НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___"_____2016 р.

Дипломна робота

першого (бакалаврського) **рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: Бібліотека наукових публікацій з використанням RDF -сховищ

Виконав (-ла): студент (-ка) _4_ курсу, групи _ДА-21
(шифр групи)

Галушко Марія Олегівна
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викладач, к.т.н. Булах Б. В
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічний розділ проф. д.е.н. Семенченко Н.В. _ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____ст. викладач Британов О.А._____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2016 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту
Галушко Марії Олегівні
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Бібліотека наукових публікацій з використанням RDF - сховищ
керівник проекту (роботи) Булах Богдан Вікторович, к.т.н., ст. викладач,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 травня 2016 р. № 50-ст

2. Строк подання студентом проекту (роботи) _____

3. Вихідні дані до проекту (роботи) _____

Бібліотека наукових публікацій з використанням RDF-сховищ, яка має:

- Містити частину публікацій кафедри Системного проектування;
- Здійснювати пошук звичайних запитів пошуку;
- Здійснювати пошук релевантної інформації по контексту.

Використання програмного забезпечення Protege.
Форма реалізації – база знань з інтерфейсом для користувача - сайт.
Середовище розробки – Protege, мова запитів в базу – SPARQL, JavaScript підключення бази знань до сайту та виконання запитів, HTML5, CSS3 – для розмітки та стилізації сайту .

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Ознайомитись із базовими поняттями онтологій, огляд існуючих рішень.
2. Розробити базу знань наукових публікацій.
3. Тестування прототипу бази знань для бібліотеки публікацій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Фрагмент онтології бази знань для бібліотеки публікацій – плакат
2. Архітектура додатку для бібліотеки публікацій – плакат.
3. Процес роботи з базою публікацій – плакат.

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічна частина	Семенченко Н. В., доцент		

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2016	
2	Збір інформації	15.02.2016	
3	Дослідження предметної області та існуючих рішень.	28.02.2016	

* Консультантом не може бути зазначено керівника дипломного проекту (роботи).

4	Вивчення підходів до побудови баз знань за допомогою RDF-сховищ	10.03.2016	
5	Розробка ієрархії класів онтології, яка б дозволяла здійснювати «розумний» пошук	15.03.2016	
6	Створення необхідних властивостей, зв’язків між об’єктами	25.03.2016	
7	Заповнення бази знань науковими публікаціями кафедри	25.04.2016	
8	Тестування прототипу бази знань для бібліотеки публікацій.	30.04.2016	
9	Оформлення дипломної роботи	31.05.2016	
10	Отримання допуску до захисту та подача роботи в ДЕК	10.06.2016	

Студент _____ Галушко М. О.
(підпис) (ініціали, прізвище)

Керівник роботи _____ Булах Б. В.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

бакалаврської дипломної роботи Галушко Марії Олегівни на тему «Бібліотека наукових публікацій з використанням RDF-сховищ»

Метою дипломної роботи є виявлення ефективних способів та засобів використання RDF-сховищ у прикладних системах, таких як бібліотеки публікацій. Об’єктом дослідження є аналіз способів і засобів використання RDF-сховищ у прикладних системах та процес створення бази знань наукових публікацій, яка б відповідала поставленим цілям роботи. Було виконано огляд існуючих аналогів готового продукту дипломної роботи – баз знань наукових публікацій та зроблено висновки щодо їх недоліків та переваг при виконанні роботи. Створено бібліотеку наукових публікацій з можливістю виконувати, пошук по контексту(«розумний пошук»). Вдосконаливши продукт можна використовувати на сайті кафедри для зручного пошуку за наявними публікаціями.

Загальний обсяг роботи: 79 сторінки, 28 ілюстрацій, 6 таблиць та 20 бібліографічних найменувань.

Ключові слова: RDF, OWL, онтологія, база знань, семантичний веб, SPARQL, середовище опису ресурсів, триплет.

АННОТАЦИЯ

бакалаврской дипломной работы Галушко Марии Олеговны на тему «Библиотека научных публикаций с использованием RDF-хранилищ»

Целью дипломной работы является выявление эффективных способов и средств использования RDF-хранилищ в прикладных системах, таких как библиотеки публикаций. Объектом исследования является анализ способов и средств использования RDF-хранилищ в прикладных системах и процесс создания базы знаний научных публикаций, соответствующей поставленным целям работы. Было выполнено обзор существующих аналогов готового продукта дипломной работы - баз знаний научных публикаций и сделаны выводы относительно их недостатков и преимуществ при выполнении работы. Создана библиотека научных публикаций с возможностью выполнять поиск по контексту («умный поиск»). Усовершенствовав продукт можно использовать на сайте кафедры для удобного поиска по имеющимся публикациям.

Общий объем работы: 79 страниц, 28 иллюстраций, 6 таблиц и 20 библиографических наименований.

Ключевые слова: RDF, OWL, онтология, база знаний, семантический веб, SPARQL, среда описания ресурсов, триплет.

ABSTRACT

to the bachelor thesis by Halushko Maria Olegivna on “Publication library based on RDF store”

The goal of the thesis is identifying effective ways and means of using the RDF-storage in application systems, such as the Publication library. The object of the study is to analyze the ways and means of using the RDF-storage in application systems and the process of creating a knowledge base of scientific publications that are fit to work goals. It was carried out a review of existing analogs of the finished product of the thesis - knowledge bases of scientific publications and conclusions about their strengths and weaknesses in the performance of work. A library of scientific publications with the ability to search by context ("smart search"). By optimizing the product can be used on the website of the Department for easy search available publications.

Includes 79 pages, 28 figures, 6 tables and 20 bibliographic items.

Keywords: RDF, OWL, ontology, base of knowledge, semantic web, SPARQL, resource description environment, triple.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Відношення	Смисловий зв'язок між двома сутностями.
RDF	(Resource Description Framework) технологія семантичної павутини, яка включає в себе середовище опису ресурсів, визначає загальну архітектуру метаданих і призначена для забезпечення сумісності метаданих за допомогою спільної семантики, структури та синтаксису.
OWL	Мова опису онтологій для семантичної павутини.
SPARQL	Мова запитів до даних, представлених по моделі RDF, а також протокол для передачі цих запитів і відповідей на них.
W3C	(англ. World Wide Web Consortium, W3C) Консорціум Всесвітньої павутини
Семантична павутина	Нова концепція розвитку Всесвітньої павутини і мережі Інтернет, яка створена і впроваджується Консорціумом Всесвітньої павутини (англ. World Wide Web Consortium, W3C)
Онтологія	Концептуальна схема, яка формалізує деяку галузь знань. Інформація як ієрархічна структура *понять*.
Клас	Поняття, які описує онтологія.
Підклас	Більш конкретне поняття, ніж клас.
Слот, роль, властивість	Властивості та атрибути поняття.
Фацет	Обмеження, які накладаються на слоти, обмеження ролей.
База знань	Онтологія разом з набором індивідуальних примірників класів.
Триплет	Твердження про ресурс. Трійка: суб'єкт, предикат, об'єкт.

ВСТУП

Розвиток наукоємних галузей людської діяльності в сучасному суспільстві супроводжується зростанням ролі комп'ютерних технологій. Зараз значно збільшується потік інформації, з'явилася необхідність пошуку нових способів її зберігання, подання, формалізації і систематизації, а також автоматичної обробки. Пошук інформації є причиною досліджень багатьох учених сьогодення. Створення семантичного вебу, штучного інтелекту чи нейронних мереж, – все це задля пошуку інформації. Тому дана тема є актуальною.

При вирішенні задач, в яких дані можуть мати довільні зв'язки, виникає непередбачувана кількість зв’язків в запитах, тому для вирішення такого плану задач, зараз найбільшого розповсюдження набули RDF-сховища. Вони базуються на стандартах комітету W3C для мови опису графів (RDF) та для обробки графових даних (SPARQL). RDF значить “середовище опису ресурсів”. Це модель даних, що представляє дані простими триплетами суб’єкт – предикат – об’єкт. Триплет також відомий як «оператор» і є базовим «явищем», або його ще можна назвати стверджуваним блоком знань в RDF. Декілька операторів комбінуються разом узгоджуючись таким чином: суб’єкт, об’єкт як вузол і предикат як ребро. Таким чином, виникає структурна мережа, також відома як RDF-граф і має вигляд вузол-ребро-вузол.

RDF визначає загальну архітектуру метаданих і призначена для забезпечення сумісності метаданих за допомогою спільної семантики, структури та синтаксису. Технологія семантичної мережі передбачає розширення можливостей інтернету завдяки механізмам надання інформації чітко визначеного значення, яке дозволяє ефективно використовувати її у спільній роботі як комп'ютерів, так і людей.

Онтологія – це загальноприйнята і загальнодоступна концептуалізація певної області знань (світу, середовища), яка містить базис для моделювання цієї

області знань і визначає шляхи для взаємодії між агентами, які використовують знання з цієї області, і, нарешті, включає домовленості про представлення теоретичних основ даної області знань.

Метою дипломної роботи є виявлення ефективних способів та засобів використання RDF-сховищ у прикладних системах, таких як бібліотеки публікацій. Поставлена мета вимагає вирішення наступних наукових задач:

1) аналіз існуючих баз знань наукових публікацій, які використовують RDF-сховища;
2) аналіз способів і засобів використання RDF-сховищ у прикладних системах;
3) створення бази знань наукових публікацій використовуючи проведений аналіз.

Об’єктом дослідження є аналіз способів і засобів використання RDF-сховищ у прикладних системах та процес створення бази знань наукових публікацій, яка б відповідала поставленим цілям роботи. Предметом дослідження є база знань.

Досягнення поставленої мети реалізовано з використанням програмного засобу Protege, для проектування бази знань та заповнення її даними та мови SPARQL, за допомогою якої можна робити запити в базу, тим самим здійснювати пошук по ній, HTML5, CSS3 та JavaScript для створення сайту для відображення результатів.

Наукова новизна дипломної роботи полягає в тому, що було створено базу знань, яка може здійснювати пошук не лише за бібліометричними даними публікації, ключовими словами як багато існуючих прототипів, але й за контекстом самої публікації, що спрощує пошук необхідної інформації.

Потенційні застосування та практична цінність результатів дипломної роботи:

1) База знань може використовуватись на сайті кафедри, щоб спростити пошук існуючих публікацій;

2) Сформульовано основні концепти, на які потрібно звернути увагу при проектуванні бази знань, яка могла б здійснювати пошук потрібної інформації

3) Розширивши базу знань, можна використовувати на вищому рівні, ніж публікації кафедри, наприклад, інституту, університету і тд.

1. БАЗОВІ ПОНЯТТЯ. ОНТОЛОГІЇ

1.1 Онтології

Онтологія – це загальноприйнята і загальнодоступна концептуалізація певної області знань (світу, середовища), яка містить базис для моделювання цієї області знань і визначає шляхи для взаємодії між агентами, які використовують знання з цієї області, і, нарешті, включає домовленості про представлення теоретичних основ даної області знань. В рамках комп’ютерних наук онтологія – це формальна назва і визначення типів, властивостей і взаємовідносин суб’єктів, що дійсно, або принципово, існують в обраному контексті (предметній області). Таким чином, вони є практичним застосуванням філософського поняття онтології, за допомогою таксономії.

Онтології виділяють змінні, необхідні для деякої множини обчислень і встановлення відносин між ними. В полі штучного інтелекту, семантичних мереж, інженерних систем, розробки програмного забезпечення, біометричної інформатики, бібліотекознавства, підприємництва та інформаційної архітектури, створюються онтології для обмеження складності і організації інформації. Онтологія може бути використана для вирішення прикладних проблем.

Історично онтології виникли з гілки філософії, відомої як метафізика, яка має справу з природою реальності – того, що існує. Це фундаментальне вчення займається аналізом різних типів або форми існування, часто з особливою увагою до відносин між особливостями і універсалів, між внутрішніми і зовнішніми властивостями, а також між сутністю та існуванням. Традиційна мета онтологічного дослідження, полягає в поділі світу (класифікації), для відкриття фундаментальних категорій або видів, в які об'єкти в світі потрапляють. У другій половині 20–го століття філософи широко обговорювали можливі методи або

підходи до побудови онтологій, без фактичного будівництва будь–яких більш–менш складних онтологій. Навпаки, програмісти будували кілька великих і надійних онтології, таких як WordNet і Cyc, з порівняно невеликим обговореннями, на основі яких вони були побудовані. З середини 1970–х років дослідники в галузі штучного інтелекту (ШІ) визнали, що захоплення знання є ключем до створення великих і потужних систем ШІ. Дослідники стверджували, що вони могли б створювати нові онтології як обчислювальні моделі, які дозволяють виконувати певні види автоматизованих досліджень. У 1980–х роках спільнота штучного інтелекту почала використовувати термін онтології для позначення теорії модельованого світу і компонента системи знань [1]. Схему використання онтологій у інформаційній системі зображено на (рис. 1.1).

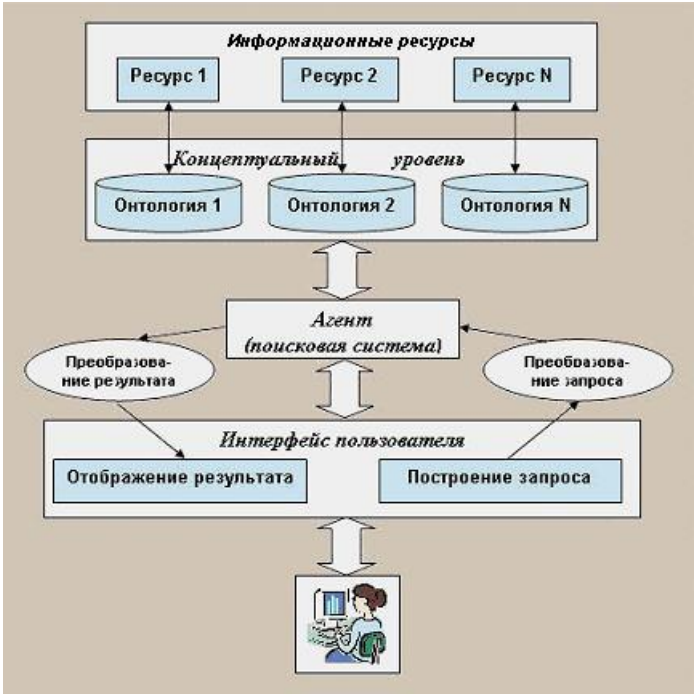

Рисунок 1.1 – Схема використання онтологій у інформаційній системі [4]

Сучасні онтології структурно подібні незалежно від мови, якою вони виражені. Як уже згадувалося вище, більшість онтології описують осіб (зразки), класи (концепції), атрибути і відносини.

