Хмельницький національний університет
Міністерство освіти і науки України

Хмельницький національний університет
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

ПАВЛОВА ОЛЬГА ОЛЕКСАНДРІВНА

УДК 004.9:004.05

Д И С Е Р Т А Ц І Я

АГЕНТНО-ОРІЄНТОВАНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОЦІНЮВАННЯ ПОЧАТКОВИХ ЕТАПІВ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ

122 Комп’ютерні науки
(шифр і назва спеціальності)

12 Інформаційні технології
(галузь знань)

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних проваджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Павлова Ольга Олександрівна
підпис

Науковий керівник Говорущенко Тетяна Олександрівна,
доктор технічних наук, професор

Хмельницький – 2020

АНОТАЦІЯ

Павлова Ольга Олександрівна. Агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки. – Хмельницький національний університет, Хмельницький, 2020.

У дисертаційній роботі розв'язана актуальна науково-прикладна задача автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення шляхом розроблення агентно-орієнтованої інформаційної технології (АОІТ) на основі онтологічного підходу, яка забезпечує: автоматизацію трудомісткого, рутинного та схильного до помилок завдання розбору специфікацій вимог та майже миттєве його виконання; підказку, де потрібна повторна робота над специфікацією вимог (користувач може переглядати відсутні атрибути та бачити області специфікації, яким потрібна додаткова увага, а також які вимоги потребують переробки); забезпечення навчання для нових розробників специфікацій, системних інженерів та менеджерів проєктів (використання цієї АОІТ допомагає їм побачити помилки, які вони можуть допускати, та допомагає їм розпізнати ці помилки в роботі інших); допомогу в розробці вимог високої якості; допомогу у виправленні та усуненні помилок у вимогах там, де вони виникають – на ранніх етапах життєвого циклу програмного забезпечення – до того, як вони стануть дорожчими для виправлення; надання інструменту для вибору більш якісних специфікацій програмних вимог; безкоштовний доступ через Інтернет в будь-який час без будь-якої реєстрації.

Об'єктом дослідження є процеси оцінювання поачткових етапів життєвого циклу програмного забезпечення (ПЗ).

Предметом дослідження є моделі, методи та засоби агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення.

У роботі виконано аналіз відомих підходів, методів, інформаційних технологій, інструментів для аналізу вимог до програмного забезпечення та оцінки достатності інформації у специфікації вимог та інтелектуальних агентів на основі онтологічного підходу, показав, що вони не розв'язують проблему оцінювання ранніх етапів життєвого циклу ПЗ. Крім цього, всі вони належать до різних методологічних підходів і не інтегруються між собою, тобто наразі відсутні інтелектуальні інформаційно-аналітичні технології для оцінювання ранніх етапів життєвого циклу ПЗ, що є однією з причин проблем у галузі інженерії ПЗ. Актуальність проблеми аналізу достатності інформації щодо якості у специфікаціях вимог до ПЗ, а також відсутність моделей, методів та засобів оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ обумовлює необхідність розроблення інтелектуальних інформаційно-аналітичних технологій для підвищення якості програмного забезпечення шляхом оцінювання достатності інформації на ранніх етапах життєвого циклу.

У дисертаційній роботі запропоновані базові (універсальні) онтологічні моделі нефункційних характеристик-складових якості ПЗ, а також онтологічні моделі нефункційних характеристик-складових якості конкретного ПЗ, які ґрунтуються на врахуванні вимог стандартів ISO 25010:2011 та ISO 25023:2016 і забезпечують підґрунтя для вибору достатнього обсягу інформації для оцінювання нефункційних характеристик ПЗ. Також розроблено модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ, яка ґрунтується на порівняльному аналізі онтологій та є теоретичним підґрунтям для розроблення методів діяльності інтелектуального агента на основі онтологічного підходу.

В роботі вперше розроблено метод діяльності інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення, який працює на основі розробленої моделі та здійснює

оцінювання достатності інформації у специфікації вимог для визначення всіх нефункційних характеристик-складових якості ПЗ, забезпечує висновок про достатність або недостатність інформації у специфікації, надає числові оцінки рівня достатності інформації для визначення кожної нефункційної характеристики ПЗ та для визначення всіх нефункційних характеристик-складових якості ПЗ разом, формує список атрибутів, якими варто доповнити специфікацію вимог для підвищення рівня достатності її інформації, тобто в комплексі дозволяє частково усунути людину з процесів опрацювання інформації та здобуття знань.

Реалізовано інтелектуальні агенти на основі онтологічного підходу для оцінювання ранніх етапів життєвого циклу ПЗ, які здійснюють оцінювання достатності інформації у специфікації вимог для визначення всіх нефункційних характеристик-складових якості ПЗ, а також для розрахунку метрик якості та складності ПЗ. Реалізовані інтелектуальні агенти забезпечують висновок про достатність або недостатність інформації у специфікації. Крім цього, вони надають числові оцінки рівня достатності інформації для визначення кожної нефункційної характеристики ПЗ та/або метрики та для визначення всіх нефункційних характеристик-складових якості ПЗ та/або метрик разом. Агентами також формується список атрибутів та показників, якими варто доповнити специфікацію вимог для підвищення рівня достатності її інформації, та візуалізація прогалин у знаннях про всі нефункційні характеристики-складові якості ПЗ та метрики якості і складності ПЗ. Отже, реалізовані інтелектуальні агенти забезпечують автоматизацію аналізу специфікацій вимог до ПЗ на предмет достатності їх інформації. Таким чином, представлені агенти дозволяють частково усунути людину з процесів опрацювання інформації та здобуття знань.

У дисертації удосконалено метод діяльності інтелектуального агента для автоматизованого семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення, який, на відміну від відомих, грунтується на врахуванні вимог стандарту ISO 25010 і на обраній номенклатурі метрик виконує парсинг специфікації, визначає кількість та відсоток відсутніх атрибутів,

відображає, яких атрибутів не вистачає для тієї чи іншої підхарактеристики нефункційної характеристики, а також формує реальну онтологію для нефункційних характеристик, яка може бути використана інтелектуальним агентом на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення

Розроблено інтелектуальні агенти для семантичного парсингу природомовних специфікацій, які виконують парсинг специфікації, визначають кількість та відсоток відсутніх атрибутів та/або показників, відображають, яких атрибутів та/або показників не вистачає для тієї чи іншої підхарактеристики нефункційної характеристики та/або метрики, а також формують реальну онтологію для нефункційних характеристик та/або метрик, яка може бути використана інтелектуальним агентом на основі онтологічного підходу для оцінювання ранніх етапів життєвого циклу ПЗ шляхом оцінювання достатності інформації для визначення нефункційних характеристик та метрик.

У дисертаційній роботі набула подальшого розвитку агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходув частині автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення, яка, на відміну від відомих, виконує оцінювання та забезпечує підвищення рівня достатності інформації вимог для визначення кожної нефункційної характеристики окремо та всіх нефункційних характеристик разом. Дана АОІТ оцінює та забезпечує приріст рівня достатності інформації у специфікації вимог для визначення нефункційних характеристик програмного забезпечення – приріст рівня достатності становить від 4,71% до 27,79% (наприклад, з 58,23% до 86,02% для специфікації вимог №1 до програмного агента для підвищення безпеки програмного забезпечення комп’ютерних систем, від 81,26% до 85,97% для специфікації №2, з 60,85% до 73,7% для специфікації №3). Перевагами розробленої АОІТ є: автоматизація трудомісткого, рутинного та схильного до помилок завдання розбору специфікацій вимог та майже миттєве його виконання; підказка, де потрібна повторна робота над специфікацією вимог (користувач може переглядати

відсутні атрибути та бачити області специфікації, яким потрібна додаткова увага, а також які вимоги потребують переробки); забезпечення навчання для нових розробників специфікацій, системних інженерів та менеджерів проєктів (використання цієї АОІТ допомагає їм побачити помилки, які вони можуть допускати, та допомагає їм розпізнати ці помилки в роботі інших); допомога у розробці вимог високої якості; допомога у виправленні та усуненні помилок у вимогах там, де вони виникають – на ранніх етапах життєвого циклу програмного забезпечення – до того, як вони стануть дорожчими для виправлення; надання інструменту для вибору більш якісних специфікацій програмних вимог; безкоштовний доступ через Інтернет в будь-який час без будь-якої реєстрації. Економічним ефектом від використання розробленої АОІТ є можливість економії бюджету програмних проєктів на обробку та виправлення (протягом життєвого циклу) дефектів та помилок, які утворюються на ранніх етапах життєвого циклу – завдяки демонстрації слабких сторін специфікацій вимог до ПЗ, які потрібно доопрацювати або переробити в той момент, коли вони виникають.

Практичне значення отриманих результатів полягає у розробленні агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу, яка: автоматизує трудомістку та схильну до помилок задачу розбору специфікації вимог; вказує на необхідність доопрацювання специфікації із зазначенням вимог, які потребують доопрацювання; забезпечує швидке навчання нових системних інженерів та керівників проєктів (використання розробленої інформаційної технології під час створення або аналізу вимог допомагає їм швидко бачити помилки, яких вони можуть припуститись, і допомагає розпізнавати ці помилки в роботі інших); допомагає виправити та усунути помилки та неточності у вимогах на ранніх етапах життєвого циклу, є інструментом для вибору більш якісної специфікації, доступна онлайн в будь-який час без реєстрації.

Результати дисертаційної роботи впроваджено у: ТОВ «ІТТ» (акт впровадження від 10.03.2020 р.), ТОВ «Деймос» (акт впровадження від 20.02.2020 р.), ГО «ІТ Кластер м. Хмельницького» (акт впровадження від 03.06.2020 р.), при

виконанні держбюджетного проєкту кафедри комп'ютерної інженерії та системного програмування "Агентно-орієнтована система підвищення безпеки та якості програмного забезпечення комп'ютерних систем" (ДР №0119U100662); у навчальному процесі Хмельницького національного університету (акт впровадження від 13.05.2020 р.) та ПВНЗ «Міжнародного науково-технічного університету імені академіка Юрія Бугая» (акт впровадження від 08.10.2020 р.).

