Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____ 2017 р.

ЗАВДАННЯ
на дипломну роботу студенту

Хачатрян Артем Артакович
(прізвище, ім'я, по батькові)

1. Тема роботи Побудова семантичних веб-додатків з використанням мов опису логічних правил
_____
_____,
керівник роботи Булах Богдан Вікторович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи _____
_____
_____
_____

4. Зміст роботи 1 Вробити огляд предметної області.
2 Вибрати засоби реалізації програмної області.
3 Сформулювати рекомендації та розробити програмний продукт.

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____ 2017 р.

ЗАВДАННЯ
на дипломну роботу студенту

Хачатрян Артем Артакович
(прізвище, ім'я, по батькові)

1. Тема роботи Побудова семантичних веб-додатків з використанням мов опису логічних правил
_____
_____,
керівник роботи Булах Богдан Вікторович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи _____
_____
_____
_____

4. Зміст роботи Проаналізувати мови опису правил для семантичного веб та розробити демонстраційний додаток
_____
_____
_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____
_____
_____
_____

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка

Студент _____ А. А. Хачатрян
(підпис) (ініціали, прізвище)

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

* Консультантом не може бути зазначено керівника дипломної роботи.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

FOAF – friend of a friend;

GUI – graphic user interface;

HTML – Hyper Text Markup Language;

IDE – Integrated Development Environment;

OWL – Web Ontology Language;

RDF – Resource Description Framework;

RDFS – RDF-Schema;

RIF – Rule Interchange Format;

RSS – Rich Site Summary;

SGML – Standard Generalized Markup Language;

SPARQL – SPARQL Protocol and RDF Query Language;

SWAD-Europe – Semantic Web Advanced Development for Europe;

SWI – Sociaal-Wetenschappelijke Informatica ("Social Science Informatics");

SWRL – Semantic Web Rule Language;

URI – уніфікований ідентифікатор ресурсу;

W3C – The World Wide Web Consortium;

Web 3.0 – Semantic Web або семантична павутина;

XML – Extensible Markup Language;

XMP – Extensible Metadata Platform;

ПП – програмний продукт.

ВСТУП

Метою даної роботи є формулювання рекомендацій для розробки прикладних веб-додатків з використанням мов опису логічних правил у складі онтологій.

Семантична павутина (Semantic Web) розглядається як одна з фундаментальних складових Web 3.0, і являє собою сукупність пов'язаних даних.

Основна ідея семантичної павутини полягає в додаванні семантичної розмітки, яка не тільки описує дані і зв'язки всередині них, а й забезпечує можливість створення зв'язків між розрізненою інформацією, що зберігається на різних веб-ресурсах. Таким чином, семантичний веб служить для поліпшення якості інформації доступною в мережі Інтернет, в деякій мірі спрощує пошук дійсно релевантних результатів на пошукові запити, з можливістю формувати відповідь на запити користувачів без переходів безпосередньо на веб-ресурси, що, звичайно, можна порахувати і недоліком . Крім того, використання семантичної павутини в довгостроковій перспективі, може вирішити проблему повторюваного контенту.

Також семантичні веб-технології дозволяють власникам веб-ресурсів більш точно вирішити проблему неоднозначностей для пошукових систем. Наприклад, зіткнувшись з двома ресурсами за запитом "замок", доступними тільки в форматі HTML, пошукова система повинна обробити дані, щоб по можливості з'ясувати, про що саме йшла мова. Семантична павутина дозволяє не тільки точно вказати пошуковій системі, що мова йде про «замок - будова», а не «замок - пристрій», а й пов'язати між собою поняття «замок - будова» і «фортеця - будова», таким чином пов'язуючи і веб-ресурси.

Очевидно, що пошуковим системам «легше» обробляти структуровану інформацію. Постійно зростає число підтримуваних пошуковими системами

форматів і здійснюється підтримка щодо їх використання веб-розробниками.

Для вирішення поставленої задачі було обрано мову програмування SWI-Prolog, що має набір інструментів, який дозволяє повністю виконати весь обсяг робіт і є реалізацією однією з популярних освітніх мов програмування.

SWI-Prolog — це вільна (відкрита) реалізація мови програмування Prolog, часто використовувана для викладання і додатків Semantic Web. Ця реалізація надає багатий набір можливостей, многопоточности, юніт-тестування, GUI, інтерфейс до багатьох мов програмування, підтримує літературне програмування, містить реалізацію веб-сервера, бібліотеки для SGML, RDF, RDFS, засоби розробника (включаючи IDE з графічними отладчиком і профілювальником), і велику документацію.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Роль в сучасних технологіях

Глобальна мережа Інтернет постійно розвивається. На сьогоднішній день орієнтація її розвитку спрямована на створення високоякісних і високо інформативних сервісів, дана концепція отримала назву Web 3.0.

Семантична павутина (Semantic Web) розглядається як одна з фундаментальних складових Web 3.0, і являє собою сукупність пов'язаних даних. Основна ідея семантичної павутини полягає в додаванні семантичної розмітки, яка не тільки описує дані і зв'язки всередині них, а й забезпечує можливість створення зв'язків між розрізненою інформацією, що зберігається на різних веб-ресурсах.

Таким чином, семантичний веб служить для поліпшення якості інформації доступною в мережі Інтернет, в деякій мірі спрощує пошук дійсно релевантних результатів на пошукові запити, з можливістю формувати відповідь на запити користувачів без переходів безпосередньо на веб-ресурси, що, звичайно, можна порахувати і недоліком . Крім того, використання семантичної павутини в довгостроковій перспективі, може вирішити проблему повторюваного контенту.

1.2 Область застосування

Також семантичні веб-технології дозволяють власникам веб-ресурсів більш точно вирішити проблему неоднозначностей для пошукових систем. Наприклад, зіткнувшись з двома ресурсами за запитом "замок", доступними тільки в форматі HTML, пошукова система повинна обробити дані, щоб по можливості з'ясувати, про що саме йшла мова. Семантична павутина дозволяє не тільки точно вказати пошуковій системі, що мова йде про «замок - будова», а не «замок - пристрій», а й пов'язати між собою поняття «замок - будова» і

«фортеця - будова», таким чином пов'язуючи і веб-ресурси.

Очевидно, що пошуковим системам «легше» обробляти структуровану інформацію. Постійно зростає число підтримуваних пошуковими системами форматів і здійснюється підтримка щодо їх використання веб-розробниками. Так в 2009 році Google оголосив про підтримку RDF і мікроформатів і продовжує розширювати цей список в тому числі і онтологіями для електронної комерції GoodRelations. Яндекс також підтримує ряд мікроформатів і семантичну розмітку.

Насправді, незважаючи на відносно повільне впровадження технологій Web 3.0, структуровані дані вже всюди, і навіть якщо не використовується RDF; то пошуковим системам передаються структуровані дані в форматі XML.

1.3 Практичне використання

Технологія семантичної павутини на даний час успішно вирішує наступні завдання:

- незалежність даних від програм;
- семантична інтеграція даних;
- створення основи для повсюдного використання комп'ютерних сервісів.

