Хмельницький національний університет
Факультет програмування та
комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерної інженерії та системного програмування

ДИПЛОМНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення»

ДРКІСПр. 015069.19.01.01 ПЗ

Виконав: студент 2 курсу, група КІм-19-1 Мартинюк Ю.С.
Підпис Ініціали, прізвище

Керівник доктор техн. наук, професор Говорущенко Т.О.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІ, д.т.н., проф.
Т.О. Говорущенко
__ _____ 2021 р.

Хмельницький, 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем

Кафедра Комп'ютерної інженерії та системного програмування

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітньо-наукова програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 07 " 09 2020 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Мартинюку Юрію Сергійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення

Керівник проекту (роботи) Говорущенко Т.О., д.т.н., професор

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.01.2021 р. № 7

2. Строк подання студентом проекту (роботи) на кафедру 25.05.2021 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз відомих методів і рішень

Моделювання процесу пошуку інформації для галузі якості програмного забезпечення на основі онтологій

Метод пошуку інформації для галузі якості програмного забезпечення на основі онтологій

Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., доцент кафедри КІСП		
Антиплагіат	Нічепорук А.О., старший викладач кафедри КІСП		

7. Дата видачі завдання «_07_»_____09_____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики ДРМ з керівником	07.09.2020	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	05.10.2020	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	09.11.2020	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	07.12.2020	виконано
5	Робота над науковою статтею	05.01.2021	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2021	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	05.04.2021	виконано
8	Оформлення пояснювальної записки згідно вимог	19.04.2021	виконано
9	Попередній захист ДРМ	05.05.2021	виконано
10	Захист ДРМ на засіданні ЕК	Травень 2021 року	

Студент _____ Мартинюк Ю.С.
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Говорущенко Т.О.
Підпис Ініціали, прізвище

РЕФЕРАТ

Тема дипломної роботи: Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення

Автор роботи: Мартинюк Ю.С.

Керівник роботи: Говорущенко Т.О.

Пояснювальна записка: 92 с., 14 рис., 4 табл., 2 дод., 69 джерел.

ПЕРЕЛІК КЛЮЧОВИХ СЛІВ: інтелектуалізована система, штучний інтелект, семантика, онтологія, якість, програмне забезпечення

Об’єктом дослідження є процеси опрацювання інформації щодо якості програмного забезпечення, технологія виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

Предметом дослідження є моделі, метод та інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення.

Метою дипломної роботи є підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення шляхом розроблення інтелектуалізованої системи на основі онтологій.

Для розв’язання поставлених задач використовувались принципи системного аналізу (ієрархічності, декомпозиції та ін.), методи аналізу та моделювання процесів, теоретико-множинні підходи, алгебра систем, методи онтологічного моделювання, апарат модельно-орієнтованих підходів, методи концептуального моделювання, загальні принципи створення інформаційних систем та систем підтримки прийняття рішень. Для побудови та візуалізації онтологій було використане вільно поширюване ПЗ Protégé 4.2.

Наукова новизна отриманих результатів:

− набули подальшого розвитку моделі процесу опрацювання інформації для галузі якості програмного забезпечення, які ґрунтуються на онтологіях предметної

галузі якості програмного забезпечення та є теоретичним підґрунтям методу пошуку інформації для галузі якості програмного забезпечення на основі онтологій;

− набув подальшого розвитку метод опрацювання інформації щодо якості програмного забезпечення на основі онтологій, який працює на основі розроблених онтологічних моделей та забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

На основі проведених досліджень розроблена інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення, яка забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації щодо якості програмного забезпечення.

Практична значущість отриманих результатів полягає у розробленні інтелектуалізованої системи на основі онтологій для галузі якості програмного забезпечення, яка забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації щодо якості програмного забезпечення.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ - програмне забезпечення

БД - база даних

ІПС – інформаційно-пошукова система

АІ(ШІ) – штучний інтелект

СЯ – семантичне ядро

ВСТУП

На сьогодні, відомі системи не здатні забезпечити сучасних потреб користувачів. Загальносвітовим трендом щодо опрацювання великих масивів інформації, що дозволяє вирішувати нові класи задач на основі наявних інформаційних ресурсів, є інтелектуалізація опрацювання інформації і даних. Ефективним інструментом для організації семантичного пошуку є онтології. Враховуючи той факт, що програмне забезпечення (ПЗ) є основою усіх сучасних напрямків господарської діяльності, а досягнення високих значень показників його якості є ключовим фактором забезпечення ефективного застосування ПЗ та однією із основних вимог користувачів і зацікавлених осіб до сучасного ПЗ, метою даного дослідження є підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення шляхом розроблення інтелектуалізованої системи на основі онтологій.

Поставлена мета досягається розв’язанням таких основних задач:

– проаналізувати відомі методи та рішення для опрацювання інформації щодо якості програмного забезпечення, технології виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення;

– розробити моделі процесу опрацювання інформації для галузі якості програмного забезпечення;

– розробити метод опрацювання інформації щодо якості програмного забезпечення на основі онтологій;

– спроєктувати та реалізувати інтелектуалізовану систему на основі онтологій для галузі якості програмного забезпечення.

Об’єктом дослідження є процеси опрацювання інформації щодо якості програмного забезпечення, технологія виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

Предметом дослідження є моделі, метод та інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення.

Наукова новизна отриманих результатів:

− набули подальшого розвитку моделі процесу опрацювання інформації для галузі якості програмного забезпечення, які ґрунтуються на онтологіях предметної галузі якості програмного забезпечення та є теоретичним підґрунтям методу пошуку інформації для галузі якості програмного забезпечення на основі онтологій;

− набув подальшого розвитку метод опрацювання інформації щодо якості програмного забезпечення на основі онтологій, який працює на основі розроблених онтологічних моделей та забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

Практична значущість отриманих результатів полягає у розробленні інтелектуалізованої системи на основі онтологій для галузі якості програмного забезпечення, яка забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації щодо якості програмного забезпечення.

Для розв’язання поставлених задач використовувались принципи системного аналізу (ієрархічності, декомпозиції та ін.), методи аналізу та моделювання процесів, теоретико-множинні підходи, алгебра систем, методи онтологічного моделювання, апарат модельно-орієнтованих підходів, методи концептуального моделювання, загальні принципи створення інформаційних систем та систем підтримки прийняття рішень. Для побудови та візуалізації онтологій було використане вільно поширюване ПЗ Protégé 4.2.

Дослідження, представлені у кваліфікаційній роботі, проводились в рамках держбюджетної НДР Хмельницького національного університету № 1Б-2019 «Агентно-орієнтована система підвищення безпеки та якості програмного забезпечення комп’ютерних систем» (номер державної реєстрації 0119U100662).

За темою дипломної роботи опублікована одна стаття у фаховому науковому виданні Computer Systems and Information Technologies [1].

1 АНАЛІЗ ВІДОМИХ МЕТОДІВ І РІШЕНЬ

1.1 Дослідження відомих підходів щодо оцінювання якості програмного забезпечення

На даний час, основними можна вважати два підходи щодо оцінювання якості та нефункційних характеристик-складових якості ПЗ. А саме – модель SQuaRe (стандарт ISO 25010:2011) і модель, що сформована на основі результатів метричного аналізу. Основна концепція моделі SQuaRe (стандарт ISO 25010:2011) заключається в тому, щоб перш за все, ми, повинні ідентифікувати незначний набір ознак абстракцій найвищого ступеню, а потім в напрямку "згори-донизу" розбити ці рекомендації на підрекомендації та набори підпорядкованих атрибутів [1]. Модель рівнів якості SQuaRe, створена в межах стандарту, визначається 8-ма характеристиками (нефункційними характеристиками-складовими якості ПЗ): функційна придатність (Functional Suitability), ефективність (performance efficiency), сумісність (compatibility), зручність використання (Usability), надійність (Reliability), захищеність (Security), супроводжуваність (Maintainability), можливість переносу (portability). Нижча міра ієрархії представляє собою властивості якості, що визначені та описані у стандарті ISO 25023:2016 [1]. Провівши дослідження над стандартом [1] ми змогли отримати можливість визначити залежність підрекомендацій ступенів якості від 203 атрибутів (в тому числі від 138 різних атрибутів).

Наступним і останнім методом до оцінювання ступенів якості ПЗ є характеристика, що сформована на базі підрахунків метричного аналізу. Сьогоденна програмна індустрія примножила до своїх баз велику кількість метрик, які рецензують окремі експлуатаційні та виробничі характеристики програмного забезпечення. Відповідно до стандарту ISO 24765 [2], метрика визначається мірилом кроку володіння характеристикою, що має числове значення. Якщо ототожнювати то, метрика ПЗ – це міра, яка надає числове значення деякої характеристики ПЗ, як обмірковане середнє арифметичне з заліком усіх характеристик свідчень, що рецензують цю метрику, та коефіцієнти їхньої

важливості. Як результат, після повного аналізу метрик складності та характеристик з точки бачення змоги їх використати на початкових фазах життєвого циклу ПЗ з отриманням точного або прогнозованого розуміння було виведено список метрик, які можуть бути використані під час етапу проектування – 10 метрик з точними значеннями на етапі проектування та 14 метрик з прогнозованими значеннями на етапі проектування [1, 3], які залежать від 72 показників (в тому числі від 42 різних показників) [4, 5].

Достатністю відомостей у специфікації запитів до ПЗ є можливість у специфікації всіх інформаційних органів , необхідних для визначення нефункційних характеристик ПЗ [1, 3].

Оцінювання достатності відомостей специфікації запитів до ПЗ забезпечує допуск вибору програмного проекту, підвищує ефективність керування проектом за фактурою обґрунтованості висновків , скорочує час на їхню розробку і прийняття, збавляє витрати на збір і обробку інформації. Недостатня кількість даних запитів до ПЗ знижує результативність та достовірність оцінювання нефункційних характеристик ПЗ [1]. Програмні запити визначають необхідні нефункційні ознаки ПЗ, і окрім того розширюють методи кількісної характеристики, які були сформульовані для конкретного оцінювання цих ознак критерії приймання. Отож, всю потрібну інформацію вкладено вже у специфікації вимог до ПЗ, і базуючись на запитах до ПЗ можна зробити оцінку відомостей для наступної розробки ПЗ [1, 6, 7, 3]. У випадку, якщо, деякі вимоги не присутні, то це може негативно сказатись на тестуванні ПЗ, тому розробник повинен додати потрібні доповнення в специфікацію [1]. Відповідно до цього, проведений аналіз впливу інформації специфікації вимог на якість і надійність ПЗ показав, що сучасні програмні системи є менш залежними до написання програмного коду, але суттєво залежать від ранніх етапів розробки. Дефекти, внесені на ранніх етапах, варто виявляти та усувати до того, як вони розпочнуть впливати на результати більш пізніх етапів життєвого циклу. Якість та успішність реалізації програмного проекту суттєво залежать від специфікацій вимог до ПЗ, відтак аналіз вимог є надзвичайно важливим та актуальним.

На даний момент було створено відмінні моделі якості з унікальними наборами під характеристик, атрибутів, та характеристик. Вказані моделі спроможні бути вигідні для дискусій, оцінювання та плануванн якості програмного забезпечення. Автори[8] проводять класифікування моделі за рекомендаціями користувача. Вони можуть виділити три повноцінних категорії моделей, відповідно до:

1) організаційного рівня;

2) рівня загального використання або конкретної галузі, що фокусується на задоволенні інтересів конкретної організації;

3) і проектний рівень, котрий застосовується до певного проекту, щоб надавати повноцінну якість;

Після 2000 року розробка програмного забезпечення почала залежати від створених компонентів, що призводить до новіших проблематик в сфері оцінки якості програмного забезпечення.

Аспекти комунікацій є і залишаються одним з основних факторів впливу на якість оцінювання програмного забезпечення [9]. Базові моделі (FURPS, ISO 9126, Mc Call, ISO 25010 Boehm, Dromey) мають структуру ієрархічного типу, тобто, мають всі можливості адаптуватись до програмного продукту довільного типу і орієнтовані на його оцінювання та вдосконалення [9, 180, 214, 11-25].

В моделі Мак-Кола (McCall) був описаний зв'язок між показниками і якісними характеристиками. Її мінусом є низька доцільність, не врахування функціональності і вимірювання якості.

Боем (Boehm) описав великомасштабні характеристики, що мають поліпшення в порівнянні з моделлю Мак-Кола, тому що додають точніші аспекти на різних кроках.

Модель Дромі (Dromey) вказує, що для найвищої якості оцінювання продукту та всіх елементів з якого вони складаються, данні компоненти мають бути найвисокоякіснішніми.

Процес оцінювання якості для кожного продукту має різний тип, тому може використовуватись динамічна оцінка. FURPS в процесах

класифікування характеристики ділиться на нефункціональні та функціональні. Модель 9126 має дві базові частини, що складаються з:

1) якості в атрибутах 72 використання;

2) атрибутів внутрішньої (системні властивості, які можуть бути оцінені без виконання) і зовнішньої (системні властивості, які можуть бути визначені спостереженням під час виконання) якості;

Модель ISO 25010 займається виділенням 8 ключових характеристик. Головною з її цілей є направлення розробки програмного забезпечення на оцінювання і специфікацію вимог [9, 10, 11, 12-16]. Базовим показником моделей адаптованого типу якості (Rawashdeh, GEQUAMO, Bertoa, Alvaro та інших) є те, що вони є специфічним для певної галузі, і також важливою функцією є зміна по відношенню до основної моделі. У відповідності до цих моделей, в значній мірі успіх програмного продукту залежить від якості компонентів.

На моделі ISO 9126 базується модель Бертоа (Bertoa). Її завданням є визначення набору певних атрибутів оцінки якості для ефективної оцінки програмного забезпечення.

Модель GEQUAMO, складається з характеристик і функцій, що були створені для певних інкапсуляційних потреб користувачів в гнучкій та динамічній формі.

Основою для сертифікації програмних компонентів вважається модель Альваро (Alvaro) – в порядку якого встановлюються елементи якісних компонентів.

Модель Равашдеха (Rawashdeh) основна мета якого - описання потреб різних типів користувачів.

Моделі адаптованого типу можуть орієнтуватись на продукт (GECUAMO), чи деякі галузі (Bertoa) або адаптуватись до точки зору користувача (Rawashdeh) [9, 10, 11-12, 17].

Модель для оцінювання якості програмних продуктів, що є безкоштовним, можуть адаптувати моделі, додаючи певні аспекти програмного забезпечення, що

вільно поширюється. Відмітимо, що найкраща модель, котра охоплює всі аспекти атрибутів якості, що вільно поширюється, поки не визначена.

CapGemini Open Source Maturity Model базована на готовності продукту та встановлюється відповідно до атрибутів зрілості. OpenBRR Model була виготовлена під впливом ISO 9126 і CapGemini. В даному контексті можемо визначити категорії, що є важливими для оцінювання безкоштовно поширюваного програмного забезпечення. Вони вміщують в собі сім категорій, що прискорює процес оцінки, 73 з яких забезпечують найкращий вибір з невеликого набору. SQO-OSS модель – це ієрархічна модель, яка може оцінити сирцевий (вихідний) код та деякі спільні процеси, що має в силах забезпечити автоматизування розрахунку показників. Відрізняється ж вона від інших в наступних аспектах: фокусування на автоматизації; відмова від оцінки функціональності; зосередження на вихідному коді; розглядання суспільних факторів, які можуть бути автоматично виміряні ядром системи безперервного моніторингу якості і забезпечує автоматичний збір метрик. Модель QualOSS вказує на те, що рівень якості в сильній залежності від контексту, в якому вона використовується і в цілях, переслідувані компанією [9, 10, 11, 12, 18]. Під час дослідження моделей якості ПЗ було виявлено, що на даний час вони досить слабко вираховують системний аспект сучасного ПЗ (в особливості, адаптовані моделі якості), і саме тому недостатню увагу приділяють ефімерним властивостям програмного забезпечення, котрі є невід'ємною якістю системи. Моделі якості програмного забезпечення (особливо адаптовані) націлені на вдосконалення та визначення якості інших компонентів, але системним аналізом було доведено, що функції даної системи не в стані бути простою сумою концепцій складових системи, а знаходження інтегративних властивостей вважається однією з найважливіших рис системи [19]. Певні з інтегративних властивостей програмних систем відображає невідповідність нефункціональним вимогами, оскільки саме вони мають деякий ступінь суб'єктивності [20], та формулюються на базовому (системному) рівні [20, 22] і відображають інтегративні властивості програмних систем. Моделі якості програмного забезпечення мають забезпечувати можливість прогнозувати якомога більшу

кількості нефункціональних вимог, що дає змогу зменшити негативний вплив інтегративних властивостей. Інші моделі якості враховують не весь перелік нефункціональних вимог.

1.2 Аналіз відомих систем пошуку інформації для галузі якості програмного забезпечення

Як ми знаємо знаходження коректних даних є самим конструктивним елементом людського функціонування. Персоні, яка періодично має діло з підбором і знаходженням певної предметної області в якийсь момент (через збільшення її обсягу) приходиться використовувати деякі регулювання класифікації і систематизації існуючих матеріалів, що надають набагато зручний і ефективний пошук.

Інформаційно-пошукові системи з’явились набагато давніше, ніж ми можемо уявити. Через деякий час і після проведення певних досліджень, ми цілком і точно маємо змогу вказати, про те що World Wide Web (гіпертекстова модель), на даний момент не може розв’язати і вирішити поставлене завдання найшвидшого знаходження даних і матеріалів у великих обсягах з різних джерел. Отже, зараз можна констатувати факт, що досі не існує інакшої методики пошуку даних більш швидкого, як не за допомогою ключових слів.

Інформаційно-пошукова система (ІПС) – це програмна система для пошуку, видачі, і збереження інформації, яку потребує користувач [23]. Інформаційно-пошукові системи – це системи автоматизованого типу, призначені для збереження, пошуку, оброблення, зібрання та видачі даних за підтримкою певних технічних засобів [24]. Користувач посилає дані до ІПС з інформаційним зверненням. Пошук інформації проводиться в спеціальному пошуковому масиві, що створюється(і за потребою оновлюється) адміністраторами чи розробниками системи. Деякі частини цього масиву створеного для пошуку заносять в інформаційно-пошукову систему основною(природною) мовою, і вже після піддаються перекладу на інформаційно-пошукову формальну мову [23].

Інформаційно-пошукові системи – це певний вид систем інформаційного типу, в котрих фінальна обробка даних не вказана за умовчанням. Подібні системи створені для знаходження текстів (фактографічних записів, документів, і т.д.) в базах даних (сховищах) за деякими формальними описами. Саме тому в виконанні ІПС можна вказати два головних кроки: перший – зберігання і збір даних, другий – видача і пошук даних для того хто відправив [25].

Інформаційно-пошукова система (ІПС) – це певна сукупність інформаційного - довідково фонду та деяких методів технічного інформаційного пошуку. В свою чергу, інформаційно-довідковий фонд (ІДФ) – це набір масивів даних (чи інформації), який пов'язаний з ними за допомогою довідково-пошукового апарату. Пошуковий образ документа – це певний текст, який може складатись з деяких лексичних елементів інформаційно-пошукової мови, яка вказує на головний суттєвий зміст документа і створений для реалізації інформаційного пошуку [26]. Ідеальна ІПС має повертати тільки документи, які відповідно релевантні до запиту. Але на практиці це зазвичай не так не працює. Серед найбільш поширених помилок ми можемо виділити мовчання ІПС (тобто, релевантні документи не видаються) та шум ІПС (видання непотрібних чи зайвих документів) [23]. За деякими правилами в ідеальної ІПС точність і повнота повинна становити 100%, а шум, відповідно, становитиме 0 (знайдені лише потрібні документи та не видано жодного зайвого). В системах реального типу даний коефіцієнт повноти досягає 70%, а коефіцієнт який вказує на точність пошуку коливається в досить широких рамках, тобто може навіть знижуватись до 10%. Величини цих коефіцієнтів можуть залежати від багатьох рядів факторів – таких як внутрішні властивості пошукової системи (характеристика і обсяг інформаційного масиву, критерію видачі, інформаційно-пошукової мови), і від інших зовнішніх умов, а саме: здатність користувача коректно вказати свої інформаційні потреби природною мовою, що також залежить від ступеня особливості інформаційних запитів, коректності побудованого конкретного запиту, і також від суб'єктивного уявлення користувача про те, яку інформацію він шукає[23].

Швидкий розвиток інформаційного суспільства неминуче з'єднаний з потребою обробки, збору, та передачі великих об'ємів інформації, що призвело до глобального переходу від суспільства індустріального типу до інформаційного. Інформатизація суспільства – це соціальний глобальний процес, особливості якого полягають в тому, що основними видами діяльності в сфері суспільного виготовлення є накопичення, продукування, збір, зберігання, використання, обробка та передача інформації, виконувані на базі сучасних методів обчислювальної техніки, і на основі різних методів інформаційного обміну. При такій швидкій інформатизації суспільства основна увага наділяється певному комплексу методик, які спрямовані на надання повного використання вичерпного , своєчасного, і достовірного знання у всіх знайомих видах людської діяльності. Інформатизація суспільства направлена на швидке навчання керування інформацією для задоволення власних потреб. Всі процеси, що виконуються у зв'язку з інформатизацією суспільства, впливають не лише прискоренню технічного-науково прогресу, інтелектуалізації усіх видів діяльності людини, а й реалізації якісно оновленого інформаційного середовища. Головними умовами інформаційного суспільства є: якість наявної в обігу інформації, ефективність і передавання та опрацювання, кількість та доступність інформації для кожного [27-30]. Наразі організації повинні застосовувати якнайбільше джерел інформації, для покращення результатів власного бізнесу, і також якості послуг та досвід, які можуть отримати клієнти. Керування інформаційними масивами стало функцією, критично необхідною для бізнесу. Нинішнє матеріальне вироблення та інші сфери діяльності збільшують свою потребу в сферах інформаційного обслуговування, перероблення великих об'ємів інформації [29, 30]. Відомі нам традиційні методи і підходи (які в більшій мірі засновані на рішеннях проаналізованих бізнес сегментів та системах управління базами даних) не мають змоги бути використані до таких скупчень даних. Використання вже відомих способів та методів до опрацювання чималих масивів даних та інформації не до кінця виправдовує очікувань розробників, приводить до перевитрат наявних ресурсів, конфліктів між очікуваннями замовників та отриманими результатами та втрат значущої

інформації [30-34]. Наразі загальносвітовим трендом щодо опрацювання громадних скупчень інформації, яка дає змогу вирішувати нові класи завдань на базі отриманих інформаційних пакетів, є підхід до інтелектуалізації отриманої інформації і даних [35, 36]. Появлення нових і потужних технічних ресурсів для зберігання та обробки великих скупчень інформації, не наявність ефективних засобів та методів для виконання цих масивів даних, розвинення сфери інтелектуалізації обробки інформації є передумовами для вступу на новий якісний рівень обробки інформації [30].

Під час розробки програмного забезпечення, організації, що розробляють їх повинні керуватись певними стандартами по відношенні до процесу розробки, при цьому не забувати про забезпечення і оцінювання якості. Відповідно, усі стандарти надають гарантію в зв'язку між розробником і користувачем, і до цього ж містять в своїй структурі ранішній досвід розробників. Однак, довготривалий час розробки стандартів і їх несвоєчасне оновлення, і пізніше оновлення, як результат, призводять до зупинення розвитку технології, деградація від практичних потреб і зупинка впровадження новацій. Відповідно до роботи [37], ми можемо виділити певні критерії для оцінки стандарту в галузі оцінювання і розробки якості програмного забезпечення:

1) довільність стандартної інтерпретації для оцінки відповідності;
2) властивості (функції) компонентів та систем;
3) визначення цілей та норм для зацікавлених сторін;
4) цілісність та критичність рівнів абстракції;
5) розділення рівнів абстракції у стандарті;
6) відповідність теорії безпеки;
7) відповідність теорії систем та системної інженерії;
8) відповідність людському фактору;
9) можливість зміни та розвитку технологій;
10) значущість керування;
11) можливість багаторазового використання процесу;
12) значущість аналізу вимог до продукції;

13) підтримка незалежної оцінки та сертифікації;

14) значущість етапу експлуатації.

Критерії, що наведені вище, були оцінені з використанням 5-бальної шкали і пізніше було виявлено, що стандарт [38] - має 10 балів, стандарт [39] - 32 бали, [40] - 25 балів.