Ключові слова: агентно-орієнтована інформаційна технологія, інтелектуальний агент на основі онтологічного підходу, нефункційні характеристики-складові якості програмного забезпечення, онтологія, програмне забезпечення.

Список публікацій здобувача за темою дисертації

Основні наукові публікації у періодичних виданнях, що індексуються в наукометричних базах SCOPUS, Web of Science:

1. Hovorushchenko T., Pavlova O., Bodnar M. Development of an Intelligent Agent for Analysis of Nonfunctional Characteristics in Specifications of Software Requirements. *Eastern-European Journal of Enterprise Technologies*. 2019. Vol. 1. No. 2 (97). Pp. 6-17. *(Scopus, Q2)*

2. Hovorushchenko T., Boyarchuk A., Pavlova O. Ontology-Based Intelligent Agent for Semantic Parsing the Software Requirements Specifications. *International Journal on Information Technologies and Security*. 2019. No. 2. Vol. 11. Pp.59-70. *(WoS, Bulgaria)*

Публікації у періодичних виданнях, що індексуються в наукометричних базах SCOPUS, Web of Science:

3. Hovorushchenko T., Pavlova O., Fedula M. Improving the input information for medical software requirements specifications using ontology-based intelligent agent. *CEUR-WS*. 2018. Vol. 2255. Pp.113-125. *(Scopus, WoS)*

4. Hovorushchenko T., Pavlova O. Method of Activity of Ontology-Based Intelligent Agent for Evaluating the Initial Stages of the Software Lifecycle. *Advances in Intelligent Systems and Computing*. 2019. Vol. 836. Pp. 169-178. *(Scopus)*

5. Hovorushchenko T., Pavlova O. Intelligent System for Determining the Sufficiency of Metric Information in the Software Requirements Specifications. *CEUR-WS*. 2019. Vol. 2353. Pp.253-266. *(Scopus)*

6. Hovorushchenko T., Boyarchuk A., Pavlova O., Bobrovnikova K. Agent-Oriented Information Technology for Assessing the Initial Stages of the Software Life Cycle. *CEUR-WS*. 2019. Vol. 2393. Pp.617-632. *(Scopus)*

7. Hovorushchenko T., Pavlova O., Boyarchuk A. Modelling of non-functional characteristics of the software for selection of accurate scope of information for their evaluation. *CEUR-WS*. 2019. Vol. 2533. Pp. 206-216. *(Scopus, WoS)*

8. Hovorushchenko T., Pavlova O., Medzatyi D. Ontology-Based Intelligent Agent for Determination of Sufficiency of Metric Information in the Software Requirements. *Advances in Intelligent Systems and Computing*. 2020. Vol. 1020. Pp. 447-460. *(Scopus)*

9. Boyarchuk A., Pavlova O., Bodnar M., Lopatto I. Approach to the Analysis of Software Requirements Specification on Its Structure Correctness. *CEUR-WS*. 2020. Vol. 2623. Pp. 85-95. *(Scopus)*

Статті у фахових наукових виданнях України:

10. Говорущенко Т.О., Іванов О.В., Павлова О.О. Метод оцінювання достатності інформації для визначення якості програмного забезпечення на основі зваженої онтології. *Вісник Хмельницького національного університету. Серія «Технічні науки»*. Хмельницький: ХНУ. 2016. №5. С.146-156.

11. Говорущенко Т. О., Павлова О. О. Сучасні проблеми оцінювання початкових етапів життєвого циклу програмного забезпечення. *Електротехнічні та комп'ютерні системи*. 2018. №27 (103). С. 165-175.

12. Говорущенко Т. О., Поморова О. В., Павлова О. О. Моделювання діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до програмного забезпечення. *Вісник Хмельницького національного університету. Серія «Технічні науки»*. 2018. № 4. С. 128-137.

13. Павлова О. О., Говорущенко Т. О., Іванов О. В. Діяльність інтелектуального агента для оцінювання інформації у специфікаціях вимог до програмного забезпечення. *Штучний інтелект*. 2018. №2. С. 66-75.

14. Говорущенко Т. О., Павлова О.О., Боднар М. А. Сучасні проблеми семантичного аналізу специфікацій вимог до програмного забезпечення. *Вчені записки Таврійського національного університету ім. В. І. Вернадського. Серія «Технічні науки»*. 2019. Том 30 (69). №1. Частина 1. С. 38-43.

15. Говорущенко Т.О., Павлова О.О., Тоненька М.М. Структура агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу. *Вісник Хмельницького національного університету. Серія «Технічні науки»*. 2020. №1. С. 77-81.

16. Павлова О.О., Боднар М.А., Гнатчук Є.Г. Метод діяльності та реалізація інтелектуального агента на основі онтологічного підходу для парсингу природомовних специфікацій вимог до програмного забезпечення. *Вісник Хмельницького національного університету. Серія «Технічні науки».* 2020. №2. С.171-175.

17. Павлова О.О., Лопатто І.Ю., Говорущенко Т.О. Метод діяльності та структура інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2020. №3. С. 61-64.

Статті в матеріалах конференцій, що індексуються в наукометричних базах SCOPUS, Web of Science:

18. Hovorushchenko T., Pavlova O. Evaluating the Software Requirements Specifications Using Ontology-Based Intelligent Agent. *Proceedings of 2018 IEEE International Scientific and Technical Conference "Computer Science and Information Technologies"* (Lviv, Ukraine, September 11-14, 2018). Vol.1. Pp.215-218. *(Scopus, WoS)*

19. Pavlova O., Hovorushchenko T., Boyarchuk A. Method of activity of intelligent agent for semantic analysis of software requirements. *Proceedings of the 2019 IEEE 10-th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (Mez, France, September 18-21, 2019). Vol.2. Pp. 902-906. *(Scopus, WoS)*

20. Hovorushchenko T., Lopatto I., Pavlova O. Concept of Intelligent Agent for Verification of Considering the Subject Area Information. *Proceedings of 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies* (Kyiv, Ukraine, May 14-16, 2020). Pp. 465-469. *(Scopus)*

Публікації у матеріалах конференцій (тези доповідей):

21. Говорущенко Т.О., Павлова О.О. Аналіз сучасного стану інформаційних технологій для галузі інженерії програмного забезпечення. *Матеріали Всеукраїнської науково-практичної конференції «Сучасні тенденції розвитку додрукарських систем»* (Львів, Україна, 19 квітня 2018 р.). С. 32-33.

22. Говорущенко Т. О., Павлова О. О., Медзатий Д. М. Інтелектуальний агент на основі онтологічного підходу для визначення достатності метричної інформації у вимогах до програмного забезпечення. *Матеріали Міжнародної наукової конференції "Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту"* (Херсон, Україна, травень 2019 р.). С. 38-40.

23. Павлова О.О. Інтелектуальна система для визначення достатності метричної інформації у вимогах до програмного забезпечення. Збірник наукових праць молодих науковців і студентів "Інтелектуальний потенціал-2019" (Хмельницький, Україна, листопад 2019 р.). С. 59-62.

Свідоцтва про реєстрацію авторського права на твір

24. А. с. 80645 Україна. Метод діяльності інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення / Т. О. Говорущенко, О. О. Павлова. 2018.

25. А. с. 89841 Україна. Інтелектуальна система для визначення достатності метричної інформації у специфікаціях вимог до програмного забезпечення / Т. О. Говорущенко, О. О. Павлова. 2019.

26. А. с. 89840 Україна. Метод діяльності інтелектуального агента на основі онтологічного підходу для семантичного парсингу природомовних специфікацій вимог до програмного забезпечення / Т. О. Говорущенко, О. О. Павлова. 2019.

27. А. с. 97014 Україна. Інтелектуальна інформаційно-аналітична технологія для підвищення якості програмного забезпечення шляхом оцінювання достатності інформації на ранніх етапах життєвого циклу / Т. О. Говорущенко, О. О. Павлова. 2020.

28. А. с. 97015 Україна. Комп'ютерна програма «Веб-орієнтована інформаційно-аналітична система оцінювання достатності інформації у специфікаціях вимог до програмного забезпечення» / О. О. Павлова, Т. О. Говорущенко. 2020.

ANNOTATION

Pavlova Olga. Agent-oriented information technology for assessing the initial stages of the software life cycle based on the ontological approach. – Manuscript copyright. Thesis on competition of scientific degree of Doctor of Philosophy by specialty 122 – Computer Science. – Khmelnytskyi National University, Khmelnytskyi, 2020.

The dissertation solves the current scientific and applied problem of automating the assessment of the initial stages of the software life cycle by developing agent-oriented information technology based on an ontological approach, which provides: automation of time-consuming, routine and error-prone task analysis requirements and its almost instantaneous implementation; a hint where the requirements specification needs rework (the user can view the missing attributes and see the areas of the specification that need extra attention, as well as which requirements need to be reworked); providing training for new specification developers, systems engineers, and project managers (using this AOIT helps them to see the mistakes they may make and helps them to recognize these mistakes in the work of others); assistance in developing high quality requirements; assistance in correcting requirements errors where they occur – in the early stages of the software life cycle - before they become more expensive to

correct; providing a tool for selecting better specifications of software requirements; free internet access at any time without any registration.

The object of research is the process of evaluating the initial stages of the software life cycle.

The subject of the research are models, methods and means of agent-oriented information technology for assessing the initial stages of the software life cycle.

The dissertation offers basic (universal) ontological models of non-functional characteristics-components of software quality, as well as ontological models of non-functional characteristics-components of quality of specific software, which are based on the requirements of ISO 25010: 2011 and ISO 25023: 2016 and provide a basis for the sufficient amount of information to assess the non-functional characteristics of the software. A model of intelligent agent activity based on ontological approach for evaluation of software requirements specifications has been developed, which is based on comparative analysis of ontologies and is a theoretical basis for developing methods of intellectual agent activity based on ontological approach.