Уже зараз зроблено багато для реалізації Semantic Web. Проект на базі пошукової системи Google надає свої ресурси для запитів агентам на виконання пошукових функцій і перевірки правопису. Також представляє інтерес проект з автоматичного створення RDF-описів і сховища метаданих, створюваний на основі Open Directory пошуковим механізмом Google. Крім того, необхідно також відзначити і проект консорціуму W3C SWAD-Europe, який займається проблемою зв'язку сховищ семантичних даних з використовуваними реляційними системами баз даних, особливо ліцензованих як Free Software/Open Source.

Проект SemanticHealthNet (SHN), що виконувався в 2012-2015 рр.,

присвячено поліпшенню семантичної сумісності змісту електронних персональних медичних карт шляхом внесення в них інтегрованого семантичного резюме. Цей підхід був продемонстрований на прикладах хронічної серцевої недостатності і серцево-судинної профілактики.

Подальшому розвитку семантичної павутини сприяє наявність вільно розповсюджуваних систем для розробки додатків Semantic Web:

- Jena Framework (Java);
- Drive RDF Parser (C#).
- В даний час вже існують:
- бібліотеки для інтерпретації стека мов RDF для всіх популярних мов програмування (Jena, Redland, RDFLib);
- редактори онтологій (Protege);
- системи міркувань над онтологіями (Racer, KAON, FACT);
- семантичні сховища (Sesame, Kowari, YARS);
- семантичні браузери (Simile, Piggy Bank, Gnowsis, Haystack);
- пошуковики семантичних даних (Swoogle);
- конвертори з різних форматів представлення даних в RDF/XML (Aperture, RDFizers, D2R);
- прикладні програми (Bibster, FOAF Explorer).

А також існуючі комерційні продукти:

- Adobe's XMP - інструментарій для створення метаописів про файлах;
- Oracle's 10.2 Database - вже має вбудовану підтримку моделі RDF;
- Tucana's Knowledge Discovery Suite - платформа для інтеграції інформації застосувань (Enterprise Information Integration, EII)

З 2003 року щорічно проводиться всесвітній конкурс Semantic Web Challenge, покликаний зібрати найостанніші напрацювання і показати світу стан справ щодо практичної реалізації ідей Semantic Web. При цьому був сформульований перелік мінімальних критеріїв, що визначають поняття

«додаток Semantic Web» в рамках конкурсу.

- По-перше, програма має використовувати інформаційні джерела, які:
  - географічно розподілені;
  - мають різних власників, що передбачає відсутність контролю за їх розвитком;
  - містять дані реального світу, тобто джерела повинні бути чимось більшим ніж умоглядні приклади.
- По-друге, програма має сприймати відкритий світ; це означає, що воно знає, що інформація ніколи не буває повною і постійно змінюється.
- По-третє, програма має використовувати деякий формальний опис значення даних.

В даний час постає завдання створення додатків другого покоління. Друге покоління додатків семантичної павутини повинні використовувати весь величезний запас вже накопиченої семантики. Додатки Semantic Web 2-го покоління повинні бути здатні використовувати:

- безліч онтологій;
- бути відкритими для семантичних ресурсів;
- бути відкритими для роботи з користувачем (user interaction).

В ідеалі вони також повинні вміти використовувати не тільки дані семантичної павутини, але і інші формати даних, отже повинні мати потужні механізми з автоматичного вилучення інформації.

1.4 Архітектура

З точки зору архітектури семантичну павутину можна розділити на три яруси (рис. 1.4.1);

1. базис, який складається з унікальної глобальної ідентифікації ресурсу, метаданих для декларування фактів про ресурси, і спільної мови для вираження метаданих і знань, який реалізований за допомогою онтологій

для загальнодоступного розуміння і загального словника метаданих і правил для додавання нових метаданих та знань;

2. базовий сервіс, наприклад логічний висновок і запити до метаданих і онтологій, роз'яснення таких висновків, управління довірою (достовірністю), агенти, пошукові системи, сервери онтологій;
3. сервіси додатків, безпосередньо клієнтська частина програми.

Рисунок 1.4.1 – Три яруси додатків семантичної павутини

1.5 Структура

Ще в 1998 році Тім Бернерс-Лі запропонував наступний логічний план побудови Semantic Web:

- синтаксис для представлення знань, який використовує посилання на онтології;
- мову опису онтологій;
- мова опису веб-сервісів;
- інструменти читання/розробки документів Semantic Web;
- мову запитів до знань, які записані з використанням синтаксис для представлення знань;
- логічний висновок знань;
- семантична пошукова система.

Сучасна модель Semantic Web трохи складніше. Ця модель складається з цілого сімейства стандартів (рис. 1.5.1)

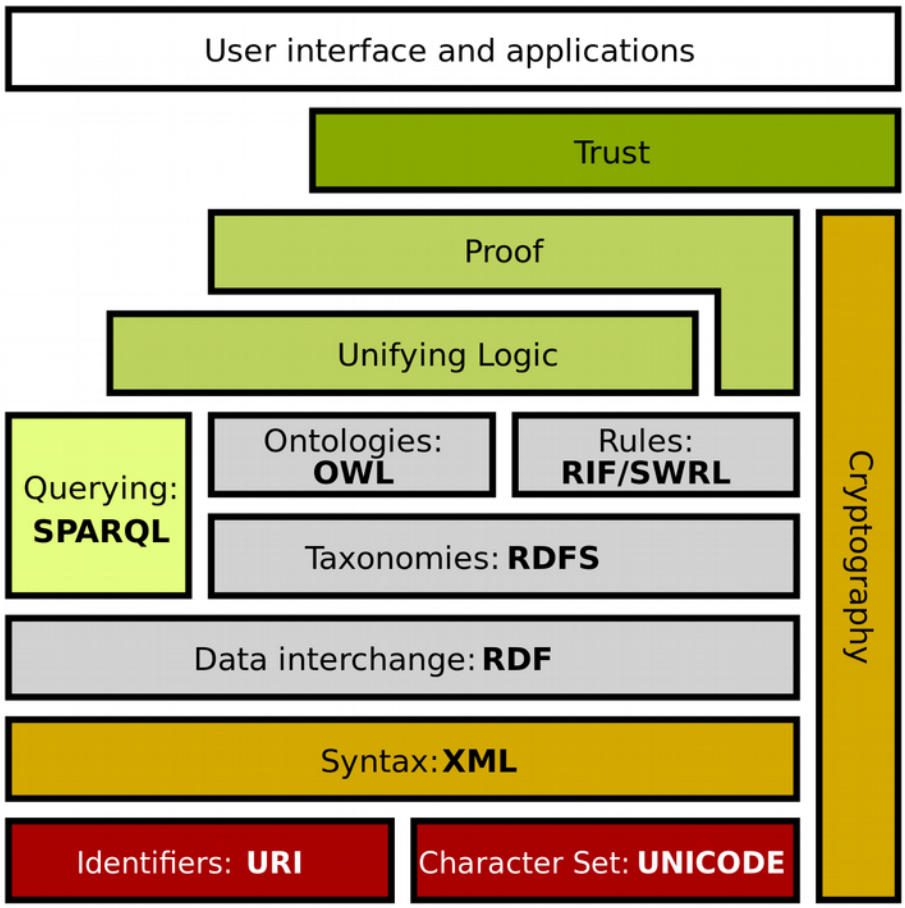

Рисунок 1.5.1 – Базовая модель Semantic Web в редакции 2014 года

На цьому малюнку добре видно три яруси семантичної павутини.