1.2 Основні компоненти онтологій

Загальні компоненти онтологій включають в себе:

- фізичні особи: випадки або об'єкти (основні або "базові" об'єкти);
- класи: набори, колекції, поняття, класи з програмування, типи об'єктів, або види речей;
- стосунки: способи, в яких класи та окремі особи можуть бути пов'язані один з одним;
- атрибути: аспекти, властивості, особливості, характеристики або параметри об'єктів, які клас може мати;
- функціональні терміни: складні структури, утворені з певних відносин, які можуть бути використані замість окремого терміну;
- обмеження: формально зазначені описи того, що повинно бути вірно для того, щоб твердження було прийняте в якості вхідного;
- правила: твердження у вигляді, if–then (попередник–наслідок) пропозиції, що описують логічні висновки, які можна зробити з твердження;
- аксіоми: твердження (у тому числі правила) в логічній формі, що разом складають загальну теорію, що онтологія описує галузі застосування. Це визначення відрізняється від «аксіом» в породженій граматиці і формальної логіки. У цих дисциплінах, аксіоми включають тільки твердження, що є апріорними знаннями. Поняття «аксіом» також включають теорію, отриману з аксіоматичних тверджень;
- події: зміна атрибутів або відносин.

Такий тип систем в значній мірі залежить від контекстної обізнаності та методів прогнозування контексту, які використовують відповідну інформацію користувача.

1.3 Задачі онтологій та їх спільне використання

Задачі онтології:

1. Спільне використання людьми або програмними агентами загального розуміння структури інформації;
2. Можливість повторного використання знань в предметній області;
3. Зробити допущення в предметній області явними;
4. Відділення знань в предметній області від оперативних знань;
5. Аналіз знань в предметній області.

Спільне використання (людьми або програмними агентами) загального розуміння структури інформації – основна мета розробки онтологій. Наприклад, нехай кілька різних веб-сайтів містять інформацію з медицини або надають інформацію про платні медичні послуги, які оплачуються через Інтернет. Якщо ці веб-сайти спільно використовують і публікують одну і ту ж базову онтологію термінів, якими вони всі користуються, то:

- Комп'ютерні агенти можуть отримувати інформацію з цих ,різних сайтів і накопичувати її.
- Агенти можуть використовувати накопичену інформацію для відповідей на запити користувачів або як вхідні дані для інших додатків.

1.4 Структура онтології

Після того як було визначено, що таке онтологія, її основні задачі перейдемо до не менш важливого питання, з чого складається онтологія. У загальному вигляді структура онтології являє собою набір елементів чотирьох категорій:

- поняття;
- відносини;
- аксіоми;
- окремі екземпляри.

Поняття розглядаються як концептуалізації класу всіх представників якоїсь сутності або явища (наприклад, Людина, Їжа). Класи (або поняття) є загальними категоріями, які можуть бути впорядковані ієрархічно. Кожен клас описує групу індивідуальних сутностей, які об'єднані на підставі наявності загальних властивостей.

Поняття можуть бути пов'язані різного роду відносинами (наприклад, Приналежність, Вага), які пов'язують воєдино класи і описують їх. Найпоширенішим типом відносин, що використовується у всіх онтологіях, є ставлення категоризації, тобто віднесення до певної категорії. Цей тип відносин має ряд інших назв [3], що зустрічається в різних дослідженнях:

- таксономічне відношення;
- ставлення IS-A;
- клас - підклас;
- лінгвістика: гіпонім - гіперонім;
- відношення; відношення a-kind-of.

Аксіоми задають умови відповідності категорій і відносин, вони виражають очевидні твердження, що зв'язують поняття і відносини. Під аксіомою можна розуміти твердження, що вводиться в онтологію у готовому вигляді, з якого можуть бути виведені інші твердження. Вони дозволяють висловити ту інформацію, яка не може бути відображена в онтології за допомогою побудови ієрархії понять і установки різних відносин між поняттями.

Як приклад аксіоми можна навести таке висловлювання: «Якщо Х смертний, то Х колись помре». Аксіоми дозволяють надалі здійснювати умовиводи в рамках онтології. Вони можуть постачати дослідників інформацією про правила, що

дозволяють автоматично додавати інформацію. Аксіоми можуть також являти собою обмеження, що накладаються на будь-які відносини, які роблять можливим виведення умовиводів [8]. Наведемо кілька прикладів таких обмежень. Понятійні обмеження вказують на те, який тип понять може виражати дане відношення (наприклад, властивість Колір може виражатися тільки поняттями категорії Колір). Прикладом числових обмежень є твердження того, що для Людини кількість біологічних батьків рівна 2. Кількість і ступінь деталізації аксіом зазвичай залежать від типу онтології, про що буде докладніше сказано далі.

Поряд із зазначеними елементами онтології в неї також входять так звані «екземпляри». У літературі вони можуть виступати також під назвами:

- конкретні екземпляри;
- інстанції;
- індивідуальні екземпляри.

Примірники – це окремі представники класу сутностей або явищ, тобто конкретні елементи будь-якої категорії (наприклад, екземпляром класу Хлопець буде Петро).

Складові онтології підкоряються своєрідній ієрархії. На нижньому рівні цієї ієрархії знаходяться екземпляри, конкретні індивіди, вище йдуть поняття, тобто категорії. На рівень вище розташовуються відносини між цими поняттями, а узагальнюючої і сполучною є ступінь правил або аксіом.

Як згадано в роботах [3] і [5], «терміну« онтологія »задовольняє широкий спектр структур, що представляють знання про ту чи іншу предметну область». Так до онтології можна віднести ряд структур, що відрізняються різним ступенем формалізованості:

- глосарій;
- проста таксономія;
- тезаурус (таксономія з термінами);
- понятійна структура з довільним набором відносин;

- повністю аксіоматизована теорія.

Однак в цих структурах не завжди представлені всі складові онтології, які описувалися в даному розділі.

1.5 Класифікація онтологій

Онтології сильно розрізняються по ряду параметрів, і дослідники виділяють різні підстави для їх класифікації. Так Е. Хові [14] говорить, що онтології розрізняються залежно від набору елементів, що містяться в них, а також типів відносин. Він виділяє так звані «термінологічні онтології» і «справжні онтології». Під першими Е.Хові [14] розуміє онтології, що включають суті, явища, властивості, зв'язки предметної області та структурні відносини, що їх об'єднують. «Справжні» ж онтології включають в себе визначальні відносини і відносини додаткової інформації. Поряд з цим в них входять аксіоми, що визначають взаємозалежності між відносинами і поняттями.

Е.Хові вибудовує детальну класифікацію різних характеристик онтологій. Він згадує, що основними параметрами можуть бути: форма (те, як формується онтологія), зміст, а також засоби використання онтології.

Можна згадати про те, що існує розбиття онтологій за кількістю і якістю понять, що включаються в них.

1.1.1 Верхні онтології

Верхня онтологія (фундаментальна онтологія) – це модель спільних об'єктів, які звичайно застосовуються в широкому діапазоні онтологій ПО. Вона, як правило, використовує ядро-словник, що містить терміни і пов'язані описи об'єктів, які вони використовують в різних відповідних наборах предметних областей. Онтології верхньої зони зазвичай налічують приблизно 100-500 концептів. У них включені найбільш абстрактні категорії, що володіють властивістю універсальності. Вони є базовим розбиттям на категорії. Зазвичай

вони будуються теоретиками, філософами. Перевагою таких онтологій є можливість їх використання у багатьох областях і навіть у багатьох мовах. Для даного роду онтологій характерний обмежений набір узагальнених відносин, які можна віднести до базових (таких як родовидові відносини, відносини частину-ціле і асоціативні відносини). У таких онтологіях типовими на верхньому рівні розбиття є такі поняття, як:

- сутність;
- явище;
- об'єкт;
- процес;
- роль.

Є кілька стандартизованих верхніх онтології, що доступні для використання, в тому числі BFO, метод Боро, Dublin Core, GFO, OpenCyc/ResearchCyc, SUMO, Unified Foundational Ontology (UFO), DOLCE. WordNet [3].

1.1.2 Середні онтології

Елементів зазвичай у цьому типі вже більше (500 - 100000 концептів). Вони представляють світ в цілому. Складність полягає в тому, що для даного виду онтологій потрібно виводити дуже велику кількість аксіом. Зазвичай виходом є використання методів автоматизованого виведення аксіом з уже існуючих онтологій. Побудовою онтологій цього рівня найчастіше займаються когнітологи і лінгвісти.

1.1.3 Низькі онтології

Зазвичай вони налічують близько 200 - 2000 концептів. Вони описують конкретні предметні області з їх специфікою. При цьому коло вирішуваних завдань і питань, на які онтологія відповідає, обмежене обраною областю. Для даного типу онтологій характерна наявність відносин, специфічних для конкретної

області [9]. Для цього типу онтологій можлива побудова великої кількості аксіом і правил. У більшості випадків цей тип онтологій будується експертами галузі знання або за їх сприяння. У зв'язку з великою специфікою кожної окремої предметної онтології її повторне використання найчастіше можливо тільки в рамках предметної області.

1.1.4 Онтології предметної області (ПО)

Онтологія предметної області (або предметно–орієнтована онтологія) являє концепцію, яка відносяться до частини світу. Конкретні значення термінів, що застосовуються в цій області надаються онтологією. Наприклад слово «карта» має багато різних значень. Онтологія з предметною областю хвороба буде моделювати симптоми, в той час як онтологія продукти буде моделювати "овочі", "фрукти", і т.д.

Так онтології (ПО) представляють концепції в дуже специфічних і часто еклектичних способах, що часто несумісні. Оскільки системи, які покладаються на онтології предметних областей розширювані, їх часто необхідно об'єднати в більш загальні онтології [6]. Це становить проблему для дизайнера онтології. Різні онтології в тій же ПО виникають через різні мови, різного напряму використання онтологій і різних уявлень про ПО (на основі культурних традицій, освіти, ідеології і т.д.) [2].

В даний час об'єднання онтологій, які не розроблені із загального фундаменту онтології в основному є ручним процесом і, отже, забирають багато часу. Онтологій, що використовують той же фундамент онтології, щоб забезпечити набір основних елементів, за допомогою яких вказуються значення елементів онтології можуть бути об'єднані автоматично. Є дослідження узагальнених методів для злиття онтологій, але ця область досліджень є значною мірою теоретичною.

1.1.5 Гібридні онтології

Gellish онтологія – приклад поєднання верхньої та онтології ПО. Онтології зазвичай кодується з використанням мов онтології. Мова опису онтологій — формальна мова, що використовується для кодування онтології. Існує кілька подібних мов :

- OWL — Ontology Web Language, стандарт W3C, мова для семантичних тверджень, розроблена як розширення RDF і RDFS;
- KIF (Knowledge Interchange Format або формат обміну знаннями) — заснований на S–виразах синтаксис для логіки;
- Common Logic — спадкоємець KIF (стандартизований — ISO/IEC 24707:2007). CycL — онтологічна мова, що використовується в проекті Cyc, заснована на численні предикатів із деякими розширеннями вищого порядку.
- DAML+OIL (FIPA)
- DOGMA (Developing Ontology–Grounded Methods and Applications — розробка методів на основі онтологій і додатків).
- Для роботи з мовами онтологій існує декілька видів технологій: редактори онтологій (для створення онтологій), DBMS онтологій (для зберігання й звертання до онтології) і сховища онтологій (для роботи з декількома онтологіями) [10].

Поряд з описаним розподілом всі онтології можуть бути розділені на:

- Поверхневі – онтології, що будуються на поверхневій семантиці, вони визначають поняття через значення слів. Однак тут виникає проблема, яка кількість смислів виділяти для кожного слова.
- Глибинні ж онтології використовують глибинну семантику.

Можна розрізняти «відсильні (також звані офф-лайн) онтології» (reference ontologies) і «здійснювані (спільно використовувані, он-лайн) онтології» (implemented (shareable) ontologies). Нескладна структура, що описує, наприклад,

лексику, може міститися он-лайн, в той час як мудрі теорії, що визначають значення термінів з лексики, можуть перебувати оф-лайн.

Онтології можуть бути також розділені на одномовні і багатомовні. Вже існує ряд онтологій, орієнтованих на представлення знань на декількох мовах, наприклад, EuroWordNet, MikroKosmos і деякі інші. Складність створення таких онтологій зазвичай полягає в тому, що можлива наявність відмінностей в понятійних систем різних мов.

В рамках роботи [16] також виділяється особливий тип онтологій - лексичні (або лінгвістичні). Відмітною властивістю таких онтологій є «фіксація в одному ресурсі понять (слів) разом з їх мовними властивостями». Основним джерелом понять в онтологіях даного типу є значення мовних одиниць. До лінгвістичних онтологій автори [3] відносять WordNet, MikroKosmos, Sensus, РуТез і інші. Коло завдань, що вирішуються такими онтологіями, тісно взаємопов'язаний з обробкою природної мови [15].

Створювана онтологія відноситься до нижньої онтології і описує конкретну предметну область. У зв'язку з цим для моєї онтології були характерні деякі відносини, які є специфічними для даної області, але є ймовірність, що онтологія може використовуватися й інших областях.

1.6 Огляд існуючих рішень

На початку XXI століття виникла могутня пошукова система наукової інформації, яка здійснює пошук по всьому Інтернету. Це Google Scholar — вільна доступна пошукова система, яка індексує повний текст наукових публікацій всіх форматів і дисциплін [7]. Вона охоплює також вихідні ресурси, які є в базах даних ISI I SCOPUS, а також менш значимі ресурси, патенти, матеріали, розміщені в електронних архівах відкритого доступу і онлайнових наукових журналів та ін. Перевагами цієї пошукової системи – розроблені Google спеціальні алгоритми

розрахунку цитованості документів, а також можливості пошуку наукових документів на сайтах наукових організацій (в 2004 році іспанська кібернетична лабораторія використовувала Google Scholar при розробці вебометричного рейтингу університетів і науково-досліджуваних центрів світу).