First time developed a method of intelligent agent based on an ontological approach to assess the initial stages of the software life cycle, which works on the basis of the developed model and evaluates the adequacy of information in the specification requirements to determine all non-functional characteristics of software. Provides a conclusion about the sufficiency or insufficiency of information in the specification, provides numerical estimates of the level of sufficiency of information to determine each non-functional characteristic of the software and to determine all non-functional characteristics-components of the software together, forms a list of attributes to supplement the specification of requirements to eliminate a person from the processes of information processing and acquisition of knowledge.

Further elaborated: method of activity of the intellectual agent for the automated semantic analysis (parsing) of specifications of requirements to the software which carries out parsing of the specification, defines quantity and percent of the missing attributes, displays which attributes are missing for this or that subcharacteristic of the nonfunctional characteristic, forms a real ontology for non-functional characteristics

that can be used by an intelligent agent based on an ontological approach to assess the initial stages of the software life cycle.

First time developed: an agent-oriented information technology for assessing the initial stages of the software life cycle based on an ontological approach, which evaluates and improves the adequacy of information requirements to determine each non-functional characteristic separately and all non-functional characteristics together. This AOIT evaluates and provides an increase in the level of adequacy of information in the requirements specification to determine the non-functional characteristics of the software - the increase in the level of adequacy is from 4.71% to 27.79% (for example, from 58.23% to 86.02% for the requirements specification № 1 to the software agent for improving the security of computer systems software, from 81.26% to 85.97% for the specification №2, from 60.85% to 73.7% for the specification №3). The advantages of the developed AOIT are: automation of time-consuming, routine and error-prone task of analysis of requirements specifications and its almost instantaneous execution; hint where a rework on the requirements specification is needed (the user can view the missing attributes and see the areas of the specification that need extra attention, as well as which requirements need to be reworked); providing training for new specification developers, systems engineers, and project managers (using this AOIT helps them see the mistakes they may make and helps them recognize these mistakes in the work of others); assistance in developing high quality requirements; assistance in correcting and correcting requirements errors where they occur - in the early stages of the software life cycle - before they become more expensive to correct; providing a tool for selecting better specifications of software requirements; free internet access at any time without any registration. The economic effect of using the developed AOIT is the ability to save budget software projects for processing and correction (during the life cycle) of defects and errors that occur in the early stages of the life cycle - by demonstrating the weaknesses of software requirements that need to be finalized or revised at that time when they occur.

Key words: agent-oriented information technology, intelligent agent based on ontological approach, non-functional characteristics-components of software quality, ontology, software.

ЗМІСТ

АНОТАЦІЯ 2
ЗМІСТ 14
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 16
ВСТУП 17
РОЗДІЛ 1. АНАЛІЗ ВІДОМИХ МОДЕЛЕЙ, МЕТОДІВ ТА ЗАСОБІВ ОЦІНЮВАННЯ ПОЧАТКОВИХ ЕТАПІВ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 27
1.1. Аналіз впливу інформації у специфікації вимог до програмного забезпечення на успішність програмного забезпечення 27
1.2. Аналіз методів та засобів оцінювання початкових етапів життєвого циклу програмного забезпечення 37
1.3. Онтології та інтелектуальні агенти на основі онтологічного підходу як перспективні засоби для оцінювання початкових етапів життєвого циклу програмного забезпечення 41
1.4. Висновки. Постановка задачі 49
РОЗДІЛ 2. МОДЕЛЮВАННЯ ДІЯЛЬНОСТІ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ 52
2.1. Моделі нефункційних характеристик-складових якості програмного забезпечення 52
2.2.Онтологічні моделі нефункційних характеристик програмного забезпечення 62
2.3. Моделювання діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до програмного забезпечення 69
2.4. Висновки 73
РОЗДІЛ 3. МЕТОДИ ДІЯЛЬНОСТІ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ 74
3.1. Методи діяльності інтелектуальних агентів на основі онтологічного підходу для семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення 74

3.2. Методи діяльності інтелектуальних агентів на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення 78
3.3. Експерименти: аналіз нефункційних характеристик у специфікаціях вимог до програмного забезпечення за допомогою інтелектуальних агентів 83
3.4. Висновки 109
РОЗДІЛ 4. АГЕНТНО-ОРІЄНТОВАНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОЦІНЮВАННЯ ПОЧАТКОВИХ ЕТАПІВ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ 112
4.1. Структура агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу 112
4.2. Реалізація та результати функціювання агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу 117
4.3. Переваги агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу 129
4.4. Висновки 137
ВИСНОВКИ 140
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 145
ДОДАТОК А. Список публікацій здобувача 162
ДОДАТОК Б. Акти впровадження 166
ДОДАТОК В. Лістинг модуля ядра програми AppKernel.php комп’ютерної програми «Агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу» 174

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ – програмне забезпечення

ПС – програмна система

ІТ – інформаційна технологія

ЖЦ – життєвий цикл

ІА – інтелектуальний агент

АОІТ – агентно-орієнтована інформаційна технологія

ВСТУП

Обґрунтування вибору теми дослідження. Оскільки специфіка та особливості предметних галузей, для яких розробляються інформаційні технології, суттєво впливають на зміст і методи опрацювання інформації, відтак наразі виправданим залишається підхід, що базується на дослідженні характеристик і особливостей предметних галузей та розробленні нових інформаційних технологій саме для конкретних галузей [1].

Практично усі сфери людської діяльності на сьогодні пов’язані з комп’ютерними системами, основою яких є програмне забезпечення (ПЗ). Галузь інженерії програмного забезпечення особливо потребує сьогодні ефективних інформаційних технологій, в тому числі для розв’язання задачі оцінювання початкових етапів життєвого циклу програмного забезпечення.

На сьогодні в світі витрачається більше 250 млрд доларів США щорічно на розроблення приблизно 175 тис. програмних проєктів. Середня вартість проєкту для великої компанії становить 2,322 млн доларів США, для середньої компанії – 1,313 млн доларів США, а для невеликої компанії – 434 тис доларів США [2, 3]. При цьому значна кількість програмних проєктів є неуспішними (з перевитратами часу, коштів, з недостатнім функціоналом або такими, що скасовуються до завершення і ніколи не використовуються). В середньому, лише 16-29% програмних проєктів виконуються в межах запланованих часу та бюджету (для великих компаній – 9-16% проєктів); проєкти, виконані найбільшими американськими компаніями, мають лише приблизно 42% від необхідних можливостей та функцій [2-4].

Значна кількість помилок вноситься у програмне забезпечення на етапі формування та формулювання вимог [1, 5-7] – за статистикою, 56% всіх дефектів програмних проєктів вносяться саме на етапі формування та формулювання вимог; близько 50% дефектів вимог є наслідком погано написаних, неясних, неоднозначних або неправильних вимог; інші 50% обумовлені неповнотою специфікації (неповні та пропущені вимоги) [8].

Чим раніше буде виявлено дефект (помилку, порушення, недолік, несправність), тим дешевше обійдеться його виправлення [1]. За статистикою, вартість виправлення дефектів та помилок програмного забезпечення, внесених на ранніх етапах життєвого циклу, експоненційно зростає з кожним наступним етапом життєвого циклу проєкту [8]. Витрати на виправлення некоректних вимог в специфікації, котрі виявлені після випуску продукту, майже в 100 разів перевищують витрати на виправлення недоліків специфікації, що виявлені на більш ранніх етапах, в тому числі на етапі формування та формулювання вимог [9]. Ризиками недостатньо відпрацьованого етапу формування та формулювання вимог є недотримання термінів проєктів та фінансові перевитрати, що можуть призвести до закриття проєкту, а то й розпаду софтверної компанії внаслідок її фінансової нестабільності [1].

Отже, критичний вплив на програмні проєкти та на успішність їх реалізації здійснюють питання, пов'язані із аналізом та оцінюванням початкових етапів життєвого циклу, а успішність реалізації програмного проєкту (як вчасне виконання програмного проєкту в рамках виділеного бюджету та з реалізацією всіх необхідних можливостей та функцій) суттєво залежить від ранніх етапів життєвого циклу ПЗ. Тоді сьогодні, коли кількість високобюджетних програмних проєктів стрімко зростає, *актуальним* є аналіз специфікації вимог до ПЗ та можливість автоматизованого оцінювання рівня відпрацювання початкових етапів життєвого циклу ПЗ, зокрема, виявлення та усунення недоліків початкових етапів життєвого циклу ПЗ та фактів недостатності інформації, котра має до них відношення (причому особливої уваги потребує інформація про нефункційні характеристики ПЗ).

Для оцінювання початкових етапів життєвого циклу значну цінність мають знання фахівців, що вже володіють досвідом оцінювання рівня відпрацювання початкових етапів для різних типів ПЗ. Інформацію щодо вимог зручно подавати у вигляді онтологій, що дають змогу відобразити причинно-наслідкові зв’язки між вимогами.

Дослідженню впливу початкових етапів життєвого циклу на успішність програмних проєктів присвячено ряд робіт українських та іноземних учених: Т. Говорущенко [1, 10, 11], О. Димо [12], Д. Маєвського [7, 13], О. Харченка [14], В. Ліпаєва [15, 16], В. Харченка [17, 18], О. Гордєєва [19-22], В. Міщенка [23, 24], В. Яковини [25, 26], М. Фузані (Fuzani) [27, 28], С. Макконнелла (McConnell) [5], Н. Левенсон (Levenson) [6, 29, 30], С. Джонса (Jones) [31-33], А. Чена (Chen) [34], А. Фатванто (Fatwanto) [35], Т. Рехмана (Rehman) [36], Л. Басса (Bass) [37], К. Вігерса (Wiegers) [38], І. Соммервилла (Sommerville) [39]. Онтологіям як перспективному засобу для оцінювання початкових етапів життєвого циклу програмного забезпечення, а також інтелектуальним агентам на основі онтологічного підходу для галузі інженерії ПЗ також присвячено ряд робіт українських та іноземних учених: Ф. Андона [40], Л. Бабенка [41], Й. Лі (Li) [42], Н. Ассавамекіна (Assawamekin) [43], К. Леоніда (Leonid) [44], Н. Бажнайд (Bajnaid) [45], А. Фрейтаса (Freitas) [46, 47], К. Оссовської (Ossowska) [48], І. Гарсіа-Магаріно (Garcia-Magarino) [49], А. Ракіба (Rakib) [50]. Однак задача автоматизованого оцінювання рівня відпрацювання початкових етапів життєвого циклу ПЗ на основі аналізу специфікацій у цих роботах не розв’язувалась.