Цей аналіз дав зрозуміти, що стандарт [39], виданий у 2003 році кращий за стандарт [40], виданий у 2008 році, що є гіршим по базовим критеріям, отже, поновлені стандарти не завжди імплементують ранній досвід розробників і більш інноваційні технології. На даний момент, існує не менш ніж десяток різноманітних підходів до забезпечення якості програмного забезпечення, та в кожен з них має певні недоліки та переваги. Total Quality Management (загальне керування якістю) – це філософія організації, що розробляє програмне забезпечення, котра була створена на прагненні покращити якість програмного забезпечення і на базі 69 практик керування, що призводять до підняття загальної якості програмного забезпечення. Цей підхід до управління процесом життєвого циклу програмного забезпечення, направлений на якість і створений при участі усіх учасників процесу та направлений на досягнення довгострокового успіху, що задовольняє потреби користувача та вигоди для людства в цілому [41, 42]. В загальному ми можемо виділити 7 факторів сумісної якості [41, 42]:

1) командна організація робіт з покращення якості;

2) проведення вимірювань;

3)проблеми, які залежать від працівників, повинні складати не більше 20%;

4) неперервне вдосконалення;

5) керування участю в роботі та відповідальністю;

6) орієнтація на процес та його результати;

7) орієнтація на споживача;

Дані фактори є безпрецедентно коректними як для програмного забезпечення так і для продукту, до того ж підходять для розробки програмного забезпечення як процесу. Головним зусиллям проектів програмного типу є ціль створення програмного забезпечення, що має деяку цінність, тобто є важливим для

вирішення певних задач або досягнення поставлених цілей [43, 44]. Замовник має власне бачення про найвищі вартісні вклади, і також деякі очікування щодо відношення до якості оцінювання програмного забезпечення. Велика кількість рішень відносно якості програмного забезпечення приймається в процесі виробництва з вимогами, але дані питання необхідно піднімати на протязі усього життєвого циклу програмного забезпечення. Варто помітити, що не існує стандартизованих правил, які можуть точно вказати, як слід приймати рішення. В основному, існує ряд інших способів, як досягти іншого рівня якості з різною ціною, тобто може мати місце суб'єктивізм і великий вплив людського фактору [45]. Певні з сучасних стандартів не можуть повністю відповідати вимогам до нинішнього програмного забезпечення, і до того ж усім потребам користувача. Була виготовлена велика кількість стандартів, що по-різному стандартизували та регламентули одні і ті ж самі процеси (як правило, рутинні масові процеси), після чого виникало неповноцінне покривання певних об'єктів стандартизування та несумісність документів, що регулюють норми різних організацій, делегація інтересів окремих організацій, що розробляють програмне забезпечення та адаптація цих стандартів розробниками до власних потреб. Вказані проблеми в галузі оцінки та забезпеченні якості оцінювання програмного забезпечення змушує експертів регулярно розробляти рішення в даній галузі, що призвело до виготовлення великого ланцюжка стандартів та методологій: ISO/IEC 33002 (SPICE) [7], ISO 12207б Capability Maturity Model (CMM) [50]. Окрім того було створено серію стандартів ISO25000 [47] – оновлена генерація нормативних документів, які пов’язані з нормуванням з оцінки якості програмного забезпечення [19, 49]. Був розроблений ланцюжок спеціалізованих моделей оцінки якості для програмного забезпечення. Модель якості ПЗ – це деякий набір характеристик і відношення між ними, які надають базу для визначення оцінки та вимог якості [51, 52]. Дана структура моделі якості може описуватись ієрархією, елементи котрої є множинами під характеристик, характеристик, атрибутів і зв’язків поміж ними. Відповідно до стандарту [51], характеристика – це деякий набір властивостей програмного забезпечення, з допомогою котрих йде опис та оцінка його якості.

Характеристики якості програмного забезпечення можуть бути уточнені на базі певних комплексних під характеристик (показників), що базуються на деякій властивості, що має задовольнити виникаючі або заявлені потреби. Показник оцінювання якості програмного забезпечення виражає себе як середній зважений арифметичний показник з урахуванням значень атрибуту, що оцінює цей показник, та коефіцієнт їхньої важливості. В свою ж чергу, атрибут – це абстрактна або фізична властивість якості оцінювання програмного забезпечення, який може бути виміряним. Рекомендації, котрі увійшли в певну модель, є базою для формування вимог до якості програмного забезпечення. Кожна модель може містить відповідну кількість ієрархічних рівнів і відмінну базову кількість показників якості.

1.3 Дослідження відомих рішень щодо застосування онтологій для галузі якості програмного забезпечення

Як певна норма інженерії знань при розробці інформаційно-пошукових систем доцільно і розумно робити звернення до онтології, які досить широко застосовують в роботі пошукових, роботизованих машин та інформаційно-пошукових систем. Онтологія – це детальна формалізація деякої галузі знань за допомогою концептуальної схеми. Ця схема, в своїй основі , може складатись з структури даних ієрархічного типу, яка містить в собі теореми та обмеження, зв’язки та всі релевантні класи об’єктів, які можуть бути прийняті у деякій предметній галузі. Звернення до онтологій у складі ІПС може допомагти вирішити певний набір проблем технологічного та методологічного виду, які з’являються під час розробки подібних систем. Наприклад, в Україні дані проблеми виникають через відсутність нестачі кваліфікованих фахівців у цій галузі, концептуальної узгодженості й цілісності окремих методів та практик інженерії знань; низької адаптивної здатності; складності впровадження ІПС, жорсткості розроблених програмних засобів та їх, що зумовлено деякими психологічними моментами. Онтології є одним з ефективних інструментів для створення системи семантичного пошуку (тобто для пошуку інформаційних об’єктів та інформаційних ресурсів,

семантична недалекість яких відноситься до інформаційних потреб користувача та виділяється з дотримання знань на рахунок об'єктів та суб'єктів пошукового процесу) [53]. Окрім того, онтології можуть надати можливість з допомогою в пошуку інформаційних ресурсів, які необхідні для користувача який вирішуватиме його поточні завдання з використовуванням знань про цього ж користувача, його закладену сферу інтересів, задачі і досвід інших користувачів з схожими інформаційними потребами [53]. Однією з найперших інформаційно-пошукових систем на базі онтологій – SHOE – надає змогу користувачам вводити запити структурованого типу з використанням онтології [53-56]. Більша кількість ІПС, яку можуть використовувати онтології, були націлені на знаходження інформації лише у Semantic Web (ONTOSEARCH2, Swoogle), це значить , що серед семантично анотованих інформаційних ресурсів [53-56]. ІПС QUICK виконує знаходження даних у всьому інформаційному просторі Web, і працює як метапошукова ІПС, тобто, спочатку отримує запит користувача, після цього, семантично переробляє його та відправляє до зовнішніх інформаційних систем (наприклад, до Yahoo), і лише після цього семантично обробляє результати, які отримала з використанням онтологій [53-56]. Враховуючи даний факт, який вказує, що програмне забезпечення (ПЗ) є базою усіх сучасних напрямків господарської діяльності, а здобуток високих показників його якості є основним фактором надання ефективного використання ПЗ та однією із базових вимог користувачів та зацікавлених осіб до сучасного ПЗ, ціллю даного дослідження є розроблення ефективної інформаційно-пошукової системи (на основі онтологій) для галузі якості програмного забезпечення.

Також важливо доповнити, що атрибути та показники якості програмного забезпечення є одними з ключових проблем, котрі зробили значний вплив на програмне забезпечення. Розглянута проблема серед практиків та дослідників була ключовим фактором у створенні високоякісної конкурентоспроможної мережі або комп'ютера, що забезпечує формування завдань до оцінки і забезпечення якості. Власне кажучи, багато ініціатив, таких як IEEE Standard Releases, ISO/IEC Releases, SPICE (удосконалення програмного забезпечення та визначення можливостей), а

також решта якісних моделей, таких як “Модель якості McCall”, “Модель якості Boehm”, “Модель якості Dromey “ та інші, розглядають якість програмного забезпечення міри за атрибутами повинні бути важливим елементом досягнення вищих рівнів управління якістю програмних програм.

Протягом останніх десятиліть багато подій вплинули на те, як ведеться бізнес. Одним з найважливіших питань, яке було виявлено, - це поява інтернету та впливу глобалізації на людей і бізнес-процесів організацій. Це створило потребу в широкому обміні інформацією та ресурсами для співпраці та ефективної конкуренції на ринку. Для досягнення цієї співпраці створюються стандарти, що забезпечують узгоджені концепції та практики, які змушують учасників уникати невідповідності їх бізнесу. Дослідники в цій сфері пояснили, що не існує єдиного стандарту, який би охоплював область вимірювань програмного забезпечення в цілому, а, скоріше, існує багато різних стандартів, орієнтованих на конкретні сфери, без розгляду всеосяжних рамок. Останнім часом багато зусиль дослідників та інститутів спрямовані на оцінку якості програмного забезпечення.

Онтологія - перед усім, використовується для досягнення зрозумілої уніфікованої семантичної структура для веб- або настільних додатків вимірювання якості ПЗ.

Онтологія, що була придумана в 1613 році, включена до багатьох філософських напрямків від метафізики Арістотеля до Росії об’єктної-теорії Олексія Мейнонга. Філософська онтологія обробляє точне використання слів як дескриптори сутностей; це дає пояснення цих слів які належать до сутностей, і тих, що не належать. В сферах інформатики, онтологія - це представлення набору понять у межах домену та взаємозв'язок між цими поняттями. Це використовується для обмірковування про властивості домену, про який йде мова, і може бути використано для визначення домену.

Нещодавно термін “онтологія” був широко включений в область інформатики. При розробці умовних рамок для подання інформації, даних та розробок систем баз знань були використані широкі спектри термінів та понять. Дослідження показали, що існує проблема невідповідності в семантиці термінів,

які використовується, наприклад, використовуються однакові мітки баз даних, але з різними значеннями, і одне і те ж значення може бути виражене з використанням різних назв. Таким чином, потрібно знайти методи вирішення термінологічних та концептуальних несумісностей. У цьому контексті онтологія - це словник термінів сформульованих в канонічному синтаксисі та з загальноприйнятому визначенні, покликані дати лексичні або таксономічні основи для представлення знань, які можуть ділитися на різні масиви інформаційних систем. Онтології використовуються в різних сучасних галузях, таких як штучний інтелект, програмна інженерія, семантика, біомедична інформатика, бібліотекознавство та інформаційна архітектура, як форма пізнання.

У цю епоху наявність послідовної глобальної інформації стало важливим питанням. У кожному домені дослідники та практикуючі користувачі повинні обмінюватися інформацією, для прискорення роботи один одного. Щоб зробити це правильно повинні бути прибрані невідповідності між термінами та поняттями. Онтологія визначає загальний для них словниковий запас. Це може містити визначення машинних інтерпретацій основних понять у домені та відносинах між ними. Деякі з причини, які спонукають дослідників та практиків розробити онтологію є:

1. Ділення загальним розумінням структури інформація серед людей або програмних агентів.
2. Включити повторне використання знань домену.
3. Зробити явними припущення щодо домену.
4. Відокремити знання домену від операційних знань.

Ще однією причиною використання онтології є те, що вона сприяє зменшення розривів серед дослідників, створених концептуальними плутанинами.

Онтологія демонструє величезний потенціал в Україні роблячи програмне забезпечення більш ефективним, адаптивним та інтелектуальним. Це визнається одним із напрямків, який принесе наступний прорив у розробці програмного забезпечення. Ідея онтології була схвалена візіонерам. Нещодавно семантична веб-ініціатива, на чолі з W3C, змінила ландшафт онтології повністю. Дослідники та

розробники об'єднали зусилля, щоб надати стандартні мові розмітки на основі семантики XML, системи управління онтологією та інші корисні інструменти. Крім того, Інтернет пропонує цікаві програми, що має вирішальне значення для повсякденного життя, такого як пошук та навігація.

1.4 Висновок

Проаналізувавши відомі методи та рішення, були виявлено описані вісім характеристик якості програмного забезпечення, відмінні моделі якості з унікальними наборами під характеристик, атрибутів і характеристик. Окрім того, аналіз показав, що вже існуючі інформаційно-пошукові системи, цілком можуть покрити потреби проектованого ПЗ. Варто додати, що онтології, що вже застосовуються до ІПС, є одним з ефективних інструментів для створення семантичного пошуку, і в майбутньому відіграють важливу роль в ІПС, що розроблюється.

2. МОДЕЛЮВАННЯ ПРОЦЕСУ ПОШУКУ ІНФОРМАЦІЇ ДЛЯ ГАЛУЗІ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ОНТОЛОГІЙ

2.1 Концепція пошуку інформації для галузі якості програмного забезпечення на основі онтологій

Перш за все ми маємо вставновити концепцію інформаційно-пошукової системи (на базі онтологій) для галузей оцінювання якості програмного забезпечення. Відповідно до стандартів ISO SWEBOK , ISO 25030 [57], 25010 [58], давайте розглянемо якість оцінювання програмного забезпечення як можливість данного програмного забезпечення задовольнити потреби, які передбачувані та заявлені, як потреби що виконуються під час його використання при деяких умовах. Дана концепція якості оцінювання ПЗ відповідно до стандартів ISO 25023 [59], ISO 25010 [58] (рисунок 2.1).

Рисунок 2.1 – Сучасна концепція оцінювання якості програмного забезпечення в рамках моделі SquaRE [21]

Явний стандарт є основною частиною збірної частини серій міжнародний стандартів SQuaRE, яка складається з наступних розділів та частин:

1) розділ "Менеджмент якості" (ISO/IEC 2500n);
2) розділ "Модель якості" (ISO 2501n);
3) розділ "Вимірювання якості" (ISO 2502n);
4) розділ "Потреби до якості" (ISO 2503n);

5) розділ "Оцінка якості" (ISO 2504n);

6) розділ "Розширення SQuaRE" (ISO 25050 - ISO 25099);

Для реалізації деяких функцій в бізнесі і в персональних цілях в теперішніх умовах більше розповсюджуються програми для високопроцесних вираховувань. Для покриття задач та цілей чи задоволення власних потреб, успіху в бізнесі або забезпечення безпеки людини, ми маємо повне право розраховувати на якісне програмне забезпечення і системи. Високоякісні програмні продукти і в переважності програмні системи для вираховування мають важливе для зацікавлених сторін значення в розробці цінностей матеріального типу і запобігання результатів які можливо матимуть негативний результат.

В певних програмних розробок і в програмних систем для вимірювання великих об’ємів інформації є досить багато сторін які зацікавлені в даних продуктах. До них ми можемо додати таких спеціалістів як клієнти, користувачі, покупці та розробники. Дана група людей в більшості використовують якраз системи для розрахунків великих об’ємів інформації. Особливо детальна оцінка якості програмного забезпечення та специфікація є ключовими та важливими в забезпеченні певної якості, а потім вже користі для осіб, які проявили інтерес до даної галузі чи продукту. Можем точно вказати, що оцінка може бути виконана на базі визначення характеристик, які були запрошені. Серед них - характеристики якості, яка суміжна з завданнями, які були поставлені сторонами зацікавленими в даному продукті та цілями системи, включаючи описи якості, що відносяться до даних і системи програмного забезпечення. Не варто забувати і про характеристики системи що впливають на зацікавлені сторони. По можливості, важливо, щоб якісні характеристики були виміряні, виведені і оцінені відповідно до широко-використовуваних чи перевірених показників й методів вимірювання. Ідентифікувати відповідні якісні характеристики, котрі можуть дальше використовуватись для виділення критерій, потреб і їх задовільнення відповідно до існуючих показників.

Модернізований сучасний стандарт був виготовлений на базі ISO 9126, що був виготовлений для задовільнення потреб і який виділив шість характеристик якості і описав конкретну модель оцінки програмного продукту.

ISO 9126 змінився двійкою інших пов’язаних між собою стандартами ISO 14598 "Програмна інженерія - оцінка продукту". та ISO 9126 "Програмна інженерія - якість продукту"

Описаний вище стандарт міжнародного типу є результатом перегляду ISO 9126-1. У ньому знаходяться ті ж якісні характеристики програмного забезпечення, але з деякими правками:

1) галузь використання якісних моделей була збільшена, для того щоб включити в свій список системи для вирахування великої кількості інформації і її якості відповідно до системної точки бачення;

2) як одну з характеристик якості, а саме при використанні програмного забезпечення був доданий такий пункт, як - “Покриття контексту” з деякими підхарактеристиками і “Повнота контексту”, ще одном з доданих пунктів була “Гибкість”;

3) не як підхарактеристики, а саме як характеристики функціональності була додана “Безпека” з такими підхарактеристиками, як - “Справжність”, “Безвідказність”, “Цілісність”, “Конфеденційність” та “Знаходження”;

4) ще однією характеристокою була додана “Суміжність” (включаючи існуючу та функціональну суміжність);

5) були додані наступні підхарактеристики: “Доступність”, “Функційна повнота”, “Ємність”, “Захист від від користувацької помилки”, “Готовність”, “Доступність”, “Можливість багатократного використання” та “Модульність”;

6) відповідностні підхарактеристики були видалені, через те що вони відносились до загальних потреб системи, а не якісними характеристиками

7) моделі зовнішньої і внутрішньої якості були об’єднані в складі моделі якості продукта;

8) там де була змога, особливі для програмного забезпечення визначення були замінені на більш універсальні;

9) для деяких підхарактеристик і характеристик були надані більш точні назви;

Вказаний стандарт міжнародного типу був створений для використання в відповідності з іншими частинами міжнародних стандартів, що відносяться до серії SQuaRE (ISO 25000 - ISO 25099) и ISO 14598, до тієї пори, поки вона не замініться серією нових стандартів ISO 2504n.

Серія стандартів SQuaRE включає в себе наступні розділи стандартів:

1. ISO 2500 n - розділ “Менеджмент якості”. Стандарти міжнародного типу, що входять в даний розділ, здатні оприділяти загальні терміни, моделі та визначення, які використовуються надалі у всіх інших міжнародних стандартах серії SQuaRE. В даному розділі також наявні методичні матеріали та вимоги, які відносять до функціоналу категорії “підтримки”, які відповідають за керування потреб до програмного забезпечення, його специфікації та оцінки.

2. ISO 2501n - розділ “Модель якості”. Стандарти міжнародного типу, котрі поміщені в даний розділ, представляють з себе деталізовані якісні моделі систем, що вираховують великі об’єми даних і програмного забезпечення, їхньої якості при використанні і даних які вони обробляють. Окрім того, заявлене практичний посібник по їх використанню.

3. ISO 2502n - розділ “Вимірювання якості”. Стандарти міжнародного типу, котрі поміщені в даний розділ, включають в себе константну чи еталонну модель вимірювання якості програмного продукту, математичні визначення показників якості та практичний посібник, до того як їх використовувати. В вказаному вище розділі наявні показники внутрішні показники програмного забезпечення, а також показники зовнішнього якісного оцінювання програмного забезпечення і показники якості при їх використанні. Окрім того, там вказані та наявні елементи показників якості, що формують базу для показників, що вказані вище.

4. ISO 2503n - розділ “Вимоги до якості”. Стандарти міжнародного типу, котрі поміщені в даний розділ, описують вимоги до якості на основі моделей якості та показників якості. Дані потреби до оцінки якості програмного забезпечення

можуть використовуватись в процесі формування вимог до якості оцінки програмного продукту ще перед розробкою, як вхідні дані для процесу оцінки.

5. ISO 2504n - розділ “Оцінка якості”. Стандарти міжнародного типу, котрі поміщені в даний розділ, формують рекомендації, методичні матеріали і вимоги для оцінки програмного продукту, який виконується як оцінниками так і замовниками чи розробниками. Окрім того, у них представлна підтримка документування показників вимірювання як модуля оцінки.

6. ISO 25050-25099 - розділ "Розширення SQuaRE". Стандарти міжнародного типу, котрі поміщені в даний розділ на даний момент включають в себе вимоги до якості готового комерційного програмного забезпечення та загальному промисловому формату для рецензій про легкість та зручність використання.

Моделі якості вказано міжнародного стандарту разом в сукупності з ISO 12207 та ISO 15288 можуть використуватись для процесів зв’язаних з визначенням вимог та верифікацій і валідацій з нагалошенням на специфікація та оцінювання вимог до якості. В ISO 25030 визначено, яким саме чином моделі якості можливо використовувати для вимог до якості програмного забезпечення, а ISO 25040 описує використання моделі якості в процесі оцінки якості програмного забезпечення.

Найголовнішими перевагами в серії стандартів SQuaRE ми можемо вказати вище перечислені базиси, котрі можуть повністю забезпечувати координацію методологій з вимірювання чи оцінці якості програмного забезпечення, існування вимог до якості від керівництва по специфікації вимог до оцінки якості програмного забезпечення та об’єднання з стандартом ISO/IEC 15939:2007 у вигляді константної форми вимірювань якості.

Організацію та взаємозв’язок серії SQuaRE можна переглянути на рисунку 2.2 [56,57]. Даний перелік стандартів SQuaRE розділений на підрозділи (групи), які в сукупності утворюють модель SQuaRE, що описані у таблиці 2.1.

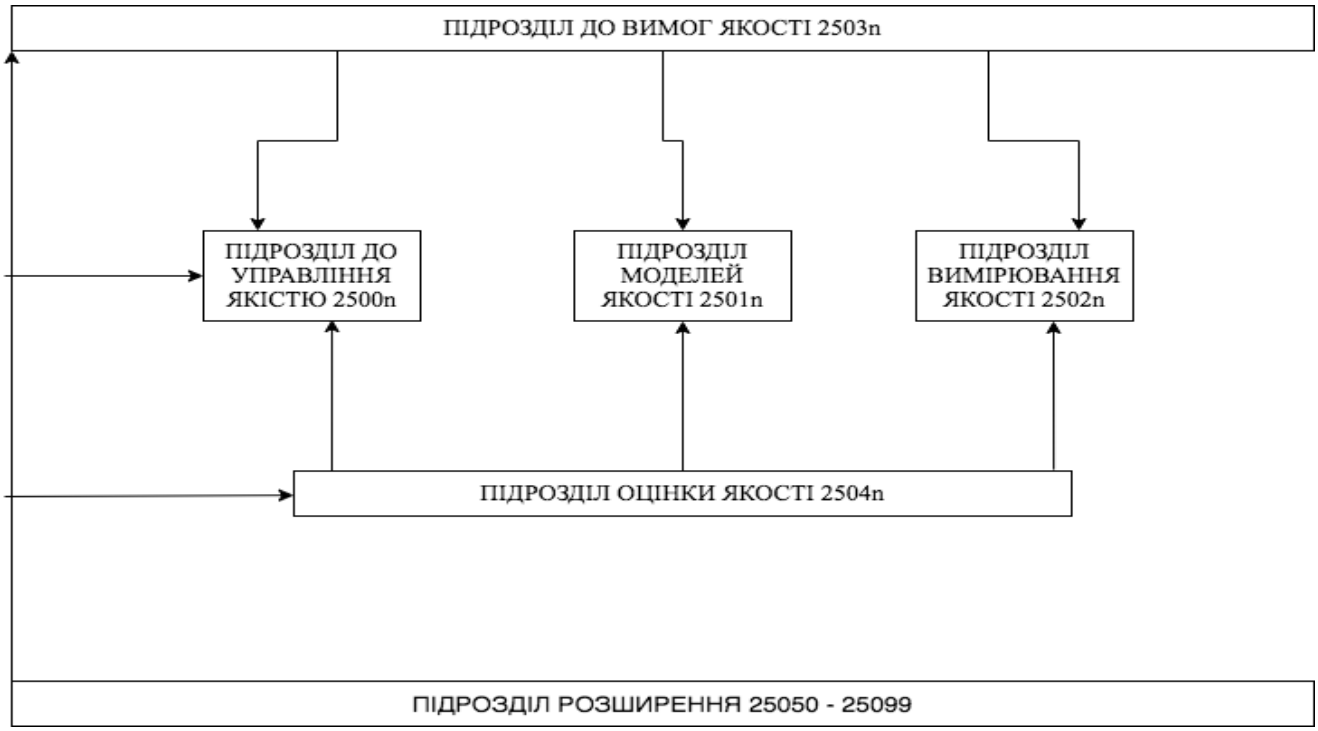


Рисунок 2.2 – Організація та взаємозв’язок серії SquaRE

Таблиця 2.1 – Підрозділи, що в сукупності утворюють модель SQuaRE

Найменування підрозділу	Цілі	Коментар
ПІДРОЗДІЛ ДО УПРАВЛІННЯ ЯКІСТЮ 2500n	Виділено базові моделі, визначення та терміни, які зазвичай використовують в інших реєстрах SQuaRE	Має в наявності методичні вказівки по використанню даних реєстрів
ПІДРОЗДІЛ МОДЕЛЕЙ ЯКОСТІ 2501n	Описано повноцінні моделі опису якості для систем комп'ютерного типу та іншого програмного забезпечення	Має в наявності методичні вказівки по використанню усіх існуючих моделей якості
ПІДРОЗДІЛ ВИМІРЮВАННЯ ЯКОСТІ 2502n	Описано модель для вимірювання якості програмного забезпечення, вказані визначені математичні міри якості і методичний вказівник, для їхнього використання	Має в наявності приклади зовнішніх чи внутрішніх програмних розробок і систем, та, окрім цього певні міри якості в використанні

Кінець таблиці 2.1 – Підрозділи, що в сукупності утворюють модель SquaRE

ПІДРОЗДІЛ ДО ВИМОГ ЯКОСТІ 2503n	Описані методики для визначення потреб до якості, базовані на моделях і обсягу якості	Вказують на те, що вимоги та потреби до якості оцінювання програмного забезпечення в здатності використовуватись під час процесу виявлення вимог до якісного розробленого програмного продукту, або як дані, що надаються для процесу оцінки
ПІДРОЗДІЛ ОЦІНКИ ЯКОСТІ 2504n	Описані потреби, керівництва та рекомендації на рахунок оцінки програмного забезпечення замовниками чи розробниками	Представлений список правил для опису документації міри у вигляді компоненту оцінки
ПІДРОЗДІЛ РОЗШИРЕННЯ 25050 - 25099	Описані потреби та вимоги до певного рівня якості коментарів до програмного забезпечення (англ.: Commercial off the shelf Software; COTS)	-

Властивості, що вимірюються, базуються на якості продукту чи системи, називаються “властивостями якості” в ISO/IEC 25010:2011. Вони ж і об’єднуються (асоціюються) з рівнями якості. Відповідно до стандарту, для того щоб, отримати рівень підхарактеристики чи характеристики якості без вимірювання даних, нам необхідно:

1. Отримати сукупуність властивостей, котрі покривають в групі підхарактеристику чи характеристику;
2. Визначити рівень якості для властивостей (кожної);
3. Згрупувати отримані рівні використавши обчислення для того, щоб отримати результуючий рівень якості, що пов’язаний з під характеристикою чи характеристикою якості.

