING, Sarrebruck, 2000. – Режим доступу: http://citeseerx.ist.psu.edu/ viewdoc/download?doi=10.1.1.668.6423&rep=rep1&type=pdf.
76. Nymble: a high-performance learning name-finder / Bikel D., Miller S., Schwartz L., Wesichedel R. // Proceedings of the Fifth Conference on Applied Natural Language Processing. – 1997. – P. 194–201.

77. Gene name extraction using FlyBase resources / Morgan A., Hirschman L., Yeh A., Colosimo M. // Proceedings of the ACL workshop on Natural language processing in biomedicine. – 2003. – P. 1–8.
78. Effective adaptation of Hidden Markov model-based named entity recognizer for biomedical domain / Shen D., Zhang J., Zhou G., Su J., Tan C.L. // Proceedings of the ACL Workshop on Natural Language Processing in Biomedicine. – 2003. – P. 49–56.
79. Tuning support vector machines for biomedical named entity recognition / Kazama L., Makinoz T., Ohta Y., Tsujiiy J. I. // Proceedings of the ACL workshop on Natural Language Processing in Biomedicine. – 2003. – P. 1–8.
80. Protein name tagging for biomedical annotation in text / Yamamoto K., Kudo T., Konagaya A., Matsumoto Y. // Proceedings of the ACL workshop on Natural Language Processing in Biomedicine. – 2003. – P. 65–72.
81. Chanlekha H, Collier N. A methodology to enhance spatial understanding of disease outbreak events reported in news articles / Chanlekha H, Collier N. // J. Med Informatics. – 2010. – №79. – P.284–96.
82. Hearst M. A. Automatic acquisition of hyponyms from large text corpora / Hearst M. A. // Proceedings of the 12th Conference on Computational Linguistics. – 1992. – P.539-545.
83. Caraballo S. Automatic construction of a hypernym-labeled noun hierarchy from text / Caraballo S. // Proceedings of the 37th Conference on Computational Linguistics, 1999. – Режим доступу: http://www.aclweb.org/anthology/C12-3062.
84. Cederberg S. Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction / Cederberg S., Widdows D. // Proceedings of the 7th Conference on Natural Language Learning. – 2003. – P. 111–118.
85. Fiszman M. Integrating a hypernymic proposition interpreter into a semantic processor for biomedical texts / Fiszman M., Rindflesch T. C., Kilicoglu H. // Proceedings of the Annual Symp. of American Medical Informatics

Association. – 2003. – P. 239–243.

86. Snow R. Learning syntactic patterns for automatic hypernym discovery / Snow R., Jurafsky D., Ng A. Y. // Advances in Neural Information Processing Systems. – 2004. – Режим доступу:http://ai.stanford.edu/~rion/papers/ hypernym_nips05.pdf.
87. Riloff E. Automatically generating extraction patterns from untagged text / Riloff E. // Proceedings of the 13th National Conference on Artificial Intelligence, 1996. – Режим доступу: dl.acm.org/citation.cfm?id=1864542.
88. Evaluation of OntoLearn, a methodology for automatic learning of domain ontologies / Velardi P., Navigli R., Cucchiarelli A., Neri F. // Proceedings of ECAI and EKAW, 2004. – Режим доступу:https://pdfs.semanticscholar.org/ 92ed/eca81f92fa0997f656be0470bc3eb23fe4b8.pdf.
89. Learning taxonomic relations from heterogenerous sources of evidence / Cimiano P., Pivk A., Schmidt-Thieme L., Stabb S. // Proceeding of EKAW, 2004. – Режим доступу: http://olp.dfki.de/ecai04/final-cimiano.pdf.
90. Exploiting technical terminology for knowledge management / Rinaldi F, Yuste E., Schneider G., Hess M., Roussel D. // Proceedings of ECAI and EKAW, 2004. – Режим доступу: http://ceur-ws.org/Vol-121/02.pdf.
91. Morin E. Automatic acquisition and expansion of hypernym links / Morin E., Jacquemin C. // Comp Human. – 2004. –№38. – P.343–362.
92. Bodenreider O. Unsupervised, corpus-based method for extending a biomedical terminology / Bodenreider O., Rindflesch T. C., Burgun A. // Proceedings of the ACL-02 Workshop on Natural Language Processing in the Biomedical Domain.– 2002. – P. 53–60.
93. Ryu P.-M. Measuring the specificity of terms for automatic hierarchy construction / Ryu P.-M., Choi K.-S. // Proceedings of the ACL-SIGLX Workshop on Deep Lexical Acquisition, Ann Arbor, June, 2005. – Michigan, 2005. – P. 63–69.
94. Alfonseca E. An unsupervised method for general named entity recognition and automated concept discovery / Alfonseca E., Manandhar S. // Proceedings of the

1st International Conference on General WordNet, 2002. – Режим доступу: http://alfonseca.org/pubs/generalne.pdf

95. Witschel H. F. Using decision trees and text mining techniques for extending taxonomies / Witschel H. F. // Proceedings of Learning and Extending Lexical Ontologies by using Machine Learning Methods, Workshop at ICML. – Bonn, 2005. – P. 69–79.
96. Berland M. Finding parts in very large corpora / M. Berland, E. Charniak // Proceedings of the 37th Conference on Computational Linguistics. – 1999. – P. 57-64.
97. Sundblad H. Automatic acquisition of hyponyms and meronyms from question corpora / Sundblad H. // Proceedings of the 15th European Conference on Artificial Intelligence. – Lyon, 2002. – P. 99–107.
98. Girju R. Learning semantic constraints for the automatic discovery of part-whole relations / Girju R., Badulescu A., Moldovan D. // Proceedings of the Human Language Technology Conference, 2003. – Режим доступу: http://www.aclweb.org/anthology/N03-1011.
99. Nenadc G. Automatic discovery of term similarities using pattern mining, COLING on COMPUTERM / Nenadc G., Spasic I., Ananiadou S. // 2nd International Workshop on Computational Terminology ACL. –2002. – P. 1–7.
100. Kavalec M. A study on automated relation labeling in ontology learning / Kavalec M, Svatek V. // Ontology learning from text: method, evaluation and applications / Buitelaar P, Cimiano P, Magnini B, editors. – Amsterdam; Berlin; Oxford; Tokyo; Washington, 2005. – P. 44–58.
101. Gulla J. A. Association rules and cosine similarities in ontology relationship learning, enterprise information systems / Gulla J. A., Brasethvik T., Kvarv G. – Berlin; Heidelberg: Springer; 2009. – P. 201–212.
102. Cherfi H. How far association rules and statistical indices help structure terminology? / Cherfi H., Toussaint Y. // Proceedings of the15th ECAI: Workshop on Machine Learning and Natural Language Processing for Ontology Engineering. – Lyon, 2002. – P. 261–272.

103. Bodenreider O. Non-lexical approaches to identifying associative relations in the gene ontology / Bodenreider O., Aubry M., Burgun A. // Pac Symp Biocomput. – 2005. – №2. – P.91–102.
104. Lin D. Automatic retrieval and clustering of similar words / Lin D. // Proceedings of COLING, 1998. – Режим доступу:www.aclweb.org/anthology/ P98-2127.
105. Blaschke C. Automatic ontology construction from the literature / Blaschke C., Valencia A. // Genome Inform. – 2002. – №13. – P.201–213.
106. Aiming to Learn as We Do, a Machine Teaches Itsel. – Режим доступу: fhttp://www.nytimes.com/2010/10/05/science/05compute.html?_r=2.
107. Open Information Extraction from the Web / Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, Oren Etzioni. – Режим доступу: http://web.eecs.umich.edu/~michjc/papers/banko_ijcai07.pdf.
108. Циганов О. В. Основи проектування систем штучного інтелекту. – Одеса: Наука і техніка, 2006. – 196 с.
109. Conceptual Structures: Logical, Linguistic, and Computational Issues, Lecture Notes in AI 1867 / Ganter, Bernhard, & Guy W. Mineau, eds. – Berlin: Springer-Verlag, 2000. – 569 p.
110. Sleator D. Parsing English with a Link Grammar / Daniel, Davy Temperley // Computer Science technical report / Carnegie Mellon University,October 1991.– Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12. 1238&rep=rep1&type=pdf.
111. The PORSCE II Framework: Using AI Planning for Automated Semantic Web Service Composition / Ourania Hatzi, Dimitris Vrakas, Nick Bassiliades, Dimosthenis Anagnostopoulos, Ioannis Vlahavas // The Knowledge Engineering Review / Cambridge University. – 2010. – Vol. 00:0. – P.1–24.