- Сервіси додатків - призначений для користувача інтерфейс і розширень;
- Базовий сервіс - логіка, доведення, довіру;
- Базис - ідентифікатори (URI), кодування символів (Unicode), синтаксис (XML), модель представлення даних (RDF), таксономія (RDF-Schema), онтологія (OWL), робота з правилами (RIF / SWRL), запити до даних і їх передача (SPARQL).

1.5.1 URI

URI — уніфікований ідентифікатор ресурсу або адреса, що використовується для вказівки посилань на будь-який об'єкт (наприклад, веб-

сторінку, файл або ящик електронної пошти). URI використовуються для іменування об'єктів. Кожен об'єкт глобальної семантичної мережі має унікальний URI.

URI однозначно називає деякий об'єкт. Окремі URI створюють як для сторінок, так і для об'єктів реального світу (людей, міст, художніх творів), а також для абстрактних понять ("ім'я", "посаду"). Завдяки унікальності URI одні й ті ж предмети можна називати однаково в різних місцях семантичної павутини. Використовуючи URI, можна збирати інформацію про один предмет з різних місць. Рекомендується починати з "http: //" або "https: //". Такий адресу можна одночасно використовувати як адреса URI і як адреса веб-сторінки (URL). На веб-сторінках, адреси URL яких збігаються з URI, W3C рекомендує розміщувати опис предмета. Опис бажано надавати в форматі:

- зручному для читання людиною;
- зручному для читання машиною.

1.5.2 Мови опису структури

Технічну частину семантичної павутини становить сімейство стандартів на мови опису, включаючи XML, XML Schema, RDF, RDF Schema, OWL, мікроданних. Маючи в своєму розпорядженні їх в порядку підвищення рівня абстракції отримуємо:

- XML надає синтаксис для визначення структури документа, що підлягає машинній обробці. Синтаксис XML не несе семантичного навантаження.
- XML Schema визначає обмеження на структуру XML-документа. Стандартний синтаксичний аналізатор мови XML в змозі перевірити довільний XML-документ на відповідність його структури так званою схемою документа, описаної в XML Schema.
- RDF являє собою простий спосіб опису екземплярність даних в форматі суб'єкт-відношення-об'єкт, в якому в якості будь-якого елементу цієї трійки використовуються тільки ідентифікатори ресурсів (за винятком

об'єкта, якому дозволено літералом). Існує стандартизоване відображення цих трійок на XML-документи певної структури, а також на інші формати представлення.

- RDF Schema описує набір атрибутів (тут їх точніше назвати відносинами), таких, як rdfs: Class, для визначення нових типів RDF-даних. Мовою підтримується також ставлення успадкування типів rdfs: subClassOf.
- OWL розширює можливості по опису нових типів (зокрема, додаванням перерахувань), а також дозволяє описувати нові типи даних RDF Schema в термінах вже існуючих (наприклад, визначати тип, який є приємним або об'єднанням двох існуючих).
- Мікродані (HTML microdata) - це міжнародний стандарт семантичної розмітки HTML-сторінок, за допомогою атрибутів, що описують зміст інформації, що міститься в тих чи інших HTML-елементах. Такі атрибути роблять контент сторінок машино зчитування, тобто дозволяють в автоматичному режимі знаходити і витягати потрібні дані.

1.5.3 Логічний висновок

Принцип "логічного висновку" дуже простий: це можливість виводити нові дані з з наявних даних на підставі відомих закономірностей і правил. У математичному сенсі, виконання запиту є однією з форм логічного висновку (наприклад, можливість вивести з маси даних деякий результат пошуку). Логічний висновок є одним з провідних принципів Semantic Web, так як він дозволяє дуже легко створювати додатки Semantic Web.

Для того, щоб Semantic Web став досить виразним і зміг допомагати людям в різних ситуаціях, виникає необхідність побудови потужного логічного мови, який підтримує логічний висновок. Дискусії про методи, а також можливості виконання цього завдання, до сих пір ведуться дуже активно.

1.5.4 RIF

Rule Interchange Format (RIF) — формат обміну правилам. Мета цього стандарту W3C - визначення формату, який би дозволив транслювати правила між різними мовами правил і завдяки цьому забезпечити обмін правилами між системами, заснованими на правилах.

Системи, засновані на правилах, набули широкого поширення в інформаційних технологіях. До їх числа відносяться, наприклад, експертні системи і системи дедуктивних баз даних. Розробки технологій Semantic Web забезпечують нове середовище використання таких систем. Тому консорціум W3C приділяє особливу увагу цій галузі. Специфікація RIF може розглядатися як складова частина комплексу стандартів Semantic Web.

1.5.5 SWRL

Правила виведення нових фактів за допомогою SWRL — Semantic Web Rule Language. Завдяки доповненню OWL мовою RuleML (підмножина Datalog) у вигляді словника SWRL з'явилася можливість використовувати диз'юнкт Хорна для явної вказівки способу виведення нових фактів з RDF-тверджень. Поки словник SWRL знаходиться в стадії стандартизації. Хоча роботи над цим рівнем Semantic Web тривають, проте в нашому розпорядженні є вже достатній набір засобів для побудови Semantic Web: твердження, цитування, так звана матеріалізація, в RDF, класи, властивості, області, документування в схемі RDF, непересічні класи, властивості однозначності і унікальності, типи даних, інверсії, еквівалентності, списки та інше.

1.5.6 Довіра и доведення

Для забезпечення цілісності і несуперечності інформації, представленої в Semantic Web, важливо забезпечити зв'язок додатків Semantic Web з контекстом, а також механізми перевірок доказів і цифрових підписів.

Додатки Semantic Web повинні враховувати контекст в цілому для того, щоб повідомляти користувачам, чи можуть вони довіряти наданими даними. Якщо користувач отримує потік RDF-даних від іншого користувача про прочитану книгу і про його оцінку цієї книги, то він повинен знати, хто ця людина, і чи можна довіряти цій інформації. Більш того, користувач може потім скористатися цією інформацією, не сумніваючись в її джерелі. Далі користувач залишає на свій розсуд наскільки йому вірити отриманим критичного відгуку про книгу.

Необхідно пам'ятати і про те, що над роздільними контекстами працюють також і групи людей. Якщо якась група розробляє в Semantic Web інформаційну службу для художників, каталогізуючи людей, їх імена і місця, де знаходяться картини цих людей, то довіра користувача до групи залежить від того, наскільки він довіряє людям, які приймають участь в цій групі.

У зв'язку з цим в Semantic Web для визначення джерела інформації пропонується використовувати цифрові підписи. Розробники Semantic Web планують, що кожен користувач або агент всі свої RDF-затвердження підписувати персональної унікальною цифровим підписом.