В зв’язці з ISO 15504, котрий відноситься до оцінки процесів програмного забезпечення, теперішній міжнародний стандарт забезпечує:

1) основи визначення якості програмного продукту в процесах “виконавець – отримувач;
2) підтримку аналізу, верифікацій та валідації та основи масової оцінки якості в процесах підтримки;
3) підтримку налаштування цілей якості в процесі керування організацією;
4) теперішній стандарт може бути використаний в зв’язку з ISO 9001,
5) підтримка визначення цілі якості;
6) підтримки аналізу, верифікації та валідації проекту.

Базовими першоджерелами інформації на періоді вибудовування і з'ясування вимог до ПО бізнес-зобов'язання клієнта, вимоги об'єктної галузі, нормативи, викладу процесу реалізації та введення аналогічного ПО і т.д., які налічують кластер вхідної інформації (АПР), що характеризує підсистеми майбутнього програмного забезпечення, а також його явища і регулювання. Бізнес-вимоги характеризують завдання формування схеми, параметрі звершення цієї мети, засадничі дотримання до підсумків і їх критерії та регулювання. Вивчимо детальніше інформацію, яка може входити до складу АПР на зразку приписів до

реєстраційної Enterprise-системи. Вся специфікація на цей продукт полягала з загальної частини, приписів, викладу розробки, сканування, посібників і планування. Загальна частина специфікації поділялась з двох підрозділів: перелік концептів і їх понять і викладу менеджмент-функцій користувачів. Будь-яка специфікація по системі, включаючи, наприклад, демонстраційні сценарії, відштовхувалася на наведені в цьому підрозділі поняття. У переліку термінів реєстраційної Enterprise-системи потрапило близько 200 бізнес- і структурних понять. Перелік менеджмент-функцій застосовується для актуалізації підгруп і функцій користувачів, призначення структурних прав, він потрібен тестувальникам, щоб перевіряти сценарій під необхідними ролями. Визначення ролей було дано на рівні символьної конфігурації базових функцій співробітників.

Підрозділ вимог специфікації реєстраційної Enterprise-системи включав бізнесвимоги (загальні сценарії, сценарії застосування, механізми повторного огляду), структурні вимоги, нефункційні вимоги, вимоги до переорієнтації, вимоги до контролера. Бізнес-вимоги описували те, що потрібно бізнес-користувачам. Наприклад, їм абсолютно не потрібен предмет системи «Користувач», але зате їм потрібно мати здатність змінювати вартість продукту в платежі і роздрукувати його. Бізнес-вимоги поділялися з загальних сценаріїв, сценаріїв застосування (use cases) і опису механізмів обробки даних. Вимоги були продемонстровані у вигляді дерева (з періодами). Тобто загальні сценарії уточнювалися сценаріями застосування, які, в свою чергу, мали виноски на перевірки і механізми. Семантична сторінка дерева вимог до реєстраційної Enterprise-системи поділялась з загальних сценаріїв, кожен з яких характеризував один з 24 бізнес-процесів, що підлягають розробки. Загальний сценарій - це черговість кроків користувача і системи для звершення відповідного завдання, основне завдання його - систематизувати сценарії впровадження і побачити, що бажає зробити користувач, і як система йому в цьому сприяє. Загальні сценарії також поміщали кроки, які користувач реалізував поза системою, оскільки потрібно було характеризувати його діяльність у всій цілісності, з усіма витками, потрібними для звершення бізнес-цілей.

Сценарій впровадження поміщав помічені кроки. Кожен крок - це, як правило, просту заяву в повному. При складанні механізмів аналітик намагався їх охарактеризувати ніж можна найбільш повно. Втім отримані тексти виявлялися погано читаються, і, як правило, все одно якісь подробиці позбавлялися. Тому аналітику варто характеризувати механізм настільки повно, наскільки це значимо в сценарії бізнес-логіки, другорядні інспекції програміст сам зобов'язання "пов'язаний передбачити в коді. Загальна частина і частина приписів специфікації на продукт входить в кластер вхідної інформації. Вся ця інформація, як видно з вищевикладеного зразка, постає у вигляді вербального викладу бізнес-процесів на показниках бізнес-підсистем і операцій для всіляких структурних частин пО, комплектів бізнес 99 правил і комплекту модифікацій об'єктної сфері, наприклад, методологічних модифікацій, сценаріїв, таблиць та ін.

Для надання якості ПО бажано влаштувати вивчення оцінок АПР з завданням виявлення і запобігання проблем і вад на початкових витках життєвого періоду ПО. У процесі такого вивчення необхідно проаналізувати, наскільки повно в УІІ, особливо, в бізнес-вимогах, характеризована інформація, що висловлює підсистеми, впливу і обмеження майбутнього програмного забезпечення, особливо ті, що характеризують його нефункційні характеристики, такі як якість, надійність, гарантоздатність і ін., і виявити факти недостатності інформації, має до них відношення. При таких вимогах актуальною проблемою розглядається характеристика обгрунтованості інформації про властивості в документації приписів до проектованого ПО.

Інформаційні потоки, що характеризують приписи, що регулюють класифікації якості, виробляються на концепції високорівневих задач організації клієнтів і завдань і цілей, а також на концепції бізнес-правил, які включають внутрікорпоративну політику і дотримання індустріальних нормативів, продемонстрованих у вигляді протоколу про предмет і межі проекту, регулювання дизайну та інтеграції. Такі інформаційні потоки поділяються з приписів АПР, які обчислюють характеристики якості - наприклад, припис «Як мінімум 25% пропускної можливості комп'ютера і оперативної пам'яті, загальнодоступною

використання, не повинні застосовуватися в умовах задуманого пікового навантаження» впорядкує таку характеристику 102 властивості, як ефективність ; припис «Тільки користувачі, які володіють пільгами підрівні Аудитор повинні здобувати здатність сканувати транзакції клієнтів» впорядкує таку характеристику якості, як захищеність. Модуль фільтрації інформації (роботизована людино-машинна підструктура) виконує вибудовування і заповнення модифікацій якості ПО на концепції інформації з АПР. Концепції властивості ПО містять інформацію про характерістікіпідхарактерістіки-атрибути якості, взяту за нормативами [53, 40], а також інформацію про метрики і коефіцієнти якості, взяту з галузевих рецензій [20, 60, 61, 62, 63]. Оскільки модифікації якості ПО застосовуються для каталогізації інформації про якість, придбаної з АПР, і наведення її до загального уніфікованого) вигляду згідно зі стандартами та публікаціями, то доцільним є розроблення таких моделей у статичному теоретико-множинному вигляді.

2.2 Онтологічні моделі якості програмного забезпечення

Проаналізувавши стандарти ISO 25023 [59] та ISO 25010, ми отримали можливість підвести висновки, на тему того, що існують атрибути, які залежні більш ніж до однієї характеристики і підхарактеристики якості оцінювання ПЗ, отже має місце бути кореляція характеристик та підхарактеристик за деякими атрибутами (виходить, що згідно з стандартом якісні підхарактеристики мають залежність від 203 атрибутів, але при цьому всього від 138 відмінних атрибутів). Для розробки програмного забезпечення на базі онологій, а саме - інтелектуалізованої системи для інформаційного-пошукової системи (в основі якої лежать онтології), що може покрити галузь оцінювання якості програмного забезпечення ми маємо перш за все виготовити онтологію предметної області, що аналізує якість програмного забезпечення, котра зможе показати відношення та семантичні зв’язки між предметно - галузними системами та ляже в базу пошуку інформації про якість програмного забезпечення, що оцінює ПЗ. А саме в базу тезаурусу розроблюваної ІПС.

Ми можемо розглянути онтологічну концепцію вказаної предметної галузі якісної оцінки ПЗ, що представлена на рисунку 2.3.

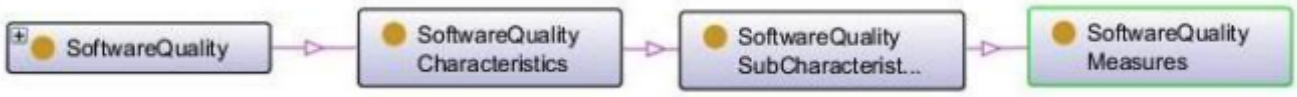


Рисунок 2.3 – Концепція онтології предметної галузі якості ПЗ [21]

Предметно - галузна онтологія якості оцінювання програмного забезпечення, утворює складові частини для:

Онтологію предметної галузі якості програмного забезпечення утворюють складові частини для: Функційної придатності (рисунок 2.4), Сумісності (рисунок 2.5), Ефективності, Можливості переносу, Зручності використання, Надійності, Захищеності, Супроводжуваності.

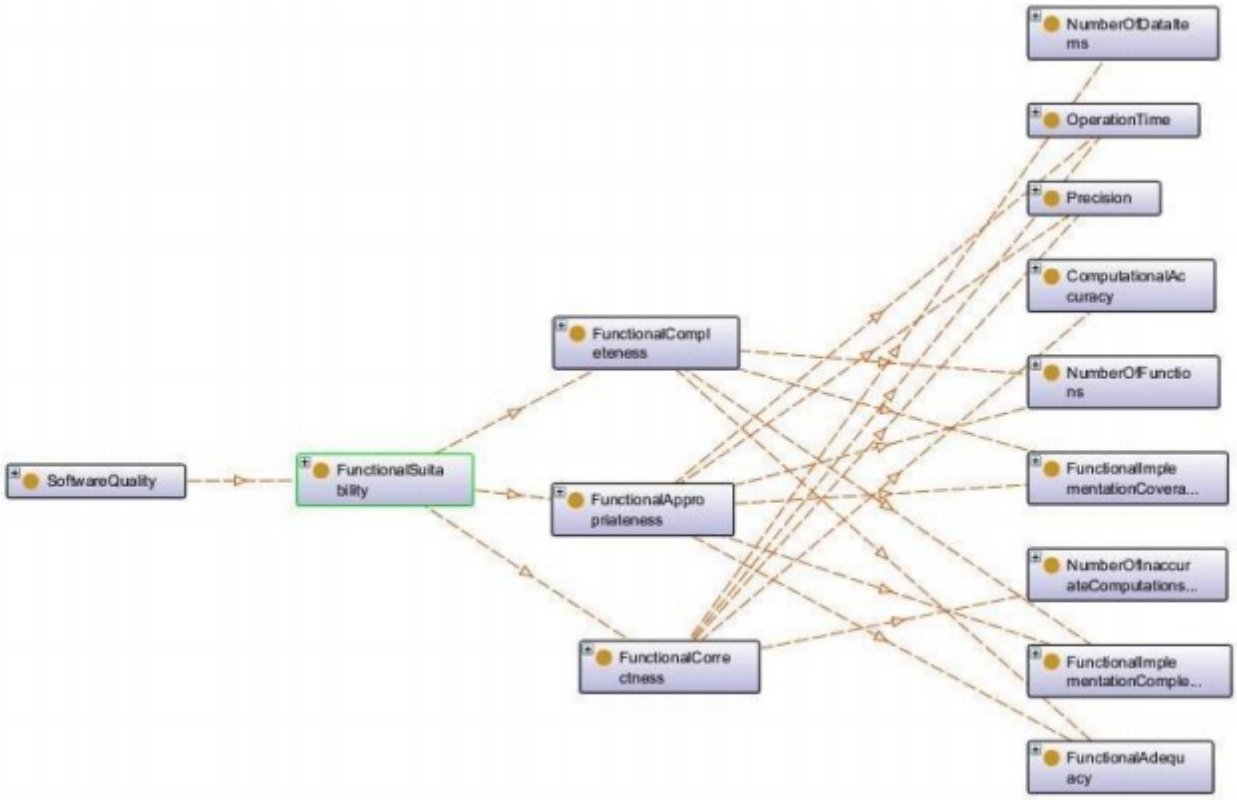


Рисунок 2.4 – Складова «Функційна придатність» онтології предметної галузі якості ПЗ [21]

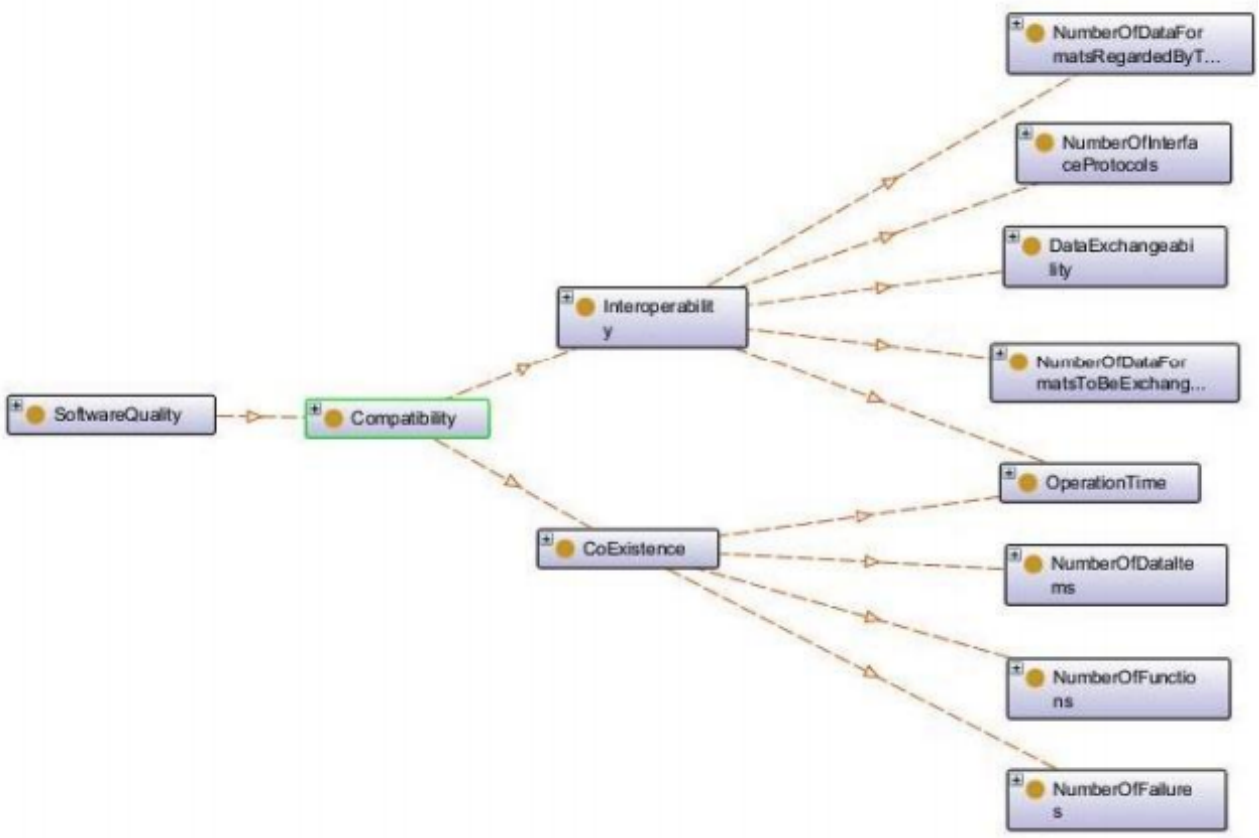


Рисунок 2.5 – Складова «Сумісність» онтології предметної галузі якості ПЗ

На сьогоднішній день оцінка показників якості, атрибутів якості та складності виконується тільки для готового програмного коду [40]. Однак, всі показники та атрибути мають бути прогнозовані вже у специфікації вимог до ПЗ (хоча б, їхні передбачені значення), відповідно, вже на початковій базі специфікацій ми маємо можливість зробити оцінку достатньої кількості інформації на рахунок якості і, в випадку, що відсутні важливі показники або атрибути, у нас з'являється можливість вносити потрібні правки в специфікаційні вимоги.

Маючи на увазі специфікаційну структуру вимог до програмного забезпечення відповідно до стандарту ISO 29148 [38], представимо в формалізованому виді специфікацію з точки бачення належності у ній атрибутів якості оцінки програмного забезпечення:

$$OIO = < O1, \ldots, O5 >, \tag{2.1}$$

в якій Оі – є множиною атрибутів якості і -го розділу специфікації вимог до ПЗ.

В черговому формалізованому вигляді представимо специфікаційні вимоги до програмного забезпечення з точки бачення належності у ній показників для вимірювання значень якості та складності програмного забезпечення:

$$\text{ОІО}_{\text{val}} = < \text{О1}_{\text{val}}, \ldots, \text{О5}_{\text{val}} > , \quad (2.2)$$

в якій ОІО_{val} – є множиною показників складності та якості j -го розділу специфікації вимог до ПЗ;

О1_{val} – один з елементів множини;

Відповідно до стандарту ISO 29148 [38], розділ 3 «Специфічні вимоги» специфікації вимог до ПЗ має підрозділ «Атрибути програмної системи», що має можливість отримувати значення усіх 138 атрибутів якості оцінювання програмного забезпечення, які є необхідними для визначення характеристик , підхарактеристик якості ПЗ за стандартом ISO 25010. Грунтуючись цими висновками та враховуючи формулу (2.2), давайте представим структурну модель специфікацій потреб щодо програмного забезпечення (з точки бачення належності атрибутів якості програмного забезпечення), то вона матиме вигляд [37, 29]:

$$\text{ОІО} = <\varnothing,\varnothing,\{\text{wnd1}, \text{wnd138}\},\varnothing,\varnothing>, \quad (2.3)$$

де wnd1,wnd138 – переелік атрибутів якості оцінювання програмного забепезчення;

Відповідно до стандарту ISO 29148 [2], розділ 1 «Вступ» специфікації вимог до ПЗ має підрозділи «Призначення», «Сфера призначення», «Огляд», що можуть мати в собі визначення деяких метрик якості та складності ПЗ, що необхідні для

визначення показників якості та складності оцінювання програмного забезпечення. Відповідно,

$$O1_{val} = \{Ryr, Zazis, Zwwfgis, Zgdwwfg, Zpzis, Rf\}\ , \quad (2.4)$$

де $Ryr, Zazis, Zwwfgis, Zgdwwfg, Zpzis, Rf$ –перелік метрик якості та складності ПЗ;

Отже, відповідно до стандарту ISO 29148, розділ 3 «Специфічні вимоги» специфікації вимог до ПЗ має підрозділи «Зовнішні інтерфейси», «Функції», «Вимоги до продуктивності», «Логічні вимоги», «Архітектурні обмеження», «Атрибути програмної системи», «Допоміжна інформація», котрі в змозі мати в собі визначення певних показників якості та складності оцінювання програмного забезпечення, що необхідні для визначення показників якості та складності ПЗ.

Відповідно до стандарту ISO 29148 [38], розділ 5 «Додатки» специфікації вимог до ПЗ має підрозділ «Припущення», що може мати в собі значення усіх показників якості чи складності оцінювання програмного забезпечення, що очікуються, та які є необхідними для визначення показників якості та складності ПЗ.

Модель онтологічного характеру предметної галузі «Інженерія програмного забезпечення» (частина «Специфікація вимог до ПЗ (атрибути якості)») виглядає як:

$$A_{OIO} = < Y_{OIO}, OY_{OIO} > , \quad (2.5)$$

де Y_{OIO} – скінченна множина концептів (розділів специфікації та атрибутів якості ПЗ),

OY_{OIO} – скінченна множина відношень між поняттями.

Враховуючи структурну модель специфікації вимог до ПЗ (з точки зору наявності атрибутів якості ПЗ), представлену формулою, а також теоретико-множинну модель якості програмного забезпечення, представлену формулою (2.6), множина концептів:

$$A_{OIO} = \{OIO, WND\} = \{y^1_{OIO}, \ldots, y^{143}_{OIO}\},$$

$$\text{де } \{y^1_{OIO}, \ldots, y^{143}_{OIO}\} = \{O1, \ldots, O5\}, \{y^1_{OIO}, \ldots, y^{143}_{OIO}\} = \{wnd1, \ldots, wnd138\}, \quad (2.6)$$

Множина відношень між поняттями складається з відношення «міститься у», тобто A_{OIO}= {"contained in"} .

Отже, беручи до уваги формулу, базова, або універсальна онтологічна модель предметної галузі «Інженерія програмного забезпечення» (частина «Специфікація вимог до ПЗ (атрибути якості)») матиме вигляд [37, 29]:

$$A_{OIO} = \{O1, \ldots, O5, wnd1, \ldots, wnd138, \text{“contained in”}\}, \quad (2.7)$$

База знань (онтологічна), котра розробляється відповідно до моделі, заповнюється на базі інформаційних пакетів, що запозичені зі стандартів [5, 29]. Онтологічна модель предметної галузі «Інженерія ПЗ» (частина «Специфікація вимог до ПЗ (показники якості)») має вигляд:

$$OIO_{val} = < Y_{OIOval} >, \quad (2.8)$$

де Y_{OIOval} – скінченна множина концептів (розділів специфікації, показників складності та якості ПЗ), OIO_{val}, Y_{OIOval} – скінченна множина відношень між поняттями.

Виконані онтологічні специфікаційні моделі вимог щодо програмного забезпечення (з точки бачення існування даної інформації на рахунок якості) по факту є шаблонними специфікаційними вимогами з точки бачення існування в даній інформації на рахунок якості, що можуть опрацьовувати як автоматизованими засобами, так і фахівцями для роботи зі специфікаціями або інтелектуальними агентами (ботами).

2.3 Висновки

Інформаційно - пошукові системи на даний час надають коефіцієнт точності пошуку – в межах 7 - 10%, а рівень коефіцієнту повноти тримається на рівні приблизно 65 - 70%. Як результат, відомі інформаційно-пошукові системи не в силах надати змогу забезпечення повноцінним функціоналом потреби нинішніх користувачів. Серед загальносвітових трендів в опрацювання великих стеків даних, що надає змогу виконувати нові ранги задач на базі існуючих інформаційних ресурсів, є опрацювання даних та інформації з приміненням інтелектуалізації. При розробці інформаційно - пошукових систем, як стандарт доцільно і необхідно використовувати онтології, що досить в широкому спектрі застосовуються в алгоритмічних процесах пошукових машин і пошуково - інформаційних систем. Можемо точно вказати, що онтології є досить надійним та ефективним інструментом для організування пошукової системи онтологічного типу. Оскільки програмного забезпечення, де факто, є базою більшості нинішніх напрямів в господарській діяльності, а отримання найвищих значень в показниках його якості є фактично ключовим значенням в наданні можливості ефективно використовувати ПЗ і однією з багатьох базових потреб зацікавлених осіб та користувачів до сучасного програмного забезпечення. Головною метою даної роботи є спроба розробки ефективної інтелектуалізованої інфораційно-пошукової систеи (онтологічно базованою) для галузі оцінки якості програмного забезпечення.

3 МЕТОД ПОШУКУ ІНФОРМАЦІЇ ДЛЯ ГАЛУЗІ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ОНТОЛОГІЙ

3.1 Правила для пошуку інформації для галузі якості програмного забезпечення

Перед тим, як описати правила для пошуку інформації для галузі якості програмного забезпечення на основі онтологій, нам необхідно описати процес виконання роботи штучного інтелекту.

Першим кроком в роботі штучного інтелекту, є сканування, або аналіз.

Сканування - це процес виконання певного алгоритму штучного інтелекту, в процесі якого він аналізує онтології існуючих проектів, для того щоб додати їх в список для індексування.

Для його роботи використовується штучний інтелект який зроблений використовуючи програмну мову Python та бібліотеки які він використовує. Вона зможе забезпечити для нас хорошу швидкість опрацювання та індексації багатьох проектів. На даному етапі будемо називати її - роботом. Даний “робот” зможе автоматично оприділяти, які проекти необхідно сканувати, і наскільки часто це потрібно робити.

Робот при індексуванні проекта в першу чергу, буде аналізувати список необхідних атрибутів, які були створені на базі попередніх сканувань і на базі тих атрибутів які були закладені в онтологічні словники при створенні ядра штучного інтелекту.

Коли робот переходить до проекту, що індексується, він знаходить в ньому атрибути і додає їх в список атрибутів, що сканується. Робот оновлюється з обліком всіх нових та змінених проектів, та також проектів, що не пройшли індексування.

В процесі сканування робот запускає проект за допомогою емуляторів, а також ініціалізувати всі скрипти, щоб переконатися, що атрибути які вказані розробниками правдиві. Якщо атрибути вказані розробниками і реальні онтології не будуть співпадати, то робот видасть відповідний результат.

В ідеалі, проводити два сканування - основне і додаткове. Даний штучний інтелект запрограмований на виконання обох.

Основне сканування проводиться роботом основного типу. При скануванні роботом основного типу він використовуватиме лише базові (найнеобхідніші) атрибути для оцінювання програмного забезпечення.

Додаткове сканування проводиться роботом додаткового типу. При скануванні роботом додаткового типу він використовуватиме додаткові атрибути для оцінювання програмного забезпечення. Це необхідно, якщо ви бажаєте отримати більш розширені результати сканування.

Як робот розуміє, які проекти йому необхідно опрацювати?

1. Якщо доступ до проекту заблокований в файлі з налаштуваннями робота, він не буде скануватись.
2. Робот не може сканувати сторінки, якщо користувач не матиме доступу до самого проекту.
3. Після занесення в індекс робот рідше сканує схожі проекти.

Як підвищити ефективність сканування робота для оцінки якості програмного забезпечення:

Спростити роботу робота для аналізування проекта можна за допомогою перерахованими нижче правилами:

1. Додати до проекту файл з необхідними атрибутами (даний документ повинен сформувати розробник).
2. Якщо ваш проект використовує модульну систему, то має сенс, оцінювати програмне забезпечення відповідного до кожного модуля окремо, а не до всього програмного забезпечення цілком.
3. Використовуйте структурно-впорядковані файли за зв’язки між компоненти.
4. Якщо проект використовує різнонаправлений роутинг, необхідно описати і його.