ДОДАТОК А. ПРОГРАМНИЙ КОД ОКРЕМИХ МОДУЛІВ ПІДСИСТЕМИ АВТОМАТИЗОВАНОЇ РОЗБУДОВИ ОНТОЛОГІЇ

```
package crocus;

import java.awt.*;

import javax.swing.*;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.event.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileInputStream;
import java.io.FilenameFilter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.lang.reflect.InvocationTargetException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URI;
import java.net.URL;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
import java.util.*;

import crocus.Is_a;
import crocus.OWLEvaluation;
import crocus.pub.Person;
import crocus.pub.Publication;
import crocus.pub.PublicationDBread;
import edu.stanford.smi.protegex.owl.ProtegeOWL;
import edu.stanford.smi.protegex.owl.jena.JenaOWLModel;
import edu.stanford.smi.protegex.owl.repository.util.XMLBaseExtractor;
import edu.stanford.smi.protegex.owl.repository.Repository;
//import java.util.Properties;
import edu.smu.tspell.wordnet.*;
import crocus.mdp.*;
public class ControlGUI extends JFrame implements ActionListener
{
   private static final int DIM = 1000;
   private JFileChooser fc = null; //hhhhhhhhhhhhhhhhhhh
   public static int processed_row;
   public static int debugging = 1;
   public static String uri = "file:///learnedTest.owl";
   public static String Uri = null;//"V:/learnedTest.owl"; // Working ontology!
   public static String LGP_text = "parser_output.txt";
   public static String language = "en"; // Default
   public static String country = "US";  // Default
   public static final String ENCODING_WIN1251 = "windows-1251";
   public static int userID = 0;  // 0 - Dosyn, 1 - Darevych, 2 - Lytvyn, 3 - Kovalevych
   public static final String ENCODING_UTF8 = "UTF-8";
   public static final String ENCODING_ISO_8859_1 = "ISO-8859-1";
   public static final String ENCODING_ISO_8859_2 = "ISO-8859-2";
   public WordNetDatabase wordNetdatabase = null;
   public Login login = null;
   public String text = new String("");
   public TextField linkNameInput = new TextField(40);
```

```
public TextArea resultsTextArea = new TextArea(10,90);
public JenaOWLModel currentOntology = null;// Онтологія owlModel - scope - клас
public static int R1 = 255, G1 = 500, B1 = 255;
public static Locale currentLocale = null;
Label semLinkNameAsk = null;
JPanel inputPanel = new JPanel();
JPanel controlPanel = new JPanel();
public JPanel resultPanel = new JPanel();
public static ImageIcon image = new ImageIcon("bubbler.png");//"book.png");
public static ResourceBundle messagesBundle = null;
private Connection conn = null;
private Statement statement = null;
private ResultSet myResult = null;
public Properties settings = new Properties();
public LinkedList<Publication> list; // Список статей - результат пошуку за ключовими словами
public int keywordID = -1;

private void close()
{
   try
   {
      if (myResult != null) {myResult.close();}
      if (statement != null) {statement.close();}
      if (conn != null) {conn.close();}
   } catch (Exception e) { }
}

public  String i18n(String key) throws UnsupportedEncodingException
{
   //Перекодування - тимчасове рішення, яке надалі слід замінити на використання текстових файлів ресурсів у форматі 'codepoint' Unicode
   String restored;
```

```
            if (ControlGUI.language == "pl") restored = new
String(ControlGUI.messagesBundle.getString(key).getBytes(ENCODING_ISO_8859_2),ENCODI
NG_UTF8);
            else restored = new
String(ControlGUI.messagesBundle.getString(key).getBytes(ENCODING_ISO_8859_1),ENCODI
NG_WIN1251); //ENCODING_UTF8
            return restored;
        }
        public  class MyFileFilter extends FileFilter
        {

            @Override
            public boolean accept(File f)
            {
                // TODO Auto-generated method stub
                return false;
            }

            @Override
            public String getDescription()
            {
                // TODO Auto-generated method stub
                return null;
            }

        }

        public ControlGUI() throws UnsupportedEncodingException
        {
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          try
           {
              File f = new File(SetupGUI.confFileAddress);
              if (!(f.exists()))
```

```
                {
                    FileDialog fd = new FileDialog(this, i18n("Select_configuration_file_here"));
                    fd.setVisible(true);
                    String confFileAddress = fd.getFile(); //Локальна String-змінна ім'я файла тексту для парсингу
                    String confFilePath = fd.getDirectory(); //Локальна String-змінна ім'я файла тексту для парсингу
                    //JOptionPane.showMessageDialog(null, """+confFilePath+"' selected");
                    SetupGUI.confFileAddress = confFilePath+confFileAddress;
                }
                settings.load(new FileInputStream(SetupGUI.confFileAddress));
            }
            catch (IOException ex)
            {
                ex.printStackTrace();
            }

            String wordNetAddress = settings.getProperty("wordNetAddress");
            File wnFile;
            if (wordNetAddress==null) wordNetAddress = "C:\\wn\\dict\\";
            wnFile = new File(wordNetAddress);
            System.setProperty("wordnet.database.dir",wordNetAddress);//C:\wn\dict\
            wordNetdatabase = WordNetDatabase.getFileInstance();
            currentLocale = new Locale(language, country); // :I18n
            messagesBundle = ResourceBundle.getBundle("MessagesBundle", currentLocale);
            // Від версії 0.2 відрізняється інтернаціоналізацією, а також можливістю коригувати результати навчання.
            // Від версії 0.3 відрізняється можливістю інформаційного пошуку наукових публікацій за ключовими словами.
            setTitle(i18n("CROCUS_project_v")+" 0.4");
            setSize(700, 300);
            //Розраховую положення центру екрану:
            Toolkit tk = Toolkit.getDefaultToolkit();
            Dimension screenSize = tk.getScreenSize();
```

```
Dimension frameSize = getSize();
int w = (screenSize.width - frameSize.width)/2;
int h = (screenSize.height - frameSize.height)/2;
setLocation(w, h);
setLayout(new BorderLayout());
//////////////////////////////////////////////////////Menu//////////////////////////////////////////////////////////////
setIconImage(image.getImage());
// Створення головного меню:
MenuBar menuBar = new MenuBar();
setMenuBar(menuBar);
// Створення елементів меню:
/*1-1*/     Menu fileMenu = new Menu(i18n("Ontology"));
/*2-1*/     Menu editMenu = new Menu(i18n("Processing"));
/*3-1*/     Menu adjustmentsMenu = new Menu(i18n("Preferences"));
/*4-1*/     Menu toolsMenu = new Menu(i18n("Tools"));
/*5-1*/     Menu statisticsMenu = new Menu(i18n("Statistics"));
/*6-1*/     Menu helpMenu = new Menu(i18n("Help"));
/*1-1*/ menuBar.add(fileMenu);
/*2-1*/ menuBar.add(editMenu);
/*3-1*/ menuBar.add(toolsMenu);
/*4-1*/ menuBar.add(adjustmentsMenu);
/*5-1*/ menuBar.add(statisticsMenu);
/*6-1*/ menuBar.add(helpMenu);
/*1-1-1*/ MenuItem newOntology = new MenuItem(i18n("New_ontology"));
/*1-1-2*/ MenuItem openOntology = new MenuItem(i18n("Open_ontology"));
/*1-1-3*/ MenuItem saveOntology = new MenuItem(i18n("Save_ontology"));
/*1-1-4*/ MenuItem verifyOntology = new MenuItem(i18n("Verify"));
/*1-1-5*/ MenuItem openText = new MenuItem(i18n("Learning_text"));
/*1-1-6*/ MenuItem saveText = new MenuItem(i18n("Save_text"));
/*1-1-7*/ MenuItem closeOntology = new MenuItem(i18n("Close_ontology"));
/*1-1-8*/ MenuItem closeProgram = new MenuItem(i18n("Close_work"));
/*1-1-1*/ fileMenu.add(newOntology);
/*1-1-2*/ fileMenu.add(openOntology);
/*1-1-3*/ fileMenu.add(saveOntology);
```

```
/*1-1-4*/ fileMenu.add(verifyOntology);
/*1-1-5*/ fileMenu.add(openText);
/*1-1-6*/ fileMenu.add(saveText);
/*1-1-7*/ fileMenu.add(closeOntology);
/*1-1-8*/ fileMenu.add(closeProgram);
/*2-1-1*/ MenuItem recalculateWeights = new MenuItem(i18n("Reestimate_weight"));
/*2-1-2*/ MenuItem removeConcept = new MenuItem(i18n("Exclude_concept"));
/*2-1-3*/ MenuItem removeRelation = new MenuItem(i18n("Exclude_link"));
/*2-1-4*/ Menu readStatistics = new Menu(i18n("Read_statistics"));
/*2-1-4-1*/ MenuItem readData1 = new MenuItem(i18n("Read_ontology_data"));
/*2-1-4-2*/ MenuItem readData2 = new MenuItem(i18n("Read_user_data"));
/*2-1-4-3*/ MenuItem readData3 = new MenuItem(i18n("Read_sources_data"));
/*2-1-5*/ Menu cleanOntology = new Menu(i18n("Clean_ontology"));
/*2-1-5-1*/ MenuItem selectSemanticLink = new
MenuItem(i18n("Select_semantic_relation"));
/*2-1-5-2*/ MenuItem selectWrongSentence = new
MenuItem(i18n("Select_wrong_learning_sentences"));
/*2-1-6*/ MenuItem addConcept = new MenuItem(i18n("Add_concept"));
/*2-1-7*/ MenuItem addRelation = new MenuItem(i18n("Add_relation"));
/*2-1-8*/ MenuItem findRelation = new MenuItem(i18n("Find_out_relations"));
/*2-1-1*/ editMenu.add(recalculateWeights);
/*2-1-2*/ editMenu.add(removeConcept);
/*2-1-3*/ editMenu.add(removeRelation);
/*2-1-4*/ editMenu.add(readStatistics);
/*2-1-5*/ editMenu.add(cleanOntology);
/*2-1-6*/ editMenu.add(addConcept);
/*2-1-7*/ editMenu.add(addRelation);
/*2-1-8*/ editMenu.add(findRelation);
/*3-1-1*/ MenuItem textEditor = new MenuItem(i18n("Text_processor"));
/*3-1-2*/ MenuItem searchArticles = new MenuItem(i18n("Article_search"));
/*3-1-3*/ MenuItem browsingDatabase = new MenuItem(i18n("Browsing_database"));
/*3-1-4*/ MenuItem recognizeConcepts = new
MenuItem(i18n("Learn_new_concepts"));
/*3-1-5*/ MenuItem buildModel = new MenuItem("Sheduling search task");
```

```
/*3-1-6*/ MenuItem compareTexts = new MenuItem(i18n("Steam"));
/*3-1-7*/ MenuItem relevantOrder = new MenuItem(i18n("Order_by_relevance"));
/*3-1-1*/ toolsMenu.add(textEditor);
/*3-1-2*/ toolsMenu.add(searchArticles);
/*3-1-3*/ toolsMenu.add(browsingDatabase);
/*3-1-4*/ toolsMenu.add(recognizeConcepts);
/*3-1-5*/ toolsMenu.add(buildModel);
/*3-1-6*/ toolsMenu.add(compareTexts);
/*3-1-7*/ toolsMenu.add(relevantOrder);
/*4-2-1*/ MenuItem extResourcesPreferences = new MenuItem(i18n("External_resources"));
/*4-2-2*/  Menu languagePreferences = new Menu(i18n("Language"));
/*4-2-2-1*/ MenuItem langUkrainian = new MenuItem(i18n("Ukrainian"));
/*4-2-2-2*/ MenuItem langEnglish = new MenuItem(i18n("English"));
/*4-2-2-3*/ MenuItem langPolish = new MenuItem(i18n("Polish"));
/*4-2-2-4*/ MenuItem langRussian = new MenuItem(i18n("Russian"));
/*4-2-3*/  Menu viewPreferences = new Menu(i18n("Dialogue_format"));
/*4-2-3-1*/ MenuItem toolButtons = new MenuItem(i18n("Dialogue_buttons"));
/*4-2-3-2*/ MenuItem textFieldDimensions = new MenuItem(i18n("Text_field"));
/*4-2-3-3*/ MenuItem inputDialog = new MenuItem(i18n("Input_field"));
/*4-2-1*/  adjustmentsMenu.add(extResourcesPreferences);
/*4-2-2*/  adjustmentsMenu.add(languagePreferences);
/*4-2-2-1*/ languagePreferences.add(langUkrainian);
/*4-2-2-2*/ languagePreferences.add(langEnglish);
/*4-2-2-3*/ languagePreferences.add(langPolish);
/*4-2-2-4*/ languagePreferences.add(langRussian);
/*4-2-3*/  adjustmentsMenu.add(viewPreferences);
/*4-2-3-1*/ viewPreferences.add(toolButtons);
/*4-2-3-2*/ viewPreferences.add(textFieldDimensions);
/*4-2-3-3*/ viewPreferences.add(inputDialog);
/*5-1-1*/ MenuItem generalStats = new MenuItem(i18n("General_Stats"));
/*5-1-2*/ MenuItem classStats = new MenuItem(i18n("Class_Stats"));
/*5-1-3*/ MenuItem relationsStats = new MenuItem(i18n("Relations_stats"));
/*5-1-4*/ MenuItem individStats = new MenuItem(i18n("Individuals_stats"));
```

```
/*5-1-1*/ statisticsMenu.add(generalStats);
/*5-1-2*/ statisticsMenu.add(classStats);
/*5-1-3*/ statisticsMenu.add(relationsStats);
/*5-1-4*/ statisticsMenu.add(individStats);
/*6-1-1*/ MenuItem tutorial = new MenuItem(i18n("Tutorial"));
/*6-1-2*/ MenuItem help = new MenuItem(i18n("Help"));
/*6-1-3*/ MenuItem aboutProject = new MenuItem(i18n("About_project"));
/*6-1-1*/ helpMenu.add(tutorial);
/*6-1-2*/ helpMenu.add(help);
/*6-1-3*/ helpMenu.add(aboutProject);
MyMenuHandler menuHandler = new MyMenuHandler(this);