Google Scholar дозволяє вести пошук з урахуванням різних логічних операторів, включаючи розширений пошук з точною фразою, із обмеженням в часі та області знань та ін.

На відміну від іншої потужної пошукової системи наукової інформації – Scirus, Google Scholar включає багато російськомовних та україномовних наукових документів.

До конкурентів Google Scholar можна віднести тільки вищезгаданий Scirus. Не дивлячись, що розробники називають його всеохоплюючим, він не веде пошук по всьому Інтернету, так як це робить Google Scholar, а має визначений, хоча і дуже обширний перелік наукових інформаційних ресурсів: близько 140 млн документів із сайтів із доменами edu, 40 млн – org, 39 млн – gov, 38 млн – com, 23 млн – ac.uk, та ін [4]. Цей пошуковий інструмент охоплює близько 410 млн наукових документів – наукові статті, книги, курси лекцій, журнали, патенти, та ін.

Sirius запущений за підтримки видання “Elsevier”. Як і Google Scholar, в процесі пошуку наукових статей відфільтровує ненаукові статі і шукає виключно, ті що рецензуються. Обидві пошукові машини йдуть глибше, ніж перші два рівня веб-сайтів, тому знаходять набагато більше релевантної інформації.

На відміну від Google Scholar, в Scirus передбачена більш дрібна градація пошуку ключових слів в різних місцях метаданих і всього документу(назва статті, назва журналу, ім’я автора, ключові слова), пошук за видом документу(книги, статі, реферати, патенти, дисертації і тд.), за форматами(pdf, HTML, ppt, word).

При розширеному пошуку автоматично видається інформація про кількість журнальних джерел, бажаних веб-ресурсах, інших веб-ресурсах, а також типах

файлів. Від цих кількісних розподілень через гіперпосилання можна відразу переглядати знайдені наукові документи. Так само як і в Google Scholar, реалізовані логічні оператори і можливість пошуку з точною фразою. У Scirus, на відміну від Google Scholar, де використовується 7 широких областей, використовується 9, хоча і ця градація не на користь Scirus, Так як логічнішою є класифікація Google Scholar.

DBpedia — співтовариство, зусилля якого спрямовані на те, щоб витягати структуровану інформацію з Wikipedia і робити цю інформацію доступною в Web. DBpedia дозволяє створювати уточнені запити до Wikipedia, і прив'язує інші набори даних у Web до даних Wikipedia.

Проект DBpedia підсилює гігантське джерело знання Wikipedia, розпаковуючии структуровану інформацію від Wikipedia і тим самим робить цю інформацію доступною в Web на основі GNU Free Documentation License. База знань DBpedia описує більш ніж 3.4 мільйонів об'єктів, вміщує 274 мільйонів уривків інформації (RDF triples).

База знань DBpedia має декілька переваг над існуючими базами знань:

- охоплює багато областей;
- відображає реальну колективну згоду;
- автоматично еволюціонує як і Wikipedia, і вона справді багатомовна.

DBpedia дозволяє вам створювати складні запити для Wikipedia, наприклад «Надати мені всі міста в Нью-Джерсі з більш ніж 10,000 жителів» або «Надати мені всіх італійських музикантів від 18-го століття» [10].

Також варто згадати про такі пошукові системи як, Google Books та Google Patents. Перша охоплює більше 10 тис. видавців і авторів, які публікують книги більше, ніж на 35 мовах; дозволяє здійснювати пошук за мовами, назвами, авторами, предметним областям, видавцям та за інтервалами часу.

Google Patents охоплює більше 7 мільйонів патентів на винайдення і торгові марки, які входять у базу даних патентного підприємства США. У цієї системи широкі можливості для пошуку (дати, автори, організації і тд.).

Існують вузькоспеціалізовані комерційні онлайнові БД наукової інформації. Десять найбільших видань наукової періодики формують бази даних ISI, мають власні онлайн-платформи зі своїми пошуковими інтерфейсами: “”Elsevier-платформа Science Direct, “Springer” – плітформа Springer Link, “Wiley” – платформа Wiley InterScience, і тд.

Найбільш відома і крупна – платформа ScienceDirect, яка охоплює більше 2,5 тис. журналів, 9,5 млн повнотекстних статей; щорічний приріст – близько 500 тис. статей.

Також варто згадати такі міжнародні галузеві БД наукової інформації: PubMedCentral, Public Library of Science, Medline, Inspec, Econlit, EBSCO Hostweb і тд, а також міжнародну БД дисертацій та тезисів – ProQuest Dissertations and Theses.

1.7 Висновок

У цьому розділі було проведено загальний огляд, що таке онтологія, база знань. Також було визначено основні компоненти онтології, наведено сфери застосування онтологій.

Проведено класифікацію онтологій і описано кожну з них. Також було окреслено задачі онтологій, які полягають у наступному:

1. Спільне використання людьми або програмними агентами загального розуміння структури інформації;
2. Можливість повторного використання знань в предметній області;
3. Зробити допущення в предметній області явними;
4. Відділення знань в предметній області від оперативних знань;

5. Аналіз знань в предметній області.

Було окреслено структуру онтологій, яка має складатись із таких компонентів:

- поняття;
- відносини;
- аксіоми ;
- окремі екземпляри.

У розділі проведено огляд існуючих прототипів бібліотек наукових публікацій, що використовують RDF-сховища, такі як DBpedia, Google Scholar та Scirus. І було зроблено висновок, що вони всі містять великий об’єм даних, є зручними, та мають звісно багато переваг, але все ж було вирішено створити нову базу знань, так як для конкретної задачі, а саме створення бібліотеки наукових публікацій, там було багато зайвих класів, властивостей тощо. А так як заповнювати базу знань потрібно вручну, то адміністратору цієї бази знань доведеться робити багато зайвої роботи. На додаток, пошук у всіх існуючих проаналізованих системах зазвичай виконувався лише за ключовими словами, у роботі було ж реалізовано саме пошук за контекстом. У наступному розділі буде детальніший опис безпосереднього створення продукту – бази знань публікацій.

4. РОЗРОБКА БАЗИ ЗНАНЬ НАУКОВИХ ПУБЛІКАЦІЙ

2.1 Основні етапи створення бази знань

Наведемо основні етапи, які потрібні для творення бази знань.

1. Визначення класів онтології:

Наприклад, для моєї онтології: Publication, Thesis, Article, Book, TechReport, Description, Branch, Method, Technology, Library, Problems і тд.

2. Створення ієрархії класів.
3. Визначення слотів (властивостей).

Наприклад, «використовує», «належить», «узагальнює», «вирішує»(проблему) і тд.

4. Заповнення онтології екземплярами та встановлення зв’язків між об’єктами.

Після цього ми можемо створити базу знань, визначивши окремі екземпляри цих класів, ввівши в певний слот значення і додаткові обмеження для слота.

А тепер опишемо детальніше процес створення бази знань.

2.1.1 Визначення області і масштабу онтології

Почнемо розробку онтології з визначення її області і масштабу. Тобто, відповімо на кілька основних питань:

1. Яку область буде охоплювати онтологія?

2. Для чого буде використовуватись онтологія?

3. На які типи питань повинна давати відповіді інформація в онтології?

4. Хто буде використовувати і підтримувати онтологію?

Відповіді на ці питання можуть змінитися під час процесу проектування онтології, але в будь-який заданий момент часу вони допомагають обмежити масштаб моделі.

Область нашої онтології – наукові публікації кафедри Системного проектування. Онтологія буде використовуватись для зручного пошуку за контекстом статті, пропонувати публікації, які мають відношення до того, що шукає користувач.

Один із способів визначити масштаб онтології - це накидати список питань, на які повинна відповісти база знань, заснована на онтології, тобто питання для перевірки компетентності. Це можуть бути такого типу питання: Чи містить онтологія досить інформації для відповіді на ці типи питань? Чи потрібна для відповідей особливий рівень деталізації або подання певної області? Ці питання для перевірки компетентності є всього лише формальними і не повинні бути вичерпнимир [5].

Для бібліотеки публікацій, для якої має бути можливість виконувати пошук по контексту, це можуть бути такі питання:

1. Які відомості мають бути про публікацію?
2. Які технології вона використовує?
3. Які проблеми вирішує?
4. Якою є об'єкт і предмет дослідження?
5. Яка новизна?

Судячи із цих питань у онтології будуть не лише бібліографічні відомості, а й за змістом статті, а саме: технології, предмет, об'єкт, на що посилається, що узагальнює і тп.

2.1.2 Доцільність використання існуючих онтологій

Повторне використання існуючих онтологій може бути необхідним, якщо нашій системі потрібно взаємодіяти з іншими додатками, які вже увійшли в окремі онтології або контрольовані словники. Багато онтології вже доступні в електронному вигляді і можуть бути імпортовані е використовуване Вами середовище проектування онтологій. Формалізм онтології часто не має значення,

тому що багато систем уявлення знань можуть імпортувати і експортувати онтології. Навіть якщо система подання знань не може працювати безпосередньо з окремим формалізмом, завдання перекладу онтології з одного формалізму в іншій зазвичай не є складною.

В літературі і всесвітній павутині є бібліотеки повторно використовуваних онтологій. Наприклад, ми можемо використовувати бібліотеку онтологій Ontolingua або бібліотеку онтологій DAML. Існує також ряд загальнодоступних комерційних онтологій (наприклад, UNSPSC, RosettaNet, DMOZ).

Було проаналізовано існуючі онтології публікацій і зроблено висновок, що онтології, яка повністю задовільняла потреби немає, але була онтологія, яка мітить об’єкт «Публікація» і відповідні властивості притаманні їй, тому можна видозмінити існуючу онтологію, для наших потреб вилучивши непотрібні класи, властивості, зв’язки і додати нові.

2.1.3 Визначення термінів, класів і ієрархіїкласів онтології

Потрібно скласти список всіх термінів, про які ми хотіли б сказати що-небудь або які хотіли б пояснити користувачеві. Які терміни ми б хотіли розглянути? Які властивості мають ці терміни? Що б ми хотіли сказати про ці термінах? Наприклад, в число важливих термінів, пов'язаних з публікаціями, входять назва статті, тип публікації, автор, проблеми, вирішення, мета, використані технології і т.д. На початку важливо отримати повний список термінів, не турбуючись про перетин понять, які вони представляють, про відносини між термінами, про можливі властивості понять або про те, чим є поняття - класами або слотами [11].

Опишемо декілька основних підходів створення онтології:

Низхідна розробка. Починається з визначення найзагальніших понять предметної області з подальшою конкретизацією понять. Наприклад, ми можемо почати з створення класів для загальних понять Публікація і Контекст. Потім ми

конкретизуємо клас Технології, створюючи його підкласи: Засоби, Бібліотеки, Програмне забезпечення.

Висхідна розробка. Починається з визначення найконкретніших класів, листя ієрархії, з наступним групуванням цих класів в більш загальні поняття [13]. Наприклад, спочатку ми визначаємо класи для публікацій Бакалаврські Тези, Магістерські Тези. Потім ми створюємо загальний надклас для двох цих класів – Тези, який, в свою чергу є підкласом Публікація.

Комбінована розробка - це поєднання низхідного та висхідного підходів. Спочатку визначаємо більш помітні поняття, а потім відповідним чином узагальнюємо і обмежуємо їх. Можна почати з кількох понять вищого рівня, таких як Публікація, і декількох конкретних понять, таких як Магістерські Тези, PhD Тези. Потім ми можемо співвіднести їх з поняттям середнього рівня, таким як Тези.

Вибір підходу значною мірою залежить від особистого погляду на предметну область.

2.1.4 Визначення властивостей слотів

Зрозуміло, що класи не дають достатньої інформації для відповіді на питання перевірки компетентності, які були визначені раніше. Після визначення деякої кількості класів ми повинні описати внутрішню структуру понять.

Вибрано класи зі списку термінів, ну і більшість з тих, що залишились будуть, ймовірно, будуть властивостями цих класів. Ці терміни включають, наприклад, автор, дата, адреса, назва, кількість сторінок, номер видання тощо.

Для кожної властивості зі списку ми повинні визначити, який клас вона описує. Ці властивості стануть слотами, прив'язаними до класів. Таким чином, у класу Публікація будуть наступні слоти: автор, назва, рік. А у класу Тези буде слот видання, журнал, і тд.

Взагалі, в онтології слотами можуть стати кілька типів властивостей об'єктів:

- «Внутрішні» властивості, такі як автор;

- «Зовнішні» властивості, такі як використання технологій, вирішення проблеми;

- Частини, якщо об'єкт має структуру; вони можуть бути як фізичними, так і абстрактними «частинами»;

- Відносини з іншими індивідуальна концептами; це відносини між окремими членами класу та іншими елементами.

Всі підкласи класу успадковують слот цього класу. Це є важлива властивість, так як саме вона допомагає здійснити саме розумний пошук.

2.1.5 Визначення фацетів слотів

Слоти можуть мати різні фацети, які описують тип значення, дозволені значення, число значень (потужність) та інші властивості значень, які може приймати слот [12].

Потужність слота визначає, скільки значень може мати слот. У деяких системах розрізняються тільки одинична потужність (можливо тільки одне значення) і множинна потужність (можливо будь-яке число значень).

Деякі системи дозволяють визначити мінімальну і максимальну потужність для того, щоб більш точно описати кількість значень слота. Мінімальна потужність N означає, що слот повинен мати не менше N значень. Наприклад, слот автор класу Публікація має мінімальну потужність 1: кожна публікація має, як мінімум, одного автора. Максимальна потужність M означає, що слот може мати максимум M значень. Максимальна потужність слота назва для статті дорівнює 1. Іноді корисно встановити максимальну потужність в 0. Ця установка буде означати, що для певного підкласу слот не може мати значень.

Фацет типу значення описує, які типи значень можна ввести в слот. Ось список найбільш загальних типів значень:

- Строка - найпростіший тип значення, який використовується в таких слотах, як назва.

- Число (іноді використовуються більш конкретні типи значень: Float (Число з плаваючою комою) і Integer (Ціле число)) описує слоти числовими значеннями.