5. Потрібно сумлінно задавати атрибути. Дане правило особливо важливе для виконання роботи робота. Якщо описати все правильно то зменшується кількість ітерацій виконання алгоритму штучного інтелекту.

6. Якщо програмне забезпечення має декілька версій, необхідно позначити це спеціальним текстовим прапорцем.

7. Аналізуйте дані звіту після аналізу.

8. Слідкуйте за тим, щоб у робота був доступ до основних файлів і важливих ресурсів проекту.

Розроблений робот, аналізує кожну модель, за описаними характеристиками чи під характеристиками. Даний штучний інтелект здатний аналізувати більшість атрибутів які вказані в специфікації, але на даній версії ще не всі.

Між процесами сканування та індексування робот перевіряє, чи є проект що перевіряється вже індексованим і занесеним в базу даних. Якщо робот помічає дублювання проекту, то він зупинить її сканування. Схожі ж проекти аналізуються з загальними правилами (програма аналізує подібні атрибути, характеристики та під характеристики, які в проектах)

Важливо звернути уваги, що робот не буде аналізувати проекти, в яких не описані атрибути в конфігураційних файлів.

Правила для того, щоб підняти ефективність оцінки якості програмного забезпечення з допомогою машинного інтелекту. Існує декілька способів спростити виконання роботом оцінки якості програмного забезпечення:

1) Якщо вам необхідно прибрати деякі модулі чи компоненти програми, що індексуються, встановіть відповідний текстовий прапорець в файлі конфігурацій.

2) Використовуйте структуровані та ієрархічно-впорядковані компоненти та модулі відповідно документації.

3) Ознайомтесь з пошуковою оптимізацією.

Після того як користувач провів аналіз програмного забезпечення, штучний інтелект показує в результатах оцінку щодо продукту, який аналізується. При цьому враховуються попередні аналізи та результати з бази даних. Основний

позитивний момент полягає в тому, що робот навчається на попередніх ітераціях, і обробляє схожі проекти швидше з кожним разом.

Ми можемо оцінити якість ПЗ з допомогою зовнішніх і внутрішніх характеристик які були описані в попередніх розділах. Різниця між ними досить розмита, тому що одні з них впливають на інші. Якщо програма не досить зрозуміла чи незручна для підтримки то в ній досить тяжко та проблемно правити помилки, що впливає на певні зовнішні характеристики, а саме - коректність та надійність. З іншої сторони, ПЗ, що є не досить гнучким, не може бути покращене з боку відповідей від користувач, що б'є по його практичності. Для того щоб краще знати взаємозв'язок між цими характеристиками та коректно описати правила для штучного інтелекту, ми повинні знати як вони взаємодіють.

На таблиці 3.1 ми можемо візуально побачити взаємозв'язок характетеристок та їхню взаємодію одна з одною.

Умовні позначеня:

1) Живучість – l.
2) Правильність – p.
3) Адаптація – a.
4) Цілісність – t.
5) Надійність – n.
6) Ефективність – e.
7) Практичність – z.
8) Коректність – c.

Таблиця 3.1 – Взаємозв’язок характеристик та їхня взаємодія одна з одною

Вплив	l	p	a	t	n	e	z	c
l	+	-	+	-	-	-	+	-
p	-	+	-		+	-		+
a	+		+	-				
t				+	+	-		
n	-	+		+	+			+
e		-	-	-	-	+		-
z		+	+				+	
c	-	+			+	+		

Як ми бачимо на таблиці покращення певних аспектів якості змінює в негативну сторону погіршення інших.

Один з важливим факторів є те, що концентруючись на одній характеристиці розробник не завжди має змогу представити компроміс для іншої. Певні характеристики щільно пов’язані між собою зворотнім взаємозв’язком, інші - прямим, а декілька з них взагалі незалежні одна від одної.

Відповідно до цих взаємозв’язків ми можемо описати певні правила, які будуть виставляти оцінку (TOTAL), для програмного забезпечення що оцінюється (в даному прикладі, знак “+” розцінюватиметься нами як значення, що є більше 0, а знак “-”, відповідно менше 0:

Живучість - l

1) l &&l > 0, TOTAL += 1;
2) l&&p < 0, TOTAL -= 1;
3) l&&a > 0, TOTAL += 1;

4) l&&t < 0, TOTAL -= 1;

5) l&&n < 0, TOTAL -= 1;

6) l&&e > 0, TOTAL += 1;

7) l&&z > 0, TOTAL += 1;

8) l&&c < 0, TOTAL -=1;

Правильність - p

1) p &&l < 0, TOTAL -= 1;

2) p&&p > 0, TOTAL += 1;

3) p&&a < 0, TOTAL -= 1;

4) p&&n > 0, TOTAL += 1;

5) p&&e < 0, TOTAL -= 1;

6) p&&z < 0, TOTAL -= 1;

7) p&&c > 0, TOTAL +=1;

Адаптація - a

1) a &&l > 0, TOTAL += 1;

2) a&&a > 0, TOTAL += 1;

3) a&&n > 0, TOTAL += 1;

4) a&&t < 0, TOTAL -= 1;

Цілісність - t

1) t&&t > 0, TOTAL += 1;

2) t&&n > 0, TOTAL += 1;

3) t&&e < 0, TOTAL -= 1;

Надійність - n

1) n &&l < 0, TOTAL -= 1;

2) n&&p > 0, TOTAL += 1;

3) n&&t > 0, TOTAL += 1;

4) n&&n > 0, TOTAL += 1;

5) n&&c > 0, TOTAL += 1;

Ефективність - e

1) e&&p < 0, TOTAL -= 1;

2) e&&a < 0, TOTAL -= 1;

3) e&&t < 0, TOTAL -= 1;

4) e&&n < 0, TOTAL -= 1;

5) e&&e > 0, TOTAL += 1;

Практичність - z

1) z &&p > 0, TOTAL += 1;

2) z&&a > 0, TOTAL += 1;

3) z&&z > 0, TOTAL += 1;

Коректність - c

1) c &&l < 0, TOTAL -= 1;

2) c&&p > 0, TOTAL += 1;

3) c&&n > 0, TOTAL += 1;

4) c&&e > 0, TOTAL += 1;

3.2 Метод пошуку інформації для галузі якості програмного забезпечення на основі онтологій

Оскільки нам необхідно автоматизувати семантичну (онтологічну), природномовну специфікацію, то першим кроком в даному процесі повинна виступати - формалізація. Даний етап необхідно виконувати, не забувши врахувати, про вимоги на видобування саме яких вимог буде направлений семантичний (онтологічний парсинг) специфікацій. Для того щоб провести парсинг специфікацій на наявність атрибутивного пошуку для виділення складових характеристик (не функційного типу) якості оцінювання програмного забезпечення, необхідно виконати формалізацію з використанням онтологій, тому що саме вони надають можливість виявити прогалини чи дублювання в знаннях на базі візуалізації відсутності зв’язків логічного типу, і до того ж мають можливість провести аналіз інформації з допомогою AI. Щоб успішно формалізувати дану специфікацію вимог, ми можемо використати специфікаційну базу вимог до

програмного забезпечення (з точки бачення існування атрибутів) відповідно до стандарту ISO 29148:2018 [152].

Онтологічні атрибути, що наведені вище, потрібні для того щоб ми могли визначити нефункційні складові характеристики якості оцінювання програмного забезпечення, котрі були наведені після врахування ділення за специфікаційними розділами, внаслідок чого, онтологія, що була розроблена є специфікаційни шаблоном вимог до програмного забезпечення з точки бачення існування атрибутів і також надає допомогу для користувача про розташування атрибутів у специфікаційних вимогах до програмного забезпечення. AI, що було розроблено на базі підходу онтологічного типу для сканування специфікаційних вимог до якості оцінювання програмного забезпечення отримує на вхід специфікаційні вимоги і після цього виконує автоматичний аналіз даних вимог, шукаючи атрибути, які потрібні для того щоб ми могли визначити нефункційні характеристики програмного забезпечення. Метод пошуку інформації AI на базі онтологічного підходу для аналізу специфікацій вимог щодо знаходження атрибутів складається з наступних кроків:

1) знаходження кожного атрибуту в специфікаційні онтології вимог, що міститься у базі знать AI;

2) у випадку, якщо даний атрибут був знайдений в специфікаційних вимогах, то даний атрибут AI може ввести в множину існуючих атрибутів;

3) у випадку, якщо даний атрибут не був знайдений в специфікаційних вимогах, то даний атрибут AI може занести в множину неіснуючих атрибутів;

- З онтологічного базису для нефункційних складових характеристик якості оцінювання якості програмного забезпечення, AI повинен видалити усі атрибути зі списку не існуючих атрибутів:

1) виконаємо перевірку, на наявність усіх атрибутів з множини існуючих атрибутів і чи залишились вони в модифікованій онтології;

2) на цьому кроці збережемо внесені зміни та отримаємо реальну онтологію для нефункційних складових характеристик якості оцінювання програмного забезпечення;

Як результат роботи даної AI, ми отримуємо реальну онтологію для нефункційних складових характеристик якості оцінювання програмного забезпечення. Додатковими результатами виконання AI для роботи в майбутньому, ми можемо використовувати ще списки існуючих та неіснуючих атрибутів в специфікаційних вимогах до якості оцінювання програмного забезпечення.

Структурна схема для даної AI на базисі онтологічного підходу для семантичного аналізу специфікацій вимог до програмного забезпечення представлена на рисунку 3.2.

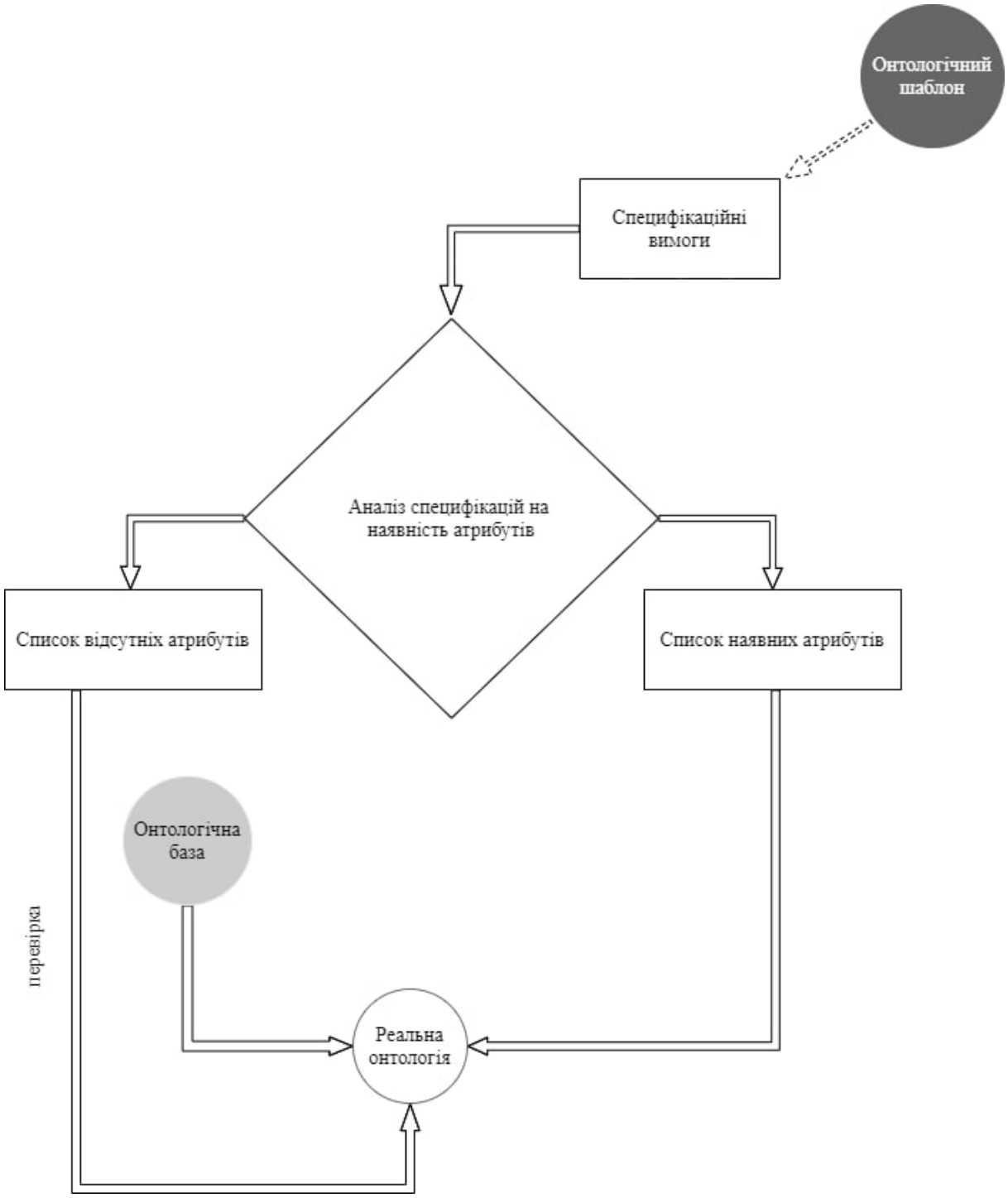


Рисунок 3.2 – структурна схема AI на базисі онтологічного підходу

Аналогічно до попереднього методу, виконаємо схематичну розробку кроків виконання AI для аналізу вимог для програмного забезпечення для пошуків метрик, що необхідні для метричного аналізу. Даний AI приймає на початок вимоги щодо якості програмного забезпечення і після цього починає виконувати автоматичний

аналіз вимог, в процесі якого шукає показники, які потрібні для вимірювання метрик програмного забезпечення.

AI, що працює на базі методу з рисунку 3.3 надає можливості проводити онтологічних аналіз специфікацій для пошуку показників, що потрібні для вимірювання метрик якості програмного забезпечення. Дані результати використовуються надалі для оцінки достатності інформаційних вимог, тому необхідно лиш вказати наявність чи відсутність кожного показника у вимогах.

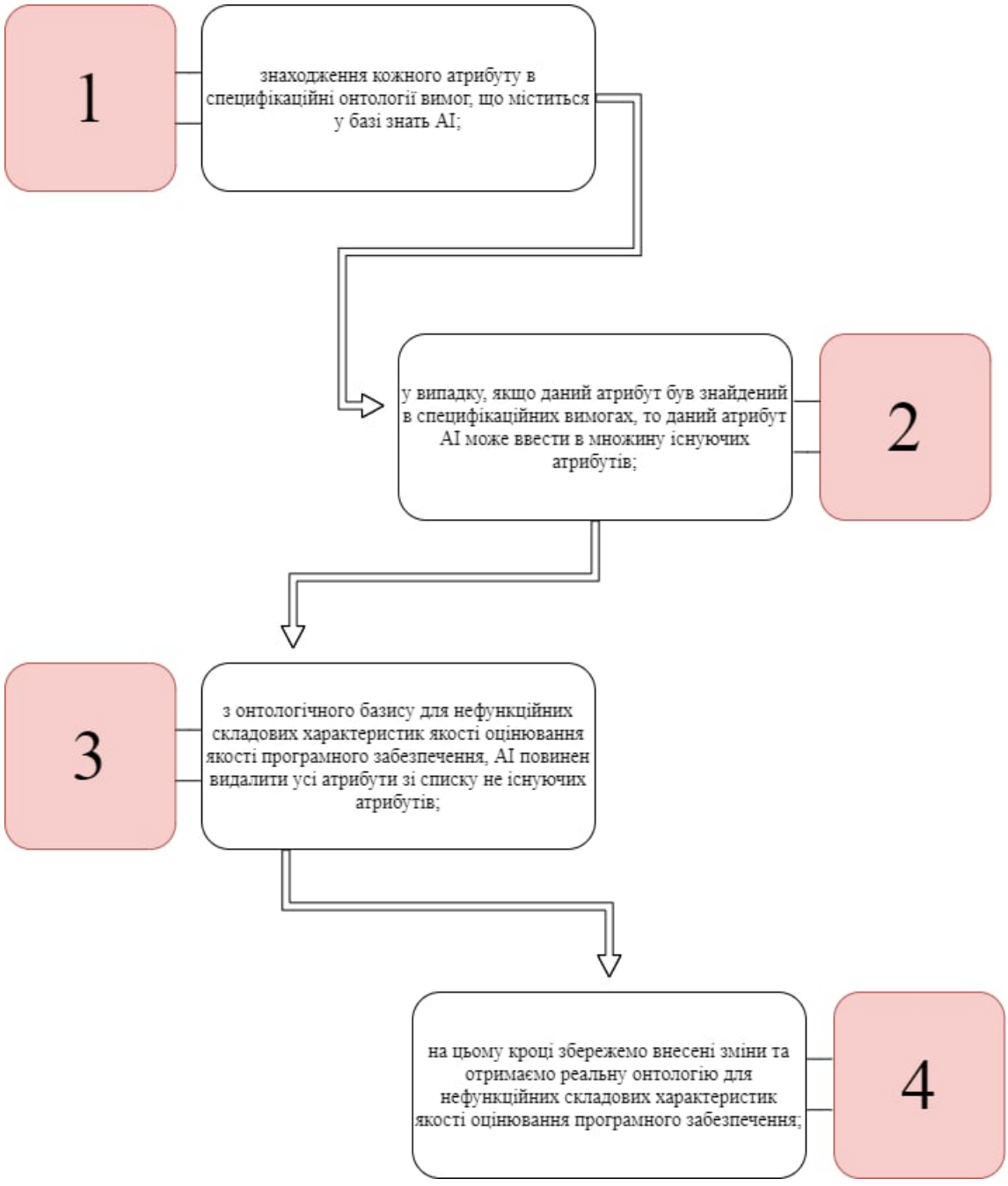


Рисунок 3.3 – Онтологічний аналіз специфікацій

Виконані схематично-розроблені AI, надають можливість проводити аналіз специфікацій з пошуком та встановленням відсутності чи існування атрибутів з показниками, що є необхідними для визначення складових характеристик якості оцінювання програмного забезпечення. Дані результати можуть пізніше використовуватися для визначення нефункційних складових характеристик якості. Для того щоб визначити достатність нам необхідна лиш інформація про відсутність або наявність атрибутів в специфікацій, тому правила аналізу для специфікацій на базі яких працює AI, є зрозумілими та простими. Ці умови надають низьку “ціну” та високу швидкість для аналізу специфікацій.

Штучний інтелект на базі онтологічного підходу, під час свого виконання, може використовувати основні онтології нефункційних складових характеристик якості програмного забезпечення, як відомі йому тезиси. Оскільки онтології, що дають нам представлення наслідково-причинних зв'язків між поняттями, надає змогу виявити неіснуючі в специфікації атрибути та встановити, котрі з нефункційних складових характеристик якості програмного забезпечення не має змоги визначити без цих атрибутів. На основі цих онтологій штучний інтелект збирає в сукупність інформацію, яку отримав з специфікаційних вимог до програмного забезпечення, котра була представлена, як реальна онтологія для надання можливостей порівняння базових та реальних онтологій. На базі даного зрівнння онтологій штучний інтелект отримує на вхід перелік неіснуючих атрибутів в специфікації, тому що саме за ними буде відрізнятись базові онтології від реальних. Штучний інтелект виконуватиме аналіз отриманого переліку атрибутів і після цього встановлюватиме, котрі нефункційні складові характеристики якості оцінювання програмного забезпечення неможливо виявити без наявних атрибутів. Окрім цього, штучний інтелект враховує кількість відсутніх нефункційних характеристик, які немає змоги вирахувати без деяких атрибутів, для генерації оцінки в числовому форматі для достатності даних в специфікаційних вимогах до програмного забезпечення. Після даного етапу, штучний інтелект проводить оцінки інформації в специфікації вимог до програмного забезпечення і

приймає рішення про наступні кроки, а саме дає рекомендацію щодо підвищення достатності інформації і висновок про неї.

Як результат, можемо вказати, що метод діяльності штучного інтелекту на базі онтологічного підходу для оцінювання якості програмного забезпечення на наявність достатності інформації у специфікації для виявлення нефункційних складових характеристик складається з наступних етапів:

1) порівняння онтологій нефункційних складових характеристик якості оцінювання програмного забезпечення (реального) з базовими онтологіями нефункційних складових характеристик якості оцінювання програмного забезпечення з ціллю виявити атрибути, що відсутні у специфікаційних вимогах до програмного забезпечення (реального), за якими були побудовані реальні онтології;

2) знаходження нефункційних складових характеристик та підхарактеристик якості програмного забезпечення, що неможливо порахувати на базі існуючих в специфікаційних вимогах до атрибутів реального програмного забезпечення;

3) створення висновку про недостатність чи достатність знань у специфікаційних вимогах для виявлення кожної нефункціної характеристики програмного забезпечення разом і окремо;

4) розрахунок оцінок (числових) про рівень достатності, що наявний у специфікаційних вимогах інфорації щодо визначення кожної характеристики програмного забезпечення нефункційного типу за формулою:

$$Aj = \frac{(hj - \Sigma \frac{zti}{zri})}{h_j}, \quad (3.1)$$

де hj- кількість підхарактеристик j-ї нефункційної характеристики програмного забезпечення (j=1…8, тому що у стандарті ISO 25010 визначено саме вісім нефункційних складових характеристик якості ПЗ);

zti- кількість відсутніх у специфікації вимог до реального ПЗ атрибутів для і-ї підхарактеристики j-ї нефункційної характеристики ПЗ;

zri - кількість необхідних атрибутів для і-ї підхарактеристики j-ї нефункційної характеристики ПЗ (визначається базовими онтологіями для кожної нефункційної характеристики-складової якості ПЗ);

5) розрахунок оцінок (числових) про рівень достатності, що наявний у специфікаційних вимогах інфорації щодо визначення кожної характеристики програмного забезпечення нефункційного типу за формулою

$$A = \frac{(k - \Sigma \frac{zto_j}{zro_j})}{h} \quad , \qquad (3.2)$$

де k – кількість нефункційних характеристик–складових якості ПЗ (k=8 відповідно до ISO 25010);

zto_j- – кількість відсутніх у специфікації вимог до реального ПЗ атрибутів для j-ї нефункційної характеристики ПЗ;

zro_j - кількість необхідних атрибутів для j-ї нефункційної характеристики ПЗ (визначається базовими онтологіями для кожної нефункційної характеристики-складової якості ПЗ);

6) представлення проблемних зон у знаннях нефункційних складових характеристик якості ПЗ.

Запропонований штучний інтелект, в змозі автоматично опрацьовувати існуючі знання (нефункційні характеристики, котрі презентовані в вигляді онтологій) і формувати нові знання (онтології для реального програмного забезпечення, рекомендації для підвищення рівня достатності). Цей штучний інтелект не в змозі працювати з нечіткими даними, тому що в задачі оцінювання достатності інформації не передбачувано нечіткості. Шукаємий атрибут, або відсутній в специфікації, або присутній у ній, після чого виконується зменшення

ступеню достатності інформації на величину, котра в залежності від кількості нефункційних характрестик, що залежні від цього атрибуту.

Аналогічно розробимо метод діяльності штучного інтелекту на базі онтологічного підходу для оцінювання програмного забезпечення на предмет оцінки достатності інформації в специфікаціях для вимірювання величин якості та складності програмного забезпечення

Окрім цього, оцінка (числова) ступеню достатності метричних даних в специфікації вираховуватиметься за формулою:

$$A = \frac{(n - \Sigma \frac{zti_j}{zri_j})}{n} \quad , \tag{3.3}$$

де n – кількість метрик (n=24, оскільки було обрано саме 24 метрики складності та якості, доступних на ранніх етапах життєвого циклу);

ztij – кількість відсутніх у реальній специфікації показників для j-ї метрики,

zrij – кількість необхідних показників для j-ї метрики (визначається базовою онтологією предметної галузі «Інженерія програмного забезпечення» (частина «Якість та складність ПЗ. Метричний аналіз»)),

j=n..k, Σ(zriij)=72 – загальна кількість показників (в т.ч. й тих, що повторюються), від яких залежать метрики ПЗ;

Описані штучні інтелекти на базі онтологічного підходу для оцінювання програмного забезпечення на можливість оцінити достатність інформації в специфікації для оцінювання нефункційних складових характеристик якості програмного забезпечення і для вимірювання ступені програмного забезпечення формує висновок про достатність або доступність інформації у специфікації, та обчислює оцінку (числову) ступеню достатності інформації. Крім цього презентують відсутні метрики або атрибути з діленням за метриками чи характеристиками для яких вони використовуються.

Досить важливим аспектом та додатковою функцією інтелектуалізованої системи є створення методів для пошуку інформації щодо дефектів, котрі

провокують зменження якості програмного забезпечення. Для цього ми можемо використати науковий метод відкладки.

Класичний науковий метод містить в собі декілька етапів, серед них:

1. Збір даних при кожній ітерації (інтелуктуалізована система, може використовувати дані зібрані з інших проектів, або дані, які були з самого початку внесені в її семантичне ядро).
2. Формування списку проблемних місць, які були зафіксовані під час перевірки, створення певного висновку, та представлення його для користувача.
3. При підтвердженні користувачем правильності висновку, інтелектуалізована система проводить додаткову ітерацію, яка підтверджує або спростовує попередній висновок.
4. В випадку невдачі, повернення до пункту (2), але вже з відпрацьованим списком.

В деяких випадках, дефект (або баг), може з’являтись не завжди, а іноді, його майже неможливо відловити. Такі непередбачувані помилки зазвичай пов’язані з якимось зовнішніми факторами, або з даними поза програмою. Коли інтелектуалізована система буде зустрічатись з даними дефектами, то вона повинна перевірити чи був ініціалізований вказівник або вказівник був звільнений відповідно до своєї області в пам’яті.