/*1-1-1*/ newOntology.addActionListener(menuHandler);
/*1-1-2*/ openOntology.addActionListener(menuHandler);
/*1-1-3*/ closeOntology.addActionListener(menuHandler);
/*1-1-4*/ verifyOntology.addActionListener(menuHandler);
/*1-1-5*/ openText.addActionListener(menuHandler);
/*1-1-6*/ saveText.addActionListener(menuHandler);
/*1-1-7*/ saveOntology.addActionListener(menuHandler);
/*1-1-8*/ closeProgram.addActionListener(menuHandler);
/*2-1-1*/ recalculateWeights.addActionListener(menuHandler);
/*2-1-2*/ removeConcept.addActionListener(menuHandler);
/*2-1-3*/ removeRelation.addActionListener(menuHandler);
/*2-1-4*/ readStatistics.addActionListener(menuHandler);
/*2-1-4-1*/ readStatistics.add(readData1);
/*2-1-4-2*/ readStatistics.add(readData2);
/*2-1-4-2*/ readStatistics.add(readData3);
/*2-1-4-1*/ readData1.addActionListener(menuHandler);
/*2-1-4-2*/ readData2.addActionListener(menuHandler);
/*2-1-4-3*/ readData3.addActionListener(menuHandler);
/*2-1-5*/ cleanOntology.addActionListener(menuHandler);
/*2-1-5-1*/ cleanOntology.add(selectSemanticLink);
/*2-1-5-2*/ cleanOntology.add(selectWrongSentence);
/*2-1-6*/ addConcept.addActionListener(menuHandler);
```

```
/*2-1-7*/ addRelation.addActionListener(menuHandler);
/*2-1-8*/ findRelation.addActionListener(menuHandler);
/*3-1-1*/ textEditor.addActionListener(menuHandler);
/*3-1-2*/ searchArticles.addActionListener(menuHandler);
/*3-1-3*/ browsingDatabase.addActionListener(menuHandler);
/*3-1-4*/ recognizeConcepts.addActionListener(menuHandler);
/*3-1-5*/ buildModel.addActionListener(menuHandler);
/*3-1-6*/ compareTexts.addActionListener(this);
/*3-1-7*/ relevantOrder.addActionListener(menuHandler);
/*4-2-1*/   extResourcesPreferences.addActionListener(menuHandler);
/*4-2-2-1*/ langUkrainian.addActionListener(menuHandler);
/*4-2-2-2*/ langEnglish.addActionListener(menuHandler);
/*4-2-2-3*/ langPolish.addActionListener(menuHandler);
/*4-2-2-4*/ langRussian.addActionListener(menuHandler);
/*4-2-3-1*/ toolButtons.addActionListener(menuHandler);
/*4-2-3-2*/ textFieldDimensions.addActionListener(menuHandler);
/*4-2-3-3*/ inputDialog.addActionListener(menuHandler);
/*5-1-1*/ generalStats.addActionListener(menuHandler);
/*5-1-2*/ classStats.addActionListener(menuHandler);
/*5-1-3*/ relationsStats.addActionListener(menuHandler);
/*5-1-4*/ individStats.addActionListener(menuHandler);
/*6-1-1*/ tutorial.addActionListener(menuHandler);
/*6-1-2*/ help.addActionListener(menuHandler);
/*6-1-3*/ aboutProject.addActionListener(menuHandler);

////////////////////////////////////////////////////////////Menu-end////////////////////////////////////////////////////////////

controlPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
controlPanel.setBackground(Color.gray);
inputPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
inputPanel.setBackground(Color.gray);

resultPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
resultPanel.setBackground(Color.LIGHT_GRAY);
```

```
 //dataPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
 //dataPanel.setBackground(Color.WHITE);

 semLinkNameAsk = new Label(i18n("Specify_relation_kind"));
 inputPanel.add(semLinkNameAsk);
 add(controlPanel,BorderLayout.NORTH);
 add(resultPanel,BorderLayout.CENTER);
 add(inputPanel,BorderLayout.SOUTH);

linkNameInput.setEditable(true);
inputPanel.add(linkNameInput);
linkNameInput.addActionListener(this);

JButton inputButton = new JButton(i18n("Input"));
inputButton.addActionListener(this);
JButton startButton = new JButton(i18n("Process_text"));
startButton.addActionListener(this);
JButton estimateButton = new JButton(i18n("Estimate"));
estimateButton.addActionListener(this);
JButton is_aButton = new JButton(i18n("Is_a_relation"));
is_aButton.addActionListener(this);
JButton teachButton = new JButton(i18n("Learning"));
teachButton.addActionListener(this);
JButton aboutButton = new JButton(i18n("Analyse"));
aboutButton.addActionListener(this);
JButton endButton = new JButton(i18n("Quit"));
endButton.addActionListener(this);
inputPanel.add(inputButton);
controlPanel.add(startButton);
controlPanel.add(teachButton);
controlPanel.add(is_aButton);
controlPanel.add(estimateButton);
controlPanel.add(aboutButton);
```

```
controlPanel.add(endButton);


resultsTextArea.append(i18n("Text_processing_explanation"));
resultsTextArea.append(i18n("Estimation_explanation"));
resultsTextArea.append(i18n("Is_a_explanation"));
resultsTextArea.append(i18n("Learning_explanation"));
resultsTextArea.append(i18n("Analysis_explanation"));
resultsTextArea.append("");
if (!wnFile.exists())resultsTextArea.append("Connection to WordNet not established! Check WordNet installation at "+wordNetAddress+"\n");
else resultsTextArea.append("Connected to WordNet"+"\n");
if (currentOntology==null)resultsTextArea.append("Ontology is undefined jet!"+"\n");
//resultsTextArea.append("Git testing - 2 \n");
resultPanel.add(resultsTextArea);
boolean retry;

String loginName = settings.getProperty("lastUser");
//int sameLogin = JOptionPane.showConfirmDialog(this, "Are you "+loginName+"?","Login", JOptionPane.YES_NO_OPTION);
int sameLogin = JOptionPane.showConfirmDialog(this, i18n("Are_you") + loginName+"?",i18n("logine"), JOptionPane.YES_NO_OPTION);
if (sameLogin == 1)
do
{
  login = new Login();
  if (!login.success) retry = JOptionPane.showConfirmDialog(null, i18n("Wrong_password_Retry"), i18n("Password_check"),0)==0;
  else retry = false;
}
while (retry);
else if (sameLogin == 0) login = new Login(loginName);

if (login.success)
```

```
                {
                   this.userID = login.user.id;
                   Person user = new Person(userID);
                   setVisible(true);
                }
             else System.exit(0);

          }
        /*************************************************************************
**/
          public void actionPerformed(ActionEvent evt)
          {
             String arg = evt.getActionCommand();
             try
             {
                if (arg.equals(i18n("Input")))// messages.getString("Input")
                   {
                      text = linkNameInput.getText();
                      semLinkNameAsk.setText(i18n("Analysis_of_the_relation"));
                   }
                ////////////////////////////////////////////////////////////////
                else if (arg.equals(i18n("Process_text")))
                   {
                   File a = new File("file:///v://");
                   String filename = File.separator+"texts";
                   String filenameURI = new File(filename).toURI().toString();
                   fc = new JFileChooser(new File(filename).toURI().getPath());
                   fc.setCurrentDirectory(a);
                   //FileFilter texts = new FileNameExtensionFilter("txt"); // ext
                   //fc.setFileFilter(new FileNameExtensionFilter("Microsoft Word (*.doc, *.docx)",
"doc", "docx"));
                   fc.setFileFilter(new          FileNameExtensionFilter(i18n("Simple_text_format")+"
(*.txt)","txt"));
                   try
```

```
                {
                    int returnVal = fc.showOpenDialog(null);
                    if (returnVal == JFileChooser.APPROVE_OPTION)
                    {
                        File file = fc.getSelectedFile();
                        String inputFileName = fc.getSelectedFile().getName();

//this.resultsTextArea.append(file.getAbsolutePath()+"*&*&*&*&*&*&*&*&*&*&*\n");

                        //Work with DB:////////////////////////////////////////////////////////////////////////////////////////////
                        try
                        {
                            conn = DriverManager.getConnection("jdbc:mysql://localhost/crocus?" +
"user=root&password=");
                            statement = conn.createStatement();
                            //String souceName = JOptionPane.showInputDialog("Please, input source
name");
                            int result1 = statement.executeUpdate("UPDATE source SET reliability =
reliability+1, comments = 'text from file' WHERE URL='"+file.toURI().getPath()+"';");
                            if (result1 == 0)
                            {
                                int result2 = statement.executeUpdate("INSERT INTO `crocus`.`source`
( `url`, `reliability`, `kind`, `comments`, `reserv`) " +
                                                                    "VALUES (
'"+file.toURI().getPath()+"','"+1+"', 'Text file','This is a test','"+ControlGUI.userID+"');");
                            }
                            close();
                            PublicationDBread newRead = new PublicationDBread(this);
                            this.resultsTextArea.repaint();
                        }
                        catch (SQLException ex)
                        {
                            // handle any errors
```

```
                    System.out.println("SQLException: " + ex.getMessage());
                    System.out.println("SQLState: " + ex.getSQLState());
                    System.out.println("VendorError: " + ex.getErrorCode());
                }
                ////////////////////////////////////////////////////////////////////////////////////////////End work with DB////////

                Parser readText = new Parser(file.getAbsolutePath(), "parser_output.txt", this);
                //PhraseParser readPhrases = new PhraseParser(file.getAbsolutePath(), "phrase_parser_output.txt", this);
                }
                else JOptionPane.showMessageDialog(null, i18n("wrong_selection"));
              }
              catch (Exception e)
              {
                JOptionPane.showMessageDialog(null, i18n("Error_of_an_opening_text_to_analyse")+i18n("wrong_selection"));
                e.printStackTrace();
              }
            }
            ////////////////////////////////////////////////////////////////
            else if (arg.equals(i18n("Estimate")))
            {
              processed_row=0;
              resultsTextArea.append("Active ontology - "+Uri+"\n");
              OWLEvaluation EvaluatedOntology = new OWLEvaluation(Uri, Uri);
              EvaluatedOntology.ShowAllGUI();
              resultsTextArea.append(Results.msg);
              setVisible(true);
            }
            ////////////////////////////////////////////////////////////////
            else if (arg.equals(i18n("Learning")))
            {
```

```
                try
                {
                  processed_row = 0;
                  String dialogText=i18n("Enter_relation_kind");
                  String learnedSemLinkName = JOptionPane.showInputDialog(dialogText,text);// Зчитуємо ім'я зв'язку
                  if (!(learnedSemLinkName == null)) /////////////// Якщо щось введено:
                  {
                    if (learnedSemLinkName.equals(""))  // Якщо нічого не введено, але натиснули "ввести":
                    {
                      JOptionPane.showMessageDialog(this, i18n("Is_a_assumption"));
                      text = "is-a";
                    }
                    else if (text.equals(""))  // Якщо не вводилося раніше
                    {
                      text = learnedSemLinkName;  // Приймаємо введене!
                    }
                    else if (!text.equals(learnedSemLinkName))  // Якщо щось вводилося раніше відмінне від нового:
                    {
                      JOptionPane.showMessageDialog(this, i18n("Was")+" - "+text+", "+i18n("Became")+" - "+learnedSemLinkName);
                      text = learnedSemLinkName;  // Приймаємо введене!
                    }
                    GSL learn = new GSL(LGP_text,currentOntology, text, this);
                    currentOntology=learn.myModel;
                    JOptionPane.showMessageDialog(this, "Current ontology updated!"); //debugging...
                  // Додано в кінці стрічки параметрів 'this' щоб всередині GSL використосувати діалогові вікна
                  learn.GSLDialog(Results.msg, text); // Виводить статистику 'msg' для сем. зв'язку 'text'
                  }
```