- Булеві слоти - це прості прапори «так - ні».

- Нумеровані слоти визначають список конкретних дозволених значень слота. У Protege-2000 нумеровані слоти мають тип Символ.

- Слоти-екземпляри дозволяють визначити відносини між індивідуальна концептами.

2.1.6 Створення екземплярів

Останнім етапом – є створення окремих екземплярів класів у ієрархії. Для визначення окремого екземпляра класу потрібно (1) вибрати клас, (2) створити окремий екземпляр цього класу і (3) ввести значення слотів. Наприклад, ми можемо створити окремий екземпляр «Grid_-_system_design_and_optimization_of_engineering_solutions» для подання певного типу Article. «Grid_-_system_design_and_optimization_of_engineering_solutions» – це екземпляр класу Article . У цього примірника визначені наступні значення слотів (рис. 2.2, приклад додавання одного із слотів на рис. 2.1):

- Назва: Grid – system design and optimization of engineering solutions

- Автор: Петренко А.І., Булах Б.В.

- Дата: 2011

- Адреса: Україна, Київ

- Видання: System Analysis and Information Technology: 13th International Scientific Conference "SAIT-2011”

- Мета – дослідження.

- Предмет – програмне забезпечення грід.
- Проблема: масштабованість, інтероперабельність.
- Рішення: сервіс, алгоритм.

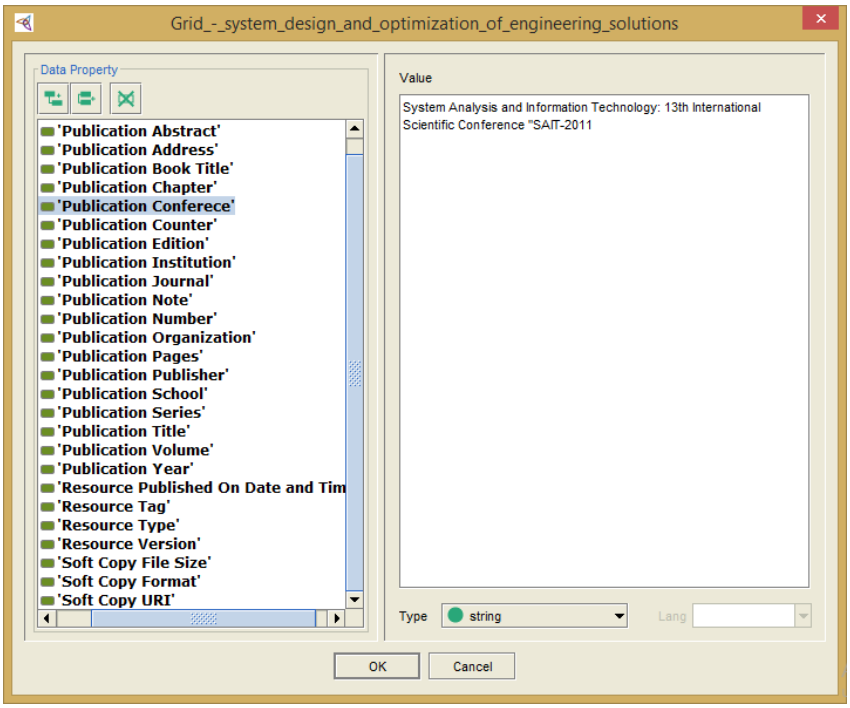

Рисунок 2.1 – Додавання властивостей екземплярів

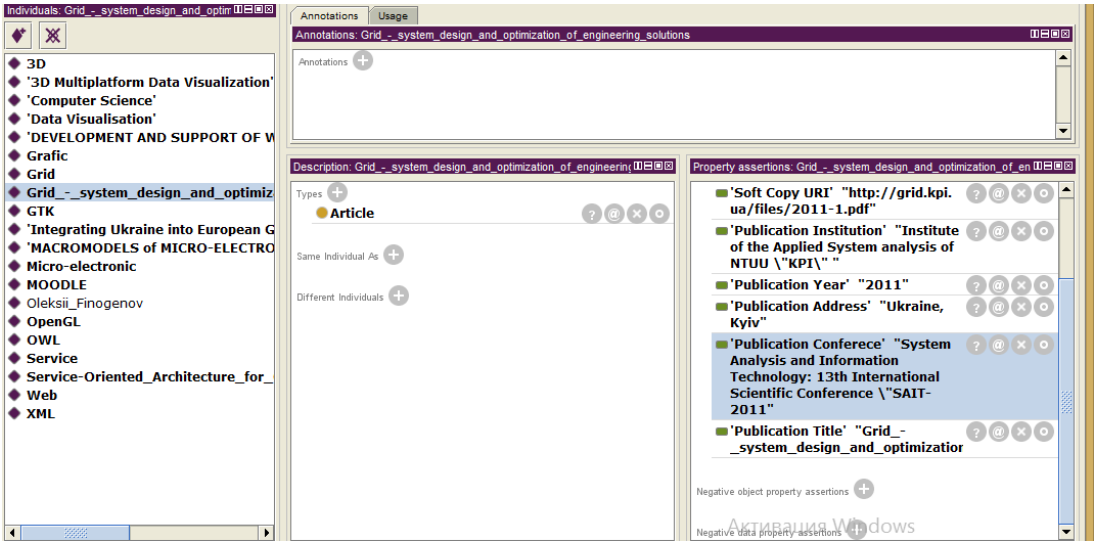

Рисунок 2.2 – Екземпляри та їхні властивості

2.2 Особливості проектування

Транзитивність ієрархічних відносин.

Відношення підкласу транзитивне:

Якщо B - це підклас A, а C - підклас B, то З - підклас A.

Іноді ми розрізняємо прямі і непрямі підкласи. Прямий підклас - найближчий підклас класу: в ієрархії між класом і його прямим підкласом немає інших класів. Тобто, між класом і його прямим надкласса в ієрархії немає інших класів.

Розвиток ієрархії класів.

Підтримка послідовної ієрархії класів може викликати складнощі по мірі того, як розвиваються предметні області.

Уникнення циклів класів

Потрібно уникати циклів в ієрархії класів. В ієрархії є цикл, коли у деякого класу A є підклас B і в той же час B - це надклас A. Створення такого циклу в ієрархії рівнозначний оголошенню того, що класи A і B еквівалентні: всі екземпляри A - це екземпляри B, а всі екземпляри B також є екземплярами A. Справді, оскільки B - підклас A, то всі примірники B повинні бути екземплярами класу A. оскільки A - підклас B, то всі примірники a також повинні бути екземплярами класу B.

Збалансованість ієрархії

У багатьох онтологіях з чіткою структурою є від двох до дюжини прямих підкласів. Звідси:

• Якщо клас має тільки один прямий підклас, то, можливо, при моделюванні допущена помилка або онтологія неповна.

• Якщо у даного класу є більше дюжини підкласів, то, можливо, необхідні додаткові проміжні категорії [17].

Введення підкласу

Одним із найскладніших рішень, яке потрібно прийняти під час моделювання, – це визначити, коли ввести новий клас або додати відмінність за допомогою різних властивостей [6]. Складно орієнтуватися як в ієрархії з дуже великим ступенем вкладеності і безліччю сторонніх класів, так і з ієрархією, де дуже мало класів, але в їх слотах міститься занадто багато інформації. Знайти відповідний баланс нелегко.

Існує декілька практичних способів визначення того, коли в ієрархію слід ввести нові класи:

Підкласи мають мати такі особливості:

1. Мають додаткові властивості, яких немає у надкласу.
2. Мають обмеження, відмінні від тих, які є у суперкласу.
3. Входять до інших відносин, на відміну від надкласу.

2.3 Автоматичні методи побудови онтологій

Розвиток онтологій починає набувати більш масовий характер, і в даний час в цій області є ряд масштабних розробок. На даному етапі з'являються ідеї використання автоматичних та напівавтоматичних методів для не тільки поновлення онтологій, але навіть для їх створення.

Існує ряд методів розширення онтологій, які специфічні для онтологій різних зон. Для розширення верхньої, найбільш загальної зони необхідно докладно описати, після чого можна приступати до побудови понять і аксіом. Для онтологій середньої зони, які відрізняються великою кількістю понять, збір понять може виконуватися автоматично за допомогою кластеризації [13]. В процесі обробки великої кількості інформації відбувається збір понять і розбиття їх за класами на підставі загальних характеристик. Існує цілий ряд методів по підвищенню точності вилучення семантично пов'язаних сімей понять. При такому

аналізі надалі можливо також встановлювати перехресні посилання всередині онтології. Однак для онтології важливо знати не тільки те, що поняття взаємопов'язані, але, і те, як саме вони взаємопов'язані. Для виявлення таких відносин між поняттями також можуть бути використані автоматичні методи перегляду і аналізу різних текстів, наприклад, як пропонують автори роботи [6], можна витягати цю інформацію зі словникових визначень. Це обумовлено тим, що існує обмежений набір фразових моделей, за характером яких можна сформулювати тип зв'язків між поняттями і ввести ці дані в онтологію.

Виявлення ж аксіоматичних знань, правил може бути вироблено на підставі Інтернету. Е.Хові вказує на те, що збір окремих прикладів відбувається за рахунок вивчення великої бази ресурсів, проте спочатку формулюється ряд параметрів, що вказують, які саме екземпляри нам потрібні [10]. Це можна пояснити на конкретному прикладі: якщо нам потрібно зібрати авторів публікацій столиць, то можна, наприклад, поставити умови пошуку з моделлю "X написав публікацію ...". При цьому можна використовувати не одну таку модель, а кілька. Таким чином, перебування окремих екземплярів буде зводитися до аналізу текстів для виявлення прикладів з даними структурами.

Такі шаблони можуть формуватися як вручну, так і автоматично за допомогою програм, що можуть самонавчатись.

Поряд з пошуком окремих екземплярів важливим є встановлення різних зв'язків між елементами онтології, класами. В цілому автори роботи [18] пропонують розділити всі методи вилучення відносин між елементами онтології на два класи: підходи, засновані на використанні шаблонів, і методи, які використовують кластеризацію. При використанні методів на основі шаблонів дослідники шукають мовні моделі, які вказують на якийсь тип відносин між класами. У більшості випадків здійснюється пошук родовідових відносин і відносин «частина-ціле».

При наявності базового переліку категорій можна нарощувати їх кількість, звертаючись до корпусу текстів. У корпусі виділяються кластери близьких за значенням елементів, потім оцінюється тіснота зв'язку між елементами, далі кожному кластеру приписується ім'я, яке асоціюється з категорією, що включається в онтологію. Таким чином, поповнення онтології новими елементами відбувається автоматично і також автоматично визначається місце нового елемента в ієрархії категорій.

Як згадувалося раніше, в онтології присутні не тільки класи і окремі екземпляри, але й аксіоми. Вони вносять великий внесок в удосконалення аналізу інформації, дають можливість комп'ютера доповнити текст додатковими знаннями, які для нас здаються тривіальними. Аксіоми повідомляють комп'ютера, наприклад, про те, що два висловлювання мають один і той же сенс або ж один факт тягне за собою наявність іншого. В роботі [16] розглядається метод автоматичного вилучення аксіом з текстів і їх подальша класифікація і перевірка на релевантність. В якості основної проблеми автоматичного виявлення аксіом автори згадують побудову помилкових припущень або занадто загальних, а також проблему визначення симетричності аксіом. Вони беруть за основу гіпотезу про схожість значень при схожості контекстів народження, доповнюючи їх своїми обмеженнями [14]. Тут ключовим стає визначення методів обчислення близькості контекстів і близькості понять. За твердженням авторів їх методика виявлення аксіом дозволяє витягувати релевантні аксіоми з маленьким відсотком помилок.

У більшості випадків проблемою автоматичного вилучення стає велика кількість «шуму», який треба ефективно відсіювати. У зв'язку з цим іноді поряд з автоматичними методами використовують наступну ручну обробку отриманого матеріалу для отримання даних більшої точності.

2.4 Застосування онтологій

Побудова онтології часто не є кінцевою метою, зазвичай онтології далі використовуються іншими програмами для вирішення практичних цілей. На даному етапі розвитку науки існує ряд завдань, де застосування онтологій може дати хороші результати. Однак зараз лише мала кількість додатків на природній мові включають в себе онтологічні бази. С. Ніренбург і В. Раскін [16] говорять про можливість використання онтологій в:

- ✓ машинному перекладі;
- ✓ питально-відповідних системах;
- ✓ інформаційний пошук;
- ✓ системах добування знань;
- ✓ загальних системах ведення діалогу між комп'ютером і людиною;
- ✓ системах розуміння мови (автоматичне реферування тексту, рубрикація і ін.)

У штучному інтелекті онтології використовуються для формальної специфікації понять і відносин, які характеризують певну область знань. Оскільки комп'ютер не може розуміти, як людина, стан речей в світі, йому необхідно подання всієї інформації в формальному вигляді. Таким чином, онтології є своєрідною моделлю навколишнього світу, а їх структура така, що легко піддаються машинній обробці і аналізу. Онтології забезпечують систему відомостями про добре описану семантику заданих слів і вказують ієрархічну будову області, взаємозв'язок елементів. Все це дозволяє комп'ютерним програмам за допомогою онтологій робити висновки з представленої інформації та маніпулювати ними.

Онтології можуть використовуватися для виведення умовиводів, необхідних для розуміння текстів на глибинно-семантичному рівні, що вимагається для

високоякісного машинного перекладу і може служити базою для розширення і уточнення інформаційного пошуку. Глибокий аналіз тексту необхідний і для систем автоматичного реферування. Варто згадати, що також онтології можуть сприяти систематизації понять. На базі онтологій може здійснюватися автоматичне анотування та розбір текстів, яке в подальшому може використовуватися в першу чергу в інформаційному пошуку, а також при різних видах аналізу інформації.

Наведемо деякі приклади існуючих систем, що містять онтологічні додатки: CROSSMARC, OntoLearn, IAMTC (Interlingual Annotation of Multilingual Text Corpora), YAWA, та ін.

2.5 Опис реалізації

Для реалізації бази знань наукових публікацій було обрано середовище Protege версії 4.3. Для реалізації сайту, де користувачі могли б спробувати здійснити пошук публікацій було використано Bootstrap, HTML5, CSS3 та JavaScript.