Виявлена помилки не завжди обмежується однією ітерацією, в деяких випадках перевірка може проводитись до тих пір, поки шукана область дефектів не зменшується до мінімальною. Тобто спростити область настільки, що змінення деякого аспекту вплинуло на поведінку помилки. Після чого, інтелектуалізована система повинна аналізувати програму та її поведінку в умовах, що контролюються - це повинна допомогти відловити помилку.

Після виявлення умов що викликають помилку, інтелектуалізована система, повинна буде їх земулювати та постаратись відловити дефект. Якщо дефект неможливо знайти, ми перевірити, якість написання коду, можливе порушення одного з патернів якісного програмування. Одним з них таких патернів є DRY. Головною ідеєю цього патерну є принцип зменшення повторюваного коду,

особливо в програмному забезпечення з великою кількістю шарів абстрагування. У випадку якщо, даний принцип використовується, то зміна одного елемента системи не буде потребувати внесення змін в інші компоненти. Цілком логічно навчити інтелектуалізовану систему працювати з іншими патернами та принципами написання якісного та оптимізованого коду, оскільки це прямим чином впливає на його якість.

В випадку, якщо інтелектуалізована система виявила неякісний код, вона може виконати наступні етапи:

1. Формуючи висновок про можливе джерело дефекту, система повинна врахувати якомога більше даних (базованих на попердніх ітераціях чи тих що були внесені в ядро на початку).

2. Деталізувати ітерації, на яких найчастіше виявляється помилка. Зміна параметрів в більш широкому діапазоні, або ж концентрації на одному з них дасть змогу звузити область пошуку до меншої.

3. Емуляція ітерацій, що раніше викликали дефект, може дати нам можливість більш точно визначити джерело проблеми. Як тільки система виявить причину дефекту, вона повинна провести ітерації, що найбільш схожі до ітерації, що викликала помилку. Якщо і це призведе до виявленя дефектів, то система повинна зробити висновок, що проблема не до кінця зафіксована. Це можливо через це що джерелом помилки є не один, а комбінація факторів, тому спроба діагностувати дефект при лише одній ітерації часто не дає змогу знайти саме корінь проблеми (рисунок 3.4).

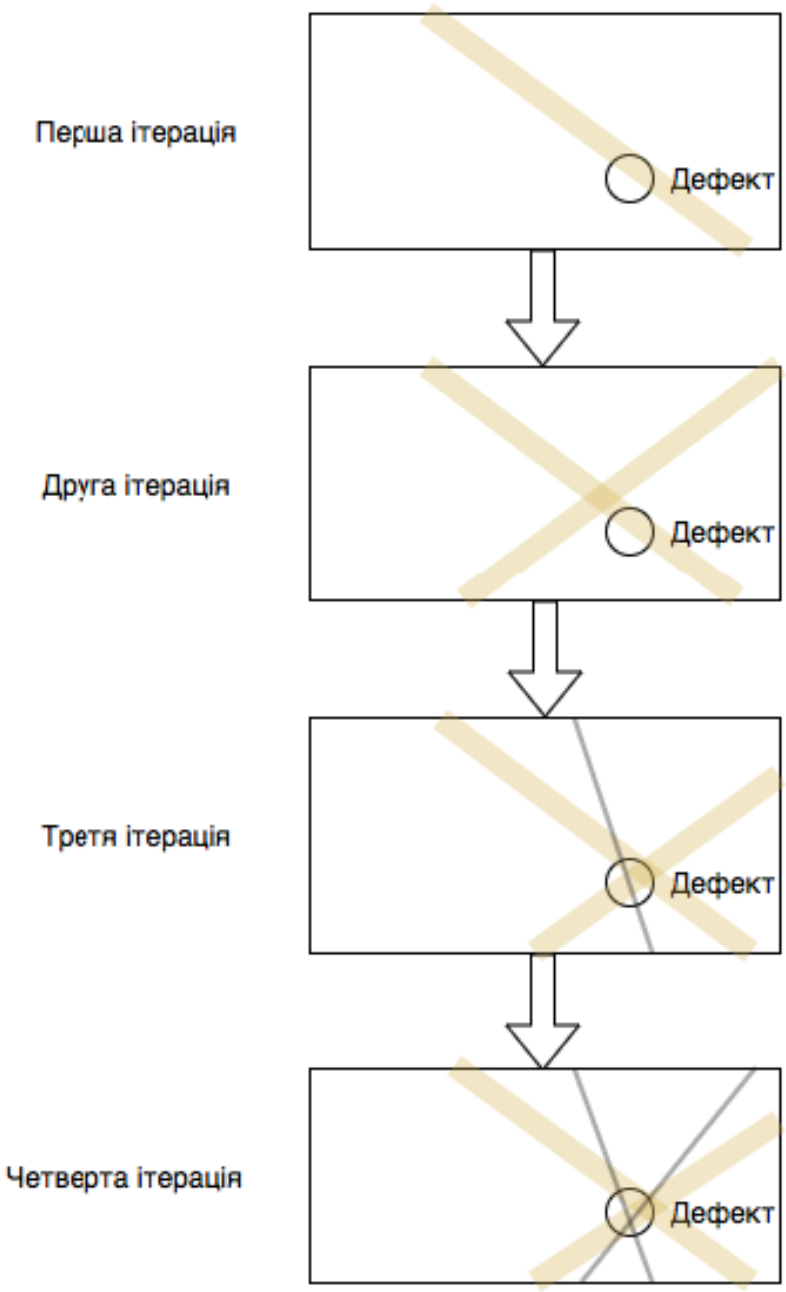


Рисунок 3.4 – Проведення ітерацій для знаходження помилки

1. Генерація даних для формування висновків, та зменшення області пошуку інформації. Базуючись на попердні ітераціях та семантичному ядрі, система зможе краще виконувати пошук.

2. Задіювати результати ітерацій в яких не було виявлено помилок.

3. Зменшити “підозрілу” область коду. Для того щоб не шукати помилку у всій програмі відразу, інтелектуалізована система повинна провести ітераційний пошук спочатку по меншому фрагменту коду. Крім того, система може мати

можливість видаляти певні компонети (перед цим зробивши резервну копію), і вже після цього робити висновки, чи знаходився дефект в цій частині коду. Система може використовувати певні алгоритми, для того, щоб зменшити час та інші ресурси для пошуку проблем. Одним з таких алгоритмів може бути - бінарний пошук. Система спочатку видалятиме половину коду, і визначати в якій з двох частин була помилка. Ітерації ділення будуть продовжуватись до тих пір, поки не буде знайдена проблема.

4. Система позначатиме прапорцями певні компоненти, які вже до цього мали дефекти та в першу чергу перевіряти їх на наявність нових дефектів.

5. Перевірка дефектів, які широко розповсюджені.

3.3 Висновки

Проведене дослідження надало для нас перелік правил та умов для коректної роботи ІПС. Окрім того були виведені поведінкові фактори, та правила, що дозволяють уникати повторень проблем та помилок. Окрім того ми змогли схематично побачити роботу AI, і отримати реальну онтологію для нефункційних складових характеристик якості оцінювання програмного забезпечення.

Були виведені методи діяльності штучного інтелекту на базі онтологічного підходу для оцінювання якості програмного забезпечення на наявність достатності інформації у специфікації для виявлення нефункційних складових характеристик та етапи з яких вони складаються.

4 Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення

4.1. Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення

Важливою ознакою розробляємого продукту передусім повинна бути реалізація та використання високо якісних методів та функцій. Оскільки дане програмне забезпечення буте проводити оцінку якості інших додатків, то воно і саме повинне відповідати цим умовам.

Зазвичай, низькоякісне програмне забезпечення - це причина неповноцінної підготовки спеціаліста який розроблятиме його. Для розробки точних і повних вимог, планування проекту та високоякісної архітектури, розробник повинен мати хорошу підготовку. У випадку, якщо вона відсутня ми не можемо розраховувати на якісний код, і навіть виконання такого в великих об’ємах не принесе нам бажаного результату. Тому ми маємо переконатись, що будуть виконані наступні підготовчі етапи:

1. Повноцінне планування.
2. Оцінка масштабу проекту.
3. Опис вимог.
4. Створення road-map проекту.
5. Створення можливих аналогій.
6. Виділення архітектурних елементів.

Як тільки дані етапі пройдені, і підготовче “ядро” було сформульоване, ми маємо перейти до більш конструктивних питань. Одним з таких є вибір мови програмування на якій, власне, і буде написана інтелектуалізована система.

Оскільки, для мови програмування, на якій буде написане програмне забезпечення, є досить важливим пунктом, ми маємо приділити цьому велику частину нашого часу для підготовки. Мова програмування, що використовуватиметься як основна, повинна бути вже знайомою для спеціаліста, і він повинен використовувати її переваги і знати про недоліки. Нижче буде

представлений перелік мов, що підходять для поставленої задачі, а саме - написання інтелектуалізованої системи:

Python. Ця мова широко використовується програмістами завдяки чистому синтаксису та логічній строго граматичній побудові програми. Python найчастіше використовується в галузі машинного навчання та створення нейронних мереж.
Сотні доступних бібліотек дають можливість реалізувати практично будь-який проект, будь то веб-сервіс, мобільний додаток, машинне навчання або наукові обчислення.
Основні характеристики:

1) відносно швидка швидкість розвитку;
2) багатий та різноманітний набір бібліотек та інструментів;
3) баланс низькорівневого та високорівневого програмування;
4) дозволяє тестувати алгоритми без необхідності їх реалізації;
5) все ще активно розробляється;

LISP. Створена ще в 1958 році, ця мова програмування високого рівня широко використовується для розробки ШІ. Гнучкий та розширюваний, він добре підходить як інструмент для вирішення конкретних завдань.
Варто згадати, що саме LISP був використаний для створення найперших роботів, які могли пересуватися, вмикати / вимикати світло та переміщати предмети.
Основні характеристики:

1) легко адаптується до рішень для конкретних потреб;
2) має макроси для реалізації різних рівнів ШІ;
3) автоматичний збір сміття;
4) підтримує швидке створення прототипів;

Prolog. Розробники ШІ цінують його за високий рівень абстракції, вбудовану пошукову систему, недетермінованість тощо. Це одна з небагатьох мов, яка представляє парадигму декларативного програмування. Крива навчання тут досить висока.

Prolog - це мова для реалізації алгоритмів з великим неявним пошуком варіантів (логічний висновок, пошук залежностей, пошук ходів), що робить його вдалим вибором для створення штучного інтелекту.

Основні характеристики:

1) потужна та гнучка структура програмування;
2) структурування даних на основі дерева;
3) опція автоматичного відкату;

Java. Це популярна, проста у вивченні і досить універсальна мова програмування, на основі якої можна створювати додатки різної складності для більшості операційних систем.

Технологія віртуальних машин Java дозволяє створити одну версію програми, яка працюватиме на всіх платформах, що підтримуються Java. Його сильними сторонами є прозорість, зручність налагодження та портативність.

Основні характеристики:

1) легке налагодження;
2) простий у навчанні;
3) незалежність платформи;
4) багата бібліотека;
5) масштабованість;

C++. Однією з найважливіших переваг цієї мови програмування є її швидкість. Це важливо для розробників, оскільки робота систем ШІ супроводжується великою кількістю розрахунків, тому швидкість може тут зіграти вирішальну роль. Це хороший вибір для проектів, заснованих на машинному навчанні та побудові нейронних мереж.

Основні характеристики:

1) хороша швидкість і продуктивність;
2) поєднання інструментів високого та низького рівня;
3) масштабованість;
4) безліч доступних бібліотек та інструментів;

В високоякісному програмному забезпеченні повинен чітко просліджуватись зв’язок між концепцією та її реалізацією. При розробці продукту, архітектурні принципи здатні вносити певний структурний баланс в код, а принципи щодо конструювання певний порядок. Даний порядок називається - конвенцією.

Що таке конвенції про кодування

Стандарт форматування коду є загальновизнаним і використовується групою розробників програмного забезпечення для рівномірного обміну кодом. Метою прийняття та використання стандарту є спрощення розуміння кодом програми людиною та мінімізація навантаження на пам’ять, мислення та зір під час читання програми.

Прикладом стандарту кодування є набір домовленостей, прийнятих у будь-якому типовому друкованому творі мовою (наприклад, стандарт C для кодування, який отримав скорочення K&R, походить із класичних книг C Кернігана та Річі). Іншим прикладом є широко використовувана бібліотека або API (наприклад, на розподіл угорської нотації явно вплинуло її використання в Windows API та більшість стандартів кодування для використання Python, в тій чи іншій мірі, стандартів PEP).

Розробники мов також випускають докладні інструкції з кодування. Наприклад, були випущені стандарти кодування C # від Microsoft та стандарти кодування Java від Oracle. Запропонований розробником або прийнятий відомими джерелами спосіб кодування більш-менш доповнений та уточнений у корпоративних стандартах.

Зазвичай стандарт конвенції кодування описує:

1) способи вибору імен та використовуваного регістру символів для імен змінних та інших ідентифікаторів:

2) написання типу змінної в її ідентифікаторі (угорська нотація) та складання символів (нижній, верхній, «верблюд», «верблюд» з малої літери), використання знаків підкреслення для відокремлення слів;

3) стиль відступу для логічних блоків - чи використовуються вкладки, ширина відступу;

4) метод розміщення дужок, що обмежують логічні блоки;

5) використання пробілів при проектуванні логічних та арифметичних виразів;

6) стиль коментування та використання документальних коментарів;

7) конвенції про іменування змінних, класів та файлів;

8) стиль висловлювання та найкращі практики їх використання;

9) організація файлів;

10) оголошення класів та інтерфейсів;

11) практики програмування;

4.2. Проектування та реалізація Інтелектуалізованої системи на основі онтологій для галузі якості програмного забезпечення

Для того щоб краще зрозуміти як буде працювати інтелектуалізована система, давайте представим її у вигляді UML – діаграми (рисунок 4.1):

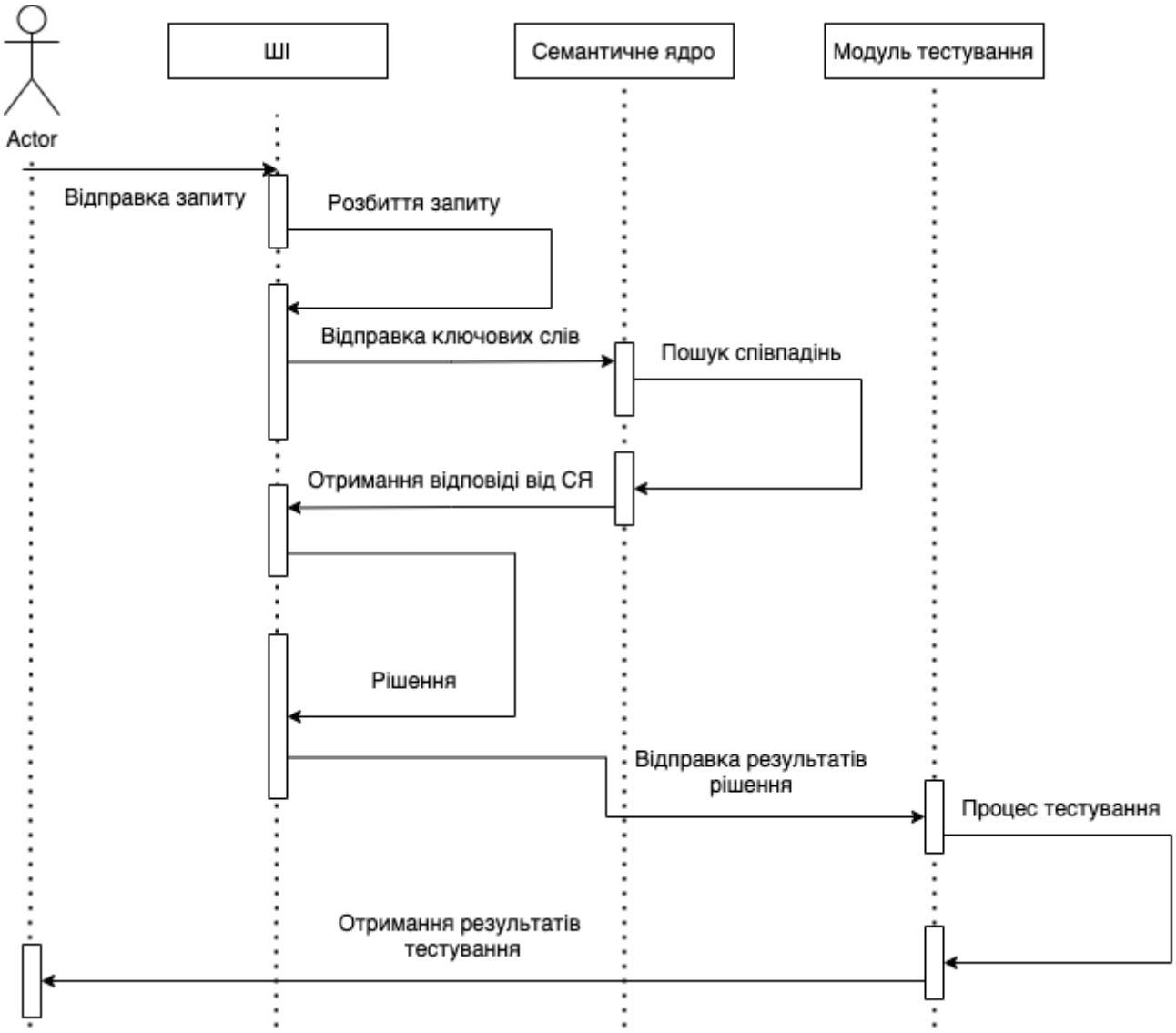


Рисунок 4.1 — Діаграма послідовностей

Для того, щоб наш штучний інтелект на базі Python, міг виконати ці дії ми можемо підключити до ПЗ наступні бібліотеки:

1) gensim — бібліотека для ЛСА python;
2) nltk — бібліотека для нормалізації слів;
3) scilitlearn – бібліотека для машинного навчання;

Gensim. Основними поняттями gensim є:

1. Документ - певний набір тексту.
2. Корпус - збірник документів.
3. Вектор - математично зручне подання документа.
4. Модель - алгоритм перетворення векторів з одного подання в інше.

Давайте розглянемо кожен із них трохи детальніше:

Документ. У Gensim документ є об'єктом типу текстової послідовності (загальновідомий як stry Python 3). Документом може бути що завгодно: коротке повідомлення із 140 символів, окремий абзац (тобто анотація статті до журналу), стаття новин або книга.

Корпус. Корпус являє собою сукупність документів об'єктів. В нього входять наступні моделі:

1. Вхідні дані для навчання моделі . Під час навчання моделі використовують цей навчальний корпус для пошуку загальних тем та тем, ініціалізуючи свої внутрішні параметри моделі. Gensim фокусується на неконтрольованих моделях , так що ні одне людське втручання, наприклад, дорогі анотації або позначки документів вручну, не потрібно.

2. Документи для організації. Після тренінгу можна використовувати тематичну модель для вилучення тем з нових документів (документів, яких не бачити в навчальному корпусі). Такі корпуси можна індексувати за запитами схожості , запитувати за семантичною подібністю, кластеризованими тощо.

Ось приклад корпусу. Він складається з 9 документів, де кожен документ є рядком, що складається з одного речення.

```
text_corpus = [
    If hairs be wires, black wires grow on her head",
    "But no such roses see I in her cheeks
    "But no such roses see I in her cheeks",
    "But no such roses see I in her cheeks",
    "I grant I never saw a goddess go",
    "My mistress when she walks treads on the ground",
    "Whose strength's abundance weakens his own heart"
]
```

Наведений приклад завантажує весь корпус в пам’ять. На практиці корпуси можуть бути дуже великими, тому завантаження їх у пам’ять може бути неможливим. Генсім розумно обробляє такі корпуси, передаючи їм по одному

документу. Це особливо невеликий приклад корпусу для ілюстративних цілей. Іншим прикладом може бути список усіх п’єс, написаних Шекспіром, список усіх статей у Вікіпедії або всі твіти певної людини, яка цікавить. Після збору нашого корпусу, як правило, ми хочемо здійснити низку етапів попередньої обробки. Ми зробимо це простіше, тому просто видалимо деякі загальновживані англійські слова (наприклад, „the“) та слова, які трапляються лише один раз у корпусі. У процесі цього ми будемо токенізувати наші дані. Токенізація розбиває документи на слова (у цьому випадку використання простору як роздільника).

Вектор. Щоб зробити висновок про приховану структуру нашого корпусу, нам потрібен спосіб представити документи, якими ми можемо маніпулювати математично. Одним із підходів є представлення кожного документа як вектора об’єктів . Наприклад, одну функцію можна розглядати як пару запитань-відповідей:

1. Скільки разів у документі з’являється слово splonge ? Нуль.
2. Скільки абзаців складається з документа? Два.
3. Скільки шрифтів використовує документ? П’ять.

Питання, як правило, представлене лише його цілочисельним ідентифікатором (наприклад, 1 , 2 та 3). Потім представлення цього документа стає низкою пар . Це відомо як щільний вектор , оскільки він містить чітку відповідь на кожне з вищезазначених питань.(1, 0.0), (2, 2.0), (3, 5.0) Якщо ми знаємо всі питання заздалегідь, ми можемо залишити їх неявними і просто представити документ як . Ця послідовність відповідей є вектором для нашого документа (в даному випадку тривимірним щільним вектором). З практичних цілей у Gensim допускаються лише запитання, відповідь на які (або їх можна перетворити) на одне число з плаваючою комою .(0, 2, 5) На практиці вектори часто складаються з багатьох нульових значень. Для економії пам'яті Gensim опускає всі векторні елементи зі значенням 0.0. Таким чином, наведений приклад стає . Це відоме як розріджений вектор або вектор мішка слів . Значення всіх відсутніх функцій в цьому рідкісному поданні може бути однозначно вирішені до нуля, .(2, 2.0), (3, 5.0)0.0. Припускаючи, що питання однакові, ми можемо порівняти вектори двох різних документів між собою. Наприклад, припустимо, що нам дано два вектори і . Оскільки вектори дуже

схожі між собою, ми можемо зробити висновок, що документи, що відповідають цим векторам, теж подібні. Звичайно, правильність такого висновку залежить від того, наскільки ми спочатку підібрали питання.(0.0, 2.0, 5.0)(0.1, 1.9, 4.9). Інший підхід до подання документа як вектора - це модель сумки слів . За моделлю "мішок слів" кожен документ представлений вектором, що містить підрахунок частоти кожного слова у словнику. Наприклад, припустимо, що у нас є словник, що містить ці слова . Тоді документ, що складається з рядка , буде представлений вектором, де записами вектора є (за порядком) випадки “кава”, “молоко”, “цукор” та “ложка” у документі. Довжина вектора - це кількість записів у словнику. Однією з головних властивостей моделі "мішок слів" є те, що вона повністю ігнорує порядок лексем у закодованому документі, звідки походить назва мішок слів.['coffee', 'milk', 'sugar', 'spoon']"coffee milk coffee"[2, 1, 0, 0]

Різниця між документом і вектором полягає в тому, що перший - це текст, а другий - це математично зручне подання тексту. Іноді люди вживають терміни як взаємозамінні: наприклад, давши якийсь довільний документ D, замість того, щоб сказати “вектор, який відповідає документу D”, вони просто скажуть “вектор D” або “документ D”. Це досягає стислості ціною двозначності. Поки ви пам’ятаєте, що документи існують у просторі документів, а вектори існують у векторному просторі, зазначена вище неоднозначність є прийнятною.

Модель. Тепер, коли ми векторизували наш корпус, ми можемо почати його трансформувати за допомогою моделей . Ми використовуємо модель як абстрактний термін, що стосується перетворення від одного представлення документа до іншого. У gensim документах представлені у вигляді векторів, тому модель може розглядатися як перетворення між двома векторними просторами. Модель дізнається подробиці цієї трансформації під час навчання, коли читає навчальний корпус .

Одним простим прикладом моделі є tf-idf . Модель tf-idf перетворює вектори з подання мішка слів у векторний простір, де відліки частот зважуються відповідно до відносної рідкості кожного слова в корпусі. Модель знову повертає список кортежів, де перший елемент є маркером ідентифікатор і другий елемент являє

собою TF-IDF зважування. Зверніть увагу, що ідентифікатор, що відповідає “системі” (що траплялося 4 рази в початковому корпусі), зважений нижче, ніж ідентифікатор, що відповідає “неповнолітнім” (що сталося лише двічі).

Після того, як ви створили модель, ви можете робити з нею всі цікаві речі. Наприклад, щоб перетворити весь корпус через TfIdf та індексувати його, готуючись до запитів подібності. А потім запитати подібність нашого документа запиту query_document щодо кожного документа в корпусі.

Scikit-learn- це бібліотека машинного навчання з відкритим кодом, яка підтримує навчання під наглядом та без нагляду. Він також надає різні інструменти для підгонки моделі, попередньої обробки даних, вибору та оцінки моделі та багатьох інших утиліт.

Встановлення та прогнозування: основи оцінювача

Scikit-learn забезпечує десятки вбудованих алгоритмів та моделей машинного навчання, які називаються оцінювачами . Кожен оцінювач може бути пристосований до деяких даних за допомогою його методу підгонки .Метод підгонки зазвичай приймає 2 входи:

Матриця X . Розмір X зазначає, що зразки представлені у вигляді рядків, а об’єкти у вигляді стовпців.(n_samples, n_features)

Цільові значення Y, які є дійсними числами для завдань регресії або цілими числами для класифікації (або будь-якого іншого дискретного набору значень). Для некерованих навчальних завдань Y не потрібно вказувати. Y - Зазвичай це 1d масив, де цей i-запис відповідає цільовому i-зразку (рядку) X.