Ще одним аспектом довірливості інформації є перевірка істинності. Необхідний мову перевірки істинності, яка дозволить проконтролювати, є чи ні твердження правдивим. Реалізація мови перевірки зазвичай складається зі списку "елементів" логічного висновку, які використовуються для отримання шуканої інформації, а також для подальшої перевірки інформації про довіру для кожного з цих елементів.

1.6 Висновки

Semantic Web - це ідея зберігання даних в інтернет таким чином, щоб вони були визначені і пов'язані для подальшої можливості автоматизованої обробки, інтеграції і повторного використання в різних додатках.

Метою Semantic Web є зробити існуючий інтернет більш машинозчитуваним з тим, щоб використовувати інтелектуальних агентів для пошуку і обробки відповідної інформації.

Концепція Semantic Web полягає в забезпеченні достатньої гнучкості для можливості подання всіх баз даних і правил логіки таким чином, щоб зв'язати їх всі разом. Однак опис Semantic Web полягає в тому, що він являє собою спробу реалізувати машинну обробку даних зокрема, трансформувати обробку інформації забезпеченням загального принципу, згідно з яким дані можуть бути отримані, пов'язані один з одним і зрозумілі. Переклад мережі інтернет від типу «великий книги з гіперпосиланнями» до великої пов'язаної базі даних.

Ідея Semantic Web полягає в звільненні людини від обтяжливих рутинних завдань з видобутку, пошуку, обліку та індексації інформації, що міститься в Web. "Semantic Web - це бачення наступного покоління Інтернет, який дозволить веб-додаткам автоматично збирати веб-документи з різних джерел, враховувати і обробляти інформацію, а також взаємодіяти з іншими додатками для виконання складних завдань".

2 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Мова опису логічних правил

Для вирішення поставленої задачі було обрано мову програмування SWI-Prolog, що має набір інструментів, який дозволяє повністю виконати весь обсяг робіт і є реалізацією однією з популярних освітніх мов програмування.

SWI-Prolog — це вільна (відкрита) реалізація мови програмування Prolog, часто використовувана для викладання і додатків Semantic Web. Ця реалізація надає багатий набір можливостей, бібліотеки для многопоточності, юніт-тестування, GUI, інтерфейс до багатьох мов програмування, підтримує літературне програмування, містить реалізацію веб-сервера, бібліотеки для SGML, RDF, RDFS, засоби розробника (включаючи IDE з графічними отладчиком і профілювальником), і велику документацію.

2.2 Висновки

В даному розділі проведено вибір засобів реалізації, що підходять для використання при розробці прикладних веб-додатків. Була обрана мова SWI-Prolog, як комплексний набір інструментів вирішення поставлених у цій роботі задач і є реалізацією однією з популярних освітніх мов програмування.

3 РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Постановка задачі на реалізацію

Під час написання дипломної роботи був створений демонстраційний семантичний додаток — база знань з класичної механіки.

Вимоги до реалізації:

- дозволяє використання логічних правил;
- інформаційне наповнення відповідає обраному і є достатнім для демонстраційного додатку.

3.2 Архітектура програми

Була використана три ярусна архітектура (дивись рис. 1.4.1):

1. базис — безпосередньо база даних з класичної механіки, яка являє собою онтологію створену і наповнену за допомогою вбудованої бібліотеки ”Semantic Web”, що підтримує RDF у SWI-Prolog;
2. базовий сервіс — повертає на запит відповідний термін і його наявні зв'язки з іншими термінами;
3. сервіси додатків — графічний інтерфейс.

3.3 Структура ПП

Кожен екземпляр-поняття онтології (бази даних) являє собою реалізацію загального класу - термінів і пов'язаний лише з близькими поняттями (рис. 3.3.1). Таке поверхневе поєднання обране задля спрощення наповнення бази даних.

?- rdf_assert(ex:speed, rdf:type, class:'term')

?- rdf_assert(ex:distance, rdf:type, class:'term')

?- rdf_assert(ex:time, rdf:type, class:'term')

?- rdf_assert(ex:speed, rdf:related, ex:distance)

?- rdf_assert(ex:speed, rdf:related, ex:time)

?- rdf(ex:speed, rdf:related, E)

E = ex:distance,

E = ex:time.

?- rdf(E, rdf:type, class:'term')

E = ex:speed,

E = ex:distance,

E = ex:time.

rdf:type – об'єкт є екземпляром класу,

rdf:related – об'єкти споріднені.

Рисунок 3.3.1 – Зв'язки екземпляру-поняття “швидкість” з близькими поняттями

Але зважаючи на те, що така структура негативно вплине на швидкодію програми при розширенні її функціоналу. Тому ця структура має бути, в подальшому, вдосконалена до рівня, який задовольнить нові потреби в

деталізації та швидкодії ПП.

Клас термінів містить загальну інформацію щодо певного терміну. А саме:

- назву,
- визначення,
- формулу,
- одиниці виміру,
- коментар.

Необхідно відзначити, що для заповнення бази знань в системі повинні використовуватися інформація з достатньо надійних джерел. Дані з джерел мають бути перевірені на достовірність.

3.4 Графічний інтерфейс

Графічний інтерфейс створювався за допомогою XPCE, яка була насамперед спроектована для розробки графічного інтерфейсу в Prolog. XPCE має динамічно типізований об'єктно-орієнтоване ядро. Методи можуть бути визначені на будь-якій мові. Графічна бібліотека дозволяє абстрактне опис компонентів інтерфейсу і підтримуються в Win32 і Unix/X11.

XPCE передбачає два способи програмування. Вбудовані і бібліотечні класи можуть просто використовувати з вашого коду Prolog, або власні класи XPCE можуть бути використані в Prolog з використанням нативного синтаксису Prolog. Перший спосіб часто використовується для “маленьких” інтерфейсів, в той час як останній забезпечує відмінну структуризацію для “великих” графічних додатків. Оскільки в даній роботі необхідно розробити невеликий демонстраційний додаток було обрано перший спосіб.

Пролог зазвичай пов'язують зі штучним інтелектом, обробкою природної мови, базами даних і подібними завданнями. Інтерактивні програми можуть мати підзадачі, для яких Prolog є інструментом для використання. Такі додатки реалізуються на інших мовах, а Prolog використовується як вбудований

механізм для вирішення цих завдань. А IDE дозволяє швидко створювати прототипи під час роботи додатків, автоматичне керування даними і пам'яттю робить розробку швидкої і надійної.