Потреба у автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення, а також відсутність інформаційної технології для оцінювання початкових етапів життєвого циклу ПЗ створюють *актуальну науково-прикладну задачу*, одним із шляхів розв’язання якої є розроблення агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу.

Мета і завдання дослідження відповідно до предмета та об’єкта дослідження. *Об’єкт дослідження* – процеси оцінювання початкових етапів життєвого циклу програмного забезпечення.

Предмет дослідження – моделі, методи та засоби агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення.

Метою дисертаційного дослідження є автоматизація оцінювання початкових етапів життєвого циклу програмного забезпечення шляхом розроблення агентно-орієнтованої інформаційної технології на основі онтологічного підходу.

Для досягнення поставленої мети необхідно вирішити такі *задачі:*

– дослідити сучасні підходи до оцінювання початкових етапів життєвого циклу програмного забезпечення;

– провести моделювання діяльності інтелектуальних агентів на основі онтологічного підходу для автоматизованого аналізу специфікацій вимог до ПЗ, а також для оцінювання специфікацій вимог до ПЗ;

– розробити методи діяльності інтелектуальних агентів на основі онтологічного підходу для автоматизованого семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення, а також для оцінювання початкових етапів життєвого циклу програмного забезпечення;

– спроєктувати та реалізувати агентно-орієнтовану інформаційну технологію для оцінювання початкових етапів життєвого циклу ПЗ;

– виконати практичне впровадження отриманих результатів при оцінюванні початкових етапів життєвого циклу ПЗ.

Методи дослідження. При вирішенні наукових задач використовувались принципи системного аналізу (ієрархічності, декомпозиції та ін.), методи аналізу та моделювання процесів. У процесі моделювання діяльності інтелектуальних агентів на основі онтологічного підходу для автоматизованого аналізу специфікацій вимог до ПЗ, а також при розробленні методів діяльності інтелектуальних агентів на основі онтологічного підходу для автоматизованого семантичного аналізу (парсингу) специфікацій вимог до ПЗ та для оцінювання початкових етапів життєвого циклу ПЗ використано теоретико-множинні підходи, алгебру систем, методи онтологічного моделювання, апарат модельно-орієнтованих підходів, методи концептуального моделювання, принципи побудови баз знань та формування логічного висновку. При проєктуванні та реалізації агентно-орієнтованої інформаційної технології для оцінювання

початкових етапів життєвого циклу ПЗ застосовувались загальні принципи створення інформаційних систем та систем підтримки прийняття рішень. Для побудови та візуалізації онтологій було використане вільно поширюване ПЗ Protégé 4.2. Для реалізації агентно-орієнтованої інформаційної технології у вигляді веб-сервісу була використана мова програмування PHP та фреймворк Symfony з архітектурою MVC.

Наукова новизна отриманих результатів полягає у розробленні агентно-орієнтованої інформаційної технології для оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу, яка забезпечила можливість автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення та усунення людського фактору з цього процесу.

Одержано такі наукові результати:

вперше розроблено:

1) модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до програмного забезпечення, яка ґрунтується на порівняльному аналізі онтологій та є теоретичним підґрунтям для реалізації інтелектуального агента на основі онтологічного підходу;

2) метод діяльності інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення, який працює на основі розробленої моделі та здійснює оцінювання достатності інформації у специфікації вимог для визначення всіх нефункційних характеристик-складових якості програмного забезпечення, забезпечує висновок про достатність або недостатність інформації у специфікації, надає числові оцінки рівня достатності інформації для визначення кожної нефункційної характеристики програмного забезпечення та для визначення всіх нефункційних характеристик-складових якості програмного забезпечення разом, формує список атрибутів, якими варто доповнити специфікацію вимог для підвищення рівня достатності її інформації, тобто в комплексі дозволяє частково усунути людину з процесів опрацювання інформації та здобуття знань;

одержала подальшого розвитку:

3) агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу в частині автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення, яка, на відміну від відомих, виконує оцінювання та забезпечує підвищення рівня достатності інформації вимог для визначення кожної нефункційної характеристики окремо та всіх нефункційних характеристик разом;

удосконалено:

4) метод діяльності інтелектуального агента для автоматизованого семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення, який, на відміну від відомих, грунтується на врахуванні вимог стандарту ISO 25010 і на обраній номенклатурі метрик та виконує парсинг специфікації, визначає кількість та відсоток відсутніх атрибутів, відображає, яких атрибутів не вистачає для тієї чи іншої підхарактеристики нефункційної характеристики, а також формує реальну онтологію для нефункційних характеристик, яка може бути використана інтелектуальним агентом на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення.

Особистий внесок здобувача полягає в розробленні нових моделей, методів, елементів інформаційної технології та інструментальних засобів, що забезпечують вирішення поставлених у дисертації задач. Всі основні наукові положення, результати, висновки і рекомендації дисертаційної роботи отримані автором особисто. Основні результати дисертації опубліковані у 28 наукових працях ([51-78] та додаток А), серед яких 9 статей у періодичних виданнях, що індексуються в наукометричних базах Scopus, Web of Science [51-59] (в тому числі 1 стаття, яку опубліковано у періодичному науковому виданні іншої держави, яка входить до Європейського Союзу [56]; 1 стаття у періодичному виданні 2-го квартилю, що індексується в наукометричній базі Scopus [53]); 8 статей у фахових наукових журналах України [60-67]; 3 статті в матеріалах конференцій, що індексуються в наукометричних базах Scopus, Web of Science

[68-70]; 3 публікації у матеріалах конференцій (тези доповідей) [71-73]; 5 свідоцтв про реєстрацію авторського права на твір [74-78].

У роботах, опублікованих у співавторстві, здобувачеві належать: дослідження сучасних підходів до оцінювання початкових етапів життєвого циклу програмного забезпечення [60, 61, 64, 70, 71]; модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ [57, 62]; метод діяльності інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення [51-53, 58, 63, 67, 68, 72, 74]; метод діяльності інтелектуального агента для автоматизованого семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення [56, 59, 66, 69, 76]; проєктування та реалізація агентно-орієнтованої інформаційної технології для оцінювання початкових етапів життєвого циклу програмного забезпечення [54, 55, 65, 73, 75, 77, 78].

Апробація матеріалів дисертації. Апробацію основних положень, ідей, висновків дисертаційної роботи проведено на міжкафедральному науково-практичному семінарі кафедри комп'ютерних наук та інформаційних технологій та кафедри комп'ютерної інженерії та системного програмування у Хмельницькому національному університеті. Наукові результати роботи доповідалися також на:

– IEEE International Scientific and Technical Conference "Computer Science and Information Technologies" CSIT (м. Львів, 2018);

– International Conference on Data Science and Intelligent Analysis of Information (м. Київ, 2018);

– International Workshop on Informatics & Data-Driven Medicine IDDM (м. Львів, 2018);

– Міжнародній науково-практичній конференції «Електротехнічні та комп'ютерні системи» ELTECS (м. Одеса, 2018);

– Міжнародній науково-технічній конференції «Штучний інтелект та інтелектуальні системи» AIIS (м. Київ, 2018);

– Всеукраїнській науково-практичній конференції «Сучасні тенденції розвитку додрукарських систем» (м.Львів, 2018);

– IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (м. Мец, Франція, 2019);

– International Conference on ICT in Education, Research, and Industrial Applications ICTERI (м. Херсон, 2019);

– International Scientific Conference “Intellectual Systems of Decision-Making and Problems of Computational Intelligence” ISDMCI (м. Херсон, 2019);

– International Workshop on Computer Modelling & Intelligent Systems CMIS (м. Запоріжжя, 2019);

– International Workshop on Digital Content & Smart Multimedia (м. Львів, 2019);

– Всеукраїнській науково-практичній конференції "Інтелектуальний потенціал-2019" (м. Хмельницький, 2019);

– IEEE International Conference on Dependable Systems, Services and Technologies DeSSerT (м. Харків, 2020);

– International Workshop on Intelligent Information Technologies and Systems of Information Security (м. Хмельницький, 2020).

Структура та обсяг дисертації. Дисертація складається з анотації, змісту, переліку умовних скорочень, вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Повний обсяг роботи становить 175 сторінок друкованого тексту, з них анотація – на 12 стор., зміст – на 2 стор., перелік умовних скорочень – на 1 стор., основний текст – на 128 стор., список із 152 використаних джерел – на 17 стор., додатки – на 14 стор. Дисертація містить 48 рисунків та 5 таблиць.

Зв’язок роботи з науковими програмами, планами, темами. Дослідження, результати яких викладено в дисертації, виконано в рамках виконання науково-дослідних робіт за держбюджетною темою Хмельницького національного університету “Агентно-орієнтована система підвищення безпеки та якості програмного забезпечення комп’ютерних систем” (ДР №0119U100662).

Роль автора в НДР, в якій вона є безпосереднім виконавцем, полягає у розробленні моделей, методів та інформаційної технології для оцінювання початкових етапів життєвого циклу програмного забезпечення.