Як x і y зазвичай , як очікується, будуть Numpy масивів або еквівалентний масив як типи даних, хоча деякі оцінювачі працюють з іншими форматами , такі як розріджені матриці.

Трансформатори та попередні процесори . Процеси машинного навчання часто складаються з різних частин. Типовий конвеєр складається з етапу попередньої обробки, який перетворює або вводить дані, та остаточного предиктора, який передбачає цільові значення.

У scikit-learn попередніх процесорах і трансформаторах використовується той самий API, що і об’єкти оцінки (насправді всі вони успадковуються від одного BaseEstimator класу). Об'єкти трансформатора не мають методу передбачення, а швидше метод перетворення, який виводить нещодавно перетворену матрицю вибірки X.

Іноді вам потрібно застосувати різні перетворення до різних функцій: ColumnTransformer призначений для цих випадків використання.

Трубопроводи: ланцюг попередніх процесорів та оцінювачів . Трансформатори та оцінювачі (предиктори) можуть бути об'єднані в один об'єднуючий об'єкт: a Pipeline. Конвеєр пропонує той самий API, що і звичайний оцінювач: його можна встановити та використовувати для прогнозування за допомогою fit та predict. Як ми побачимо пізніше, використання конвеєра також запобіжить вам витік даних, тобто розкриття деяких даних тестування у ваших навчальних даних.

Оцінка моделі . Пристосування моделі до деяких даних не означає, що вона буде добре передбачати невидимі дані. Це потрібно безпосередньо оцінити. Ми щойно бачили train_test_split помічника, який розділяє набір даних на тренувальні та тестові набори, але scikit-learn надає багато інших інструментів для оцінки моделей, зокрема для перехресної перевірки .

Автоматичний пошук параметрів . Усі оцінювачі мають параметри (які в літературі часто називають гіперпараметрами), які можна налаштувати. Потужність узагальнення оцінювача часто критично залежить від кількох параметрів. Наприклад, a RandomForestRegressorмає n_estimators параметр, який визначає кількість дерев у лісі, і max_depthпараметр, який визначає максимальну глибину кожного дерева. Досить часто незрозуміло, якими мають бути точні значення цих параметрів, оскільки вони залежать від наявних даних.

Scikit-learn надає інструменти для автоматичного пошуку найкращих комбінацій параметрів (за допомогою перехресної перевірки). У наступному прикладі ми виконуємо випадковий пошук по простору параметрів випадкового лісу з RandomizedSearchCV об'єктом. Коли пошук закінчується,

RandomizedSearchCV поводиться як той RandomForestRegressor, який був забезпечений найкращим набором параметрів.

На практиці ми майже завжди хочемо шукати за конвеєром , а не за одним оцінювачем. Однією з основних причин є те, що якщо ми застосовуєм крок попередньої обробки до цілого набору даних без використання конвеєра, а потім виконуєм будь-яку перехресну перевірку, то ми порушуємо основне припущення про незалежність між даними навчання та тестування. Дійсно, оскільки ми попередньо обробили дані, використовуючи весь набір даних, деяка інформація про тестові набори доступна для поїзних комплектів. Це призведе до надмірної оцінки сили узагальнення оцінювача.

4.3. Функціонування Інтелектуалізованої системи на основі онтологій для галузі якості програмного забезпечення

Для того щоб отримати хоча б уявну модель процесу функціонування інтелектуалізовані системи на основі онтологій для галузі якості програмного забезпечення, переглянемо створений концепт – макет (рисунок 4.2).

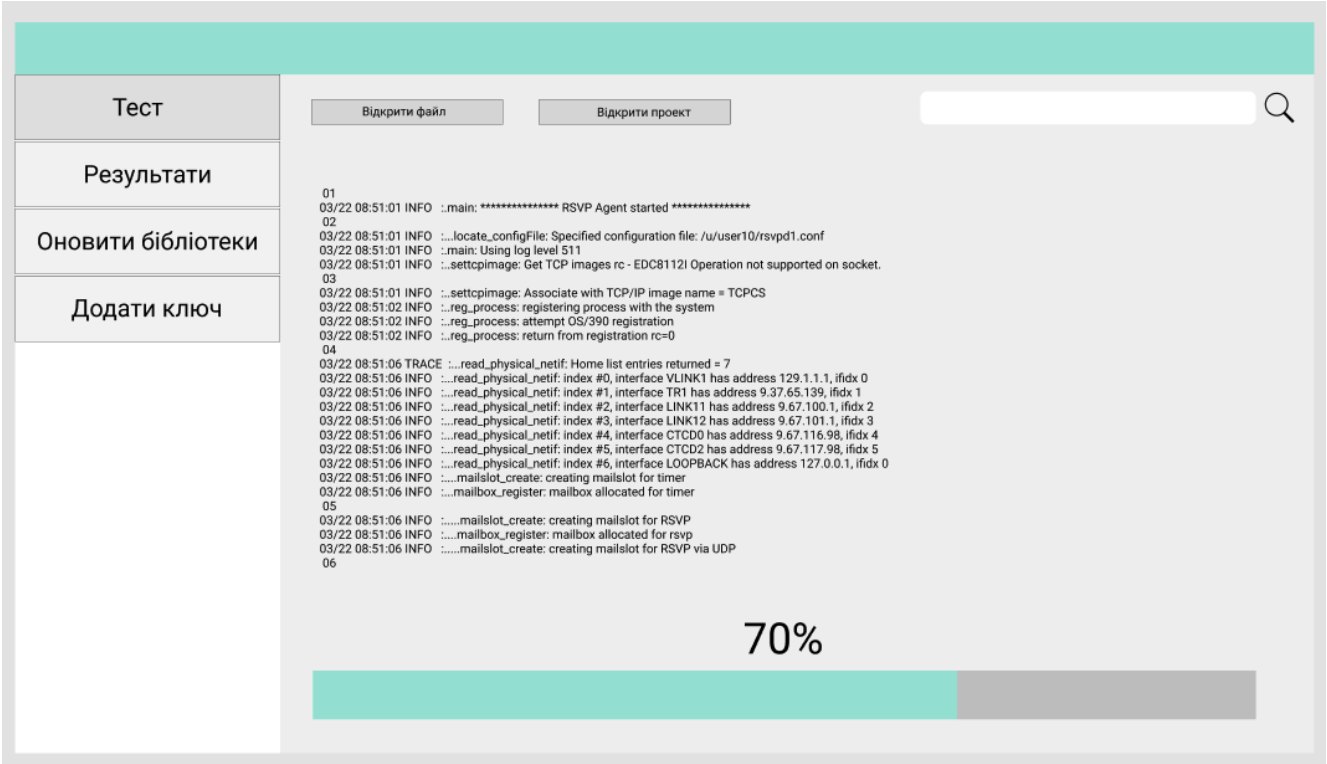


Рисунок 4.2 —Концепт інтерфейсу на першому екрані

На рисунку 4.2 ми можемо з вами візуально переглянути інтерфейс майбутнього програмного забезпечення. На даному етапі для користувача доступні лише чотири пункти меню, однак з підключенням нових бібліотек, він зможе отримувати більший функціонал, відповідно пунктів меню буде ставати все більше.

На першому (головному) екрані, для користувача надається змога завантажити окремий файл, якщо йому необхідно перевірити певний модуль чи компонент, або ж завантажити проект. Справа від цих елементів керування знаходиться поле для введення ключових слів які буде обробляти програмне забезпечення. Нижче користувач може бачити log - рядки, якщо в нього виникає потреба їх перевіряти, і більш простий progressbar, для відстежування процесу перевірки модулю або проекту.

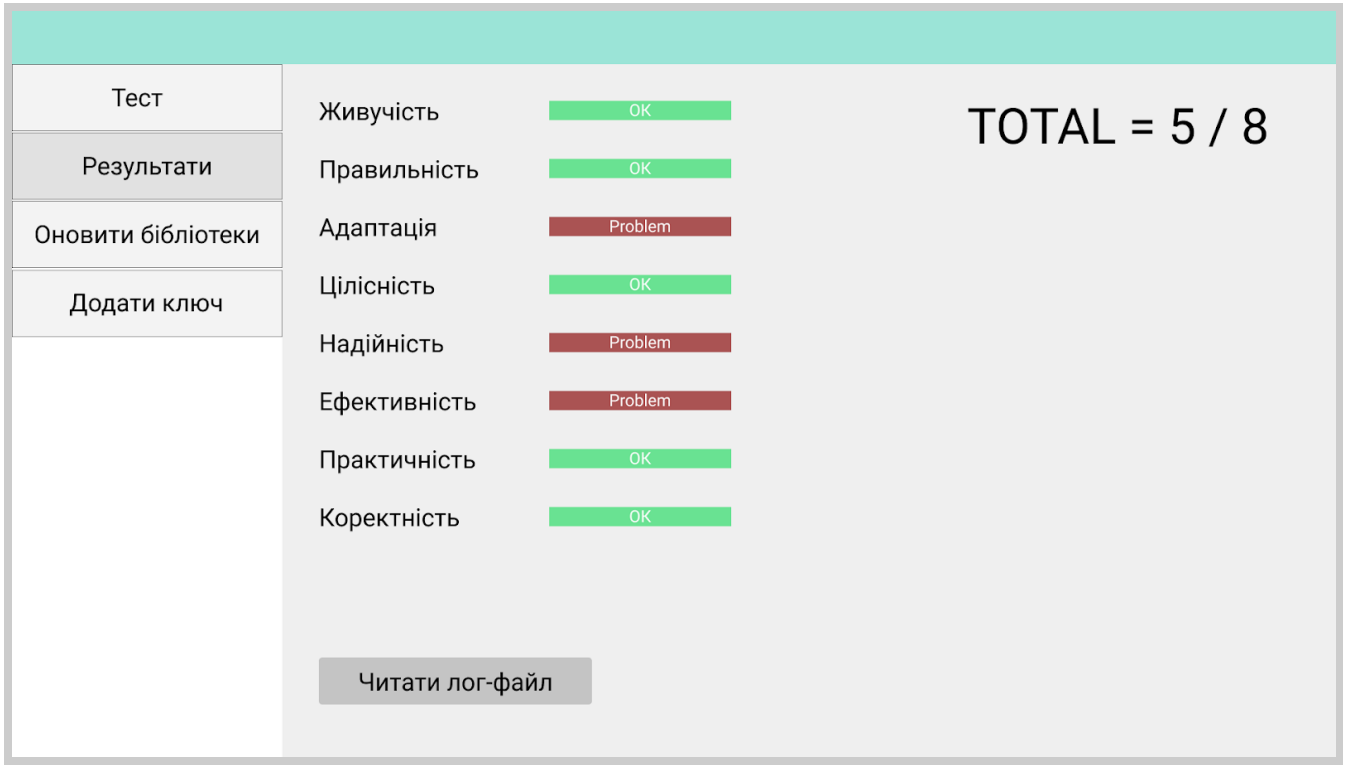


Рисунок 4.3 — Концепт інтерфейсу результатів перевірки

На даному екрані (рисунок 4.3), користувач має змогу побачити результати перевірки, та отримати загальну оцінку перевіреного файла або проекту.

Поля зліва відповідають наступним характеристикам:

1. Живучість – можливість програми продовжувати свою роботу, навіть після введення недопустих даних.

2. Правильність – визначена ступінь програмного забезпечення, особливо в ситуації, що програма має опрацьовувати велику кількість даних. Окрім того, правильність, надає характеристику про виконання системи її функцій, а не, а не те чи створена вона коректно.

3. Надійність - можливість системи виконувати функції в прогнозуємих умовах;

4. Цілісність – можливість системи запобігати некоректних чи неавторизованих викликів що запрошують доступ до даних. Концепція цілісності полягає в тому, щоб обмежити доступ до програмного забезпечення для неаутинтифікованих користувачів.

5. Адаптуємість – можливість використовувати систему без зміни її компонентів, для об’єктів для яких вона не була орієнтована;

6. Ефективність - міра, яка описує ступінь використання системних ресурсів, в цій характеристиці враховуються такі фактори, як швидкодія і використовуваний об’єм пам’яті.

У випадку, якщо компонент або ж перевіямий проект пройшов перевірку відповідно до пункту, що зазначений вище він отримує певний статус - “OK”, якщо проблем не знайдено, або ж “Problem”, якщо були знайдені певні помилки.

Якщо користувач хоче прочитати звіт програми про знайдені проблеми, він може натиснути на кнопку “Читати лог-файл” і отримати текстовий документ з посиланнями на помилки в програмному кодові.

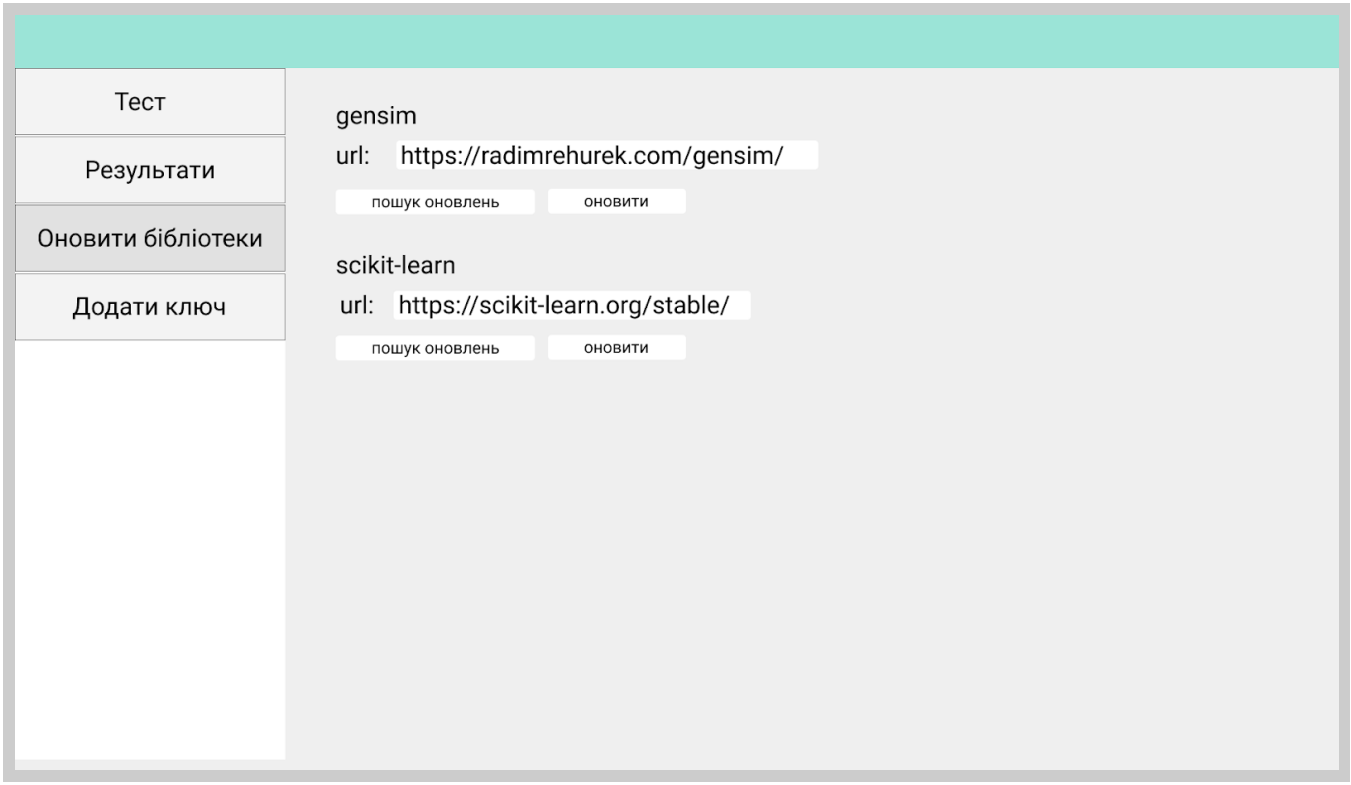


Рисунок 4.4 — Концепт інтерфейсу оновлення бібліотек

На рисунку 4.4 ми можемо переглянути інтерфейс, що надає змогу для користувача оновити бібліотеки, що використовує дане програмне забезпечення. Пізніше буде надана змога заміняти їх, або ж підключати нові.

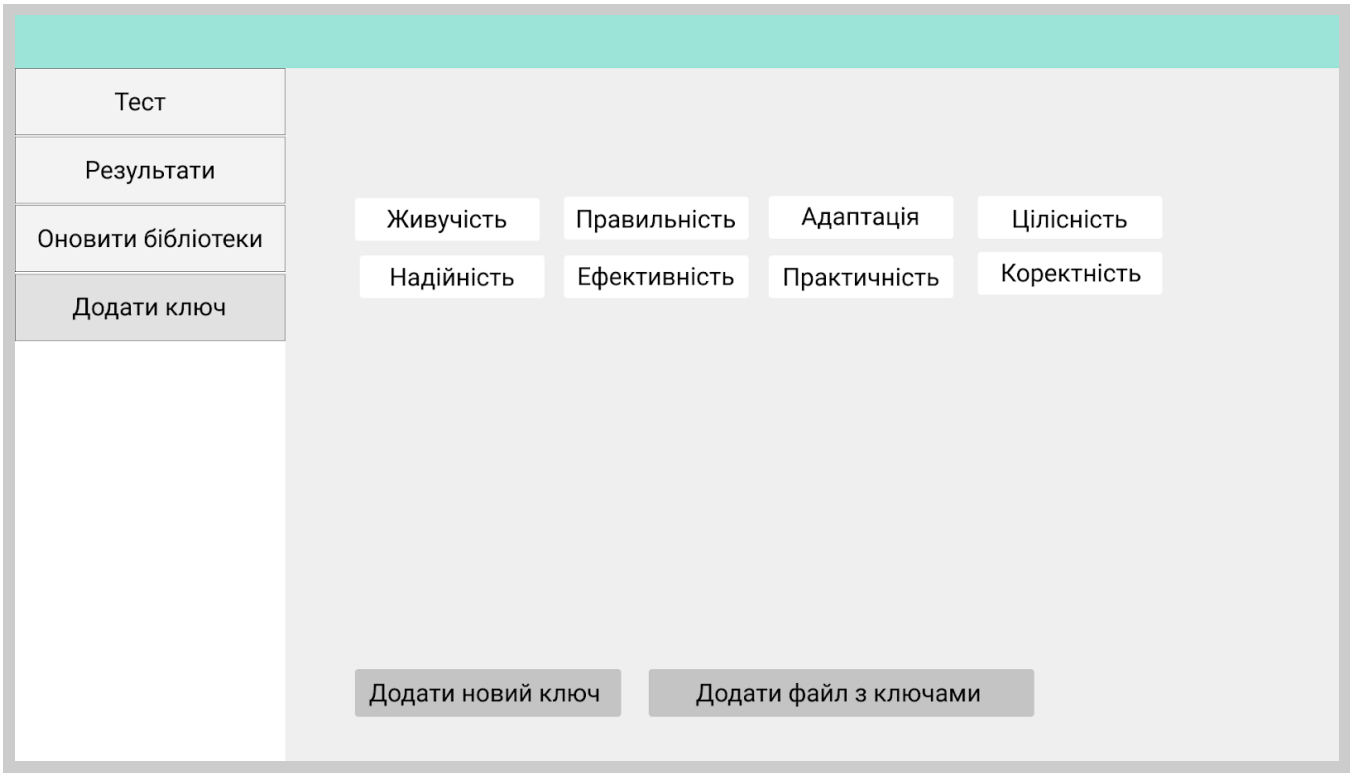


Рисунок 4.5 — Концепт інтерфейсу додавання ключів

На рисунку 4.5 ми можемо переглянути інтерфейс, що надає змогу для користувача додавати нові ключові слова, для збільшення семантичного ядра програми. Він зможе додати нове специфічне слово, або ж окремий файл з ключами.

4.4 Висновки

Був проведений аналіз мов програмування для проектуємого ПЗ, аналіз існуючих бібліотек для мови Python. Окрім того, були представлені концепт – інтерфейси для майбутнього програмного забезпечення.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено інтелектуалізовану систему на основі онтологій для галузі якості програмного забезпечення.

У першому розділі проведений аналіз відомих методів і рішень для опрацювання інформації щодо якості програмного забезпечення, технології виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

У другому розділі був змодельований процес опрацювання інформації для галузі якості програмного забезпечення на основі онтологій, а також набули подальшого розвитку моделі процесу опрацювання інформації для галузі якості програмного забезпечення, які ґрунтуються на онтологіях предметної галузі якості програмного забезпечення та є теоретичним підґрунтям методу пошуку інформації для галузі якості програмного забезпечення на основі онтологій.

У третьому розділі був розроблений метод опрацювання інформації щодо якості програмного забезпечення на основі онтологій, який працює на основі розроблених онтологічних моделей та забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

У четвертому розділі спроєктовано та реалізовано інтелектуалізовану систему на основі онтологій для галузі якості програмного забезпечення, яка забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації щодо якості програмного забезпечення.

За темою дипломної роботи опублікована одна стаття у фаховому науковому виданні Computer Systems and Information Technologies [1].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Говорущенко Т.О., Мартинюк Ю.С. Концепція інформаційно-пошукової системи (на основі онтологій) для галузі якості програмного забезпечення. Computer Systems and Information Technologies. 2020. №1. С.6-11 // http://csitjournal.khmnu.edu.ua/index.php/csit/article/view/5

2. ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality. [Introduced 31.03.2016]. Geneva (Switzerland), 2016. 45 p. (International standard).

3. ISO/IEC/IEEE 24765:2017. Systems and software engineering. Vocabulary. [Introduced 01.10.2017]. Geneva (Switzerland), 2017. 522 p. (International standard).

4. Говорущенко Т. О. Методологія оцінювання достатності інформації для визначення якості програмного забезпечення: монографія. Хмельницький: Хмельницький національний університет, 2017. 310 с.

5. Fenton N., Bieman J. Software metrics: a rigorous and practical approach. CRC Press, 2014. 617 p.

6. Мищенко В. О. CASE-оценка критических программных систем: в 3 т. Т. 1. Качество: монография / Мищенко В. О., Поморова О. В., Говорущенко Т. А. ; под ред. В. С. Харченка. Харьков: Нац. аэрокосмический университет «ХАИ», 2012. 201 с.

7. Маєвський Д. А. Теоретичні та прикладні основи забезпечення якості динамічних інформаційних систем: дис. ... доктора техн. наук: 05.13.06. Одеса, 2013. 440 с.

7. Маевский Д., Козина Ю. Где и когда формируется качество программного обеспечения? *Электротехнические и компьютерные системы.* 2015. № 18. С. 55-59.

8. Klas M., Lampasona C., Munch J. Adapting software quality models: practical challenges, approach, and first empirical results. *The 37-th EUROMICRO Conference on*

Software Engineering and Advanced Applications: Proceedings (Oulu, August 30 – September 2, 2011). Oulu (Finland), 2011. Pp. 341-348.

9. Miguel J. P., Mauricio D., Rodríguez G. A review of software quality models for the evaluation of software products. *International Journal of Software Engineering & Applications*. 2014. Vol. 5. No. 6. Pp. 31-54.

10. Gordieiev O., Kharchenko V., Fusani M. Evolution of software quality models: green and reliability issues. *CEUR-WS*, 2015. Vol. 1356. Pp. 432-445.

11. Dubey S. K., Ghosh S., Rana A. Comparison of software quality models: an analytical approach. *International Journal of Emerging Technology and Advanced Engineering.* 2012. Vol. 2. Issue 2. Pp. 111-119.

12. Relating system quality and software architecture / I. Mistrik and others. Elseiver Inc., 2014. 379 p.

13. Software quality models: a comparative study / A. B. AL-Badareen and others. *The 2-nd International Conference on Software Engineering and Computer Systems*: Proceedings (Pahang, June 27-29, 2011). Pahang (Malaysia), 2011. Pp. 46-55.

14. Ghayathri J., Priya E. M. Software quality models: a comparative study. *International Journal of Advanced Research in Computer Science and Electronics Engineering*. 2013. Vol. 2. Issue 1. Pp. 42-51.

15. Al-Qutaish R. E. Quality models in software engineering literature: an analytical and comparative study. *Journal of American Science*. 2010. Vol.6. Pp. 166-175.

16. Suman M. W. A comparative study of software quality models. *International Journal of Computer Science and Information Technologies*. 2014. Vol. 5. Pp. 5634-5638.

17. Kocak S. A., Alptekin G. I., Bener A. B. Evaluation of software product Evaluation of software product quality attributes and environmental attributes using ANP decision framework. *CEUR-WS*. 2014. Vol. 1216. Pp. 37-44.

18. Adewumi A., Misra S., Omoregbe N. Evaluating open source software quality models against ISO 25010. *IEEE Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure*

Computing; Pervasive Intelligence and Computing: Proceedings (Liverpool, October 26-28, 2015). Liverpool (United Kingdom), 2015. Pp. 872-877.

19. Говорущенко Т. О. Методологія оцінювання достатності інформації для визначення якості програмного забезпечення: монографія. Хмельницький: Хмельницький національний університет, 2017. 310 с

20. Guillerm R., Demmou H., Sadou N. Safety evaluation and management o complex systems: A system engineering approach. *Concurrent Engineering: Research and Applications*. 2012. Vol. 20 (2). Pp. 149-159.

21. Говорущенко Т. О., Мартинюк Ю. С. Концепція інформаційно-пошукової системи (на основі онтологій) для галузі якості програмного забезпечення. – 2020.

22. Pizzi N. J. Mapping software metrics to module complexity: a pattern classification approach. *Journal of Software Engineering and Applications*. 2011. Vol. 4. Pp. 426-432.