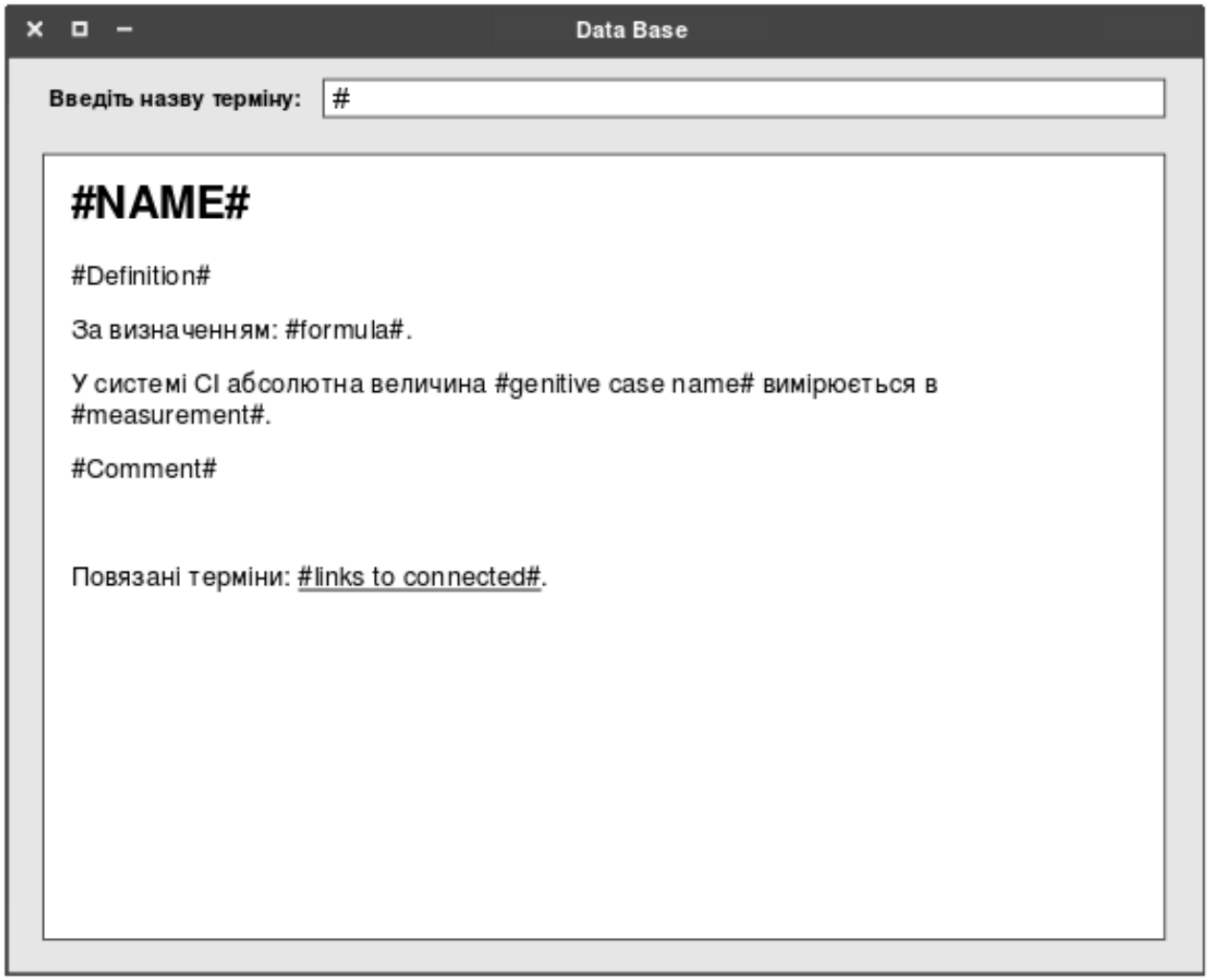

Рисунок 3.4.1 – Вигляд вікна програми з зображенням макету

Сторінка містить наявну інформацію щодо терміну: назву, визначення, формулу, одиниці виміру, коментар і посилання на пов'язані безпосередньо з поточним терміном.

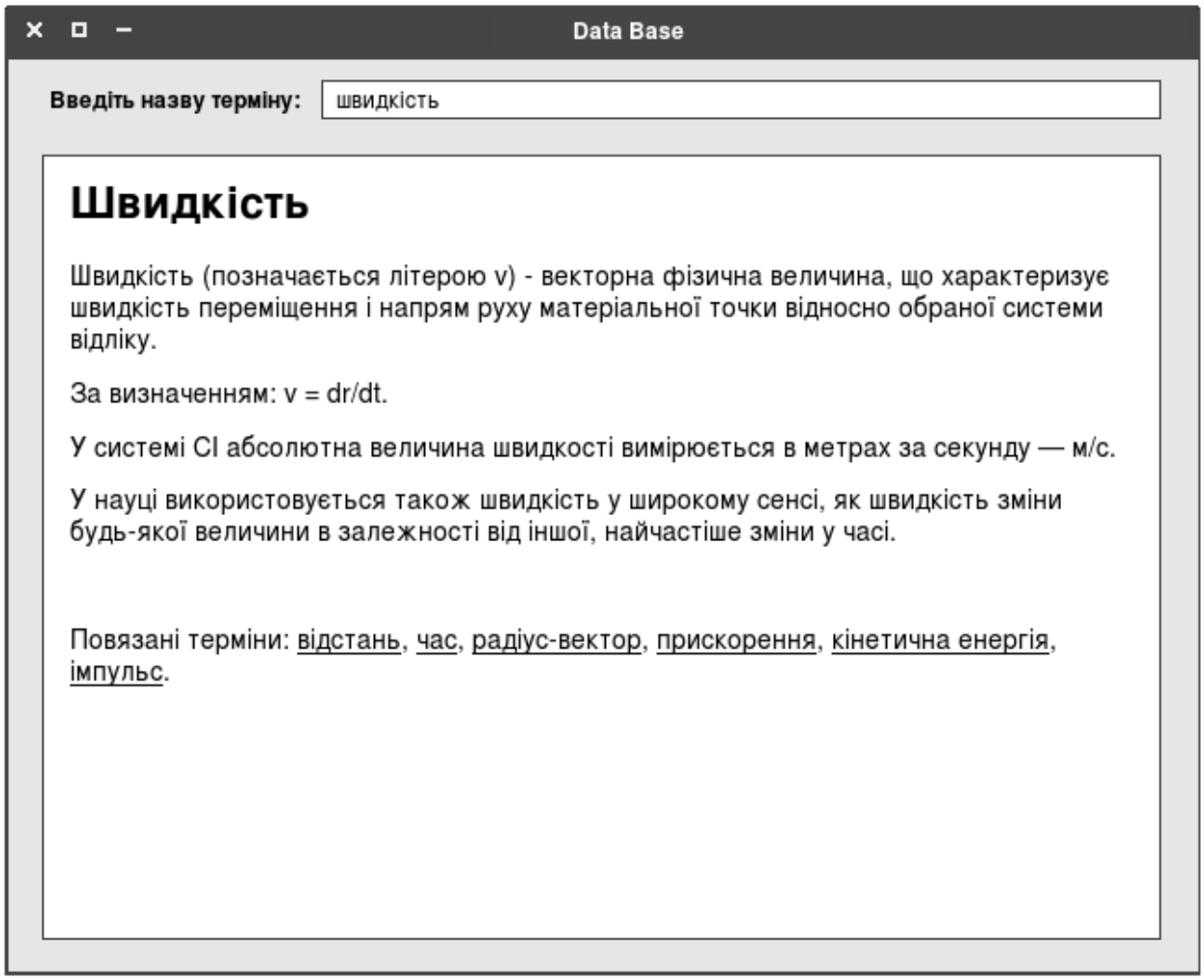

Рисунок 3.4.2 – Приклад вікна програми з наповненням

3.5 Подальший розвиток

В подальших дослідженнях планується розширити базу знань, а також реалізувати можливість проведення автоматизованих розрахунків, генерації формул для спеціалізованих обчислень і додання веб-частини додатку додаванням можливості роботи не лише з віддаленою базою даних, а й необхідності наявності і оновлювання локальної бази даних.