Практичне значення отриманих результатів. Практичне значення отриманих результатів полягає в доведенні теоретичних положень дисертації до реалізації, рекомендацій та безпосередньому використанні на підприємстві. Була реалізована агентно-орієнтована інформаційна технологія для оцінювання початкових етапів життєвого циклу програмного забезпечення, яка: автоматизує трудомістку та схильну до помилок задачу розбору специфікації вимог; вказує на необхідність доопрацювання специфікації із зазначенням вимог, які потребують доопрацювання; забезпечує швидке навчання нових системних інженерів та керівників проєктів (використання розробленої інформаційної технології під час створення або аналізу вимог допомагає їм швидко бачити помилки, яких вони можуть припуститись, і допомагає розпізнавати ці помилки в роботі інших); допомагає виправити та усунути помилки та неточності у вимогах на ранніх етапах життєвого циклу; є інструментом для вибору більш якісної специфікації; доступна онлайн в будь-який час без реєстрації.

Результати дисертаційної роботи впроваджено у (додаток Б):

- ТОВ «ІТТ» (акт впровадження від 10.03.2020 р.) при оцінюванні специфікацій вимог до програмного забезпечення (ПЗ) обліку надання послуг доступу до мережі Інтернет;

- ТОВ «Деймос» (акт впровадження від 20.02.2020 р.) при розробленні автоматизованої системи управління виробничими процесами;

- ГО «ІТ Кластер м. Хмельницького» (акт впровадження від 03.06.2020 р.) при оцінюванні специфікацій вимог до програмного агента для підвищення безпеки програмного забезпечення комп’ютерних систем;

- при виконанні держбюджетного проєкту Хмельницького національного університету “Агентно-орієнтована система підвищення безпеки та якості програмного забезпечення комп’ютерних систем” (ДР №0119U100662);

- у навчальному процесі Хмельницького національного університету (акт впровадження від 13.05.2020 р.) та ПВНЗ «Міжнародного науково-технічного університету імені академіка Юрія Бугая» (акт впровадження від 08.10.2020 р.).

Подяка. Автор висловлює глибоку подяку науковому керівнику, д.т.н., професору, завідувачці кафедри комп'ютерної інженерії та системного програмування Хмельницького національного університету Тетяні Олександрівні Говорущенко, завідувачу кафедри комп'ютерних наук та інформаційних технологій Хмельницького національного університету, д.т.н., професору Руслану Володимировичу Сорокатому за корисні поради та зауваження, підтримку і віру під час підготовки даної дисертаційної роботи.

РОЗДІЛ 1.
АНАЛІЗ ВІДОМИХ МОДЕЛЕЙ, МЕТОДІВ ТА ЗАСОБІВ ОЦІНЮВАННЯ ПОЧАТКОВИХ ЕТАПІВ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Аналіз впливу інформації у специфікації вимог до програмного забезпечення на успішність програмного забезпечення

Розроблення програмного забезпечення – це наукомістка діяльність, яка вимагає детального вивчення предметної галузі та повного розуміння цілей продукту, що розробляється [1].

У галузі розроблення ПЗ і досі існують проблеми – великі проєкти виконуються з відставанням від графіка або з перевищенням кошторису витрат, розроблений продукт не має необхідних функціональних можливостей [1-9, 79-87].

Аналіз статистики успішності програмних проєктів за 1994-2019 роки, за даними The Standish Group International (Chaos reports) [2-4, 80-86], представлений на рис.1.1. Успішними вважаються проєкти, які були виконані вчасно, в рамках бюджету, з необхідними можливостями та функціями. Проблемними вважаються проєкти, які мали перевищення термінів, перевитрати або не мали необхідних можливостей та функцій. Провальними вважаються проєкти, які були скасовані до завершення, або були доставлені, але ніколи не використовуються.

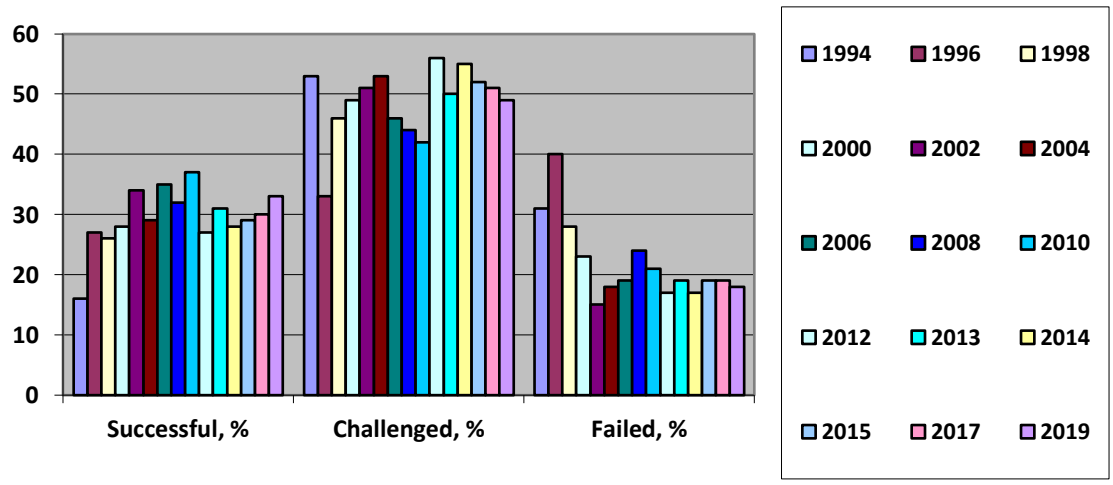

Рис.1.1 – Статистика успішності програмних проєктів у 1994-2019 рр.

Аналіз даних рис. 1.1 дав можливість побачити, що кількість проблемних проєктів є досить сталою величиною та в останні роки складає приблизно 50% проєктів.

На сьогодні, успішність програмних проєктів в залежності від методології представлена на рис. 1.2 [85].

Рис.1.2 – Статистика успішності програмних проєктів у 2019 році в залежності від методології [85]

Як видно з рис. 1.2, Agile-проєкти є більш успішними, ніж Waterfall-проєкти, але методологія Agile підходить не для всіх типів програмних проєктів (наприклад, для ПЗ критичного застосування прийнятною є лише методологія Waterfall). Крім того, і серед Agile-проєктів, і серед Waterfall-проєктів половина (1/2) всіх проєктів є проблемними, тобто, як правило, мають затримки в часі розробки, перевищення бюджету або не мають необхідних характеристик.

Дослідження The Standish Group International доводять, що 31,1% програмних проєктів будуть скасовані до їх завершення, а 52,7% проєктів коштуватимуть 189% від їхніх початкових оцінок. Але такі перевитрати при розробленні – це лише «вершина айсберга». Альтернативні витрати не завжди можна виміряти, але вони можуть сягати трильйонів доларів. Так, наприклад, відсутність надійного ПЗ для обробки багажу в аеропорту м. Денвер коштує місту 1,1 млн. доларів на день [2, 3, 81, 82].

В середньому, лише 16-29% програмних проєктів виконуються в межах запланованих часу та бюджету (для великих компаній цей обсяг становить лише 9-16% проєктів). Крім цього, проєкти, виконані найбільшими американськими компаніями, мають лише приблизно 42% від необхідних можливостей та функцій (для малих компаній 78,4% програмних проєктів мають принаймні 74,2% початкових можливостей та функцій) [2, 3, 81, 82] – рис. 1.3.

	CHAMPIONS	UNDER-PERFORMERS
Average percentage of projects completed on time	88%	24%
Average percentage of projects completed within budget	90%	25%
Average percentage of projects that meet original goals/business intent	92%	33%
Average percentage of projects experiencing scope creep	28%	68%
Average percentage of projects deemed failures	6%	24%
Average percentage of budget lost when a project fails	14%	46%

Рис.1.3 – Середні показники ефективності проєктів [83]

Дослідження міжнародної компанії McKinsey & Company [88] спільно з University of Oxford також показали, що половина масштабних програмних проєктів з загальним бюджетом понад 15 мільйонів доларів значно перевищують заплановані витрати, зокрема: середнє перевищення витрат на проєкт складає 66%, середнє перевищення часу реалізації проєкту – 33%, а середнє число втрат прибутку – 17% (рис. 1.4, 1.5 [88]).

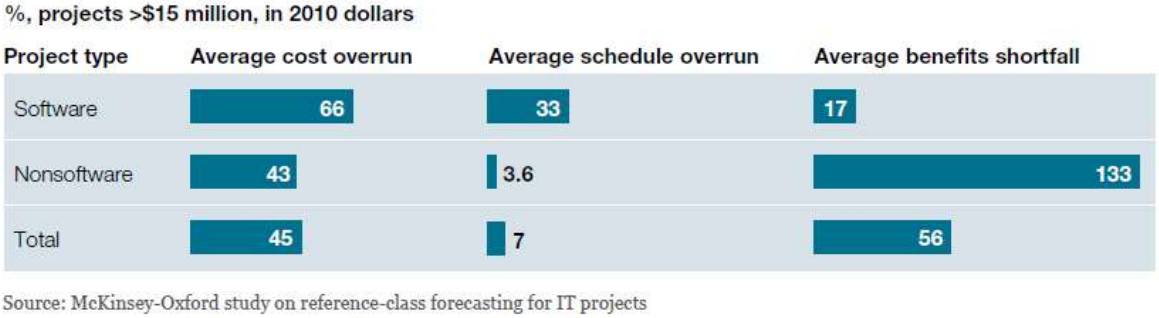

Рис.1.4 – Різниця у продуктивності різних типів проєктів [88]