else JOptionPane.showMessageDialog(this, "Hi, то ні."); // Якщо нічого не введено - навчання не виконується.

```
    } catch (Exception e)
    {
       System.out.println("GSL Діагноз: "+e.getMessage());
       e.printStackTrace();
    }
    //OWLEvaluation.DumpOWLModel("file:///"+Uri, owlModel);
    if(ControlGUI.debugging>=1)System.out.println("---GSL processed!");
    resultsTextArea.setText("");
    resultsTextArea.setText(Results.msg);
    setVisible(true);
  }
  ////////////////////////////////////////////////////////////////////
  else if (arg.equals(i18n("Is_a_relation")))
  {
    try
    {
       //owlModel = ProtegeOWL.createJenaOWLModelFromURI(URI);
       processed_row=0;
       Is_a LGP_result = new Is_a(LGP_text, currentOntology);
       OWLEvaluation.DumpOWLModel("file:///"+Uri, currentOntology);
        //*Отримання протоколу результатів роботи підпрограми 'resultsIs':
       resultsTextArea.setText(Results.msg);
       setVisible(true);
       if(ControlGUI.debugging>=1)System.out.println("---Is-A links processed!");
    } catch (Exception e)
    {
       System.out.println("Is-a Діагноз: "+e.getMessage());
       e.printStackTrace();
```

```
    }
  }
  ////////////////////////////////////////////////////////////////////
  else if (arg.equals(i18n("Analyse")))
  {
    CognitionGUI ab = new CognitionGUI(currentOntology, this);
  }
  ////////////////////////////////////////////////////////////////////
  else if (arg.equals(i18n("Quit"))) System.exit(0);
  //setBackground(color);
  ////////////////////////////////////////////////////////////////////
  else if (evt.getID()==Event.WINDOW_DESTROY) dispose();
  else if (arg.equals(i18n("Steam")))
  {
    String language = new Locale("en").getDisplayLanguage(Locale.ENGLISH);
    try
    {
      Stemmer  myStemmer = new Stemmer("en");
      String sentence = JOptionPane.showInputDialog(this, "Test the stemmer - write
a word!");

      System.out.println(myStemmer.stem(sentence));
    }
    catch (IllegalArgumentException e)
    {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    catch (IllegalAccessException e)
    {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    catch (InvocationTargetException e)
    {
```

```
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                catch (SecurityException e)
                {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (NoSuchMethodException e)
                {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
            ////////////////////////////////////////////////////////////////////////
        }
        catch (HeadlessException e) // Обробка НС (надзвичайної ситуації) при заміні терміна
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (UnsupportedEncodingException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        //setBackground(color);
        repaint();
    }
    /**
     * @throws UnsupportedEncodingException
**********************************************************************************
     *
     * */
```

```
public static void main(String[] args) throws UnsupportedEncodingException
{
            if (args.length>0) language = args[0];
            if (args.length>1) country = args[1];

   javax.swing.SwingUtilities.invokeLater(new Runnable()
   {
      public void run()
      {
         JFrame mainControlWindow;
         try
         {
          mainControlWindow = new ControlGUI();
         } catch (UnsupportedEncodingException e)
         {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      }
   }                              ); // end of invokeLater()
   //System.exit(0);
}

public void windowClosing(WindowEvent e)
{
   System.exit(0);
}

public void itemStateChanged(ItemEvent ie)
{
   Object item = ie.getItem();
   if (item.equals(R1)||item.equals(G1)||item.equals(B1))
   {
     this.repaint();
```

```
        }
    }
}

package crocus.pub;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.HeadlessException;
import java.awt.Insets;
import java.awt.TextArea;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.UnsupportedEncodingException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedList;
import java.awt.color.*;

import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
import javax.swing.JPanel;
import javax.swing.BorderFactory;
import javax.swing.JScrollPane;

import crocus.ControlGUI;
import edu.stanford.smi.protegex.owl.model.OWLDatatypeProperty;
import edu.stanford.smi.protegex.owl.model.OWLIndividual;
import edu.stanford.smi.protegex.owl.model.OWLObjectProperty;
public class BrowsingGUI extends JFrame implements ActionListener
{
   private ControlGUI mainControlWindow;
   private JButton btnClose = new JButton(i18n("Quit"));
   //private JLabel explainLabel1 = new JLabel("There you can choose keyword:");
   private JPanel upperPanel = new JPanel();
   private JPanel bottomPanel = new JPanel();
   private static int maxLabelsCount = 2000;
   private JPanel centralPanel = new JPanel();
   public JLabel articleName;
   public final JComboBox comboBox = new JComboBox();
   private LinkedList<Keyword> keywordsArray = new LinkedList<Keyword>();
   private LinkedList<String> articlesList = new LinkedList<String>();
   private JPanel articlesListPanel  = new JPanel();
   private JScrollPane scrollView = null;
   private JFrame mainFrame = new JFrame();

   private Connection conn = null;
   private Statement statement = null;
   private ResultSet myResult = null;

   public String i18n(String key) throws UnsupportedEncodingException
   {
      String restored = new
String(ControlGUI.messagesBundle.getString(key).getBytes(ControlGUI.ENCODING_ISO_8859_
1),ControlGUI.ENCODING_WIN1251);
```

```
        return restored;
    }
    private void close()
    {
        try
        {
            if (myResult != null) {myResult.close();}
            if (statement != null) {statement.close();}
            if (conn != null) {conn.close();}
        } catch (Exception e) { }
    }

    public BrowsingGUI(final ControlGUI mcw)  throws UnsupportedEncodingException
    {
        System.out.println();
        System.out.println("Browsing starts...");


        mainFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        this.mainControlWindow = mcw;

        mainFrame.setTitle("Articles that corresponds to selected keywords:");

        bottomPanel.add(btnClose);
        //this.add(upperPanel, BorderLayout.NORTH);
        //this.add(centralPanel, BorderLayout.CENTER);
        mainFrame.add(bottomPanel, BorderLayout.SOUTH);
        btnClose.addActionListener(this);
        //articlesListPanel.setAutoscrolls(true);
        articlesListPanel.setLayout(new BoxLayout(articlesListPanel, BoxLayout.Y_AXIS));

        ////////////////////////////////////////////////////////////////////////// JFrame setup
////////////////////////////////////////////////
        Toolkit tk = Toolkit.getDefaultToolkit();
```

```
Dimension screenSize = tk.getScreenSize();
mainFrame.setSize(new Dimension(1200, 800));
Dimension frameSize = mainFrame.getSize();
int w = (screenSize.width - frameSize.width)/2;
int h = (screenSize.height - frameSize.height)/2+100;
mainFrame.setIconImage(ControlGUI.image.getImage());
mainFrame.setLocation(w, h);
mainFrame.validate();
mainFrame.setLayout(new BorderLayout());
scrollView = new JScrollPane(articlesListPanel);
scrollView.setPreferredSize(new Dimension(900,210));
mainFrame.add(scrollView, BorderLayout.CENTER);
mainFrame.add(upperPanel, BorderLayout.NORTH);
mainFrame.pack();
mainFrame.setVisible(true);

//Work with DB://////////////////////////////////////////////////////////////////////////////////////////////
try
{
  conn = DriverManager.getConnection("jdbc:mysql://localhost/crocus?" +
"user=root&password=");
  // Do something with the Connection conn.
  statement = conn.createStatement();
  myResult = statement.executeQuery("select * from
PUBLICATIONS.KEYWORDS");

  while (myResult.next())
  {
    Keyword key = new Keyword(mcw);
    key.keywordName = myResult.getString("keyword");
    key.keywordID = myResult.getInt("keyword_id");
    key.usedCases = myResult.getInt("used_cases");
    comboBox.addItem(key.keywordName);
    keywordsArray.add(key);
```

```
                }
                close();
            }
            catch (SQLException ex)
            {
                // handle any errors
                System.out.println("Browsing SQLException: " + ex.getMessage());
                System.out.println("Browsing SQLState: " + ex.getSQLState());
                System.out.println("Browsing VendorError: " + ex.getErrorCode());
            }
            ////////////////////////////////////////////////////////////////////////////////////////////End work with DB////////

            upperPanel.add(comboBox);
            comboBox.addActionListener(this);
            this.setVisible(true);

        }
        public void actionPerformed(ActionEvent evt)
        {
            Object selected = comboBox.getSelectedItem();
            System.out.println("Selected Item  = " + selected);
            String command = evt.getActionCommand();
            articlesListPanel.repaint();
            System.out.println("Action Command = " + command);
            if (("comboBoxChanged".equals(command))&&(!(selected==null)))
            {
                String keyWord = selected.toString();
                articlesListPanel.removeAll();
                //Work with DB:////////////////////////////////////////////////////////////////////////////////////////////
                try
                {
```

```
                conn = DriverManager.getConnection("jdbc:mysql://localhost/crocus?" +
"user=root&password=");
                // Do something with the Connection conn.
                statement = conn.createStatement();
                int count = 0;
                JLabel[] paperLabel = new JLabel[maxLabelsCount];
                myResult = statement.executeQuery("select * from
PUBLICATIONS.PUBLICATION WHERE pub_id = ANY (SELECT pub_id from
PUBLICATIONS.keywords_publications_users WHERE keyword_id = ANY (SELECT
keyword_id from PUBLICATIONS.keywords WHERE keyword = '"+keyWord+"'))");
                while (myResult.next())
                {
                   count++;
                   Publication pub = new Publication(mainControlWindow);
                   pub.name = myResult.getString("pub_name");
                   pub.summary = myResult.getString("pub_summary");
                   pub.id = myResult.getInt("pub_id");
                   pub.url = myResult.getURL("pub_URL");
                   pub.authors = pub.getAuthors(pub.id);
                   articlesList.add(pub.name);
                   if (count<maxLabelsCount)
                   {
                      if (pub.url==null) paperLabel[count] = new JLabel (pub.name);
                      else paperLabel[count] = new JLabel ("<html><A
HREF="+pub.url.toString()+">"+pub.name+"</A>"+" - "+pub.authors+" "+"</html>");
                      //paperLabel[count].setToolTipText(pub.summary);
                      articlesListPanel.add(paperLabel[count]);
                   }
                }
                close();
             }
             catch (SQLException ex)
             {
                // handle any errors
```

```
                System.out.println("Browsing action perform SQLException: " +
ex.getMessage());
                System.out.println("Browsing action perform SQLState: " + ex.getSQLState());
                System.out.println("Browsing action perform VendorError: " +
ex.getErrorCode());
            }
            ////////////////////////////////////////////////////////////////////////////////////////End work with
DB////////
            //resultLabel.setText(" ");
            //System.out.println("User has selected an item =" + selected + "= from the combo
box.");//Test output
        }
        try
        {
            if (("comboBoxChanged".equals(command)))
            {
                mainFrame.setVisible(false);
                mainFrame.setVisible(true);
                mainFrame.repaint();
            }
            else if (command.equals(i18n("Quit"))) this.dispose();
        }
        catch (HeadlessException e) // Обробка НС (надзвичайної ситуації) при заміні
терміна
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (UnsupportedEncodingException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        //setBackground(color);
```

```
            repaint();

        }
    }

    package crocus.pub;

    import java.io.UnsupportedEncodingException;
    import java.net.URL;
    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.ResultSet;
    import java.sql.SQLException;
    import java.sql.Statement;
    import java.text.SimpleDateFormat;
    import java.util.Calendar;

    import javax.swing.JLabel;

    import crocus.ControlGUI;
    import crocus.Sub;

    //import crocus.Article;
    public class Publication
    {
        public int id; // Унікальний цифровий ідентифікатор публікації
        public String name = "Unset"; // Назва статті (книги etc.)