В подальшому, використовуючи результати цієї роботи, планується створити додаток який втілюватиме бачення розробником зручного і інформативного інструменту для зберігання, пошуку і часткового

форматування інформації.

3.6 Висновки

В даному розділі було розроблено і реалізовано демонстраційний семантичний додаток — база знань з класичної механіки, як засобу для створення рекомендації і прикладу використання цих рекомендацій.

Обрана структура онтології є достатньою для задоволення вимог до ПП. Не зважаючи на негативний вплив на швидкодію програми при великій онтології. Тому, при розширенні функціоналу, ця структура має бути, в подальшому, вдосконалена до рівня, який задовольнить нові потреби в деталізації та швидкодії додатку. Необхідно відзначити, що для заповнення бази знань в системі повинні використовуватися інформація з достатньо надійних джерел. Дані з джерел мають бути перевірені на достовірність.

Під час реалізації було сформульовано рекомендації для майбутніх розробників щодо створення семантичних веб-додатків з використанням мов опису логічних правил, цілі було досягнуто.

Створено рекомендації для майбутніх розробників щодо створення семантичних веб-додатків з використанням мов опису логічних правил. Вирішення поставленої задачі було виконано за допомогою мови програмування SWI-Prolog, що має набір інструментів, який дозволяє повністю виконати весь обсяг робіт і є реалізацією однією з популярних освітніх мов програмування. Планується подальше покращення результатів цієї роботи.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Вступ

У даному розділі проводиться оцінка основних характеристик програмного продукту, семантичного додатку — бази знань з класичної механіки. Програмний продукт був розроблений за допомогою мови логічного програмування SWI-Prolog.

Програмний продукт призначено для використання на персональних комп'ютерах під управлінням операційної системи Linux-Ubuntu.

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.

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 техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестаціонарних процесів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

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 F0 – розробка програмного продукту. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F1 – вибір мови програмування;

F2 – наповнення бази даних;

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F1:

а) мова програмування SWI-Prolog;

б) мова програмування Java.

Функція F2:

а) наповнення вручну.

б) автоматизоване наповнення.

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).

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

Таблиця 4.2.2.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	Переваги	Недоліки
F1	А	Краще підходить для малих проектів	Складність освоєння
	Б	Краще підходить для великих проектів	Складність освоєння
F2	А	Доцільно для одноразового використання	Необхідність перебудови при багаторазовому використанні
	Б	Доцільно для багаторазового використання	Нераціональні витрати ресурсів для одноразового використання

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. Ці варіанти відзначені у морфологічній карті.

Функція F1:

Оскільки проект є малим доцільніше використовувати варіант а), до того ж недоцільно спеціально вивчати іншу мову програмування, тому варіант б) має бути відкинутий.

Функція F2:

Для демонстраційного дотатку буде достатньо ручного моделювання. Тому зараз варіант б) має бути відкинутий

Таким чином, будемо розглядати такі варіанти реалізації ПП:

- F1а – F2а
- F1б – F2а

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 – швидкодія програми;
- X2 – об’єм пам’яті для збереження даних;
- X3 – точність результату;
- X4 – потенційний об’єм програмного коду;
- X5 – час розрахунку значення функції.

X1: Відображає швидкодію операцій залежно від обраної евристики.

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 наскільки велике відхилення знайденого шляху від ідеального.

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

X5: Відображає час , який витрачається на розрахунок значення функцій.

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**
Швидкодія програми	X1	мс	6000	4200	800
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Точність результату	X3	%	10	5	0
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1000	500
Час розрахунку значення функції	X5	мкс	200	70	20

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.6.