23. Інформаційно-пошукова система [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/socialnafnformatika/home/informacijno-posukovi-sistemi

24. Інформаційно-пошукові системи [Електронний ресурс]. – Режим доступу: http://esu.com.ua/search_articles.php?id=12483

25. Поняття і класифікація інформаційно-пошукових систем [Електронний ресурс]. – Режим доступу:

https://library.if.ua/book/100/6867.html

26. Вовк Н. Архівні інформаційно-пошукові системи: шляхи оптимізації пошуку текстової інформації / Н. Вовк. – Бібліотекознавство. Документознавство. Інформологія. – 2018. – №3. – С. 37-42.

27. Пиріг С. Інформаційні технології та їх використання на підприємствах України / С.Пиріг, О.Нужна. – Економічний форум. – 2014. – № 3. – С. 190–195.

28. Винничук Р. Особливості розвитку ІТ-ринку в Україні: стан та тенденції / Р.Винничук, Т.Склярук. – Вісник Національного університету «Львівська політехніка». Серія «Логістика». – 2015. – № 833. – С. 3-8.

29. Hovorushchenko T. Information Technology for Assurance of Veracity of Quality Information in the Software Requirements Specification. *Advances in Intelligent Systems and Computing*. 2018. Vol. 689. Pp. 166-185.

30. Говорущенко Т.О. Теоретичні та прикладні засади інформаційної технології оцінювання достатності інформації щодо якості у специфікаціях вимог до програмного забезпечення: дис. ... доктора техн. наук: 05.13.06. – Львів, 2018. – 441 с. у специфікаціях вимог до програмного забезпечення: дис. ... доктора техн. наук: 05.13.06. – Львів, 2018. – 441 с.

31. Sivarajah U. Critical analysis of Big Data challenges and analytical methods / U.Sivarajah, M.Kamal, Z.Irani, V.Weerakkody. – *Journal of Business Research*. – 2017. – Vol. 70. – Pp. 263-286.

32. Luo Y. How valuable is information and communication technology? A study of emerging economy enterprises / Y.Luo, J.Bu. – *Journal of World Business*. – 2016. – Vol. 51. – Issue 2. – Pp. 200-211.

33. Tian X. Big data and knowledge management: a case of deja vu or back to the future? / X. Tian. – *Journal of Knowledge Managament*. – 2017. – Vol. 21. – Issue 1. – Pp. 113-131.

34. Mauerhoefer T. The impact of information technology on new product development performance / T.Mauerhoefer, S.Strese, M.Brettel. – *Journal of Product Innovation Management*. – 2017. – Vol. 34. – Issue 6. – Pp. 719-738.

35. Press G. Top 10 hot artificial intelligence (AI) technologies [Electronic resource] / G. Press. – Access mode: https://www.forbes.com/sites/gilpress/2017/01/23/top-10-hot-artificial-intelligence-ai-technologies/#1d5da9561928

36. Литвин В.В. Методи та засоби опрацювання інформаційних ресурсів на основі онтологій / В.В. Литвин, В.А. Висоцька, Д.Г. Досин. – Львів: ЛА «Піраміда», 2016. – 404 с.

37. Говорущенко Т. О. Методологія оцінювання достатності інформації для визначення якості програмного забезпечення: монографія. Хмельницький: Хмельницький національний університет, 2017. 310 с

38. 2011. Systems and software engineering. Life cycle processes. Requirements engineering. [Introduced 01.12.2011]. Geneva (Switzerland), 2011. 28 p. (International standard). 39. ISO/IEC 15504-2:2003. Information technology. Process assessment. Part 2: Performing an assessment. [Introduced 15.10.2003; revised by ISO/IEC 33002:2015]. Geneva (Switzerland), 2003. 16 p. (International standard).

40. Маевский Д., Козина Ю. Где и когда формируется качество программного обеспечения? *Электротехнические и компьютерные системы*. 2015. № 18. С. 55-5941. Черников Б. В. Управление качеством программного обеспечения. Москва: Изд. «Форум», 2012. 240 с.

42. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. [Introduced 01.03.2011]. Geneva (Switzerland), 2011. 34 p. (International standard).

43. Bourque P., Fairley R. E. Guide to the software engineering body of knowledge (SWEBOK): Version 3.0. IEEE Computer Society, 2014. 335 p.

44. ISO/IEC TR 19759:2015. Software Engineering. Guide to the software engineering body of knowledge (SWEBOK). [Introduced 01.10.2015]. Geneva (Switzerland), 2015. 336 p. (International standard).

45. Selection of third party software in Off-The-Shelf-based software development – An interview study with industrial practitioners / C. Ayala and others.

46. ISO/IEC 33002:2015. Information technology. Process assessment. Requirements for performing process assessment. [Introduced 01.03.2015]. Geneva (Switzerland), 2015. 16 p. (International standard).

47. Chrissis M., Konrad M., Shrum S. CMM: Guidelines for process integration and product improvement: 3rd edition. Addison-Wesley Professional, 2011. 688 p

48. ISO/IEC 25000 series of standards. Web-site. URL: http://iso25000.com/index.php/en/iso-25000-standards (Last accessed: December 7, 2017).

49. Говорущенко Т. О. Дослідження відомих моделей оцінювання характеристик програмного забезпечення. *Вісник Хмельницького національного університету*. Серія «Технічні науки». 2013. № 1. С. 117-121.

50. Chrissis M., Konrad M., Shrum S. CMM: Guidelines for process integration and product improvement: 3rd edition. Addison-Wesley Professional, 2011. 688 p

51. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. [Introduced 01.03.2011]. Geneva (Switzerland), 2011. 34 p. (International standard).

52. Dubey S. K., Ghosh S., Rana A. Comparison of software quality models: an analytical approach. *International Journal of Emerging Technology and Advanced Engineering*. 2012. Vol. 2. Issue 2. Pp. 111-119.

53. ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality. [Introduced 31.03.2016]. Geneva (Switzerland), 2016. 45 p. (International standard).

54. Рогушина Ю. Використання онтологій для персоніфікації семантичного пошуку / Ю. Рогушина. – Проблеми програмування. – 2017. – №3. – С. 52- 67.

56. Рогушина Ю. Семантичний пошук у Web на основі онтологій: розробка моделей, засобів і методів / Ю. Рогушина. – Мелітополь: МДУ-ПУ ім. Б. Хмельницького, 2015. – 291 с.

57. ISO/IEC 25030:2019. Systems and software engineering. Systems and software quality requirements and evaluation (SQuaRE). Quality requirements framework [Introduced 01.09.2019]. – Geneva (Switzerland), 2019. – 46 p. (International standard).

58. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. [Introduced 01.03.2011]. – Geneva (Switzerland), 2011. – 34 p. (International standard).

59. ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality. [Introduced 31.03.2016]. Geneva (Switzerland), 2016. 45 p. (International standard).

60. Липаев В. В. Программная инженерия: методологические основы. Москва-Берлин: Директ-Медиа, 2015. 608 с.

61. Fenton N., Bieman J. Software metrics: a rigorous and practical approach. CRC Press, 2014. 617 p.

201662. Липаев В. В. Основные понятия, факторы и стандарты, определяющие качество крупномасштабных программных средств. Москва-Берлин: ДиректМедиа, 2015. 237 с.

63. Лаврищева Е. М. Software Engineering компьютерных систем. Парадигмы, технологии и CASE-средства программирования. Киев: Наукова думка, 2013. 283 с.

64. Jones C., Bonsignour O. The economics of software quality. Boston: Pearson Education, 2012. 588 p.

65. Hill P. Practical software project estimation: a toolkit for estimating software development effort & duration. McGraw-Hill Education, 2010. 312 p.

66. Abran A. Software project estimation: the fundamentals for providing high quality information to decision makers. Wiley-IEEE Computer Society Press, 2015. 288 p.

67. Bird Ch., Menzies T., Zimmermann Th. The art and science of analyzing software data. Morgan Kaufmann, 2015. 672 p.

68. ДСТУ 3230-95. Управління якістю та забезпечення якості. Терміни та визначення. [Чинний від 01.07.1996]. Київ, 1996. 37 с. (Національний стандарт України).

69. International Function Point Users Group. Web-site. – URL: http://www.ifpug.org/ (Last accessed: December 7, 2017).

ДОДАТОК А

(обов'язковий)

ОПУБЛІКОВАНА СТАТТЯ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

УДК 004.9: 004.05

Т. О. ГОВОРУЩЕНКО, Ю. С. МАРТИНЮК
Хмельницький національний університет

КОНЦЕПЦІЯ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ (НА ОСНОВІ ОНТОЛОГІЙ) ДЛЯ ГАЛУЗІ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті запропоновано концепцію інформаційно-пошукової системи (на основі онтологій) для галузі якості програмного забезпечення, зокрема, розроблено онтологію предметної галузі якості програмного забезпечення, яка відображає семантичні відношення між концептами предметної галузі та ляже в основу пошуку інформації про якість програмного забезпечення, зокрема, в основу тезаурусу майбутньої інформаційно-пошукової системи.

Ключові слова: інформаційно-пошукова система, пошук інформації,інформаційні ресурси, онтологія, якість програмного забезпечення (ПЗ).

T. O. HOVORUSHCHENKO, Y. S. MARTYNYUK
Khmelnytskyi National University

CONCEPT OF INFORMATION-SEARCH SYSTEM (ON THE BASIS OF ONTOLOGIES) FOR THE DOMAIN OF SOFTWARE QUALITY

Abstract – The search for information is one of the main components of human activity. The ideal information retrieval system should issue only documents that are relevant to the request. Today, real information retrieval systems provide a completeness factor of 70%, and a search accuracy factor – at a level sometimes even 10%. Thus, the well-known information retrieval systems are currently unable to meet the modern needs of users.

The global trend in the processing of large arrays of information, which allows you to solve new classes of problems based on available information resources, is the intellectualization of information and data processing. As a standard of knowledge engineering in the development of information retrieval systems, it is worthwhile to use ontologies that are widely used in the work of search engines and information retrieval systems, as ontologies are an effective tool for organizing a semantic search. The use of ontologies as part of information retrieval systems helps to solve a number of methodological and technological problems that arise during the development of such systems.

Given the fact that software is the basis of all modern areas of business, and achieving high values of its quality is a key factor in ensuring the effective use of software and one of the main requirements of users and stakeholders for modern software, the purpose of this study is to develop an effective information retrieval system (based on ontologies) for the software quality industry.

The paper proposes the concept of information retrieval system (based on ontologies) for the field of software quality, in particular, the ontology of the subject area of software quality is developed, which reflects the semantic relationships between the concepts of the subject area and will form the basis of a search of information about software quality, in particular, the basis of the thesaurus of the future information retrieval system.

Keywords: information retrieval system, information retrieval, information resources, ontology, software quality.

Вступ

Пошук інформації є однією з основних складових людської діяльності. Людині, яка часто має справу з підбором і пошуком якої-небудь тематичної інформації, рано чи пізно (зі зростанням її обсягу) доводиться застосовувати деякі принципи систематизації і класифікації наявних даних, що забезпечують більш зручний і ефективний пошук. Інформаційно-пошукові системи з'явилися досить давно. Гіпертекстова модель World Wide Web не вирішує проблему пошуку інформації у великих обсягах різнорідних документів. На сьогоднішній день немає іншого способу швидкого пошуку даних, крім пошуку за ключовими словами [1].

Інформаційно-пошукова система (ІПС) – програмна система для збереження, пошуку і видачі інформації, необхідної користувачу [1]. Інформаційно-пошукові системи – автоматизовані системи, призначені для зібрання, пошуку, оброблення, збереження та видачі інформації за допомогою технічних засобів [2]. Користувач звертається до ІПС з інформаційним запитом. Пошук інформації ведеться в пошуковому масиві, що формується (і за необхідності оновлюється) розробниками чи адміністраторами системи. Елементи пошукового масиву вводяться в

інформаційно-пошукову систему природньою мовою, а потім звичайно піддаються перекладу на формальну інформаційно-пошукову мову [1].

Інформаційно-пошукові системи – це різновид автоматизованих інформаційних систем, в яких завершальна обробка даних не передбачається. Ці системи призначені для пошуку текстів (документів, їх частин, фактографічних записів) в сховищах (базах даних) за формальними характеристиками. Тому в роботі ІПС можна виділити два основних етапи: перший – збір і зберігання інформації, другий – пошук і видача інформації користувачам [3].

Інформаційно-пошукова система (ІПС) – це сукупність довідково-інформаційного фонду і технічних засобів інформаційного пошуку в ньому. У свою чергу, довідково-інформаційний фонд (ДІФ) – це сукупність інформаційних масивів і пов’язаного з ними довідково-пошукового апарату (тобто даних про адреси зберігання документів з певними пошуковими образами документа). Пошуковий образ документа – це текст, що складається з лексичних одиниць інформаційно-пошукової мови, що виражає основний смисловий зміст документа і призначений для реалізації інформаційного пошуку [4].

Ідеальна ІПС повинна видавати виключно документи, змістовно релевантні запиту. Однак на практиці це зазвичай не досягається – спостерігаються мовчання ІПС (невидача деякої кількості релевантних документів) і шум ІПС (видача зайвих документів) [1]. В ідеальної ІПС повнота і точність становить 100%, а шум становить 0 (знайдені всі документи і не видано жодного зайвого). У реальних системах коефіцієнт повноти досягає 70%, а коефіцієнт точності пошуку коливається в дуже широких межах, іноді знижуючись до 10%. Величини цих коефіцієнтів залежать від цілого ряду факторів – як від внутрішніх властивостей пошукової системи (обсягу і характеристик інформаційного масиву, інформаційно-пошукової мови, критерію видачі), так і від багатьох "зовнішніх" умов: ступеня специфічності інформаційних запитів, здатності користувача правильно сформулювати свої інформаційні потреби природною мовою, правильності побудови конкретного запиту, а також від суб'єктивного представлення користувача про те, яка інформація йому потрібна [1].

Розвиток інформаційного суспільства безпосередньо пов'язаний з необхідністю збору, обробки і передачі величезних об'ємів інформації, що стало причиною глобального переходу від індустріального суспільства до інформаційного. Інформатизація суспільства – це глобальний соціальний процес, особливість якого полягає в тому, що домінуючим видом діяльності в сфері суспільного виробництва є збір, накопичення, продукування, обробка, зберігання, передача та використання інформації, здійснювані на основі сучасних засобів обчислювальної техніки, а також на базі різноманітних засобів інформаційного обміну. При інформатизації суспільства основна увага приділяється комплексу заходів, спрямованих на забезпечення повного використання достовірного, вичерпного і своєчасного знання у всіх видах людської діяльності. Інформатизація суспільства спрямована на якнайшвидше оволодіння інформацією для задоволення потреб. Процеси, що відбуваються у зв'язку з інформатизацією суспільства, сприяють не тільки прискоренню науково-технічного прогресу, інтелектуалізації всіх видів людської діяльності, а й створенню якісно нового інформаційного середовища. Основними критеріями інформаційного суспільства є: кількість і якість наявної в обігу інформації, ефективність її передавання та опрацювання, доступність інформації для кожного [5-8]. Зараз компанії змушені використовувати якнайбільше джерел інформації, щоб покращити результати свого бізнесу, а також якість послуг і досвід, які отримують клієнти. Управління інформацією стає функцією, критично важливою для бізнесу. Сучасне матеріальне виробництво та інші сфери діяльності все більше потребують інформаційного обслуговування, переробки значних об’ємів інформації [7, 8]. Традиційні способи та підходи (здебільшого засновані на рішеннях бізнес-аналітики та системах управління базами даних) не можуть бути застосовані до таких масивів інформації. Застосування відомих методів та засобів до опрацювання великих масивів інформації не виправдовує очікувань розробників, призводить до перевитрат ресурсів, втрат значущої інформації та конфліктів між очікуваннями замовників та отриманими результатами [8-12]. Загальносвітовим трендом щодо опрацювання великих масивів інформації, що дозволяє вирішувати нові класи задач на основі наявних інформаційних ресурсів, є інтелектуалізація опрацювання інформації і даних [13, 14]. Поява потужних технічних ресурсів для накопичення та опрацювання значних масивів інформації, відсутність ефективних методів та засобів для опрацювання таких масивів інформації, розвиток інтелектуалізації опрацювання інформації є передумовами для переходу на новий якісний рівень опрацювання інформації [8].

Як стандарт інженерії знань при розробленні інформаційно-пошукових систем варто і доцільно використовувати онтології, які широко застосовуються в роботі пошукових машни та інформаційно-пошукових систем [14]. Онтологія – це детальна формалізація деякої галузі знань за допомогою концептуальної схеми. Така схема, зазвичай, складається з ієрархічної структури даних, що містить всі релевантні класи об’єктів, їх зв’язків, теорем та обмежень, які прийняті у певній предметній галузі. Використання онтологій у складі ІПС допомагає вирішити низку проблем методологічного та технологічного характеру, які виникають під час розроблення таких систем. Зокрема, в Україні такі проблеми полягають у відсутності концептуальної цілісності й узгодженості окремих прийомів та методів інженерії знань; нестачі кваліфікованих фахівців у цій галузі; жорсткості розроблених програмних засобів та їх низької адаптивної здатності; складності впровадження ІПС, що зумовлено психологічними аспектами [14]. Онтології є ефективним інструментом для організації семантичного пошуку (пошук інформаційних ресурсів та інформаційних об’єктів, семантична близькість яких до інформаційних потреб користувача визначається із застосуванням знань щодо суб’єктів і об’єктів пошукової процедури) [15]. Крім цього, онтології можуть допомогти забезпечити пошук інформаційних ресурсів, необхідних користувачеві для вирішення його поточної задачі з використанням знань про самого користувача, його сферу інтересів, його задачі та про досвід іншихкористувачів із подібними інформаційними потребами [15].

Одна з перших інформаційно-пошукових систем на основі онтологій – SHOE – дозволяє користувачам задавати структуровані запити з використанням онтології [15-17].

Більшість ІПС, що використовують онтології, орієнтовані на пошук лише у Semantic Web (ONTOSEARCH2, Swoogle), тобто серед семантично анотованих інформаційних ресурсів [15-17].

ІПС QUICK здійснює пошук у всьому інформаційному просторі Web, працює як метапошукова ІПС: отримує запит користувача, семантично переробляє його та спрямовує до зовнішніх інформаційних систем (наприклад, до Google), а потім семантично обробляє отримані результати з використанням онтологій [15-17].

Враховуючи той факт, що програмне забезпечення (ПЗ) є основою усіх сучасних напрямків господарської діяльності, а досягнення високих значень показників його якості є ключовим фактором забезпечення ефективного застосування ПЗ та однією із основних вимог користувачів і зацікавлених осіб до сучасного ПЗ, *метою даного дослідження* є розроблення ефективної інформаційно-пошукової системи (на основі онтологій) для галузі якості програмного забезпечення.

Концепція інформаційно-пошукової системи (на основі онтологій) для галузі якості програмного забезпечення

Згідно зі стандартами ISO 25010 [18], ISO 25030 [19], SWEBOK [20], роглядатимемо якість програмного забезпечення як здатність програмного забезпечення задовольнити заявлені і передбачувані потреби при його використанні за певних умов. Концепція оцінювання якості ПЗ за стандартами ISO 25010 [18], ISO 25023 [21] представлена на рис. 1 [8].

Рис. 1. Сучасна концепція оцінювання якості програмного забезпечення в рамках моделі SQuaRE

Аналіз стандартів ISO 25010 [18], ISO 25023 [21] дав можливість зробити висновок, що є атрибути, від яких залежать більше однієї підхарактеристики та характеристики якості ПЗ, тобто має місце кореляція підхарактеристик та характеристик за певними атрибутами (так, згідно зі стандартом [21] підхарактеристики якості залежать від 203 атрибутів, але всього від 138 різних атрибутів).

Для розроблення інформаційно-пошукової системи (на основі онтологій) для галузі якості програмного забезпечення слід спочатку розробити онтологію предметної галузі якості програмного забезпечення, яка відображатиме семантичні відношення між концептами предметної галузі та ляже в основу пошуку інформації про якість ПЗ, зокрема, в основу тезаурусу майбутньої ІПС.

Концепція онтології предметної галузі якості ПЗ представлена на рис. 2 [8].

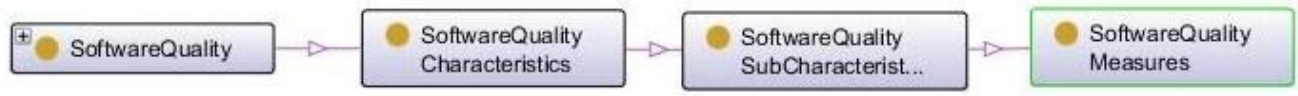


Рис. 2. Концепція онтології предметної галузі якості ПЗ

Онтологію предметної галузі якості ПЗ утворюють складові частини для: Функційної придатності (рис. 3), Сумісності (рис. 4), Ефективності, Можливості переносу, Зручності використання, Надійності, Захищеності, Супроводжуваності [8].

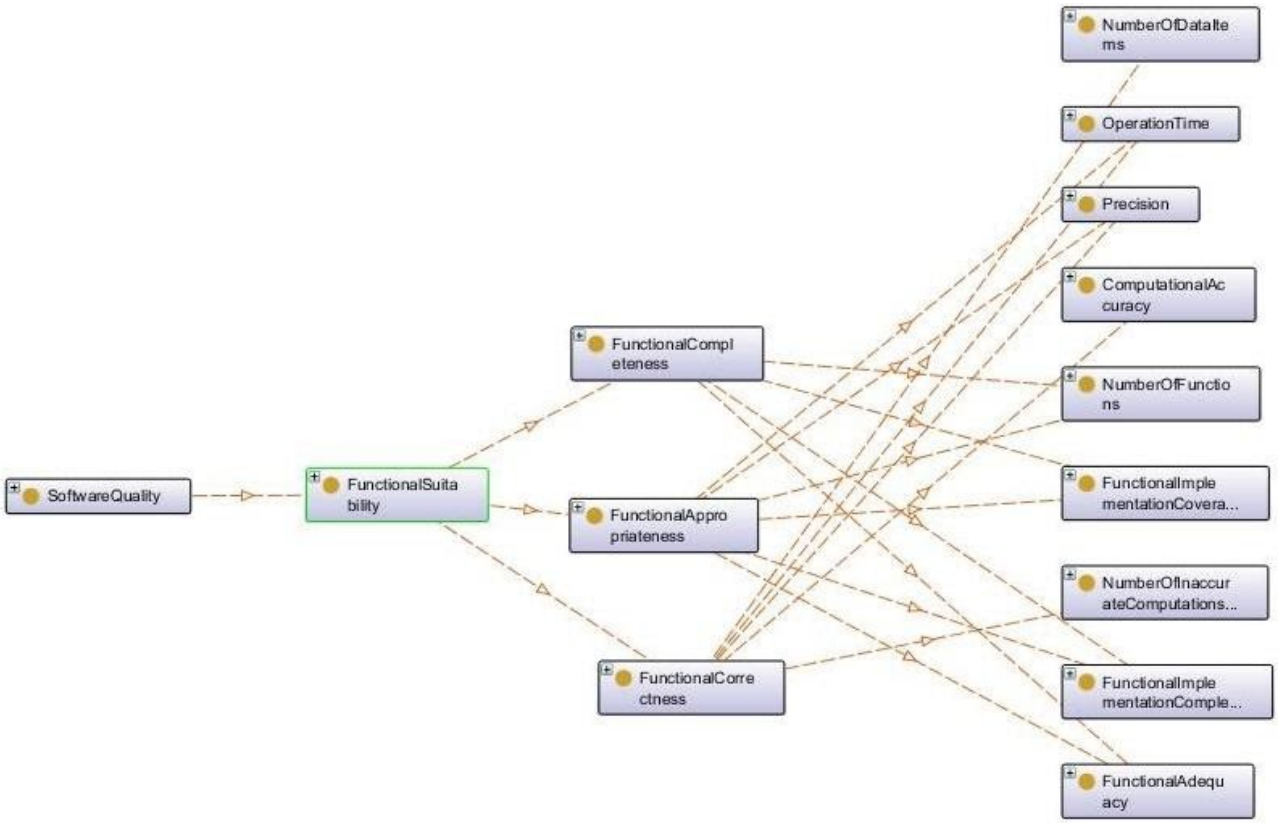


Рис. 3. Складова «Функційна придатність» онтології предметної галузі якості ПЗ

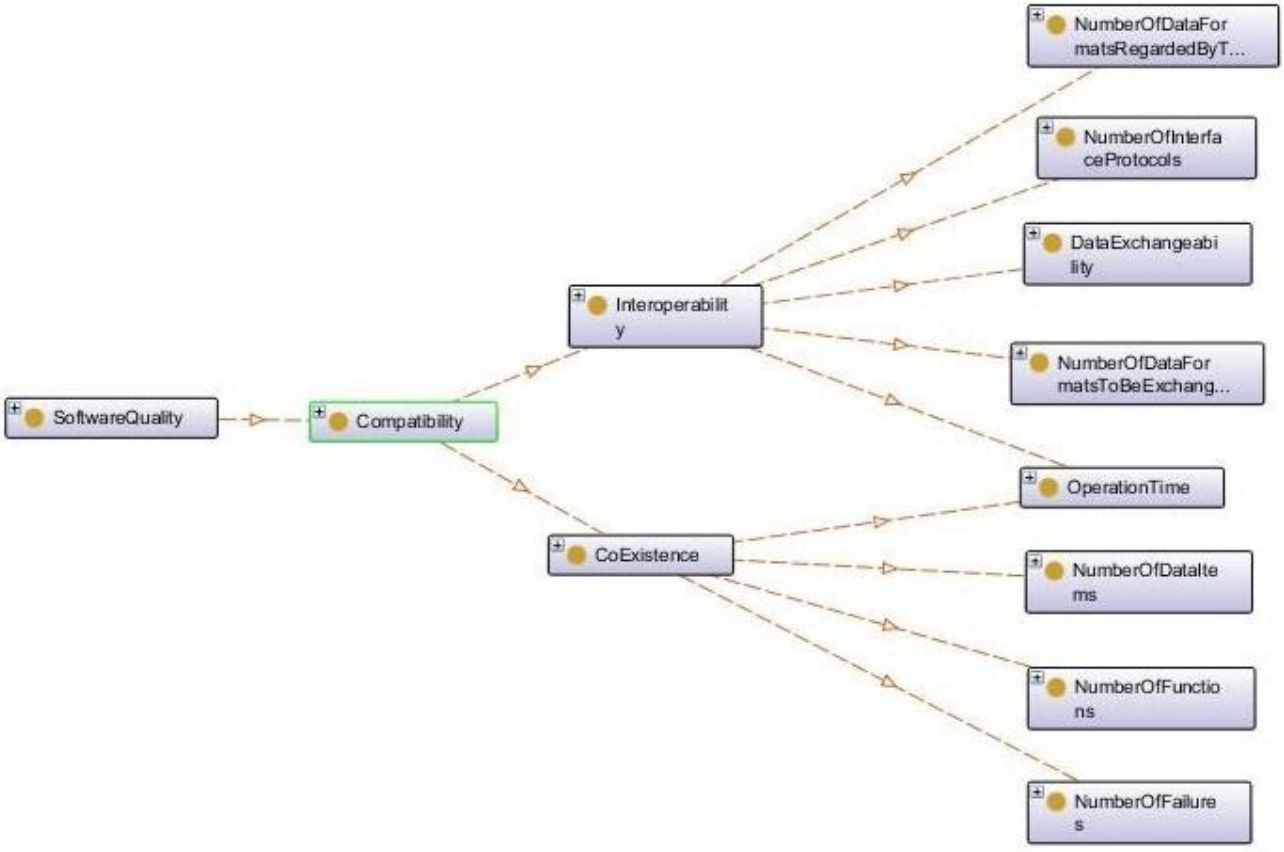


Рис. 4. Складова «Сумісність» онтології предметної галузі якості ПЗ

Висновки

На сьогодні реальні інформаційно-пошукові системи забезпечують коефіцієнт повноти на рівні 70%, а коефіцієнт точності пошуку – на рівні іноді навіть 10%. Отже, відомі ІПС наразі не здатні забезпечити сучасних потреб користувачів.