Protege – локальна Java-програма, розроблена групою медичної інформатики Стенфордського університету. Protege вільний, відкритий редактор онтологій і фреймворк для побудови баз знань.

Платформа Protege підтримує два основних способи моделювання онтологій за допомогою редакторів Protege-Frames і Protege-OWL. Онтології, побудовані в Protege, можуть бути експортовані в безліч форматів, включаючи RDF (RDF Schema), OWL і XML Schema.

Protege має відкриту, архітектуру, що легко розширюється за рахунок підтримки модулів розширення функціональності.

Protege підтримується значним співтовариством, що складається з розробників і вчених, урядових і корпоративних користувачів, що використовують

його для вирішення завдань, пов'язаних зі знаннями, в таких різноманітних галузях, як біомедицина, збір знань і корпоративне моделювання.

Protege доступний для вільного скачування з офіційного сайту разом з плагінами і онтологіями.

Структура онтології зроблена аналогічно ієрархічній структурі каталогу. На основі сформованої онтології Protege може генерувати форми отримання знань для введення примірників класів і підкласів.

Protege заснований на фреймовій моделі представлення знань і забезпечений рядом плагінів, що дозволяє адаптувати його для редагування моделей в різних форматах (стандартний текстовий, бази даних, UML, мов XML, XOL, SHOE, RDF и RDFS,DAML+OIL, OWL).

Було створено такі класи: Publication, Article, Book, Thesis, Diploma, MasterThesis, Technology, Problem, Branch,, Solution, та ін.. Також було створено такі зв’язки: use, solving, contain, isAbout, is.

На рис. 2.3, зображена отримана онтологія.

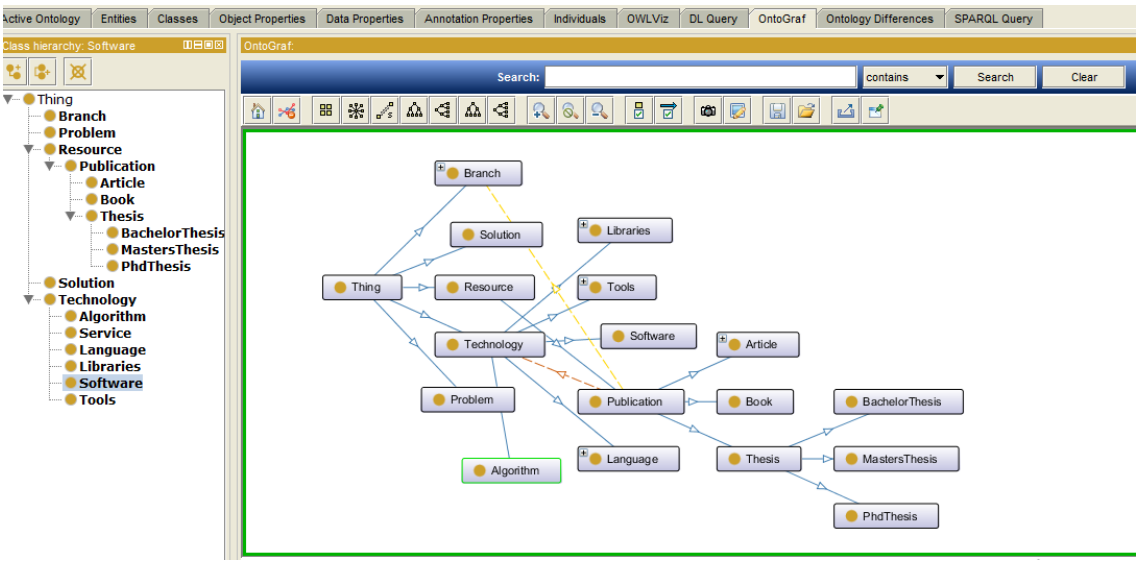

Рисунок 2.3 – Онтологія наукових публікацій

Заповнено онтологію було статтями нашої кафедри, на рис. 2.4, зображена база знань(її фрагмент, так як вся база досить велика).

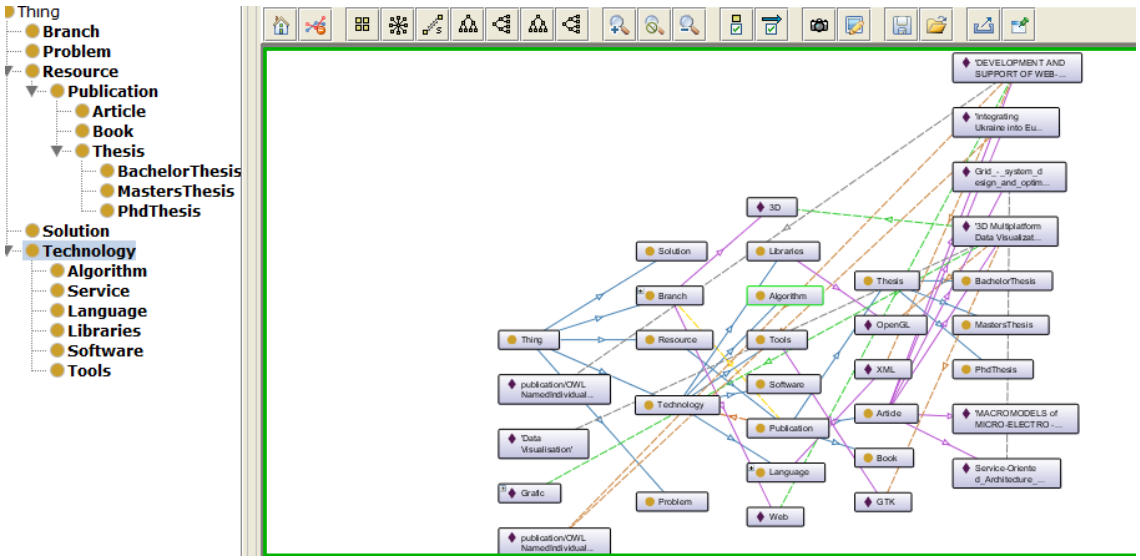

Рисунок 2.4 – Фрагмент бази знань наукових публікацій

Основна сторінка сайту наукових публікацій зображена на рис. 2.5:

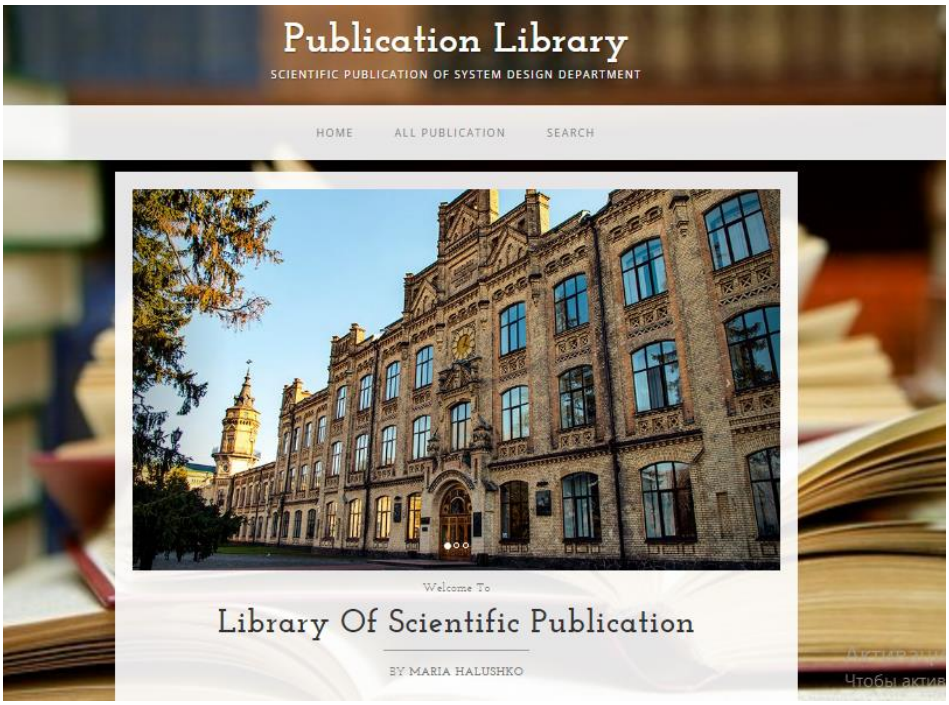

Рисунок 2.5 – Головна сторінка сайту наукових публікацій

Сторінка з усіма публікаціями, а також посиланнями на самі публікаціями (інформація взята із сайту grid.kpi.ua) зображена на рис. 2.6:

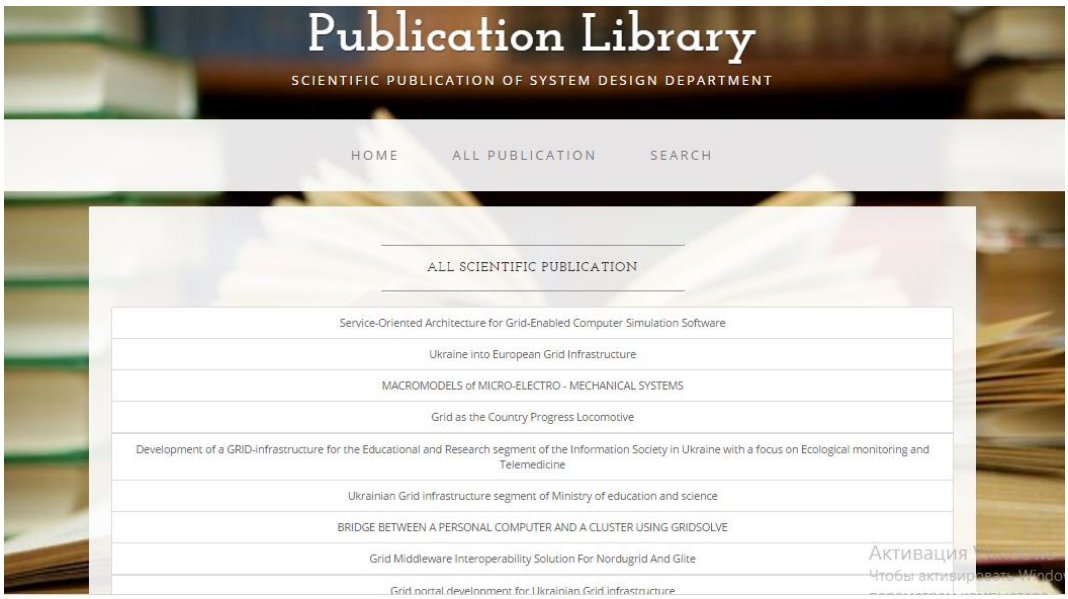

Рисунок 2.6 – Сторінка зі списком усіх наукових публікацій

2.6 Висновок

У розділі було описано загальний процес створення бази знань наукових публікацій з використанням RDF-сховищ. Основоположні правила розробки онтології автори формулюють таким чином:

1) Не існує єдиного правильного способу моделювання предметної області - завжди існують життєздатні альтернативи. Краще рішення майже завжди залежить від передбачуваного застосування та очікуваних розширень.

2) Розробка онтології - це обов'язково ітеративний процес.

3) Елементи онтології повинні бути близькі до об'єктів (фізичним або логічним) і відносинам в цікавій для вас предметної області. Швидше за все, вони відповідають іменником (об'єкти) або дієслів (відносини) в пропозиціях, які описують вашу предметну область. Тому проектування онтології - це творчий процес і дві онтології, розроблені різними людьми, ніколи не будуть однаковими.

Онтологія суб'єктивна. Перевірити правильність бази знань можна різонером, у Protege є стандартний різонер, який перевірить чи немає суперечностей в базі знань. запропонованих алгоритмів. А перевірити чи правильно виконується пошук по базі можна за допомогою запитів SPARQL.

3 ОГЛЯД РЕЗУЛЬТАТІВ ТЕСТУВАННЯ РОЗРОБЛЕНОЇ БАЗИ ЗНАНЬ

3.1 SPARQL, як засіб тестування бази знань

SPARQL (від англ. SPARQL Protocol and RDF Query Language) — мова запитів до даних, представлених по моделі RDF, а також протокол для передачі цих запитів і відповідей на них. SPARQL є рекомендацією консорціуму W3C і одною з технологій семантичної павутини. Представлення SPARQL-точок (рис. 3.1) доступу (SPARQL endpoint) є рекомендованою практикою при публікації даних у всесвітній павутині.

На додаток до мови, W3C також визначені:

- ✓ Протокол SPARQL для RDF Специфікація: він визначає протокол віддаленої видачі SPARQL запити, щоб отримувати результати.
- ✓ Результати запиту XML-специфікацією формату: вона визначає XML формат документів для представлення результатів SPARQL.

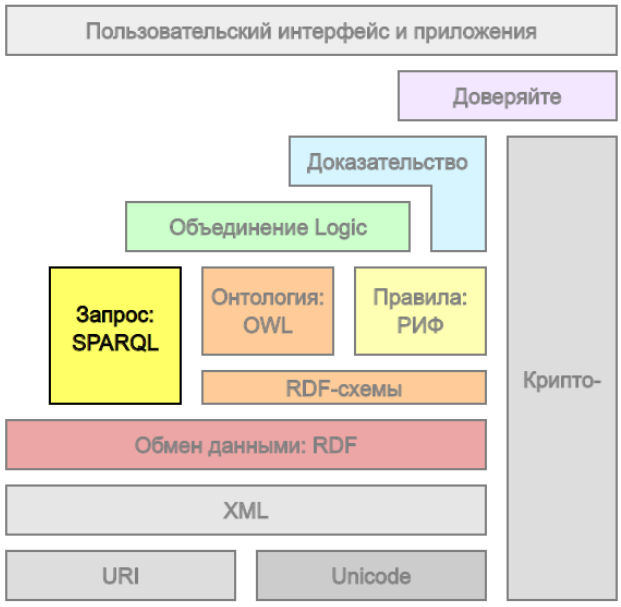

Рисунок 3.1 – Схема засобів, які використовуються в базах знань [6]

Технологія SPARQL дозволяє отримувати дані з розподілених джерел і може бути як засіб інтеграції різнорідної інформації. У специфікації SPARQL відсутні недоліки, властиві традиційним мовам запитів, зокрема, не накладаються обмеження на формат даних. Завдяки цьому стає можлива взаємодія між ресурсами різного типу. Намагатися використовувати семантичну мережу без SPARQL — це все одно, що працювати з реляційною базою даних без мови структурованих запитів SQL. Тобто, SPARQL перетворює доступ до даних у деяку подібність Web-сервісу.