        public String authors = "Unset"; // Автори публікації
        public String summary = "Unset"; // Анотація
        public int freeAccess = 0; // ознака безкоштовного доступу до повного тексту публікації
        public PubType type = null; // тип публікації
        public String language; // мова публікації (повного тексту)
        public Person owner; // Особа - власник авторських прав на публікацію
```

```
public float price; // ціна отримання повнотекстової публікації
public URL url; // URL повнотекстової публікації
public String text; // повний текст публікації
public String date; // дата опублікування
public String lastAccess; // дата останнього доступу за допомогою цієї системи до цієї публікації
public int lastUser; // id користувача, який останнім звертався до цієї публікації
public int accessedTimes = 0; // загальна кількість доступів до цієї публікації
public Institution ownerInst; // установа - власник публікації
public String udkIndex; // УДК публікації
public String isbnIndex; // ISBN публікації
public Journal journal; // Журнал, в якому міститься ця публікація
public String volume; // номер тому журналу, в якому міститься ця публікація
public String year; // рік видання тому журналу, в якому міститься ця публікація
public String pages; // сторінки журналу, на яких розташована публікація
private Connection connect = null; // Служб. атрибут для налагодження зв'язку з БД
private Statement statement, statement2 = null; // Служб. атрибут для налагодження зв'язку з БД
private ResultSet result = null; // Служб. атрибут для налагодження зв'язку з БД
private ResultSet authorResult = null; // Служб. атрибут для налагодження зв'язку з БД
private ControlGUI mainControlWindow = null; // Служб. атрибут для перевірки чи публікація вже є в БД
private int existingPubID;

public String ekran(String input)
{
   String text;
   int k = 0;
   StringBuffer myString = new StringBuffer("");
   for (int j = 0; j < input.length(); j++)
   {

      if (input.charAt(j) == '\"')
```

```
            {
               myString.append("\\'");
               //System.out.println("******************************** " + input);

            } else
            {
               myString.append(input.charAt(j));
            }
            k++;
         }
         text = "" + myString;
         return text;
      }

      public String i18n(String key) throws UnsupportedEncodingException
      {
         String restored = new
String(ControlGUI.messagesBundle.getString(key).getBytes(ControlGUI.ENCODING_ISO_8859_
1), ControlGUI.ENCODING_WIN1251);
         return restored;
      }

      private void close()
      {
         try
         {
            if (this.result != null) this.result.close();
            if (this.statement != null) this.statement.close();
            if (this.connect != null) this.connect.close();
         }
         catch (Exception e)
         {
            System.out.println("Problem with closing "+ e);
         }
```

```
        }

        public int getID(ControlGUI mcw)
        {
           int id = -1; // not exist
           mainControlWindow = mcw;
           try
           {
              Connection connect =
DriverManager.getConnection("jdbc:mysql://localhost/publications?" +
"user=crocus&password=sucorc");
              Statement statement = connect.createStatement();
              ResultSet result = null;
              connect = DriverManager.getConnection("jdbc:mysql://localhost/publications?" +
"user=crocus&password=sucorc");
              statement = connect.createStatement();
              result = statement.executeQuery("select pub_id from PUBLICATIONS.publication
WHERE pub_name = '" + ekran(this.name) + "'");
              while (result.next()) id = result.getInt("pub_id");
              //System.out.println("pubID ="+id);
              connect.close();
              statement.close();
              result.close();
           }
           catch (SQLException ex)
           {
              // handle any errors
              System.out.println("SQLException for getID: " + ex.getMessage());
              System.out.println("SQLState for getID: " + ex.getSQLState());
              System.out.println("VendorError for getID: " + ex.getErrorCode());
           }
           return id;
        }
```

```
public Publication(ControlGUI mcw)
{
   Calendar currentDate = Calendar.getInstance();
   SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
   String dateNow = formatter.format(currentDate.getTime());
   // System.out.println("Now the date is :=>  " + dateNow);
   this.name = "Unset";
   this.summary = "Unset";
   this.authors = "Unset";
   this.freeAccess = 0;
   this.type = null;
   this.language = "English";
   this.owner = null;
   this.price = 30;
   this.url = null;
   this.text = "";
   this.date = dateNow;
   this.lastAccess = dateNow;
   this.lastUser = mcw.login.user.id;
   this.accessedTimes = 0;
   this.ownerInst = null;
   this.udkIndex = "";
   this.isbnIndex = "";
   this.journal = null;
   this.volume = "";
   this.year = "";
   this.pages = "";
   //System.out.println("Created a new publication(Initialization without name done)");
}

public Publication(int lastUserId)
{
   Calendar currentDate = Calendar.getInstance();
   SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
```

```
            String dateNow = formatter.format(currentDate.getTime());
            // System.out.println("Now the date is :=>  " + dateNow);
            this.name = "Unset";
            this.summary = "Unset";
            this.authors = "Unset";
            this.freeAccess = 0;
            this.type = null;
            this.language = "English";
            this.owner = null;
            this.price = 30;
            this.url = null;
            this.text = "";
            this.date = dateNow;
            this.lastAccess = dateNow;
            this.lastUser = lastUserId;
            this.accessedTimes = 0;
            this.ownerInst = null;
            this.udkIndex = "";
            this.isbnIndex = "";
            this.journal = null;
            this.volume = "";
            this.year = "";
            this.pages = "";
            //System.out.println("Created a new publication(Initialization without name done and
used temporary lastUserId=0)");
        }

        public Publication(String pubName, ControlGUI mcw)
        {
            Calendar currentDate = Calendar.getInstance();
            SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            String dateNow = formatter.format(currentDate.getTime());
            // System.out.println("Now the date is :=>  " + dateNow);
            this.name = pubName;
```

```
        this.summary = "Unset";
        this.authors = "Unset";
        this.freeAccess = 0;
        this.type = null;
        this.language = "English";
        this.owner = null;
        this.price = 30;
        this.url = null;
        this.text = "";
        this.date = dateNow;
        this.lastAccess = dateNow;
        if (!(mcw == null))
        {
           this.lastUser = mcw.login.user.id;
        } else
        {
           this.lastUser = -1;
        }
        this.accessedTimes = 0;
        this.ownerInst = null;
        this.udkIndex = "";
        this.isbnIndex = "";
        this.journal = null;
        this.volume = "";
        this.year = "";
        this.pages = "";
        //System.out.println("Created a new publication(Initialization with name done)");
    }
public void copy(Publication oldPub, ControlGUI mcw)
{

    this.name = oldPub.name;
    this.summary = oldPub.summary;
    this.authors = oldPub.authors;
```

```
            this.freeAccess = oldPub.freeAccess;
            this.type = oldPub.type;
            this.language = oldPub.language;
            this.owner = oldPub.owner;
            this.price = oldPub.price;
            this.url = oldPub.url;
            this.text = oldPub.text;
            this.date = oldPub.date;
            this.lastAccess = oldPub.lastAccess;
            this.lastUser = oldPub.lastUser;
            this.accessedTimes = oldPub.accessedTimes;
            this.ownerInst = oldPub.ownerInst;
            this.udkIndex = oldPub.udkIndex;
            this.isbnIndex = oldPub.isbnIndex;
            this.journal = oldPub.journal;
            this.volume = oldPub.volume;
            this.year = oldPub.year;
            this.pages = oldPub.pages;
        }
        /**************************************************************************
*************************************************
         * Method savePubKeywordUserLink adds to database record that links publication,
keyword and user
         *
         * @author Dmytro Dosyn
         * @param int keywordId ,
         * @param int userId,
         * @param ControlGUI mcw
         * @return true if success
         */
        public boolean savePubKeywordUserLink(int keywordId, int userId, ControlGUI mcw)
        {
            boolean success = false;
            Calendar currentDate = Calendar.getInstance();
```

```
            SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            String dateNow = formatter.format(currentDate.getTime());
            try
            {
               Connection connect = DriverManager.getConnection("jdbc:mysql://localhost/publications?" + "user=crocus&password=sucorc");
               Statement statement = connect.createStatement();
               ResultSet result = null;
               connect = DriverManager.getConnection("jdbc:mysql://localhost/publications?" + "user=crocus&password=sucorc");
               statement = connect.createStatement();
               String request = "INSERT IGNORE INTO PUBLICATIONS.keywords_publications_users (`pub_id`, `keyword_id`, `user_id`, `relevancy`, `reliability`) VALUES ('"+this.getID(mainControlWindow)+"', '"+keywordId+"', '"+userId+"', '1', '1' )";
               //System.out.println(" = " + request);
               int succ = statement.executeUpdate(request);
               success = (succ>=1);
               statement.close();
               connect.close();
            }
            catch (SQLException ex)
            {
               // handle any errors
               System.out.println("SQLException: " + ex.getMessage());
               System.out.println("SQLState: " + ex.getSQLState());
               System.out.println("VendorError: " + ex.getErrorCode());
            }

            return success;
         }
```

```
/**********************************************************************************************
*********************************************
        * Method exists checks existing of previous versions of this publication
        * Remark: don't check by author there fore don't allow two pubs with same name but with different authors... Must be done!
        * @author Dmytro Dosyn
        * @param mcw
        * @return true if exist
        */
        public boolean exists(ControlGUI mcw)
        {
          boolean yes = false; // already exist
          this.mainControlWindow = mcw;

          try
          {
            Connection connect = DriverManager.getConnection("jdbc:mysql://localhost/publications?" + "user=crocus&password=sucorc");
            Statement statement = connect.createStatement();
            ResultSet result = null;
            connect = DriverManager.getConnection("jdbc:mysql://localhost/publications?user=crocus&password=sucorc");
            statement = connect.createStatement();
            result = statement.executeQuery("select * from PUBLICATIONS.publication");
            Publication existingPub = new Publication(mcw);
            while (result.next())
            {
              existingPub.id = result.getInt("pub_id");
              existingPub.name = result.getString("pub_name");
              if (this.name.equals(existingPub.name))
              {
```

```
                    if (true) // (this.name.equals(existingPub.name))
                    {
                       yes = true;
                       this.existingPubID = existingPub.id;
                    }
                 }
              }
              connect.close();
              statement.close();
              result.close();
           }
           catch (SQLException ex)
           {
              // handle any errors
              System.out.println("SQLException: " + ex.getMessage());
              System.out.println("SQLState: " + ex.getSQLState());
              System.out.println("VendorError: " + ex.getErrorCode());
           }
           return yes;
        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///
        public boolean add2DB (ControlGUI mcw)
        {
           boolean updated = false;
           try
           {
              //Calendar currentDate = Calendar.getInstance();
              //SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
              //String dateNow = formatter.format(currentDate.getTime());
              Connection                                    connect                                =