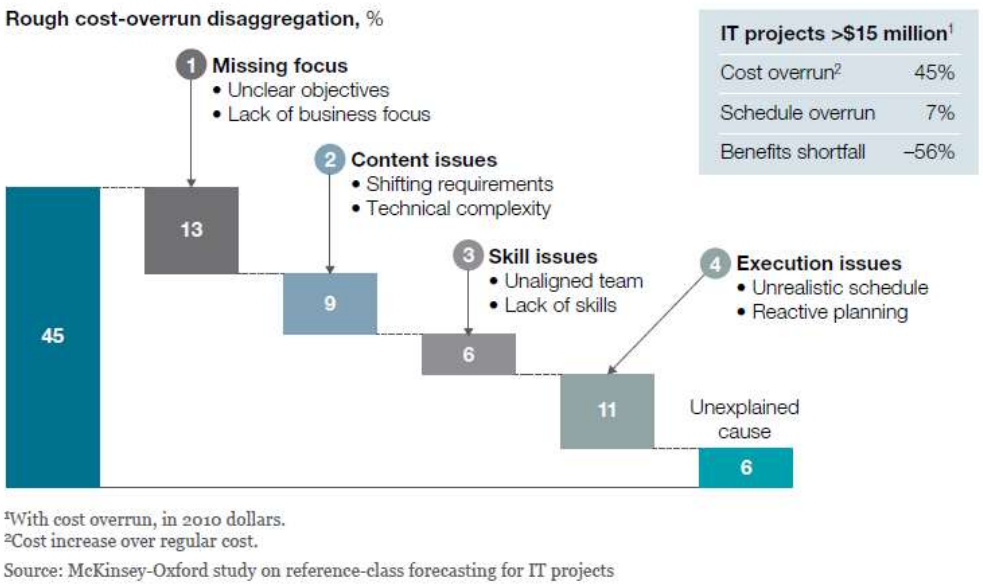

Рис.1.5 – IT-керівники визначають 4 групи проблем, які викликають більшість збоїв у проєкті [88]

В останні роки акцент змістився з виробництва автономних програмних продуктів на виробництво програмної системи як широкої системи систем (system-of-systems), інтегрованої з великої кількості компонентів (підсистем) з інтерфейсами взаємодії між ними [89, 90]. Оскільки програмне забезпечення стає дедалі складнішим та масштабнішим, розроблення саме програмної системи буде відігравати все більш важливу роль в діяльності розробника [1].

Однією з найбільш вагомих причин незадовільної якості великих програмних проєктів дослідники The Standish Group International називають збільшення кількості компонентів (підсистем) та інтерфейсів між ними, а також неконтрольовану складність програмної системи [2, 3, 82]. Дослідження The Standish Group International (CHAOS reports) [2, 3, 82] доводять, що статистика успішності малих та великих програмних проєктів суттєво різна. Серед малих проєктів 62% є успішними в той час, як серед великих проєктів успішними є лише 6%, а серед надвеликих проєктів успішними є лише 2%, тобто малі проєкти в десятки разів (на порядок) успішніші за великі. Такий висновок підтверджує і статистика проєктів на основі використання функційних точок, як основних одиниць вимірювання розміру ПЗ [1, 39, 79] – рис.1.6.

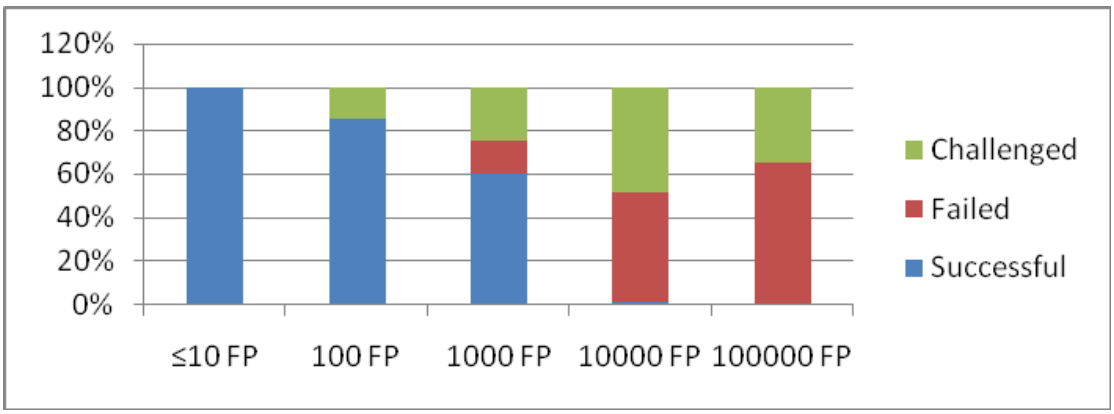

Рис.1.6 – Статистика успішності програмних проєктів обсягом у 10, 100, 1000, 10000, 100000 функційних точок [1]

Значна кількість помилок вноситься у програмне забезпечення на етапі збору вимог [83, 91-94]. Помилки формування та формулювання вимог і проєктування архітектури складають 25-55% всіх помилок, причому чим більший обсяг ПЗ, тим більше помилок вноситься саме на ранніх етапах [5]. Статистика показує, що збір неточних вимог є однією з основних причин відмов програмного забезпечення (рис. 1.7) [83, 93]. В ідеалі всі проблеми, що виникають на етапі вимог, повинні бути вирішені до початку проєктування. Пізнє виявлення помилок вимог є найдорожчим для виправлення [95].

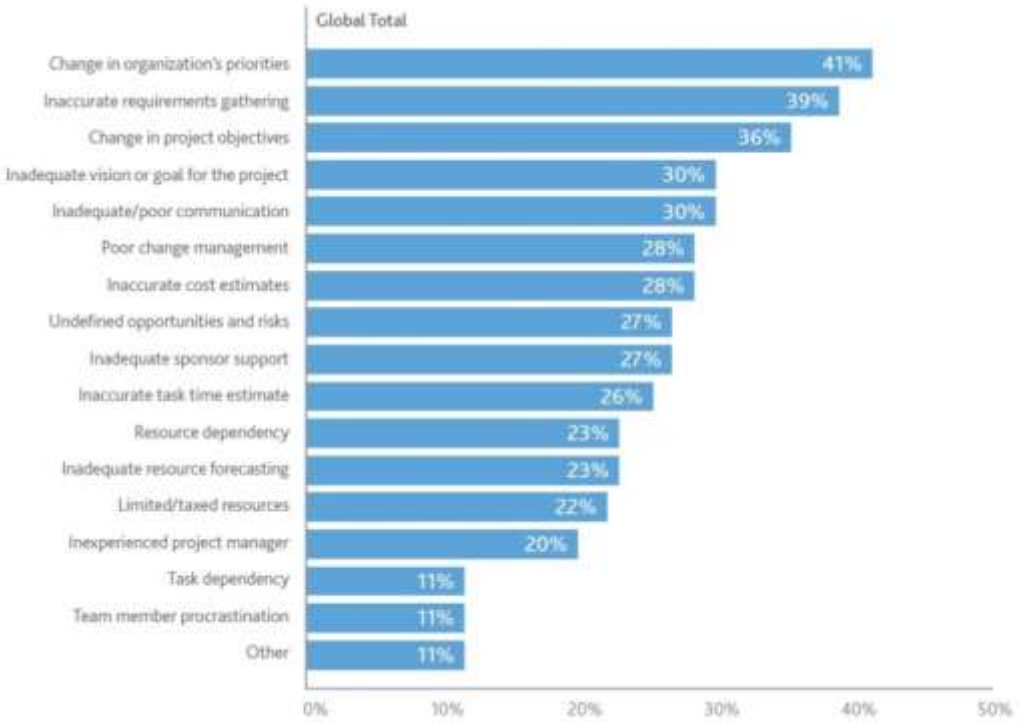

Рис.1.7 – Першочергові причини помилок у ПЗ [83]

Фактори успіху програмного забезпечення представлені на рис. 1.8 [86]. Очевидно, що чіткі вимоги мають ваговий коефіцієнт у 13% серед факторів успіху проєкту.

Рис.1.8 – Фактори успіху програмних проєктів у 2020 році [86]

Основною причиною невдач та збоїв програмного забезпечення є неякісна проєктна документація, особливо на системному рівні [9, 96] – рис. 1.9. Неякісна документація спричиняє багато помилок та знижує ефективність на кожному етапі розробки та використання програмного продукту, що призводить до збою або проблем програмного проєкту [84].

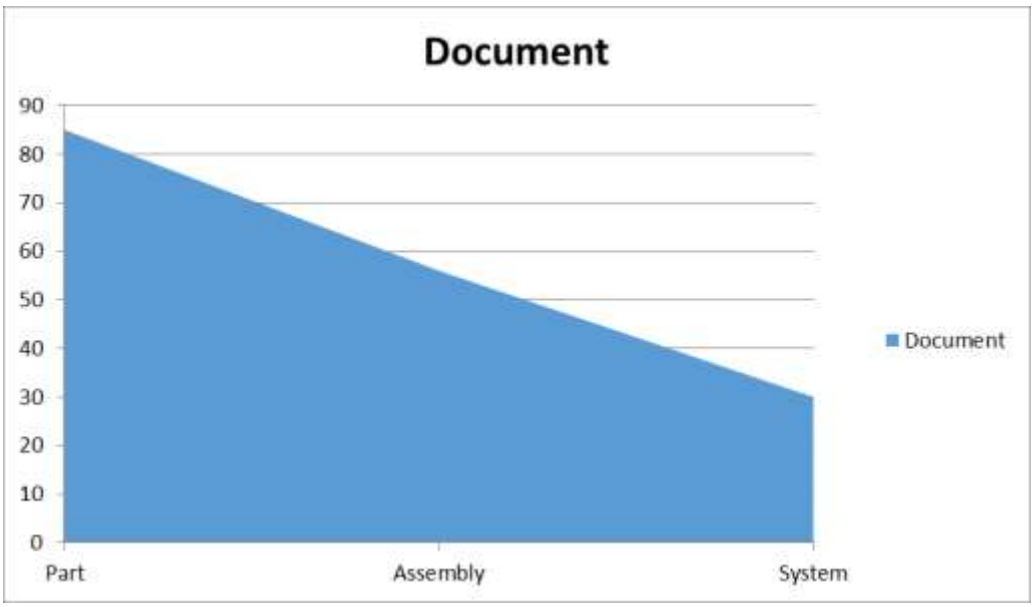

Рис.1.9 – Відсоток документації знань на різних етапах розроблення ПЗ

В роботах [6, 97] підтверджується факт, що причини майже всіх інцидентів та катастроф, пов’язаних з ПЗ, криються у специфікації вимог до ПЗ. Переважна більшість аварій, пов’язаних із ПЗ, виникли через помилкові вимоги, а не через помилки кодування. У [97] було встановлено, що версії ПЗ, написані різними розробниками за однаковими вимогами, містили ряд спільних помилок, пов’язаних із помилками або неточностями вимог (специфікації).