Загальна схема SPARQL-запиту зображена на рис. 3.2:

```
PREFIX foo: <http://example.com/resources/>
# префіксні оголошення
FROM ...
# джерела запиту
SELECT ...
# пункт результату
WHERE {...}
# критерії запиту
ORDER BY ...
# модифікатори запиту
```

Рисунок 3.2 – Схема SPARQL-запиту [4]

- ✓ *Префіксні оголошення* слугують для скорочення універсальних ідентифікаторів ресурсу.
- ✓ *Джерела запиту* визначають, які RDF графи запитуються.
- ✓ *Пункт результату* повертає набір даних (вибірку), які задовольняють умові.
- ✓ *Критерії запиту* визначає, що запитувати в базовому наборі даних
- ✓ *Модифікатори запиту* обмежують, упорядковують, і інакше перетворюють результати запиту.

Мова SPARQL визначає чотири варіанта запитів для різних цілей:

SELECT запит

Використовується для того, щоб витягати необроблені значення із точки SPARQL, результати повертаються у вигляді таблиці.

CONSTRUCT запит

Використовується для того, щоб витягати інформацію з точки доступу SPARQL в форматі RDF і перетворювати результати до визначеної форми.

ASK запит

Використовується для створення запитів типу Істина/Брехня

DESCRIBE запит

Використовується для того, щоб отримати опис RDF-ресурсу. Реалізація поведінки DESCRIBE-запитів визначається розробником SPARQL-точки доступу.

Кожна з цих форм запиту включає в себе блок WHERE, щоб обмежити запит, хоча у випадку запиту DESCRIBE — WHERE не є обов'язковим.

Нижче наведена частина часто використовуваних ключових слів в SPARQL запитів. Повний список є в офіційній документації.

PREFIX — слугує для скорочення URI.

OPTIONAL — визначає необов'язковий шаблон.

GRAPH — за допомогою нього формують запит, який застосовує шаблон до іменованих графів.

DISTINCT — вказує, що кожне рішення в відповіді на запит буде унікальним.

LIMIT — задає максимальну кількість виведених результатів.

OFFSET — дозволяє не показувати в результаті перші n рішень.

ORDER BY — дозволяє відсортувати результат за збільшенням (ASC()) або спаданням (DESC()).

А тепер перейдемо до прикладів на створеній базі знань та протестуємо як вона працює.

3.2 Тестові запити до розробленої бази знань

Для виконання SPARQL-запитів буде використано програмний засіб Protégé, так як в ньому є можливість не лише створювати, графічно відображати базу знань, але й писати SPARQL-запити.

Напишемо декілька простих запитів і подивимось результати.

Спершу перевіримо чи виводяться всі запити нашої бази знань, для цього введемо на сайті «All» у поле «Article», результат зображений на рис. 3.3:

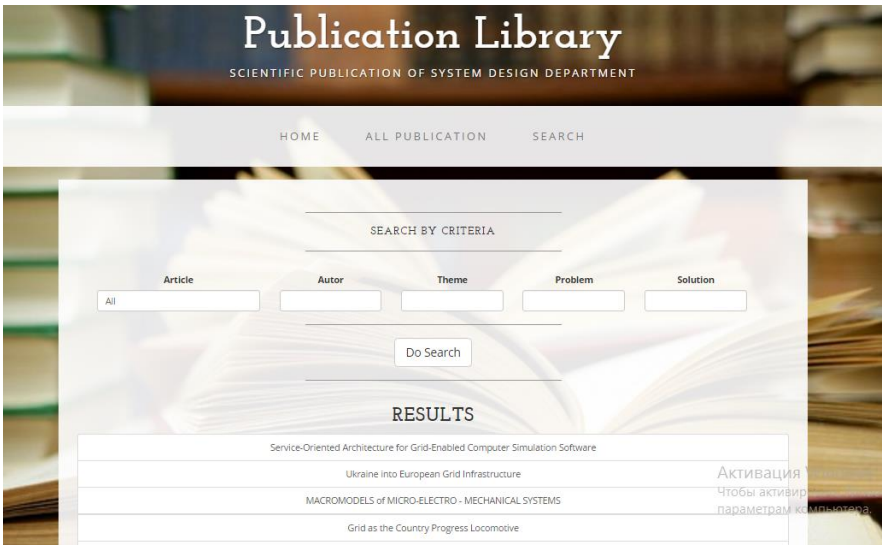

Рисунок 3.3 – Список всіх статті, які є у нашій базі знань

Другий запит виводить статі, які стосуються Грід(рис. 3.4):

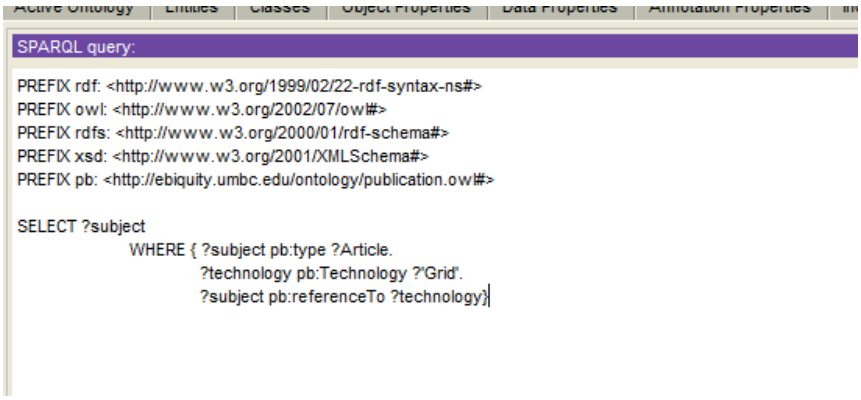

Рисунок 3.4 – SPARQL-запит, що виводить статі, які стосуються Грід

Результат, який отримали у Protege на рис. 3.5 та на сайті (рис. 3.6).

'DEVELOPMENT AND SUPPORT OF WEB-SITE FOR GRID PROJECT'
Grid_-_system_design_and_optimization_of_engineering_solutions
Service-Oriented_Architecture_for_Grid-Enabled_Computer_Simulation_Software
'Integrating Ukraine into European Grid Infrastructure'

Рисунок 3.5 – Статті, які стосуються Грід, відображені у Protege

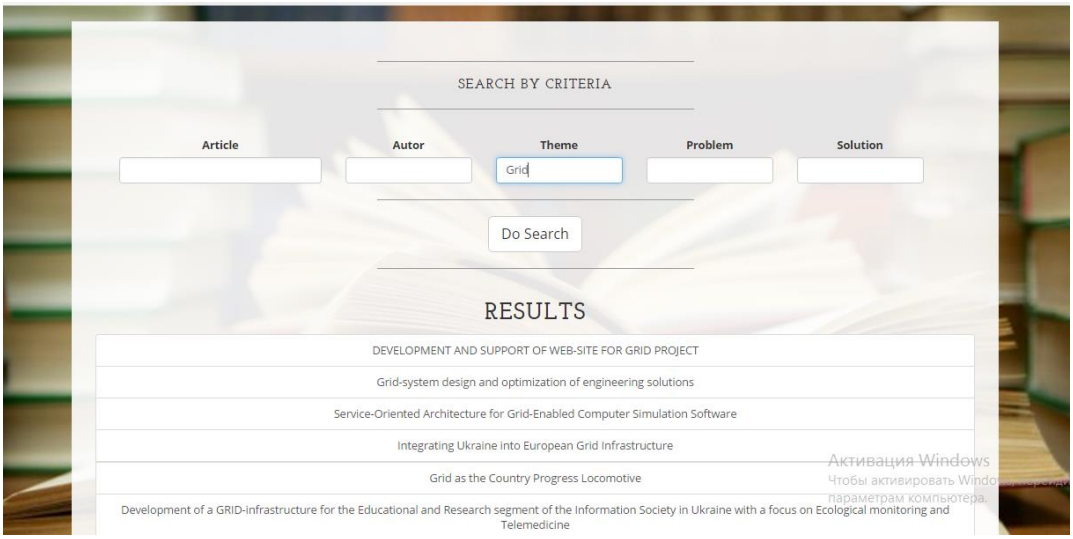

Рисунок 3.6 – Статті, які стосуються Грід, відображені сайті

Наступний запит виводить статі, проблема яких вирішується алгоритмом (рис. 3.7).

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX pb: <http://ebiquity.umbc.edu/ontology/publication.owl#>

SELECT ?subject
        WHERE {
                ?subject pb:type ?Article.
                ?problem pb:type ?Problem.
                ?algo pb:type ?Algorithm.
                ?subject pb:has ?problem.
                ?problem pb:resolve ?algo
        }
```

Рисунок 3.7 – SPARQL-запит, що виводить статі, проблема яких вирішується алгоритмом

Результат, який отримали у Protege на рис. 3.8 та на сайті (рис. 3.9).

'DEVELOPMENT AND SUPPORT OF WEB-SITE FOR GRID PROJECT'
Grid_-_system_design_and_optimization_of_engineering_solutions
Service-Oriented_Architecture_for_Grid-Enabled_Computer_Simulation_Software

Рисунок 3.8 – Статті, проблема яких вирішується алгоритмом, відображені у Protege

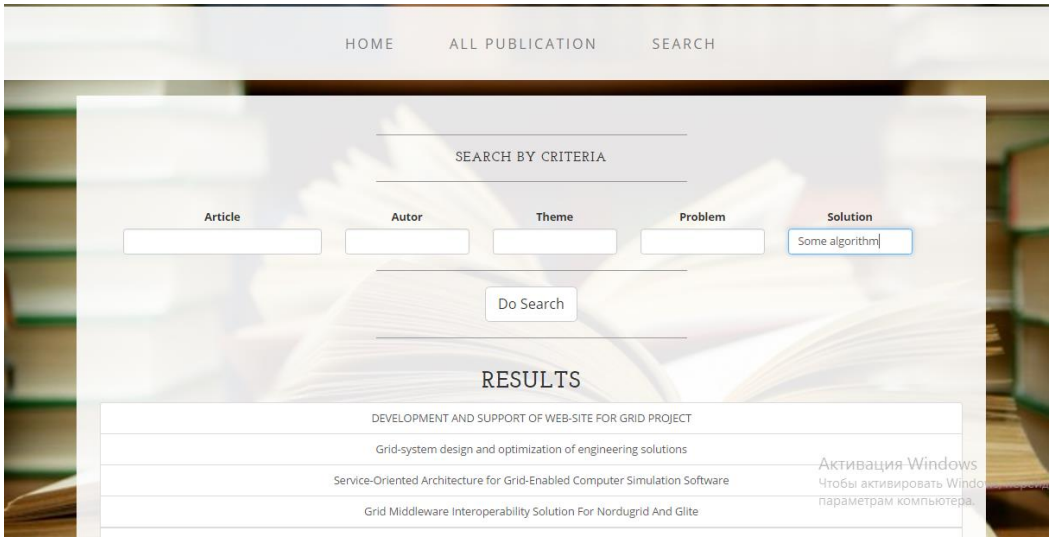

Рисунок 3.9 – Статті, проблема яких вирішується алгоритмом, відображені сайті

Ще один запит, що виводить статті, які посилаються на галузь(рис. 3.10)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX lp: <http://ebiquity.umbc.edu/ontology/publication.owl#>

SELECT ?X
WHERE {
        ?X rdf:type lp:Article.
        ?Y rdf:type lp:Branch.
        ?Y lp:reference ?Y
}
```

Рисунок 3.10 – SPARQL-запит, що виводить статті, які посилаються на галузь

Результат запиту з рис. 3.10, бачимо на рис. 3.11:

```
'DEVELOPMENT AND SUPPORT OF WEB-SITE FOR GRID PROJECT'
'MACROMODELS of MICRO-ELECTRO - MECHANICAL SYSTEMS (MEMS)'
'3D Multiplatform Data Visualization'
Grid_-_system_design_and_optimization_of_engineering_solutions
Service-Oriented_Architecture_for_Grid-Enabled_Computer_Simulation_Software
'Integrating Ukraine into European Grid Infrastructure'
```

Рисунок 3.11 – Статті, які посилаються на галузь

Запит, який виводить статі, що посилаються на на галузь, використовуючи певні засоби (рис. 3.12).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX lp: <http://ebiquity.umbc.edu/ontology/publication.owl#>