DriverManager.getConnection("jdbc:mysql://localhost/publications?"                                +
"user=crocus&password=sucorc");
```

```
Statement statement = connect.createStatement();
ResultSet result = null;
String request1 = ""; // перша половина запиту - "куди вносити"
String request2 = ""; // друга половина запиту - "що вносити"
String request = "";  // результуючий запит
if (!this.exists(mcw))
{
   updated = true;
   request1 = request1.concat("INSERT INTO publications.publication (`accessed` "); request2 = request2.concat(") VALUES ('0 '");
   if (!(this.name.equals("Unset")||(this.name==null))) { request1 = request1.concat(", `pub_name`"); request2 = request2.concat(", '" + ekran(this.name) + "'"); }
   if (!(this.summary.equals("Unset")||(this.summary==null))) { request1 = request1.concat(", `pub_summary`"); request2 = request2.concat(", '" + ekran(this.summary) + "'"); }
   if (!(this.freeAccess==0)) { request1 = request1.concat(", `free_access`"); request2 = request2.concat(", '"+this.freeAccess+"'"); }
   if (!(this.type==null)) if (!(this.type.equals("Unset"))) { request1 = request1.concat(", `pub_type`"); request2 = request2.concat(", '"+this.type.id+"'"); }
   if (!(this.language==null)) if (!(this.language.equals("Unset"))) { request1 = request1.concat(", `pub_lang`"); request2 = request2.concat(", '"+this.language+"'"); }
   if (!(this.authors==null)) if (!(this.authors.equals("Unset"))){ request1 = request1.concat(", `pub_owner_persons`"); request2 = request2.concat(", '" + ekran(this.authors) + "'"); }
   if (!(this.price==-1)) { request1 = request1.concat(", `pub_price`"); request2 = request2.concat(", '"+this.price+"'"); }
   if (!(this.url==null)) if (!(this.url.equals("Unset"))) { request1 = request1.concat(", `pub_URL`"); request2 = request2.concat(", '"+this.url.toString()+"'"); }
   if (!(this.text==null)) if (!(this.text.equals("Unset"))) { request1 = request1.concat(", `pub_text`"); request2 = request2.concat(", '" + ekran(this.text) + "'"); }
   if (!(this.date==null)) if (!(this.date.equals("Unset"))) { request1 = request1.concat(", `pub_date`"); request2 = request2.concat(", '"+this.date+"'"); }
   if (!(this.lastAccess==null)) if (!(this.lastAccess.equals("Unset"))) {request1 = request1.concat(", `last_access`"); request2 = request2.concat(", NOW() "); }
```

```
                if (!(this.lastUser==-1))                    { request1 = request1.concat(", `last_user`");     request2 = request2.concat(", '"+this.lastUser+"'"); }
                if (!(this.ownerInst==null)) if (!(this.ownerInst.equals("Unset"))) { request1 = request1.concat(", `pub_owner_inst`");     request2 = request2.concat(", " + this.ownerInst.id + "'"); }
                if (!(this.udkIndex==null)) if (!(this.udkIndex.equals("Unset")))  { request1 = request1.concat(", `udk_index`");     request2 = request2.concat(", '"+this.udkIndex+"'"); }
                if (!(this.isbnIndex==null)) if (!(this.isbnIndex.equals("Unset"))) { request1 = request1.concat(", `isbn_index`");     request2 = request2.concat(", '"+this.isbnIndex+"'"); }
                if (!(this.journal==null)) if (!(this.journal.equals("Unset")))     { request1 = request1.concat(", `journal`");     request2 = request2.concat(", '" + this.journal.id + "'"); }
                if (!(this.volume==null)) if (!(this.volume.equals("Unset"))) { request1 = request1.concat(", `volume`");     request2 = request2.concat(", '"+this.volume+"'"); }
                if (!(this.year==null)) if (!(this.year.equals("Unset")))       { request1 = request1.concat(", `year`");     request2 = request2.concat(", '"+this.year+"'"); }
                if (!(this.pages==null)) if (!(this.pages.equals("Unset")))     { request1 = request1.concat(", `pages`");     request2 = request2.concat(", '"+this.pages+"'"); }
                request = request1.concat(request2 + ")");

            }
            else
            {
                request = request.concat("UPDATE publications.publication SET  `last_access` = now() ");
                if (!(this.name==null)) if (!(this.name.equals("Unset")))           { request = request.concat(", `pub_name` = '" + ekran(this.name) + "'"); }
                if (!(this.summary==null)) if (!(this.summary.equals("Unset"))) { request = request.concat(", `pub_summary` = '" + ekran(this.summary) + "'"); }
                if (!(this.freeAccess==0))                   { request = request.concat(", `free_access` = '" + this.freeAccess+"'"); }
                if (!(this.type==null)) if (!(this.type.equals("Unset")))           { request = request.concat(", `pub_type` = '" + this.type.id+"'"); }
                if (!(this.language==null)) if (!(this.language.equals("Unset"))) { request = request.concat(", `pub_lang` = '" + this.language+"'"); }
```

```
                if (!(this.authors==null)) if (!(this.authors.equals("Unset"))) { request =
request.concat(", `pub_owner_persons` = '" + ekran(this.authors)+"'"); }
                if (!(this.price==-1)) { request = request.concat(",
`pub_price` = '" + this.price+"'"); }
                if (!(this.url==null)) if (!(this.url.equals("Unset"))) { request =
request.concat(", `pub_URL` = '" + this.url.toString()+"'"); }
                if (!(this.text==null)) if (!(this.text.equals("Unset"))) { request =
request.concat(", `pub_text` = '" + ekran(this.text) + "'"); }
                if (!(this.date==null)) if (!(this.date.equals("Unset"))) { request =
request.concat(", `pub_date` = '" + this.date+"'"); }
                if (!(this.lastUser==-1)) { request = request.concat(",
`last_user` = '" + this.lastUser+"'"); }
                if (!(this.ownerInst==null)) if (!(this.ownerInst.equals("Unset"))) { request =
request.concat(", `pub_owner_inst` = '" + this.ownerInst.id+"'"); }
                if (!(this.udkIndex==null)) if (!(this.udkIndex.equals("Unset"))) { request =
request.concat(", `udk_index` = '" + this.udkIndex+"'"); }
                if (!(this.isbnIndex==null)) if (!(this.isbnIndex.equals("Unset"))) { request =
request.concat(", `isbn_index` = '" + this.isbnIndex+"'"); }
                if (!(this.journal==null)) if (!(this.journal.equals("Unset"))) { request =
request.concat(", `journal` = '" + this.journal.id+"'"); }
                if (!(this.volume==null)) if (!(this.volume.equals("Unset"))) { request =
request.concat(", `volume` = '" + this.volume+"'"); }
                if (!(this.year==null)) if (!(this.year.equals("Unset"))) { request =
request.concat(", `year` = '" + this.year+"'"); }
                if (!(this.pages==null)) if (!(this.pages.equals("Unset"))) { request =
request.concat(", `pages` = '" + this.pages+"'"); }
                request = request.concat(" WHERE pub_id = "+this.existingPubID);
                //System.out.println("If this publication was in DB before, the request to DB will
be: " + request);
            }
            int succ = statement.executeUpdate(request);
            if (succ >= 1) this.existingPubID = getID(mainControlWindow);
            else System.out.println("Publication not added, pubID is undefined: " +
this.existingPubID);
```

```
                //if (succ >= 1) System.out.println("Last access date changed for pub Id " +
this.existingPubID);
                result = statement.executeQuery("SELECT @accessed:=accessed FROM
publication WHERE pub_id = " + this.existingPubID + ";");
                int succe = statement.executeUpdate("UPDATE publication SET accessed =
@accessed+1 WHERE pub_id = " + this.existingPubID + ";");
                if (!(this.authors==null)) if (!(this.authors.equals("Unset")))
saveAuthors(this.authors);
                connect.close();
                statement.close();
            }
            catch (SQLException ex)
            {
                System.out.println("SQLException Publication: " + ex.getMessage());
                System.out.println("SQLState: " + ex.getSQLState());
                System.out.println("VendorError: " + ex.getErrorCode());
            }
            return updated;
        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
        public boolean saveAuthors(String setOfAuthors)
        {
            boolean success = false;
            Connection connect = null;
            Statement statement = null;
            ResultSet result = null;
            try
            {
                connect = DriverManager.getConnection("jdbc:mysql://localhost/publications?" +
"user=crocus&password=sucorc");
                statement = connect.createStatement();
                Sub authors = new Sub(setOfAuthors);
                for (int i = 0; i < authors.arrayOfNames.length; i++)
```

```
                {
                  Person newPerson = new Person(authors.arrayOfNames[i].trim()); // Чи тут прізвище вже внесене до БД? Ні!
                  newPerson.Person2DB(newPerson.name);
                  newPerson.id = newPerson.getID();
                  String request = null;
                  request = "INSERT IGNORE INTO PUBLICATIONS.authors_publications (`pub_id`, `author_id`, `count`) VALUES ('"+this.getID(mainControlWindow)+"', '"+newPerson.id+"', '1' )";
                  statement.executeUpdate(request);
                }
                statement.close();
                connect.close();
              }
              catch (SQLException ex)
              {
                System.out.println("SQLException Publication: " + ex.getMessage());
                System.out.println("SQLState: " + ex.getSQLState());
                System.out.println("VendorError: " + ex.getErrorCode());
              }
              return success;
            }

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
            public String getAuthors(int pubID)
            {
              Connection conn = null;
              Statement statement = null;
              ResultSet myResult = null;
              String authors = "";
              Connection connect = null;

              try
              {
```

```
                connect = DriverManager.getConnection("jdbc:mysql://localhost/publications?" +
"user=crocus&password=sucorc");
                statement = connect.createStatement();
                myResult = statement.executeQuery("select name from PUBLICATIONS.PERSON
WHERE person_id = ANY (SELECT author_id from PUBLICATIONS.authors_publications
WHERE pub_id = '"+pubID+"')");
                while (myResult.next()) authors = authors.concat(",
").concat(myResult.getString("name"));
                statement.close();
                connect.close();
                myResult.close();
            }
            catch (SQLException ex)
            {
                System.out.println("SQLException Publication: " + ex.getMessage());
                System.out.println("SQLState: " + ex.getSQLState());
                System.out.println("VendorError: " + ex.getErrorCode());
            }
            return authors;
        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
        @Override
        public String toString()
        {
            StringBuilder st = new StringBuilder();
            st.append("Publication: ");
            st.append("\nid = ");
            st.append(this.id);
            st.append("\nname = ");
            st.append(this.name);
            st.append("\nsummary = ");
            st.append(this.summary);
            st.append("\n PublicationType = ");
```

```
        st.append(this.type);
        st.append("\nlanguage = ");
        st.append(this.language);
        st.append("\nurl = ");
        st.append(this.url);
        st.append("\ntext = ");
        st.append(this.text);
        st.append("\ndate = ");
        st.append(this.date);
        st.append("\nauthors = ");
        st.append(this.authors);
        st.append("\njournal = ");
        if (this.journal != null)
        {
           st.append(this.journal.name);
        }
        st.append("\nlastAccess = ");
        st.append(this.lastAccess);
        st.append("\nvolume = ");
        st.append(this.volume);
        st.append("\npages = ");
        st.append(this.pages);
        st.append("\nyear = ");
        st.append(this.year);

        return st.toString();
    }
}

package crocus;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.Iterator;
```

```
import javax.swing.JOptionPane;

import edu.stanford.smi.protegex.owl.jena.JenaOWLModel;
import edu.stanford.smi.protegex.owl.model.*;
import crocus.OWLEvaluation;

/**==FindSemanticRelations==
 *
 * @author Dmytro Dosyn
 * Проста (спрощена) процедура додавання семантичного зв'язку між поняттями domain і range
 * Процедура запускається з меню "Додати зв'язок"
 * @see MyMenuHandler.java
 * @param  String inputFileName - file with sentence to recognize a relation
 * @param JenaOWLModel inMyModel - ontology to write...
 * @return void
 */
public class AddSemanticRelation
{
   private OWLDatatypeProperty Wo = null;
   private OWLDatatypeProperty Ws = null;
   private OWLDatatypeProperty Lisa = null;
   private ParsedData data;
   private GrammarObjectArr objectsArray;
   private SOLinksArr linksArray.
```