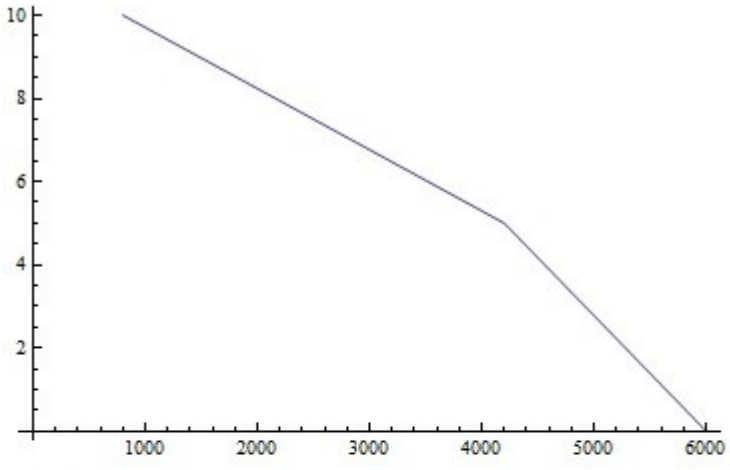

Рисунок 4.1 – X1, швидкодія програми

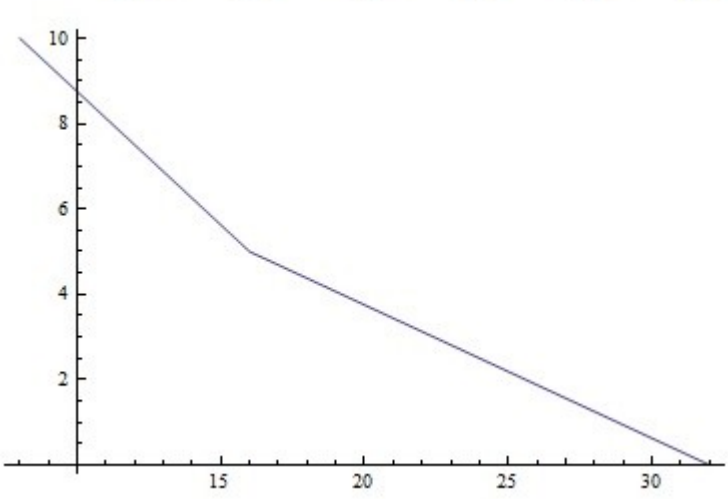

Рисунок 4.2 – X2, об’єм пам’яті для збереження даних

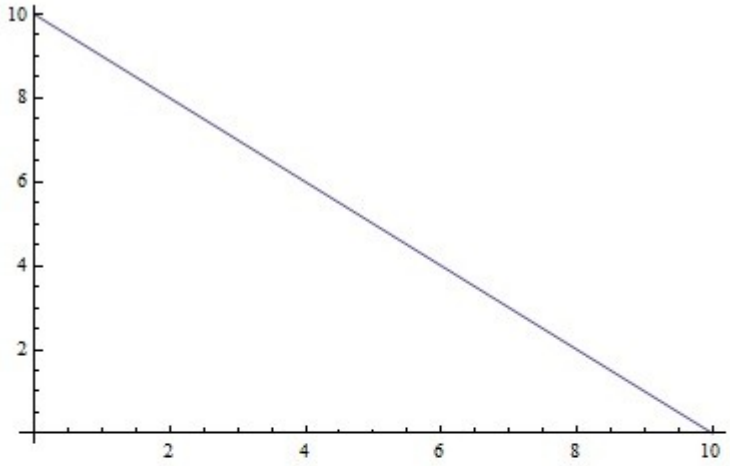

Рисунок 4.3 – X3, точність результату

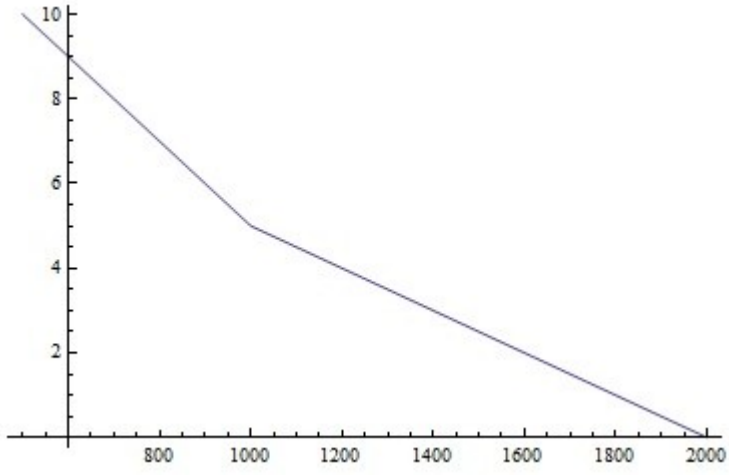

Рисунок 4.4 – X4, потенційний об’єм програмного коду

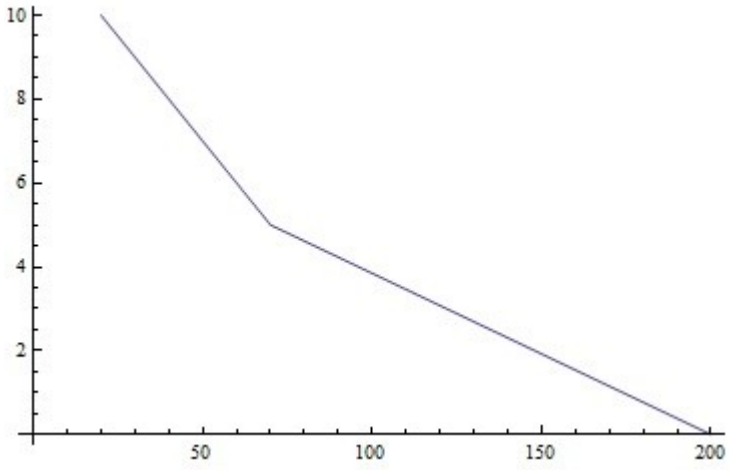

Рисунок 4.5 – X5, час розрахунку значення функції

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.

Таблиця 4.3

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	Швидкодія програми	мс	5	6	5	6	5	5	6	38	17	89
X2	Об'єм пам'яті для збереження даних	Мб	3	2	3	3	3	4	2	20	-1	1
X3	Точність результату	%	2	2	3	2	2	2	3	16	-5	25
X4	Потенційний об'єм програмного коду	кількість строк коду	3	2	2	2	3	2	1	15	-6	36
X5	Час розрахунку значення функції	мкс	2	3	2	2	2	2	3	16	-5	25
Разом			5	5	5	5	5	5	5	105	0	376

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

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

де N=7 – число експертів, n=5 – кількість параметрів;

б) середня сума рангів:

$$\mathbf{T=\frac{1}{n}R_{ij}=21.} \tag{2}$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\mathbf{\Delta_i=R_i-T} \tag{3}$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$\mathbf{S=\sum_{i=1}^{N}\Delta_i^2=376.} \tag{4}$$

Порахуємо коефіцієнт узгодженості:

$$\mathbf{W=\frac{12S}{N^2\left(n^3-n\right)}=\frac{12\cdot 376}{7^2\left(5^3-5\right)}=0{,}767>W_k=0{,}67} \tag{5}$$

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	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X1 і X5	>	>	>	>	>	>	>	>	1,5
X2 і X3	>	=	=	>	>	>	<	>	1,5
X2 і X4	=	=	>	=	=	>	>	=	1
X2 і X5	>	<	>	>	>	>	<	>	1,5
X3 і X4	<	=	>	=	<	=	>	=	1
X3 і X5	=	<	>	=	=	=	=	=	1
X4 і X5	>	<	=	=	>	=	<	=	1

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається за формулою:

$$\mathbf{a_{ij}} = \begin{cases} \mathbf{1.5\ при\ X_i > X_j} \\ \mathbf{1.0\ при\ X_i > X_j} \\ \mathbf{0.5\ при\ X_i > X_j} \end{cases} \qquad (6)$$

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

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$\mathbf{K_{в\,i}} = \frac{\mathbf{b_i}}{\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{b_i}}, \qquad (7)$$

$$\text{де } \mathbf{b_i} = \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{a_{ij}}. \qquad (8)$$

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:

$$\mathbf{K_{в\,i}} = \frac{\mathbf{b_i'}}{\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{b_i'}}, \qquad (9)$$

$$\text{де } \mathbf{b_i'} = \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{a_{ij}} \mathbf{b_j}. \qquad (10)$$

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j					Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	X5	b_i	$K_{вi}$	b_i^1	$K_{вi}^1$	b_i^2	$K_{вi}^2$
X1	1,0	1,5	1,5	1,5	1,5	7,0	0,28	34	0,287	160,75	0,287
X2	0,5	1,0	1,5	1,0	1,5	5,5	0,22	25,5	0,215	120,25	0,215
X3	0,5	0,5	1,0	1,0	1,0	4,0	0,16	18,75	0,158	88,75	0,158
X4	0,5	1,0	1,0	1,0	1,0	4,5	0,18	21,5	0,181	101,5	0,181
X5	0,5	0,5	1,0	1,0	1,0	4,0	0,16	18,75	0,158	88,75	0,158
Всього:						25	1	118,5	1	560	1

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

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) та *X1* (швидкодія програми) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*точність результату) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 800 мс або варіанту б) 80мс.

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

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

де n – кількість параметрів; $K_{вi}$ – коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

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	А	500	6,8	0,191	1,241
	Б	800	10	0,277	2,850
F2	А	3	7,2	0,168	1,147

За даними з таблиці 4.6 за формулою

$$\mathbf{K_K = K_{TY}\left[F_{1k}\right] + K_{TY}\left[F_{2k}\right] + ... + K_{TY}\left[F_{zk}\right]}, \quad (2)$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 1,241+1,147=2,388;

K_{K2} = 2,850+1,147=3,997.

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

4.5 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка логічної частини програмного продукту;

2. Розробка візуальної частини програмного продукту.

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 3.