SELECT ?X
WHERE {
        ?X rdf:type lp:Article.
        ?Y rdf:type lp:Tools.
        ?Z rdf:type lp:Branch.
        ?X lp:using ?Y.
        ?Y lp:reference ?Z
}
```

Рисунок 3.12 – Статті, які посилаються на галузь, використовуючи певні засоби

Результат запиту з рис. 3.12, бачимо на рис. 3.13:

X
'3D Multiplatform Data Visualisation'

Рисунок 3.13 – Статті, які посилаються на галузь, використовуючи певні засоби

А тепер напишемо цікавіший запит: «Знайти авторів публікацій, де рішенням проблеми є веб-сервіс» (рис. 3.14).

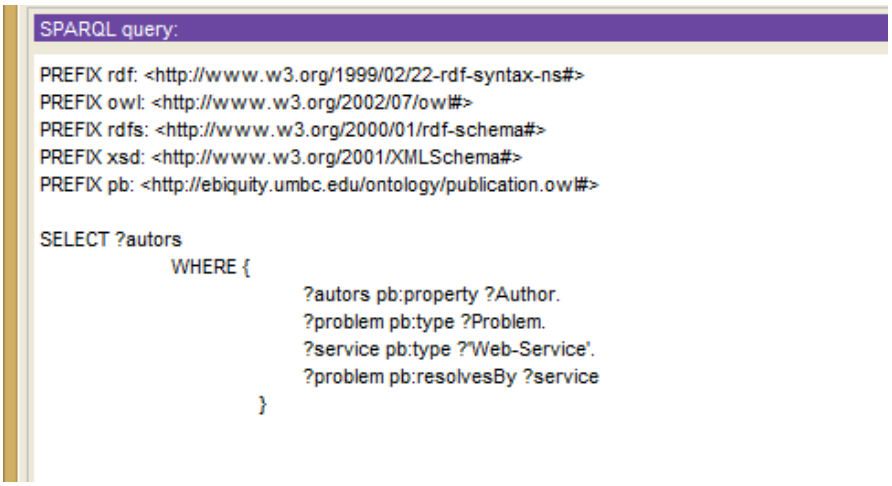

Рисунок 3.14 – Запит, що шукає авторів публікацій, де рішенням проблеми є веб-сервіс

Результат зображено на рис. 3.15 та на сайті (рис. 3.16).

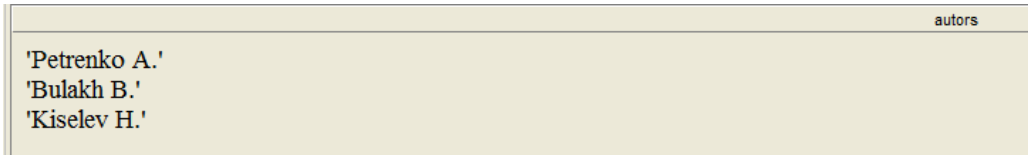

Рисунок 3.15 – Автори публікацій, де рішенням проблеми є веб-сервіс, відображені у Protégé

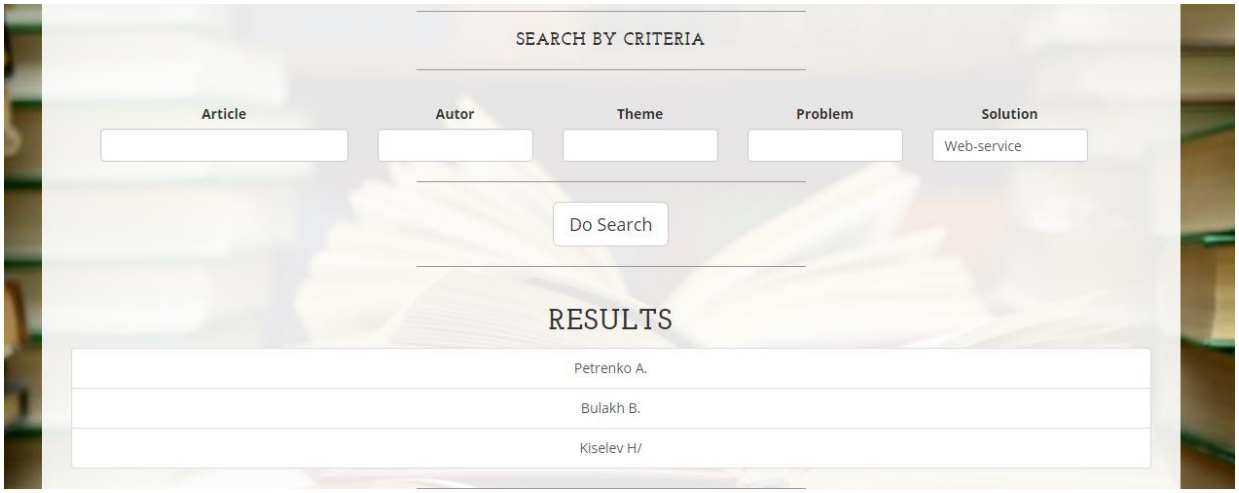

Рисунок 3.16 – Автори публікацій, де рішенням проблеми є веб-сервіс, відображені на сайті

Як бачимо, не всі отримані результати можна було б отримати за допомогою SQL бази даних, а для зручності пошуку користувач не завжди може вказати всі необхідні параметри, щоб отримати відповідну інформацію з SQL-баз даних, а з використанням онтологій цей пошук виконати стає можливим.

3.3 Висновок

Було показано, що використання онтологій робить можливим пошук по контексту, також інші запити є простішими і швидшими, ніж SQL-запити, тому дають виграш як у відповідності інформації, яку потрібно було знайти, так і в швидкодії виконання самих запитів. На даний момент база знань не є дуже велика, тому швидкість була б майже не помітна, але в майбутніх дослідженнях можна це продемонструвати заповнивши її більшою кількістю даних.

Звісно, є і недоліки використання онтологій, так як потрібно створювати чи змінювати і заповняти її вручну. Як згадувалось у попередніх розділах є напівавтоматичні системи, які дозволяють частково автоматизувати процес, але в швидкому майбутньому можливо це не буде вже проблемою.

Також було помічено, що збільшення кількості зв’язків між класами, самих класів та більшої кількості даних в базі знань позитивно впливає на пошукові результати. Чим краще описані суб’єкти, тим «розумнішим» може бути пошук.

4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Вступ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для перевірки методів прогнозування місцезнаходження об’єктів у контекстно-залежних системах керування енергоспоживанням. Програмний продукт був розроблений за допомогою мови програмування OWL, SPARQL – мова запитів до онтологій та програмного середовища Protege.

Програмний продукт є крос-платформенним та рекомендується для використання на персональних комп’ютерах під управлінням операційних систем Windows, Linux чи Mac.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями [10].

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі

витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат;

– для кожної функції визначаються повні річні витрати й кількість робочих часів;

– для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат;

після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.2 Постановка задачі

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи аналізу місцезнаходження об’єктів та його прогнозування. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для створення бази знань наукових публікацій з використанням RDF-сховищ.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

4.2.1 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка бази знань наукових публікацій. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір середовища проектування;

F_2 – використання існуючих онтологій;

F_3 – спосіб пошуку у базі знань.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) середовище проектування Protege;

б) середовище проектування Fluent Editor;

Функція F_2:

а) створення своєї онтології;

б) доповнення існуючої онтології;

Функція F_3:

а) пошук за бібліометричними даними;

б) пошук за контекстом, ключовими словами.

4.2.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

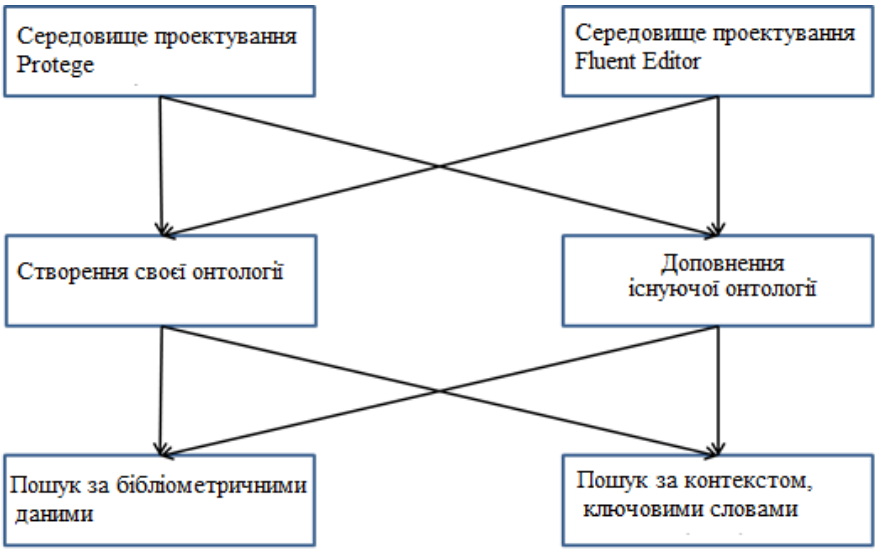

Рисунок 4.1 – Морфологічна карта

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Займає менше часу при проектуванні онтології і створенні бази знань	Використання більше ресурсів системи
	Б	Використання менше ресурсів системи	Займає більше часу при проектуванні онтології і створенні бази знань
F2	А	Онтологія відповідає всім поставленим задачам	Займає більше часу при проектуванні
	Б	Займає менше часу при проектуванні	Надлишок/Недостатність певних класів, властивостей
F3	*А*	Легша реалізація	Дає гірші результати
	Б	Дає кращі результати	Важча реалізація

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам.

Функція *F1:*

Оскільки потрібно спроектувати онтологію і заповнити великою кількістю публікацій, тому потрібно швидко це зробити, тому варіант б) має бути відкинутий.

Функція *F2:*

Оскільки методи написані вручну та за допомогою готових бібліотек будуть давати однакові результати, вважаємо варіанти а) та б) гідними розгляду.

Функція *F3:*

Точна оцінка важливіша для даної роботи, ніж складність обчислень, тому варіант а) має бути відкинутий.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3б
2. F1а – F2б – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів ПП

4.3.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня. Для того, щоб

охарактеризувати програмний продукт, будемо використовувати наступні параметри: *X1* – релевантність пошукових результатів, *X2* – об'єм пам'яті для збереження даних, *X3* – час обробки даних, *X4* – потенційний об'єм програмного коду.

X1: Відображає відповідність результатів пошуку операцій залежно від обраної архітектури онтології.

X2: Відображає об'єм пам'яті в оперативній пам'яті персонального комп'ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

4.3.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			Гірші	середні	кращі
Відповідність результатів пошуку	X1	Бали	1	3	5
Об'єм пам'яті для збереження даних	X2	Мб	32	16	8
Час обробки запиту до бази знань	X3	мс	600	320	75
Потенційний об'єм програмного коду	X4	кількість строк коду	1855	1325	985

За даними таблиці 3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.4.

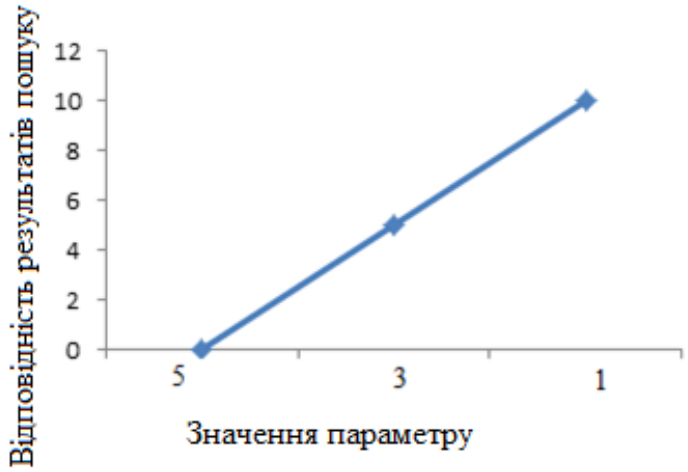

Рисунок 4.2 – Відповідність результатів пошуку

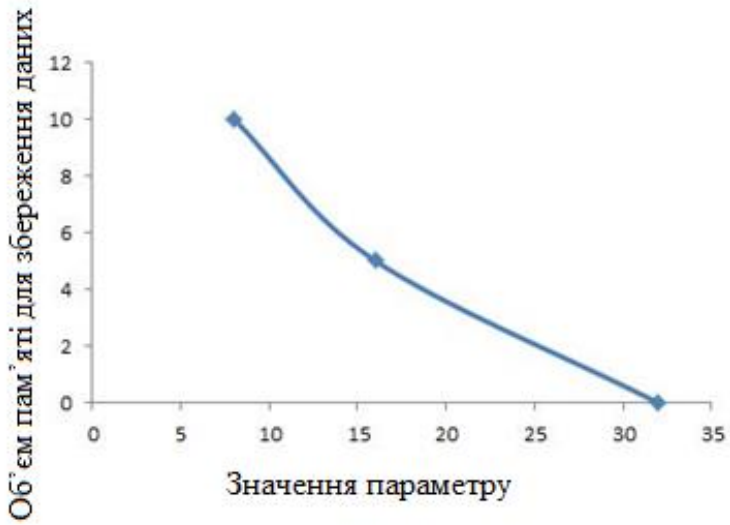

Рисунок 4.3 – Об’єм пам’яті для збереження даних

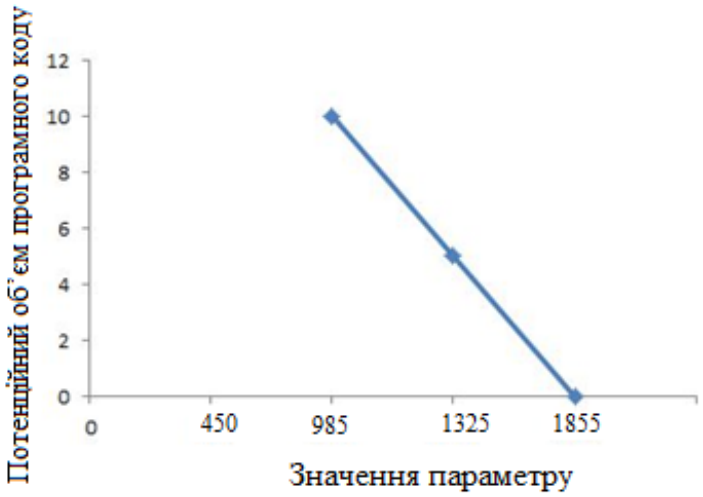

Рисунок 4.4 – Потенційний об’єм програмного коду

4.3.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів за формулою (4.1):

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105, \qquad (4.1)$$

де N – число експертів, од;

n – кількість параметрів, од

R_{ij} – ранг, од.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Відповідність результатів пошуку	Бали	4	3	4	4	4	4	4	27	0.75	0.56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1.25	1.56
X3	Час обробки даних алгоритмом	Мс	2	2	1	2	1	2	2	12	-14.25	203.06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14.75	217.56
	Разом		15	15	15	15	15	15	15	105	0	420.75

б) середня сума рангів за формулою (4.2):

$$T = \frac{1}{n} R_{ij} = 26.25. \quad (4.2)$$

де n – кількість параметрів, од;

R_{ij} – ранг, од.

в) відхилення суми рангів кожного параметра від середньої суми рангів за формулою (4.3):

$$\Delta_i = R_i - T \quad (4.3)$$

де Т - середня сума рангів, од;

R_i – – сума рангів кожного параметра, од.

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення за формулою (4.4):

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420.75 \quad (4.4)$$

де Δ_i^2 – відхилення, од;

N – кількість експертів, од.

Порахуємо коефіцієнт узгодженості за формулою (4.5):

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 420.75}{7^2(5^3-5)} = 1{,}03 > W_k = 0.67 \quad (4.5)$$

де S - загальна сума квадратів відхилення, од;

W_k – нормативний коефіцієнт узгодженості.

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0.5
X1 і X3	<	<	<	<	<	<	<	<	0.5
X1 і X4	>	>	>	>	>	>	>	>	1.5
X2 і X3	<	<	<	<	<	<	<	<	0.5
X2 і X4	>	>	>	>	>	>	>	>	1.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі (4.6):

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \qquad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами(4.7, 4.8):

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \qquad (4.7)$$

де $b_i = \sum_{i=1}^{N} a_{ij}$. (4.8)

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами (4.9):

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j. \qquad (4.9)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1.0	0.5	0.5	1.5	3.5	0.219	22.25	0.216	100	0.215
X2	1.5	1.0	0.5	1.5	4.5	0.281	27.25	0.282	124.25	0.283
X3	1.5	1.5	1.0	1.5	5.5	0.344	34.25	0.347	156	0.348
X4	0.5	0.5	0.5	1.0	2.5	0.156	14.25	0.155	64.75	0.154
Всього:					16	1	98	1	445	1

4.4 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) *X1* (відповідність результатів пошуку) та *X3* відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X4 (*потенційний об’єм програмного коду) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 1800 або варіанту б) 1200.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6) за формулою (4.10):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} , \qquad (4.10)$$

де n – кількість параметрів, од;

$K_{вi}$– коефіцієнт вагомості i–го параметра. од;

B_i – оцінка i–го параметра в балах, од.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3.6	0.215	0.774
F2(X3)	А, Б	800	2.4	0.348	0.835
F2(X4)	А	1800	2	0.154	0.308
	Б	1200	8	0.154	1.232
F3(X2)	Б	16	3.4	0.283	0.962

За даними з таблиці 4.6 за формулою (4.11)

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}] \qquad (4.11)$$

де $K_{\text{ТУ}}[F_{ik}]$ – коефіцієнт рівня якості i функції, визначаємо рівень якості кожного з варіантів:

K_{K1} = 0.774 + 0.835 + 0.308 + 0.962= 2.879

K_{K2} = 0.774 + 0.835 + 1.232 + 0.962 = 3.803

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.5 Економічний аналіз варіантів розробки ПП

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

При цьому варіант 3 має додаткове завдання: реалізація пошуку за контекстом;
А варіант 4 має інше додаткове завдання: обробка інтерфейсу готових бібліотек.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 2.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється за формулою (4.12)

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \qquad (4.12)$$

де T_P – трудомісткість розробки ПП, од;

$K_П$ – поправочний коефіцієнт, од;

$K_{СК}$ – коефіцієнт на складність вхідної інформації, од;

K_M – коефіцієнт створення ієрархії класів і відносин, од;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм, од;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення, од.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ = 90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$Т_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм другої групи складності, степінь новизни Б), тобто $Т_Р$ = 27 людино-днів, $К_П$ = 0.9, $К_{СК}$ = 1, $К_{СТ}$ = 0.8:

$Т_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Для третього завдання (використовується алгоритм другої групи складності, ступінь новизни Г з використанням перемінної інформації):

Тр = 14 людино-днів;

Кп = 0.72; Кст = 0.8;

То = $14 \cdot 0.72 \cdot 0{,}8 = 7.05$.

Для четвертого завдання (використовується алгоритм третьої групи складності, ступінь новизни Г):

Тр = 9 людино-днів;

Кп = 0.6; Кст = 1;

То = $9 \cdot 0.6 \cdot 1 = 5.4$.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

T_I = (122.4 + 19.44 + 7.05) · 8 = 1195 людино-годин;

T_{II} = (122.4 + 19.44 + 5.4) · 8 = 1181.22 людино-годин;

Найбільш високу трудомісткість має варіант І.

В розробці беруть участь два програмісти із окладом 6000 грн., один фінансовий аналітик з окладом 9000 грн. Визначимо зарплату за годину за формулою (4.13):

$$С_Ч = \frac{М}{T_m \cdot t}, \quad (4.13)$$

де М – місячний оклад працівників, грн;

T_m – кількість робочих днів тиждень, од;

t – кількість робочих годин в день, год.

$$С_Ч = \frac{6000 + 6000 + 9000}{3 \cdot 21 \cdot 8} = 41.67 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою (4.14)

$$С_{ЗП} = С_Ч \cdot Т_i \cdot К_Д \quad (4.14)$$

де $С_Ч$ – величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 41.67· 1195 · 1.2 = 59584.76 грн.

II. $С_{ЗП}$ = 41.67· 1181.22 · 1.2 = 58672.69 грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 59584.76· 0.22 = 13111.05 грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 58672.69 · 0.22 = 12975.35 грн.

Тепер визначимо витрати на оплату однієї машино-години($С_М$).

Так як одна ЕОМ обслуговує одного програміста з окладом 6000 грн., з коефіцієнтом зайнятості 0.2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 6000 \cdot 0.2 = 14400$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 14400 \cdot (1 + 0.2) = 17280$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 17280 \cdot 0.22 = 3801.6$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн за формулою (4.15).

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300 \text{ грн.}, \qquad (4.15)$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо за формулою (4.16):

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460 \text{ грн.}, \qquad (4.16)$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4 \text{ годин},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою (4.17):

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706.4 \cdot 0.156 \cdot 0.2 \cdot 2.0218 = 107.64 \text{ грн.}, \quad (4.17)$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0.67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть за формулою (4.18):

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н \quad (4.18)$$

$С_{ЕКС} = 17280 + 3801.6 + 2300 + 460 + 107.64 + 5360 = 29309.24$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 29309.24 / 1706.4 = 17.18$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає за формулою (4.19):

$$С_М = С_{М\text{-}Г} \cdot Т \quad (4.19)$$

I. $С_М = 17.18 \cdot 1190 = 20444.2$ грн.;

II. $С_М = 17.18 \cdot 1173.12 = 20154.2$ грн.;

Накладні витрати складають 67% від заробітної плати:

$С_Н = С_{ЗП} \cdot 0.67$

I. $С_Н = 59504.76 \cdot 0.67 = 39868.19$ грн.;

II. $С_Н = 58660.69 \cdot 0.67 = 39302.66$ грн.;

Отже, вартість розробки ПП за варіантами становить за формулою (4.20):

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н \qquad (4.20)$$

I. $С_{ПП} = 59504.76 + 13091.05 + 20444.2 + 39868.19 = 132908.2$ грн.;

II. $С_{ПП} = 58660.69 + 12905.35 + 20154.2 + 39302.66 = 131022.9$ грн.;

4.6 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою (4.21):

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \qquad (4.21)$$

$К_{ТЕР1} = 2.879 / 132908.2 = 0.22 \cdot 10^{-4}$;

$К_{ТЕР2} = 3.803 / 131022.9 = 0.29 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0.29 \cdot 10^{-4}$.

4.7 Висновок

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості

було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0.29 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

- ✓ середовище проектування Protege;
- ✓ створення своєї онтології;
- ✓ пошук за контекстом, ключовими словами.

ВИСНОВКИ

В ході роботи було досліджено створення баз знань з використанням RDF-сховищ. Складність роботи полягала у тому, що теоретична база хоч і активно розвивається, і є перспективною, проте не надто добре вивчена.

Створення онтологій є перспективним напрямком сучасних досліджень по обробці інформації, що подається на природній мові. В рамках роботи було висвітлено поняття онтології, баз знань, наведено класифікацію онтологій, та описано різницю між ними.

Також були розглянуті основні складові онтологій та кроки по створенню онтологій, що дозволяють уникнути ряду типових проблем при розробці баз знань.

Враховуючи універсальність формату RDF для опису не лише бібліографічних даних, а й смислових конструкцій змісту публікацій, було запропоновано розробити базу публікацій, що не лише включає набір ключових слів, а й описує основні положення тієї чи іншої публікації, що, в свою чергу, робить можливим постановку складніших за смислом пошукових запитів. Це може дати помітний виграш відносно існуючих пошукових механізмів, що здійснюють пошук інформації без урахування семантики слів, які входять до запиту, а також контексту, в якому вони використовуються. Так як це може дати помітний виграш відносно існуючих пошукових механізмів, що здійснюють пошук інформації без урахування семантики слів, які входять до запиту, а також контексту, в якому вони використовуються. У такого підходу є декілька недоліків: створення специфічних SPARQL-запитів та необхідність вручну “семантично анотувати” наявні у бібліотеці ресурси, і дослідження засобів автоматизації такої роботи є окремою актуальною задачею.

Було названо і проаналізовано ряд існуючих прототипів бібліотек наукових публікацій, що використовують RDF-сховища, такі як DBpedia, Google Scholar та Scirus. Не зважаючи на наявні переваги існуючих рішень, була поставлена задача

розробки власної системи, яка б не включала "зайві" для бібліотеки публікацій класи, зв'язки та властивості, а також дозволяла б пошук за контекстом, а не лише ключовими словами. При цьому, при заповненні бази знань адміністратор може сконцентруватися на основних поняттях, не відволікаючись на заповнення безлічі другорядних атрибутів, які найчастіше не використовуються у пошукових запитах.

Було створено таку ієрархію класів та створено відповідні між ними зв1язки, заповнено необхідними даними (науковими публікаціями) базу знань, що все ж вдалось не використовуючи при описі самого суб’єкта деякі терміни, звертаючись в пошуку по ним отримати потрібний результат.

Досить перспективною є автоматизація створення онтологій, однак на даному етапі ще не розроблені ефективні процедури, застосування яких дозволить скоротити частку помилок. Але це не було метою дослідження, тому в подальшому можна буде продовжувати роботу і знайти зручний спосіб автоматизувати цей процес.

В майбутньому планую розвивати науково-дослідницьку роботу в напрямку роботи з онтологіями, оскільки в перспективі можливості створеного апарату прийняття рішень над структурованими даними є необмеженими. Шляхи реалізації такого потенціалу : починаючи від електронних бібліотек, до розумних перекладачів, закінчуючи штучним інтелектом. На даний момент це ще не докінця досліджена галузь, поєднавши з іншими технологіями думаю, можна досягнути нових, суспільно-корисних результатів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Basu D. Smart Doorplate / Basu D. // Journal of Ontologies. – 2003. – №25. – С. 201-216.
2. Bayardo B.L. Swoogle — Semantic Web Search Engine [Електронний ресурс] – Режим доступу: http://swoogle.umbc.edu/ . –Дата доступа : 19.06.2016.
3. Benjamins V.R. An Intelligent Brokering Service for Knowledge-Component Reuse on the World – Wide Web / Benjamins V.R. // Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management, June 2009, Чикаго, США , матеріали. — Ч.: W360, 2009 С. 125-129.
4. Cardenas A.F. Data Base Communication in a heterogeneous data base management system network / Cardenas A.F. — K. : Information Systems, 2010, 253 с.
5. Garcia-Molino H. N. RDF in Semantic Web/ Garcia-Molino H. N. — K. : NGITS, 2013. — 110 с.
6. Garcia-Molino H. N. The TSIMMIS Approach to Mediation: Data Models and Languages / Garcia-Molino H. N. — K. : NGITS, 2011. — 196 с.
7. Gruber, T.R. A translation approach to portable ontology specifications / Gruber, T.R. — С. : Knowledge Acquisition, 2015. — 321 с.
8. Guarino N. H. The Ontological Level / Guarino N. H. — L. : Philosophy and the Cognitive Sciences, 2012. — 198 с.
9. Jasco. P. Google Scolar: The Pros and Cons / Jasco. P. // Journal of Information technolodgies. – 2005. – № 9. – С. 208-214.
10. Noruzi A. Google Scholar: The New Generation of Citation Indexes/ Noruzi A. – Valencia : Burjassot, 2010. – 185 с.
11. Tim B.L. World Wide Web: Proposal for HyperText Project [Електронний ресурс] – Режим доступу: http://www.w3.org/Proposal.html. –Дата доступа : 14.12.2016.

12. Бажанова А.И. Разработка онтологической модели для семантического поиска информации в электронной библиотеке [Электронный ресурс] – Режим доступу: http://masters.donntu.org/2011/fknt/bazhanova/diss/index.htm. – Дата доступа : 21.05.2016.
13. Богданюк В.Є. Методичні вказівки до виконання організаційно-економічного розділу дипломних проектів / Богданюк В.Є. – К. : Політехніка, 2009. – 66 с.
14. Демитренко М. О. Бази знань у прикладних системах їх значення та застосування [Електронний ресурс] – Режим доступу: http://gerontology–explorer.ru/aa5ada45–6fd3–42c3–8af3–02247f604571.html. – Дата доступа : 20.05.2016.
15. Пашин В.П. Управление качеством изделий на основе функционально-стоимостного анализа / Пашин В.П. // Технология и организация производства. — 1989. – № 12.— С. 17-19.
16. Пашін В.П. Методичні вказівки до виконання економіко-організаційного розділу дипломних проектів (робіт) бакалаврів і спеціалістів для студентів Інституту прикладного системного аналізу з дисципліни “Економіка та організація виробництва” для студентів спеціальностей 7.080204 - “Соціальна інформатика”, 7.080203 - “Системний аналіз і управління”, 7.080404 – “Інтелектуальні системи прийняття рішень”, 7.080402 - “Інформаційні технології проектування” / Пашін В.П., Романов В.В., Єгорова Н.В. – К. : НТУУ “КПІ”, 2011. – 118 с.
17. Пашін В.П. Оцінка конкурентоспроможності електронних пристроїв на стадії проектування : Пашін В.П., Бровкін А. Г.,. Павлуша І. А. // Економічний вісник. — 2006. — № 4. — С. 56-65.
18. Розпутний М.В. Засоби здійснення логічних висновків над онтологіями / Розпутний М.В. // Системний аналіз та інформаційні технології: 10-а міжнародна науково-технічна конференція «САІТ-2012», 20–24 травня 2012, Київ, Україна : матеріали. – К. : НТУУ «КПІ», 2012. – С. 171.

19. Щербак С. В. Руководство по созданию онтологий / Щербак С. В. – К. : Техника, 2014. – 302 с.
20. Щербін Я. О. Онтології як сховища даних / Щербін Я. О. – К. : Техника, 2016. – 235 с.