ДОДАТОК Б. ФРАГМЕНТ ПРОГРАМНОГО КОДУ ОНТОЛОГІЇ МАТЕРІАЛОЗНАВСТВА

```
<?xml version="1.0"?>
        <rdf:RDF
          xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
          xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
          xmlns:owlapi="http://www.semanticweb.org/owlapi#"
          xmlns:swrlx="http://swrl.stanford.edu/ontologies/built-ins/3.3/swrlx.owl#"
          xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
          xmlns:query="http://swrl.stanford.edu/ontologies/built-ins/3.3/query.owl#"
          xmlns:temporal="http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl#"
          xmlns:tbox="http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl#"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:owl="http://www.w3.org/2002/07/owl#"
          xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
          xmlns="http://localhost/owl#"
          xmlns:abox="http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl#"
          xmlns:swrl="http://www.w3.org/2003/11/swrl#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
          xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
         xml:base="http://localhost/owl">
         <owl:Ontology rdf:about="">

         <owl:Class rdf:ID="_356.0">
          <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
          >0.0012207031</Ws>
          <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
          >1.0</Lisa>
          <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >false</isCounted>
          <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
>4.8828125E-4</Wo>
<rdfs:subClassOf>
<owl:Class rdf:about="#_3xx.x_series__aluminum-silicon_plus_copper_or_magnesium"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_1146_cold_drawn">
<Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>0.0</Ws>
<Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>1.0</Lisa>
<isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
>false</isCounted>
<rdfs:subClassOf>
<owl:Class rdf:ID="_1146"/>
</rdfs:subClassOf>
<Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>2.4414062E-4</Wo>
</owl:Class>
<owl:Class rdf:ID="C35600_various_conditions">
<Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>0.0</Ws>
<Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>1.0</Lisa>
<Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>2.4414062E-4</Wo>
<isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
>false</isCounted>
<rdfs:subClassOf>
<owl:Class rdf:ID="C35600"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_319.0_T5">
<Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_319.0"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:ID="A356.0_T61">
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:subClassOf rdf:resource="#A356.0"/>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:ID="limit-card_fence">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >9.765625E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="combined_documents"/>
  </rdfs:subClassOf>
```

```
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >лімітно-забірні карти</translate>
 </owl:Class>
 <owl:Class rdf:ID="scientific_journal">
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >науковий журнал</translate>
  <rdfs:subClassOf>
   <owl:Class rdf:ID="journal"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.001953125</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
 </owl:Class>
 <owl:Class rdf:ID="_7075_T76__T7651">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#_7075"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
 </owl:Class>
 <owl:Class rdf:ID="phase_change">
  <rdfs:subClassOf rdf:resource="#natural_process"/>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
  >0.046875</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0625</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="Ti-7Al-4Mo_various_conditions_">
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:subClassOf rdf:resource="#Ti-7Al-4Mo"/>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:about="#non-metal">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0625</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >неметал</rdfs:comment>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="element"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.1484375</Ws>
</owl:Class>
```

```
<owl:Class rdf:ID="_6063_T4">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_6063"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="agreement_on_liability">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.00390625</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#suppliers_of_supply_document"/>
  </rdfs:subClassOf>
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >договір на матеріальну відповідальність</translate>
</owl:Class>
<owl:Class rdf:ID="suggestions">
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >пропозиції</translate>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
```

```
  >false</isCounted>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#personal_document"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.001953125</Wo>
 </owl:Class>
 <owl:Class rdf:ID="Ti-3Al-8V-6Cr-4Mo-4Zr_as_rolled">
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Ti-3Al-8V-6Cr-4Mo-4Zr"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
 </owl:Class>
 <owl:Class rdf:about="#C11300">
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0012207031</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Coppers"/>
  </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:about="#watt">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#derivatives"/>
  </rdfs:subClassOf>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.00390625</Wo>
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >ват</translate>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <owl:disjointWith>
    <owl:Class rdf:about="#weber"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#_5154">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0017089844</Ws>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_5xxx_series__aluminum-magnesium"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="Ti-8Al-1Mo-1V_duplex_annealed">
  <rdfs:subClassOf>
```

```
    <owl:Class rdf:about="#Ti-8Al-1Mo-1V"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
 </owl:Class>
 <owl:Class rdf:ID="C50500_H08">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#C50500"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
 </owl:Class>
 <owl:Class rdf:ID="_440C_Q_T">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_440C"/>
  </rdfs:subClassOf>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
```

```
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
</owl:Class>
<owl:Class rdf:ID="construction_steel">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >9.765625E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#alloyed_steel"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.001953125</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >конструкційна сталь</translate>
</owl:Class>
<owl:Class rdf:ID="_208.0">
  <rdfs:subClassOf rdf:resource="#_2xx.x_series__aluminum-copper"/>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >7.324219E-4</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:about="#_1035">
  <rdfs:subClassOf rdf:resource="#nonresulfurized"/>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
  >0.0012207031</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="_7106_T6351__Obsolete_Alloy_">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_7106"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:ID="_1021_cold_drawn">
  <rdfs:subClassOf rdf:resource="#_1021"/>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
</owl:Class>
<owl:Class rdf:ID="_6063_T5">
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_6063"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AS41A_various_conditions">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="AS41A"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:ID="_1039_hot_rolled">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_1039"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
```

```
</owl:Class>
<owl:Class rdf:ID="_2618">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <rdfs:subClassOf rdf:resource="#_2xxx_series__aluminum-copper"/>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0012207031</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:ID="ASTM_Grade_1_annealed">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ASTM_Grade_1"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_336.0_various_conditions">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_336.0"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:ID="_2018_O">
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_2018"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="_1022_as_rolled">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <rdfs:subClassOf rdf:resource="#_1022"/>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
</owl:Class>
<owl:Class rdf:ID="_308.0_F">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_308.0"/>
  </rdfs:subClassOf>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
```

```
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="_430FSe_annealed_and_cold_worked">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_430FSe"/>
  </rdfs:subClassOf>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
</owl:Class>
<owl:Class rdf:ID="cgi">
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >CGI</translate>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.015625</Wo>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="metric"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
```

```
</owl:Class>
<owl:Class rdf:ID="Youngs_modules">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#factor"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >модуль Юнга</rdfs:comment>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.125</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:ID="C14310_various_conditions">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="C14310"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="_6063_T6">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_6063"/>
```

```
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
</owl:Class>
<owl:Class rdf:ID="C48500_H01">
  <rdfs:subClassOf rdf:resource="#C48500"/>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
</owl:Class>
<owl:Class rdf:ID="_2317_T4">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_2317"/>
  </rdfs:subClassOf>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="uranium">
  <rdfs:subClassOf>
```

```
    <owl:Class rdf:about="#uranium_metals"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.00390625</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >уран</rdfs:comment>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
 </owl:Class>
 <owl:Class rdf:ID="_6063_O">
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_6063"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
 </owl:Class>
 <owl:Class rdf:ID="_5050_H38">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_5050"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
```

```
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
 </owl:Class>
 <owl:Class rdf:ID="A607_Grade_50_Class_1_cold_rolled">
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <rdfs:subClassOf>
   <owl:Class rdf:about="#A607_Grade_50_Class_1"/>
  </rdfs:subClassOf>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
 </owl:Class>
 <owl:Class rdf:ID="_1037">
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf rdf:resource="#nonresulfurized"/>
 </owl:Class>
 <owl:Class rdf:ID="C78200_H01">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
```

```
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:subClassOf rdf:resource="#C78200"/>
 </owl:Class>
 <owl:Class rdf:ID="Ti-6Al-4V_solution_treated_and_overaged">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Ti-6Al-4V"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
 </owl:Class>
 <owl:Class rdf:ID="C48500_H02">
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf rdf:resource="#C48500"/>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
 </owl:Class>
 <owl:Class rdf:ID="_1070__cold_drawn_and_annealed">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
```

```
    <owl:Class rdf:ID="_1070"/>
  </rdfs:subClassOf>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="AZ61A_F">
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="AZ61A"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_1030_as_rolled">
  <rdfs:subClassOf rdf:resource="#_1030"/>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
</owl:Class>
<owl:Class rdf:ID="C23000_annealed">
```

```
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#C23000"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="_7177">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_7xxx_series__aluminum-zinc"/>
  </rdfs:subClassOf>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="C78200_H02">
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:subClassOf rdf:resource="#C78200"/>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
  >1.0</Lisa>
 </owl:Class>
 <owl:Class rdf:ID="_1038">
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.8828125E-4</Wo>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf rdf:resource="#nonresulfurized"/>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0014648438</Ws>
 </owl:Class>
 <owl:Class rdf:ID="C69400_H00">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="C69400"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
 </owl:Class>
 <owl:Class rdf:ID="customer">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#person"/>
  </rdfs:subClassOf>
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >покупець</translate>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
```

```
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.125</Wo>
 </owl:Class>
 <owl:Class rdf:ID="C21000_spring_temper">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#C21000"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
 </owl:Class>
 <owl:Class rdf:ID="action">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#process"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.3046875</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.125</Wo>
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >дія</translate>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
```

```
</owl:Class>
<owl:Class rdf:ID="_1018_cold_drawn">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_1018"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
</owl:Class>
<owl:Class rdf:ID="gas">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="matter"/>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >ɑɜ</rdfs:comment>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.125</Wo>
</owl:Class>
<owl:Class rdf:ID="_5456_H343">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
```

```
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_5456"/>
  </rdfs:subClassOf>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
</owl:Class>
<owl:Class rdf:ID="_6063_T1">
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_6063"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="knowledge">
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0625</Wo>
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >знання</translate>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#plan"/>
  </rdfs:subClassOf>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
```

```
  >false</isCounted>
 </owl:Class>
 <owl:Class rdf:ID="C37700_M30_extruded__cold_drawn">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#C37700"/>
  </rdfs:subClassOf>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</Ws>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
 </owl:Class>
 <owl:Class rdf:ID="Ti-5Al-6Sn-2Zr-1Mo-0.25Si">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#alpha_and_near-alpha_titanium"/>
  </rdfs:subClassOf>
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.001953125</Ws>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >9.765625E-4</Wo>
 </owl:Class>
 <owl:Class rdf:ID="C63800_H10">
  <Lisa rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</Lisa>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.4414062E-4</Wo>
  <Ws rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
  >0.0</Ws>
  <isCounted rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isCounted>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="C63800"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="kind">
  <owl:disjointWith>
    <owl:Class rdf:ID="part"/>
  </owl:disjointWith>
  <Wo rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.03125</Wo>
  <translate rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >вид</translate>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="concept"/>
```