Загальносвітовим трендом щодо опрацювання великих масивів інформації, що дозволяє вирішувати нові класи задач на основі наявних інформаційних ресурсів, є інтелектуалізація опрацювання інформації і даних. Як стандарт інженерії знань при розробленні інформаційно-пошукових систем варто і доцільно використовувати онтології, які широко застосовуються в роботі пошукових машни та інформаційно-пошукових систем, оскільки онтології є ефективним інструментом для організації семантичного пошуку.

Враховуючи той факт, що програмне забезпечення (ПЗ) є основою усіх сучасних напрямків господарської діяльності, а досягнення високих значень показників його якості є ключовим фактором забезпечення ефективного застосування ПЗ та однією із основних вимог користувачів і зацікавлених осіб до сучасного ПЗ, метою даного дослідження є розроблення ефективної інформаційно-пошукової системи (на основі онтологій) для галузі якості програмного забезпечення.

В статті запропоновано концепцію інформаційно-пошукової системи (на основі онтологій) для галузі якості програмного забезпечення, зокрема, розроблено онтологію предметної галузі якості програмного забезпечення, яка відображає семантичні відношення між концептами предметної галузі та ляже в основу пошуку інформації про якість ПЗ, зокрема, в основу тезаурусу майбутньої ІПС.

Література

1. Інформаційно-пошукова система [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/socialnafnformatika/home/informacijno-posukovi-sistemi
2. Інформаційно-пошукові системи [Електронний ресурс]. – Режим доступу: http://esu.com.ua/search_articles.php?id=12483
3. Поняття і класифікація інформаційно-пошукових систем [Електронний ресурс]. – Режим доступу: https://library.if.ua/book/100/6867.html
4. Вовк Н. Архівні інформаційно-пошукові системи: шляхи оптимізації пошуку текстової інформації / Н. Вовк. – Бібліотекознавство. Документознавство. Інформологія. – 2018. – №3. – С. 37-42.
5. Пиріг С. Інформаційні технології та їх використання на підприємствах України / С.Пиріг, О.Нужна. – Економічний форум. – 2014. – № 3. – С. 190–195.
6. Винничук Р. Особливості розвитку ІТ-ринку в Україні: стан та тенденції / Р.Винничук, Т.Склярук. – Вісник Національного університету «Львівська політехніка». Серія «Логістика». – 2015. – № 833. – С. 3-8.
7. Минухин С. Формирование информационного обеспечения системы управления бизнес-процессами предприятия / С. Минухин. – Актуальні проблеми економіки. – 2006. – № 10. – С. 170–178.
8. Говорущенко Т.О. Теоретичні та прикладні засади інформаційної технології оцінювання достатності інформації щодо якості у специфікаціях вимог до програмного забезпечення: дис. ... доктора техн. наук: 05.13.06. – Львів, 2018. – 441 с.
9. Sivarajah U. Critical analysis of Big Data challenges and analytical methods / U.Sivarajah, M.Kamal, Z.Irani, V.Weerakkody. – Journal of Business Research. – 2017. – Vol. 70. – Pp. 263-286.
10. Luo Y. How valuable is information and communication technology? A study of emerging economy enterprises / Y.Luo, J.Bu. – Journal of World Business. – 2016. – Vol. 51. – Issue 2. – Pp. 200-211.
11. Tian X. Big data and knowledge management: a case of deja vu or back to the future? / X. Tian. – Journal of Knowledge Managament. – 2017. – Vol. 21. – Issue 1. – Pp. 113-131.
12. Mauerhoefer T. The impact of information technology on new product development performance / T.Mauerhoefer, S.Strese, M.Brettel. – Journal of Product Innovation Management. – 2017. – Vol. 34. – Issue 6. – Pp. 719-738.
13. Press G. Top 10 hot artificial intelligence (AI) technologies [Electronic resource] / G. Press. – Access mode: https://www.forbes.com/sites/gilpress/2017/01/23/top-10-hot-artificial-intelligence-ai-technologies/#1d5da9561928
14. Литвин В.В. Методи та засоби опрацювання інформаційних ресурсів на основі онтологій / В.В. Литвин, В.А. Висоцька, Д.Г. Досин. – Львів: ЛА «Піраміда», 2016. – 404 с.
15. Рогушина Ю. Теоретичні засади застосування онтологій для семантизації ресурсів Web [Електронний ресурс] / Ю. Рогушина. – Режим доступу: http://irtc.org.ua/image/app/webroot/Files/presentations/IPS/Rogushinadocl_seminar_29_05_2018_dd_20.pdf
16. Рогушина Ю. Використання онтологій для персоніфікації семантичного пошуку / Ю. Рогушина. – Проблеми програмування. – 2017. – №3. – С. 52- 67.
17. Рогушина Ю. Семантичний пошук у Web на основі онтологій: розробка моделей, засобів і методів / Ю. Рогушина. – Мелітополь: МДУ-ПУ ім. Б. Хмельницького, 2015. – 291 с.

18. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. [Introduced 01.03.2011]. – Geneva (Switzerland), 2011. – 34 p. (International standard).
19. ISO/IEC 25030:2019. Systems and software engineering. Systems and software quality requirements and evaluation (SQuaRE). Quality requirements framework [Introduced 01.09.2019]. – Geneva (Switzerland), 2019. – 46 p. (International standard).
20. ISO/IEC TR 19759:2015. Software Engineering. Guide to the software engineering body of knowledge (SWEBOK). [Introduced 01.10.2015]. – Geneva (Switzerland), 2015. – 336 p. (International standard).
21. ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality. [Introduced 31.03.2016]. Geneva (Switzerland), 2016. 45 p. (International standard).

References

1. Informaciyno-poshukova systema. Web-site. URL: https://sites.google.com/site/socialnafnformatika/home/informacijno-posukovi-sistemi (Last accessed: July 1, 2020).
2. Informaciyno-poshukovi systemy. Web-site. URL: http://esu.com.ua/search_articles.php?id=12483 (Last accessed: July 1, 2020).
3. Ponyattya i klasyfikaciya informaciyno-poshukovykh system. Web-site. URL: https://library.if.ua/book/100/6867.html (Last accessed: July 1, 2020).
4. Vovk N. Arkhivni informaciyno-poshukovi systemy: shlyakhy optymizatsiyi poshuku tekstovoyi informaciyi. – Bibliotekoznavstvo. Dokumentoznavstvo. Informologiya. – 2018. – No. 3. – S. 37-42.
5. Pyrig S., Nuzhna O. Informaciyni tekhnologiyi ta ih vykorystannya na pidpryemstvakh Ukrainy. – Ekonomichnyi froum. – 2014. – No. 3. – S. 190–195.
6. Vynnychuk R., Sklyaruk T. Osoblyvosti rozvytku IT-rynku v Ukrayini: stan ta tendenciyi. – Visnyk Natsionalnogo universytetu “Lvivska politekhnyka”. Seriya “Logistyka”. – 2015. – No. 833. – S. 3-8.
7. Minukhin S. Formirovaniye informatsionnogo obespecheniya sistemy upravleniya biznes-protsessami predpriyatiya. – Aktualni problem ekonomiky. – 2006. – No. 10. – S. 170–178.
8. Hovorushchenko T.O. Teoretychni ta prykladni zasady informaciynoyi tekhnologiyi otsinuvannya doststnosti informaciyi schodo yakosti u specifikatsiyakh vymog do programnogo zabezpechennya: dis. … doktora tekhnichnykh nauk: 05.13.06. – Lviv, 2018. – 441 s.
9. Sivarajah U., Kamal MM., Irani Z., Weerakkody V. Critical analysis of Big Data challenges and analytical methods. – Journal of Business Research. – 2017. – Vol. 70. – Pp. 263-286.
10. Luo Y., Bu J. How valuable is information and communication technology? A study of emerging economy enterprises. – Journal of World Business. – 2016. – Vol. 51. – Issue 2. – Pp. 200-211.
11. Tian X. Big data and knowledge management: a case of deja vu or back to the future? – Journal of Knowledge Managament. – 2017. – Vol. 21. – Issue 1. – Pp. 113-131.
12. Mauerhoefer T., Strese S., Brettel M. The impact of information technology on new product development performance. – Journal of Product Innovation Management. – 2017. – Vol. 34. – Issue 6. – Pp. 719-738.
13. Press G. Top 10 hot artificial intelligence (AI) technologies. Web-site. URL: https://www.forbes.com/sites/gilpress/2017/01/23/top-10-hot-artificial-intelligence-ai-technologies/#1d5da9561928 (Last accessed: July 1, 2020).
14. Lytvyn V. V., Vysotska V. A., Dosyn D. G. Metody ta zasoby opratsuvannya informaciynykh resursiv na osnovi ontologiy. – Lviv: LA «Piramida», 2016. – 404 s.
15. Rogushyna Y. Teoretychni zasady zastosuvannya ontologiy dlya semantizatsiyi resursiv Web Теоретичні засади застосування онтологій для семантизації ресурсів Web. Web-site. URL: http://irtc.org.ua/image/app/webroot/Files/presentations/IPS/Rogushinadocl_seminar_29_05_2018_dd_20.pdf (Last accessed: July 1, 2020).
16. Rogushyna Y. Vykorystannya ontologiy dlya personifikatsiyi semantychnogo poshuku. – Problemy programuvannya. – 2017. – No. 3. – S. 52- 67.
17. Rogushyna Y. Semantychnyi poshuk u Web na osnovi ontologiy: rozrobka modeley, metodiv i zasobiv. – Melitopol: MDU-PU im. B. Khmelnytskogo, 2015. – 291 s.
18. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. [Introduced 01.03.2011]. – Geneva (Switzerland), 2011. – 34 p. (International standard).
19. ISO/IEC 25030:2019. Systems and software engineering. Systems and software quality requirements and evaluation (SQuaRE). Quality requirements framework [Introduced 01.09.2019]. – Geneva (Switzerland), 2019. – 46 p. (International standard).
20. ISO/IEC TR 19759:2015. Software Engineering. Guide to the software engineering body of knowledge (SWEBOK). [Introduced 01.10.2015]. – Geneva (Switzerland), 2015. – 336 p. (International standard).
21. ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality. [Introduced 31.03.2016]. – Geneva (Switzerland), 2016. – 45 p. (International standard).

Рецензія / Peer review: 01.07.2020
Надрукована / Printed:
Рецензент: д.т.н., проф., декан ФПКТС ХНУ, О.С.Савенко

ДОДАТОК Б

(обов’язковий)

ПРЕЗЕНТАЦІЯ ДОПОВІДІ

Презентація до дипломної роботи на тему:
«Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення»

ВИКОНАВ: МАРТИНЮК Ю.С.

НАУКОВИЙ КЕРІВНИК: ГОВОРУЩЕНКО Т.О.

ХМЕЛЬНИЦЬКИЙ 2020

Об’єктом дослідження є процеси опрацювання інформації щодо якості програмного забезпечення, технологія виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

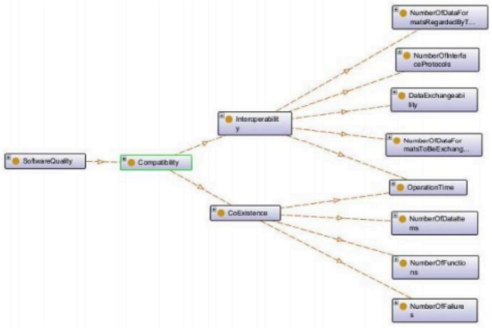
Предметом дослідження є моделі, метод та інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення.

Метою дипломної роботи є підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення шляхом розроблення інтелектуалізованої системи на основі онтологій.

План

1. Аналіз відомих методів та рішень
2. Моделювання процесу пошуку інформації для галузі якості програмного забезпечення на основі онтологій
3. Метод пошуку інформації для галузі якості програмного забезпечення на основі онтологій
4. Інтелектуалізована система на основі онтологій для якості програмного забезпечення

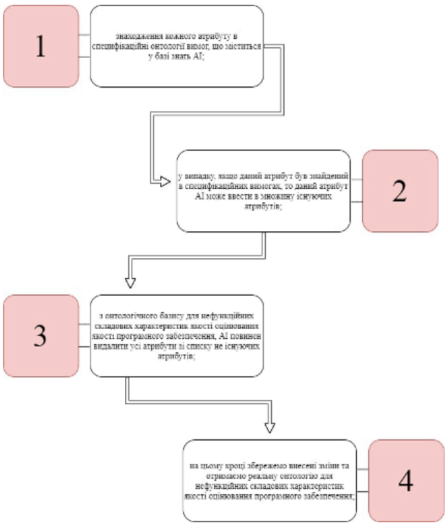
Концепція онтології предметної галузі якості ПЗ



Складова «Сумісність» онтології предметної галузі якості ПЗ

Онтологічний аналіз специфікацій

Виконані схематично-розроблені AI, надають можливість проводити аналіз специфікацій з пошуком та встановленням відсутності чи існування атрибутів з показниками, що є необхідними для визначення складових характеристик якості оцінювання програмного забезпечення. Дані результати можуть пізніше використовуватися для визначення нефункційних складових характеристик якості. Для того щоб визначити достатність нам необхідна лиш інформація про відсутність або наявність атрибутів в специфікацій, тому правила аналізу для специфікацій на базі яких працює AI, є зрозумілими та простими. Ці умови надають низьку “ціну” та високу швидкість для аналізу специфікацій.



Метод діяльності штучного інтелекту на базі онтологічного підходу для оцінювання якості програмного забезпечення на наявність достатності інформації у специфікації для виявлення нефункційних складових характеристик складається з наступних етапів:

1) порівняння онтологій нефункційних складових характеристик якості оцінювання програмного забезпечення (реального) з базовими онтологіями нефункційних складових характеристик якості оцінювання програмного забезпечення з ціллю виявити атрибути, що відсутні у специфікаційних вимогах до програмного забезпечення (реального), за якими були побудовані реальні онтології;

2) знаходження нефункційних складових характеристик та підхарактеристик якості програмного забезпечення, що неможливо порахувати на базі існуючих в специфікаційних вимогах до атрибутів реального програмного забезпечення;

3) створення висновку про недостатність чи достатність знань у специфікаційних вимогах для виявлення кожної нефункціної характеристики програмного забезпечення разом і окремо;

4) розрахунок оцінок (числових) про рівень достатності, що наявний у специфікаційних вимогах інфорації щодо визначення кожної характеристики програмного забезпечення нефункційного типу за формулою:

$$Aj = \frac{(hj - \Sigma \frac{zti}{zri})}{h_j}$$

Взаємозв'язок характеристик та їхня взаємодія одна з одною

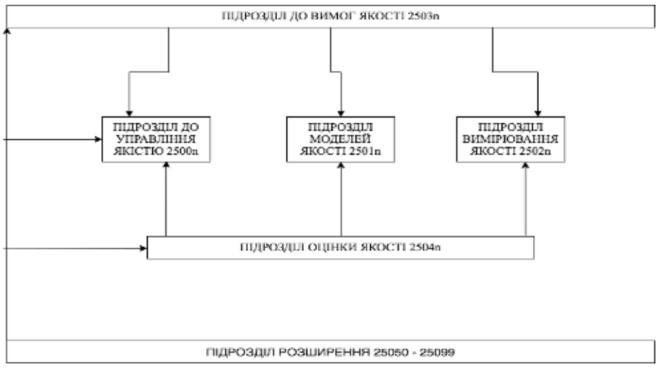
Умовні позначеня:

1) Живучість – l.
2) Правильність – p.
3) Адаптація – a.
4) Цілісність – t.
5) Надійність – n.
6) Ефективність – e.
7) Практичність – z.
8) Коректність – c.

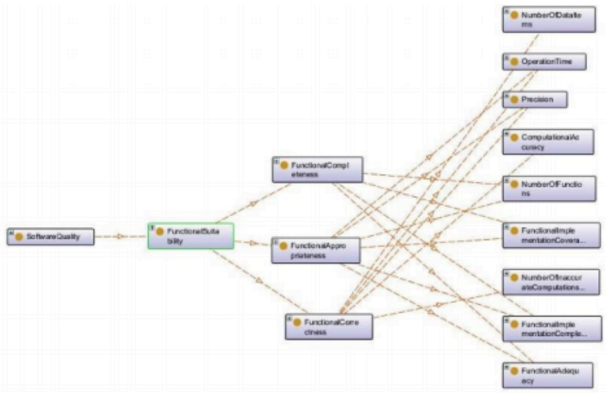
Вплив	l	p	a	t	n	e	z	c
l	+	-	+	-	-	-	+	-
p	-	+	-		+	-		+
a	+		+	-				
t				+	+	-		
n	-	+		+	+			+
e		-	-	-	-	+		-
z		+	+				+	
c	-	+			+	+		

Сучасна концепція оцінювання якості програмного забезпечення в рамках моделі SquaRE

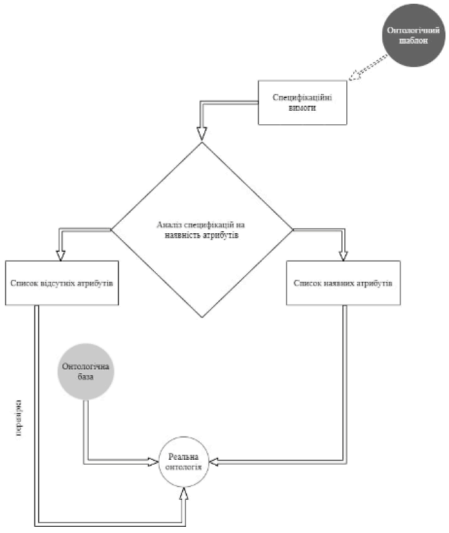
Організація та взаємозв'язок серії SquaRE



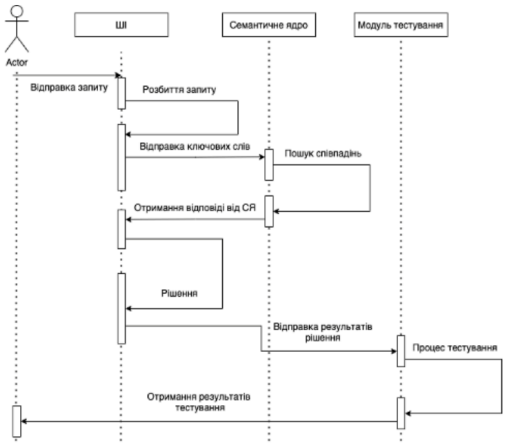
Концепція онтології предметної галузі якості ПЗ



Структурна схема AI на базисі онтологічного підходу



Діаграма послідовностей роботи AI



Отримана наукова новизна

– набули подальшого розвитку моделі процесу опрацювання інформації для галузі якості програмного забезпечення, які ґрунтуються на онтологіях предметної галузі якості програмного забезпечення та є теоретичним підґрунтям методу пошуку інформації для галузі якості програмного забезпечення на основі онтологій;

– набув подальшого розвитку метод опрацювання інформації щодо якості програмного забезпечення на основі онтологій, який працює на основі розроблених онтологічних моделей та забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

Отримана практична цінність

Практична значущість отриманих результатів полягає у розробленні інтелектуалізованої системи на основі онтологій для галузі якості програмного забезпечення, яка забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації щодо якості програмного забезпечення.

Висновки

У першому розділі проведений аналіз відомих методів і рішень для опрацювання інформації щодо якості програмного забезпечення, технології виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

У другому розділі був змодельований процес опрацювання інформації для галузі якості програмного забезпечення на основі онтологій, а також набули подальшого розвитку моделі процесу опрацювання інформації для галузі якості програмного забезпечення, які ґрунтуються на онтологіях предметної галузі якості програмного забезпечення та є теоретичним підґрунтям методу пошуку інформації для галузі якості програмного забезпечення на основі онтологій.

У третьому розділі був розроблений метод опрацювання інформації щодо якості програмного забезпечення на основі онтологій, який працює на основі розроблених онтологічних моделей та забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

У четвертому розділі спроєктовано та реалізовано інтелектуалізовану систему на основі онтологій для галузі якості програмного забезпечення, яка забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації щодо якості програмного забезпечення.

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1008009368

Дата перевірки:
25.05.2021 14:59:07 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
25.05.2021 15:01:17 EEST

ID користувача:
100005591

Назва документа: Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечен...

Кількість сторінок: 74 Кількість слів: 14595 Кількість символів: 110632 Розмір файлу: 1.13 MB ID файлу: 100810036

5.06%
Схожість

Найбільша схожість: 2.52% з Інтернет-джерелом (http://elar.khnu.km.ua/jspui/bitstream/123456789/9851/1/%D0%94%D..

4.49% Джерела з Інтернету 55 ..... Сторінка 76

0.88% Джерела з Бібліотеки 68 ..... Сторінка 76

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 67

Tue May 25 16:33:00 EEST 2021, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 1.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 9%**

ID: 91253 Название: Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення Добавлено в БД: 2021-05-25 Авторы: Мартинюк Ю.С. Руководители: Говорущенко Т.О. Консультанты: Опоненты:	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
	104181	723	1554 (1%)	21 (3%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник: Мартинюк Юрій Сергійович

Тема: Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг дипломної роботи:

Кількість сторінок записки ___

1. Короткий зміст роботи та прийнятих рішень: Метою дипломної роботи є підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення шляхом розроблення інтелектуалізованої системи на основі онтологій

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведений аналіз відомих методів і рішень для опрацювання інформації щодо якості програмного забезпечення, технології виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення. У другому розділі був змодельований процес опрацювання інформації для галузі якості програмного забезпечення на основі онтологій, а також набули подальшого розвитку моделі процесу опрацювання інформації для галузі якості програмного забезпечення, які грунтуються на онтологіях предметної галузі якості програмного забезпечення та є теоретичним підгрунтям методу пошуку інформації для галузі якості програмного забезпечення на основі онтологій. У третьому розділі був розроблений метод опрацювання інформації щодо якості програмного забезпечення на основі онтологій, який працює на основі розроблених онтологічних моделей та забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення. У четвертому розділі

спроєктовано та реалізовано інтелектуалізовану систему на основі онтологій для галузі якості програмного забезпечення, яка забезпечує підвищення ефективності виконання інтелектуальних обчислень для підбору інформації щодо якості програмного забезпечення.

4. Позитивні сторони роботи: отримання двох пунктів наукової новизни.

5. Негативні сторони роботи: недостатня увага технології виконання інтелектуальних обчислень для підбору інформації в галузі якості програмного забезпечення.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Клюєв Юрій Павлович, к.т.н., доцент, завідувач кафедри КБКСН ХНУ

"25" 05 2021 р. _____ (підпис)

Завідувачу кафедри КІСП
д-р.техн.наук, проф. Говорущенко Т. О.

Мартинюк Т О С.
ПІБ здобувача вищої освіти

ФПКТС, 2 курсу, групи КІ2м-19-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

28.05.2021
дата

підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Інтелектуалізована система на основі онтологій для галузі якості програмного забезпечення

Автор: Мартинюк Юрій Сергійович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Говорущенко Т.О. – доктор техн.наук, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) Деякі методи та моделі були запозичені, оскільки були необхідні, як готові рішення в дослідженні;
2) усі запозичення фрагментарні, або мають належним чином оформлені посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-40 джерелами на один фрагмент речення;
4) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості, складає 5.06% і адресується до 123 першоджерел, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Т. О. Говорущенко

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КІСП _____ Т. О. Говорущенко