Дефекти вимог бажано виявляти та усувати до того, як вони почнуть впливати на більш пізні етапи розроблення. Ранні етапи життєвого циклу впливають на якість ПЗ більше, ніж пізні, тому час, витрачений на контроль якості на ранніх етапах, забезпечує можливість знизити рівень дефектів, скоротити терміни розроблення та знизити витрати на більш пізніх етапах [10].

Чим раніше буде виявлено дефект (помилка, порушення, недолік, несправність), тим дешевше обійдеться його виправлення. Як показано на рис.1.10, витрати на виправлення дефектів, виявлених після випуску продукту, майже в 100 разів перевищують витрати на виправлення, якщо недоліки були виявлені в процесі формування та формулювання вимог [9]. Якщо розглянути альтернативні варіанти дизайну та провести аналіз впливу помилок у вимогах ще до створення системи як такої, це дасть суттєву економію коштів на корекцію помилок. І якщо на дослідження витрат зазвичай витрачаються години, а то й дні, то в даному випадку рахунок ітиме на хвилини [9].

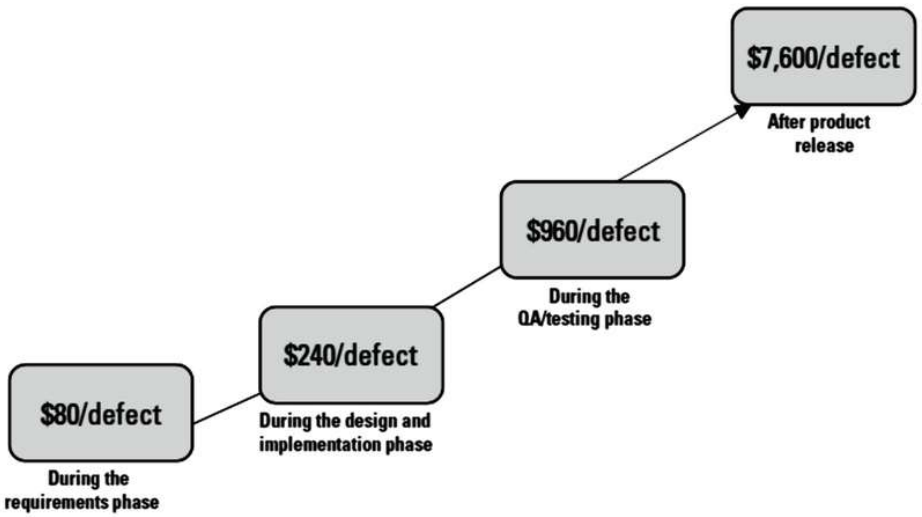

Рис.1.10 – Вартість виправлення дефектів різко зростає протягом усього процесу розробки [9]

Із збільшенням інтервалу між моментами внесення та виявлення дефекту вартість його виправлення сильно зростає. Чим довше помилка зберігається у ланцюгу розроблення ПЗ, тим сильніше вона проникає в інші частини ПЗ, тим більше шкоди завдає на наступних етапах, і тим більше коштів доведеться витратити на її усунення.

В процесі формування та формулювання вимог можуть відбуватись інформаційні втрати через неповне та різне розуміння потреб та контексту інформації – особливо такі втрати суттєві для програмних проєктів, які розробляються на стику предметних галузей, коли враховувати потрібно як стандарти щодо розроблення ПЗ, так і стандарти предметної галузі, для якої розробляється ПЗ. Таку кількість стандартів важко імплементувати, а ще важче оцінити ступінь врахування рекомендацій цих стандартів [1].

Всі наявні знання та інформацію про програмну систему представляють у вигляді діаграми, в якій є сектор, що відображає об’єм недостатньої (невідомої) інформації (розрив у знаннях) – рис. 1.11 [1].

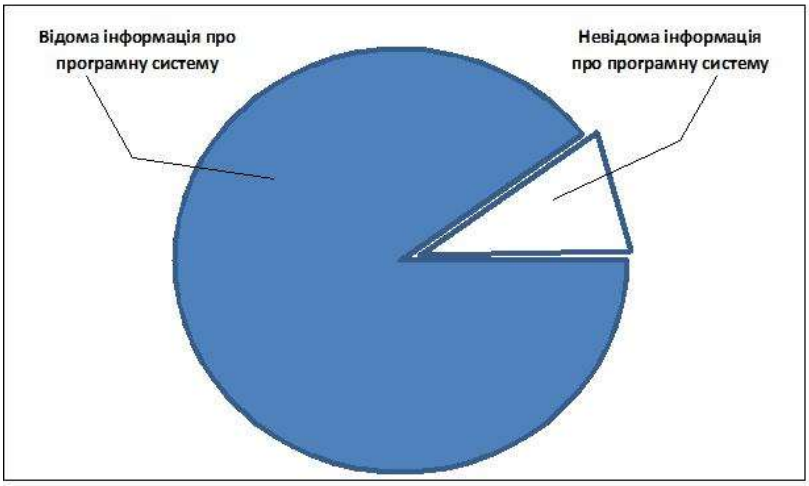

Рис.1.11 – Галузь знань про програмну систему із сектором невідомої (або нерозглянутої) інформації [1]

Сектор з невідомою інформацією слід звужувати. Чим меншим буде розмір сектору невідомої інформації, тим якіснішою буде програмна система та тим безпечніше вона працюватиме. Отже, актуальним є підхід до зменшення частки невідомої інформації про програмну систему [1].

З визначення якості ПЗ як ступеня задоволеності користувачів або ступеня відповідності ПЗ потребам замовників [98-100] слідує, що, якщо цілі проєкту не відповідають потребам користувачів, то ПЗ неможливо визнати успішним та якісним. Отже, якість та успішність реалізації програмного проєкту суттєво залежать від специфікації вимог. Таке експериментальне свідчення безпосередньо призводить до необхідності поглиблення аналізу специфікації вимог до ПЗ та необхідності тестування документації та вимог [10, 11, 101].

Враховуючи сучасні рівні і типи вимог, описані К. Вігерсом у [38], при подальшому дослідженні оцінюватимемо, наскільки повно у бізнес-вимогах відображається інформація щодо функцій та обмежень майбутнього ПЗ, причому особливу увагу буде присвячено нефункційним характеристикам (вимогам) та атрибутам якості.

На сьогодні, ключовими є 2 підходи до оцінювання якості та нефункційних характеристик-складових якості ПЗ – за моделлю SQuaRE (стандарт ISO 25010:2011 [98]) та на основі результатів метричного аналізу [1].

Загальний підхід до оцінювання якості ПЗ за моделлю SQuaRE (стандарт ISO 25010:2011 [98]) полягає в тому, щоб спочатку ідентифікувати невеликий набір характеристик найвищого рівня абстракції, а потім в напрямку "згори-донизу" розбити ці характеристики на підхарактеристики та набори підлеглих атрибутів [1].

Модель якості SQuaRE, створена в межах стандарту [98], 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). Кожна характеристика якості ПЗ є функцією від декількох підхарактеристик якості (всього 31 підхарактеристика, згідно зі стандартом [98]). Нижній рівень ієрархії представляють атрибути якості, визначені та описані у стандарті ISO 25023:2016 [102]. Аналіз стандарту [102] дав можливість визначити залежність підхарактеристик якості від 203 атрибутів (в тому числі від 138 різних атрибутів).

Другим підходом до оцінювання якості ПЗ є оцінювання якості на основі результатів метричного аналізу. Сучасна програмна індустрія накопичила велику кількість метрик, які оцінюють окремі виробничі та експлуатаційні властивості програмного забезпечення. Згідно зі стандартом ISO 24765 [103], метрика визначається як міра ступеня володіння властивістю, яка має числове значення. Взагалі, метрика ПЗ – це міра, яка надає числове значення деякої властивості ПЗ як зважене середнє арифметичне з урахуванням значень показників, що оцінюють цю метрику, та коефіцієнтів їхньої вагомості. В результаті аналізу метрик складності та якості з точки зору можливості їх застосування на ранніх етапах життєвого циклу ПЗ з одержанням точного або прогнозованого значення було виділено ряд метрик, які можуть бути використані на етапі проєктування – 10 метрик з точними значеннями на етапі проєктування та 14 метрик з прогнозованими значеннями на етапі проєктування [1, 104], які залежать від 72 показників (в тому числі від 42 різних показників) [105, 106].

Важливими властивостями якісних вимог [38] є їх коректність за завершеність з позицій достатності інформації у вимогах про атрибути та показники якості.

Достатністю інформації у специфікації вимог до ПЗ є наявність у специфікації всіх інформаційних елементів, необхідних для визначення нефункційних характеристик ПЗ [1, 104].

Оцінювання достатності інформації специфікації вимог до ПЗ забезпечує можливість вибору програмного проєкту, підвищує ефективність управління проєктом за рахунок обґрунтованості рішень, скорочує час на їх розробку і прийняття, зменшує витрати на збір і обробку відомостей. Недостатність інформації специфікації вимог до ПЗ знижує результативність та достовірність оцінювання нефункційних характеристик ПЗ [1].

Програмні вимоги визначають потрібні нефункційні характеристики ПЗ, а також впливають на методи кількісного оцінювання та сформульовані для оцінювання цих характеристик критерії приймання. Отже, всю необхідну інформацію закладено вже у специфікації вимог до ПЗ, тобто вже на основі

специфікації вимог до ПЗ можна оцінити достатність інформації для подальшого розроблення ПЗ [1, 7, 13, 104]. Якщо деякі інформаційні елементи відсутні, то у специфікації вимог недостатньо інформації, і розробники повинні внести необхідні доповнення у специфікацію [1].