ДОДАТОК В. АКТИ ВПРОВАДЖЕННЯ

ЗАТВЕРДЖУЮ

Директор Золочівського коледжу
Національного університету
«Львівська політехніка»
_____ Я.Є.Жулин

АКТ
про впровадження в навчальний процес результатів
кандидатської дисертаційної роботи
Вовнянки Романа Володимировича

Цей акт складено про те, що результати кандидатської дисертаційної роботи Вовнянки Романа Володимировича впроваджено у навчальний процес підготовки фахівців за спеціальностями 5.05010301 «Розробка програмного забезпечення» та 5.05010101 «Обслуговування програмних систем і комплексів» Золочівського коледжу Національного університету «Львівська політехніка».

Впровадження результатів кандидатської роботи полягає в їхньому використанні при викладанні начальних дисциплін як окремих розділів лекційних курсів, так і в циклах лабораторних робіт.

У лекційному курсі «Основи баз даних та знань» для студентів освітньо-кваліфікаційного рівня «молодший спеціаліст», що навчаються за спеціальністю 5.05010101 «Обслуговування програмних систем і комплексів», використано такі результати:
- системи зберігання знань
- основи побудови інтелектуальних агентів;
- моделі функціонування інтелектуальних агентів планування дій.

При викладанні дисципліни «Об'єктно-орієнтоване програмування» для студентів освітньо-кваліфікаційного рівня «молодший спеціаліст», що навчаються за спеціальністю 5.05010301 «Розробка програмного забезпечення» використано такі результати:
- редактор онтології Protégé;
- практична реалізація ІСППР на основі онтологій.

Заступник директора з навчальної роботи _____ Болюбаш Ю.Я.

Викладач комп'ютерних дисциплін _____ Олійник Б.П.

«16» березня 2016 р.

ЗАТВЕРДЖУЮ
Заступник директора з НДР
ФМІ ім. Г.В. Карпенка НАН України
д.т.н.,ст.н.с. О.І. Яськів
"07" квітня 2016 р.

АКТ
про впровадження результатів дисертаційної роботи
здобувача кафедри «Інформаційні системи та мережі »
Національного університету "Львівська політехніка"
Вовнянки Романа Володимировича

Цей акт підтверджує, що окремі результати кандидатської дисертаційної роботи Р.В. Вовнянки, були використані під час розроблення інформаційних технологій автоматизованого синтезу онтології матеріалознавства.

Впровадження дисертаційних досліджень Р. В. Вовнянки полягало у наступному:

- здійснено аналіз концептуальних графів анотацій англомовних наукових текстів з тематики технічної діагностики фізико-хімічної механіки матеріалів, отриманих на основі опрацювання програмним засобом Link Grammar Parser; здійснено в них пошук відношень, які зустрічаються в науковій літературі з заданої тематики на основі розроблених шаблонів;
- розроблено процедури автоматизованого видобування відношень із природомовних наукових тестів з галузі технічної діагностики на основі побудованих шаблонів;
- розроблено модуль оцінювання релевантності текстового документа до інформаційних потреб користувача системи інформаційного пошуку, який базується на побудові моделі інформаційних потреб у формі оптимального плану інтелектуального агента, оцінці його очікуваної корисності;
- розроблено архітектуру мовно-інформаційної системи для роботи з науковими публікаціями та науковими школами.

Вказані результати кандидатських дисертаційних досліджень, що втілені у складі автоматизованого робочого місця наукового співробітника, дали змогу:

- налаштовувати онтологію матеріалознавства під задачі науковця;
- аналізувати текстову інформацію про нові технології обробки виробів тривалої експлуатації та перспективність їх використання.

Голова комісії:

зав. лаб. системного аналізу
науково-технічної інформації, к.т.н., ст.н.с. Д. Г. Досин

Члени комісії:

Провідний інженер лабораторії системного аналізу науково-технічної інформації В. М. Ковалевич

Інженер !-ї кат. лабораторії системного аналізу науково-технічної інформації А.О. Яценко

"ЗАТВЕРДЖУЮ"
Перший проректор
Митник М.М.

А К Т
про впровадження результатів дисертаційного дослідження
Вовнянки Романа Володимировича

Цей акт підтверджує, що основні теоретичні та практичні результати дисертаційного дослідження використовуються у навчальному процесі кафедр «Інформатики і математичного моделювання» (ММ) та «Комп'ютерних наук» (КН) факультету комп'ютерно-інформаційних систем та програмної інженерії (ФІС) Тернопільського національного технічного університету імені Івана Пулюя. Окремі результати дисертаційного дослідження Р.В.Вовнянки використовуються при викладанні дисциплін «Організація баз даних та знань», «Інтелектуальні системи аналізу консолідованої інформації», «Сховища даних», «Інтелектуальні системи аналізу даних», «Інформаційні технології організації бізнесу».

У розробці, в якій п. Р.В. Вовнянка виконував роботи з проектування та розроблення програмно-алгоритмічних моделей та макетів, відображено та реалізовано:

– моделі функціонування інтелектуальних агентів планування дій, ядром баз знань яких є онтології предметних областей, що дозволило використати вже існуючі бази знань під час побудови ефективних прикладних систем;

– алгоритми пошуку даних за запитом користувача та валідації отриманих результатів з метою перевірки якості функціонування інтелектуальних систем.

Зазначені результати дисертаційного дослідження дають змогу:

– підвищити ефективність функціонування інтелектуальних агентів, достовірність пропонованих системою рішень;

– своєчасно виявляти та відхиляти неякісні переходи між станами в процесі функціонування інтелектуального агента.

Окрім того у навчальний процес впроваджено розроблені методи планування дій інтелектуальних агентів на основі онтології предметної області.

Декан ФІС доц. Мацюк О.В.

Зав.кафедри КН проф. Приймак М.В.
Зав.кафедри ММ доц. Михайлишин М.С.

"20" квітня 2016р.

"Затверджую"
Проректор з наукової роботи
Національного університету
"Львівська політехніка"
_____ проф. Н.І. Чухрай
"10" 2016 р.

А К Т

про використання результатів дисертаційної роботи "Методи та засоби планування дій спеціалізованих інтелектуальних агентів на основі онтологічного підходу" здобувача кафедри інформаційних систем та мереж Вовнянки Романа Володимировича, представленої на здобуття наукового ступеня кандидата технічних наук, при виконанні науково-дослідної роботи Національного університету "Львівська політехніка"

Ми, що нижче підписалися, начальник НДЧ, к.т.н., доц. Жук Л.В. та члени комісії: завідувач відділу науково-організаційного супроводу наукових досліджень, к.т.н. Лазько Г.В., заступник начальника планово-фінансового відділу Чулой Т.М. та завідувач кафедри інформаційних систем та мереж, д.т.н., проф. Литвин В.В. цим актом підтверджуємо, що результати дисертаційного дослідження здобувача кафедри інформаційних систем та мереж Вовнянка Р.В. використано під час виконання науково-дослідної роботи Національного університету "Львівська політехніка": "Розроблення інтелектуальних розподілених систем на основі онтологічного підходу з метою інтеграції інформаційних ресурсів" (2014-2016 рр., номер державної реєстрації 0115U004228).

В рамках науково-дослідної роботи Вовнянка Р.В. розробив математичне забезпечення планування дій спеціалізованих інтелектуальних агентів з використанням онтологічного підходу (побудова простору станів та пошук в ньому шляху переходу); розробив метод автоматизованого наповнення онтологій та на основі нього реалізував відповідне програмне забезпечення; розробив архітектуру програмного комплексу планування дій спеціалізованих інтелектуальних агентів; провів апробацію отриманих результатів шляхом розроблення та впровадження прикладних спеціалізованих інтелектуальних агентів планування дій.

Використання розроблених Вовнянка Р.В. методів функціонування інтелектуальних агентів планування дій на основі онтологічного підходу дало змого підвищити ефективність функціонування таких агентів. Зокрема, врахування міри довіри до джерела інформації дає змогу зменшити простір станів, в яких здійснюється пошук шляху розв'язку задачі планування; застосування процедур автоматизованої розбудови онтологій природномовними текстами суттєво

розширює сферу використання таких онтологій та зменшує затрати на їх реалізацію. Розроблене програмне забезпечення функціонування інтелектуальних агентів планування дій, яке ґрунтується на побудованих моделях, методах та алгоритмах, дало можливість реалізувати окремі компоненти та функціональні модулі інтелектуальних агентів планування дій на основі онтологій, ядром баз знань яких є онтологія. Розроблено ІА в галузі діагностики та експлуатації виробів тривалої експлуатації, центральною компонентою якого є онтологія матеріалознавства.

Начальник НДЧ,
канд. техн. наук, доц. Л.В. Жук

Члени комісії:
зав. відділу НОСНД,
канд. техн. наук Г.В. Лазько

Заст. нач. ПФВ Т.М. Чулой

Зав. кафедри
інформаційних систем та мереж,
д. т. н., проф. В.В. Литвин