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.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М} (5.1)$$

де T_P – трудомісткість розробки ПП; $K_П$ – поправочний коефіцієнт; $K_{СК}$ – коефіцієнт на складність вхідної інформації; $K_М$ – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =27 людино-днів, $K_П$ =0.9, $K_{СК}$ = 1, $K_{СТ}$ =0.8:

$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

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

$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328{,}64$ людино-годин;

T_{II} = (122.4 + 19.44 + 6.91 + 19.44) · 8 = 1345.52 людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 20000 грн., один фінансовий аналітик з окладом 25000 грн. Визначимо зарплату за годину за формулою:

$$\text{С Ч} = \frac{\text{М}}{\mathbf{T_m \cdot t}}\ \mathbf{грн.}, \tag{1}$$

де М – місячний оклад працівників; T_m – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$\text{С Ч} = \frac{\mathbf{20000 + 20000 + 25000}}{\mathbf{3 \cdot 21 \cdot 8}} = \mathbf{128{,}97\ грн.} \tag{2}$$

Тоді, розрахуємо заробітну плату за формулою

$$\text{С ЗП} = \mathbf{C_ч \cdot T_i} \cdot \text{К Д}, \tag{3}$$

де $С_ч$– величина погодинної оплати праці програміста; T_i – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 128,97· 1128,64 · 1.2 = 155622,86 грн.

II. $С_{ЗП}$ = 128,97· 1145.52 · 1.2 = 158238,06 грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $С_{ВІД} = С_{ЗП}$ · 0.22 = 158238,06 · 0.22 = 64407,53 грн.

II. $С_{ВІД} = С_{ЗП}$ · 0.22 = 208238,06 · 0.22 = 65569,13 грн.

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

Так як одна ЕОМ обслуговує одного програміста з окладом 20000 грн., з

коефіцієнтом зайнятості 0,2, то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 20000 \cdot 0,2 = 48000$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 48000 \cdot (1 + 0.2) = 57600$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 57600 \cdot 0,22 = 20179,52$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 30000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 30000 = 8625$ грн.,

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 як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 30000 \cdot 0.05 = 1725$ грн.,

де $К_Р$ –відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

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

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 1,94 = 1057,58$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнт зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 30000 \cdot 0,67 = 20100$ грн.

Тоді, річні експлуатаційні витрати будуть:

$C_{ЕКС} = C_{ЗП} + C_{ВІД} + C_А + C_Р + C_{ЕЛ} + C_Н$

$C_{ЕКС} = 57600 + 20179,52 + 8625 + 1725 + 1057,58 + 20100 = 109287$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$C_{М\text{-}Г} = C_{ЕКС} / T_{ЕФ} = 109287/1706,4 = 64,05$ 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:

$C_М = C_{М\text{-}Г} \cdot T$

I. $C_М = 64,05 \cdot 1328,64 = 85404,98$ грн.;

II. $C_М = 64,05 \cdot 1345,52 = 86490,03$ грн.

Накладні витрати складають 67% від заробітної плати:

$C_Н = C_{ЗП} \cdot 0,67$

I. $C_Н = 205622,86 \cdot 0,67 = 137767,32$ грн.;

II. $C_Н = 208238,06 \cdot 0,67 = 139519,50$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н$

I. $C_{ПП} = 205622,86 + 64407,53 + 85404,98 + 137767,32 = 493202,69$ грн.

II. $C_{ПП} = 208238,06 + 65569,13 + 86490,03 + 139519,50 = 499816,72$ грн.

4.6 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$K_{ТЕРj} = K_{Кj} / C_{Фj}$,

$K_{ТЕР1} = 3,997 / 493202,69 = 0,80 \cdot 10^{-5}$;

$K_{ТЕР2} = 2,388 / 499816,72 = 0,48 \cdot 10^{-5}$.

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{ТЕР1} = 0,80 \cdot 10^{-5}$.

4.7 Висновки

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

$К_{ТЕР}$ = 0,80·10-5.

Функція F1: а) мова програмування SWI-Prolog;

Функція F2: а) наповнення вручну.

Даний варіант виконання програмного комплексу дає користувачу необхідну швидкодію без втрат точності та зміни кількості коду.

ВИСНОВКИ

В цій роботі була вивчена предметна область, розроблені рекомендації щодо створення семантичних веб-додатків.

Концепція семантичної павутини має на меті зробити існуючий інтернет більш машинозчитуваним з тим, щоб забезпечити достатню гнучкість для можливості подання всіх баз даних і правил логіки таким чином, щоб зв'язати їх всі разом. Однак опис Semantic Web полягає в тому, що він являє собою спробу реалізувати машинну обробку даних зокрема, трансформувати обробку інформації забезпеченням загального принципу, згідно з яким дані можуть бути отримані, пов'язані один з одним і зрозумілі. Переклад мережі інтернет від типу «великий книги з гіперпосиланнями» до великої пов'язаної базі даних.

Ця робота включає створення демонстраційного додатку, як прикладу для формування рекомендацій. Темою цієї частини було обрано базу знань з класичної механіки. Обрана структура онтології є задовільною щодо вимог до додатку, не зважаючи на погіршення швидкодії програми при великій онтології. Тому, при розширенні функціоналу, ця структура має бути, в подальшому, вдосконалена до рівня, який задовольнить нові потреби в деталізації та швидкодії програми. Відзначимо, що для заповнення бази знань в системі повинні використовуватися інформація з достатньо надійних джерел. Дані з джерел мають бути перевірені на достовірність. Для створення цього додатку було використано мову логічного програмування SWI-Prolog, яка має комплексний набір інструментів для вирішення поставлених у цій роботі задач і є популярною освітньою мовою програмування.

На основі кроків виконаних під час створення прототипу веб-додатку було складено рекомендації і застереження щодо створення семантичних додатків. В подальших дослідженнях планується розширити базу знань, а також реалізувати веб-частину додатку з підтримкою можливості оновлювання

локальної бази даних або роботи лише з віддаленою базою даних.

Також базуючись на результаті цієї роботи планується створити додаток який, в подальшому, втілюватиме бачення розробником зручного і інформативного інструменту для зберігання, пошуку і часткової обробки/форматування інформації.

СПИСОК ЛІТЕРАТУРИ

1. Верхова О. Г. Семантична павутина і пошукова оптимізація: особливості взаємодії / Верхова О. Г., Іванова Н. О. // Постулат. – 2016. – № 6.
2. Н.С. Константинівна Онтології як системи зберігання знань / Н.С. Константинова, О.А. Мітрофанівна
3. Андрєєв А.М., Березкін Д.В., Римар В.С., Симаков К.В. Використання технології Semantic Web в системі пошуку невідповідностей в текстах документів. – Режим доступу: http://www.inteltec.ru/publish/articles/textan/rimar_RCDL206.shtml – Дата доступу: 21.05.2017
4. Официальный сайт Semantic Web – Режим доступу: http://www.w3.org/2001/sw/ – Дата доступу: 15.05.2017
5. Ритор М.В. Розробка високоефективних засобів створення додатків Semantic Web. / Ритор М.В., Березін В.В., Накумов Н.Н. – К. :ОИ-Луч, 2010. – 146 с.
6. Петренко А.І. Електронна охорона здоров'я (eHealth) / Петренко А.І. Системний аналіз та інформаційні технології : 18-а міжнародна науково-технічна конференція «САІТ-2016», 30 травня – 2 червня 2016, Київ, Україна : матеріали. – К. : НТУУ «КПІ», 2016. – С.17.