Отже, як показав проведений аналіз впливу інформації специфікації вимог на успішність ПЗ, сучасні програмні системи є менш залежними від написання програмного коду, але суттєво залежать від ранніх етапів. Дефекти, внесені на ранніх етапах, варто виявляти та усувати до того, як вони почнуть впливати на результати більш пізніх етапів життєвого циклу. Якість та успішність реалізації програмного проєкту суттєво залежать від специфікації вимог до ПЗ, відтак аналіз вимог є надзвичайно важливим та актуальним.

1.2. Аналіз методів та засобів оцінювання початкових етапів життєвого циклу програмного забезпечення

На сьогодні відомі наступні підходи до оцінки достатності інформації вимог до програмного забезпечення – таблиця 1.1 [55].

Огляд відомих інформаційних технологій та інструментів для аналізу вимог до програмного забезпечення наведено в таблиці 1.2 [55].

Таблиця 1.1

Огляд відомих підходів до оцінки достатності інформації вимог до ПЗ

Підхід	**Обмеження підходу**
Модель для перевірки достатності вимог безпеки, зосереджуючи увагу на достатності ідентифікації небезпек, аналізу небезпек та простежуваності вимог безпеки програмного забезпечення [107, 108]: було введено	На основі запропонованої метрики можна оцінити лише достатність кількості вимог безпеки, але не достатність їхньої інформації; неможливість інтерпретації шляхом порівняння запропонованої метрики для

кілька різних показників, зокрема: відсоток вимог до безпеки ПЗ – як співвідношення кількості вимог безпеки програмного забезпечення до загальної кількості вимог програмного забезпечення; автори пропонують порівняти цю метрику з аналогічною метрикою для реалізованого програмного забезпечення, на основі цього порівняння робиться висновок про достатність вимог безпеки програмного забезпечення	принципово нового програмного забезпечення; не існує інструменту для автоматичного визначення та обчислення вимог безпеки у специфікації
Оцінка достатності тестування як досягнення рівня охоплення тестами, рекомендованими або передбаченими стандартами безпеки та галузевими настановами [109]	Підхід спрямований лише на перевірку відповідності ПЗ та вимог, але не на перевірку відповідності розробленого ПЗ та потреб клієнтів; підхід використовує специфікацію виключно як вхідні дані для розробленого інструменту, але не перевіряє вимоги специфікації на їх достатність

Таблиця 1.2

Огляд відомих інформаційних технологій та інструментів для аналізу вимог до ПЗ

Інформаційна технологія або засіб	**Обмеження ІТ або засобу**
CORE: дозволяє користувачеві витягувати вимоги з вихідної документації, а потім аналізує їх на предмет повноти, послідовності та можливості тестування [110]	Перевіряє повноту специфікації відповідно до бізнес-вимог, але не передбачає перевірки бізнес-вимог на їх повноту

Visure – перевірка якості вимог за допомогою обробки природної мови та семантичного аналізу [111]	Не забезпечує підтвердження відповідності вимог специфікації потребам замовника
Accompa: забезпечує автоматичний збір вимог; автоматично виявляє та відстежує залежності між вимогами [111]	Комерційний інструмент – коштує 199 доларів США на місяць без установки та обслуговування [112]; не перевіряє, чи всі потреби та вимоги користувача були відображені в специфікації
Innoslate: аналізує вимоги з використанням технології обробки природної мови [111]	Комерційний інструмент – Innoslate Cloud коштує 49 доларів США на користувача за місяць, а Innoslate Enterprise – 199 доларів США за користувача на місяць [112]; не надає кількісних оцінок властивостей інформації специфікації
ReqView: упорядковує вимоги в ієрархію дерев, використовує формати тексту для опису вимог [111]	Не перевіряє та не підтверджує відповідність вимог специфікацій потребам клієнтів
Modern Requirements4TFS: проводить аналіз відстежуваності для забезпечення якості та виявлення прогалин або залежностей у вимогах [111]	Не визначає потреб користувача, які не були відображені у вимогах; не забезпечує візуалізацію прогалин у вимогах
Natural language processing (NLP) Requirements Analysis Tools (наприклад, QVscribe): автоматизують і значно пришвидшують пошук можливих помилок у природомовних вимогах [8]	Не виявляє інформаційних втрат при формуванні вимог

Requirements Analysis Tool: використання визначених користувачем глосаріїв для видобування структурованого вмісту; семантичні веб-технології використовуються для глибшого семантичного аналізу витягнутого структурованого вмісту для пошуку різних видів проблем у документах вимог [113]	Обмеження глосарію, на якому базується інструмент
QARCC (Quality Attribute Risk and Conflict Consultant): це інструмент для підтримки виявлення конфліктів та узгодження вимог; це заснований на знаннях інструмент, що використовується для виявлення та аналізу конфліктів на початку життєвого циклу [114]	Не перевіряє достатність вимог (зокрема, нефункційних вимог)
Requirements Assistant: визначає відсутні вимоги та невідповідність вимог; виявляє відсутність деяких типів нефункціональних вимог [115]	Комерційний інструмент; не надає кількісних оцінок (показників) щодо відсутніх нефункційних вимог
QuARS Requirements Analysis Tool: забезпечує перевірку вимог щодо послідовності, повноти; виявляє 37% дефектів вимог [114, 116]	Комерційний інструмент
DESIRe: забезпечує дотримання правил повноти, неоднозначності та зрозумілості [117]	Комерційний інструмент
RQV Tool (Requirement Quality Verification Tool) [117]: напівавтоматичний інструмент перевірки специфікації на основі комплексної моделі якості; може допомогти у верифікації якості специфікації на основі ISO 29148:2011	Надає оцінки якості інформації специфікації, але не її повноти або достатності

Огляд відомих підходів до оцінки достатності інформації вимог, а також інформаційних технологій та інструментів для аналізу специфікацій показав, що існує ряд ефективних рішень, але всі вони не розв'язують задачі оцінювання початкових етапів життєвого циклу ПЗ. Крім того, вони належать до різних методологічних підходів, розроблені для різних завдань і не поєднуються одне з одним, а також передбачають людинно-машинну взаємодію на всіх етапах опрацювання інформації, під час якої всю інформацію інтерпретує людина, що часто призводить до втрат істотної інформації.

У [1, 104] вперше розроблено теоретичні та прикладні засади інформаційної технології оцінювання достатності інформації щодо якості у специфікаціях вимог до програмного забезпечення, в якій людина частково усувається з процесів опрацювання інформації та здобуття знань, проте ця ІТ також передбачає аналіз специфікації людиною. Отже, в даний час розроблення потребує інформаційна технологія для оцінювання початкових етапів життєвого циклу ПЗ, яка повністю усуватиме людину з процесів опрацювання інформації та здобуття знань.

1.3. Онтології та інтелектуальні агенти на основі онтологічного підходу як перспективні засоби для оцінювання початкових етапів життєвого циклу програмного забезпечення

Для підвищення достовірності оцінювання нефункційних характеристик ПЗ значну цінність мають знання фахівців, що вже володіють досвідом оцінювання нефункційних характеристик для різних типів ПЗ.

Інформацію щодо оцінювання якості та нефункційних характеристик-складових якості ПЗ за стандартом ISO 25010:2011, а також щодо визначення якості та складності програмного проєкту та ПЗ на основі результатів метричного аналізу зручно подавати у вигляді онтологій, що дають змогу відобразити причинно-наслідкові зв'язки між поняттями [1].

Онтологія – це сукупність концепцій, які здатні моделювати терміни лексики в знаннях предметної галузі [118]. Вона забезпечує краще розуміння

контекстуальних знань. З точки зору інформатики, онтологія визначається як концепція для моделювання структури системи. Онтологія, як правило, має основні компоненти форми, тобто клас, екземпляр та відношення [119].

Перевагами використання онтологій є системний підхід до вивчення предметної галузі, можливість цілісного подання відомої інформації предметної галузі, виявлення дублювань та прогалин у знаннях на основі візуалізації відсутніх логічних зв'язків [1, 120, 121].

Ідея використання онтологій як перспективних засобів для оцінювання початкових етапів життєвого циклу програмного забезпечення не є новою. Так, для здійснення трасування вимог до ПЗ вченими використовувались зважені онтології [42-44, 122]. Метою роботи [123] є використання доменної онтології для аналізу ПЗ та засобів реінжинірингу. Робота [45] представляє онтологічну модель для опису та визначення предметних і операційних знань щодо забезпечення якості ПЗ.

Методи та засоби побудови програмних систем на основі онтологічних моделей задач запропоновані у [120, 121]. Автори [120, 121] пропонують використовувати онтологічні моделі на всіх етапах життєвого циклу ПЗ за рахунок концептуалізації предметної галузі в контексті розв’язуваних задач. Тобто у предметній галузі виділяються необхідні сутності, визначаються їх атрибути та обмеження, залежності між ними і будується онтологія предметної галузі, яка надалі може використовуватись розробниками. Авторами [120, 121] доведено, що використання такої онтології предметної галузі спрощує процес початкової концептуалізації при створенні ПЗ, дозволяє уникнути помилок концептуалізації на ранніх етапах життєвого циклу ПЗ. Але використання такого підходу є неможливим, якщо ПЗ розробляється для предметної галузі, для якої ще не розроблено онтологію. Крім цього, при розробленні ПЗ слід враховувати вимоги та стандарти не лише предметної галузі, але й стандарти щодо розроблення ПЗ, що жодним чином не враховується в описаному у [120, 121] підході. Також методи та засоби побудови програмних систем на основі онтологічних моделей задач не дають можливості автоматизовано аналізувати специфікації вимог до ПЗ, зокрема, на предмет достатності їх інформації, хоча саме автоматизоване опрацювання знань забезпечує мінімізацію інформаційних втрат.

Автори [43] пропонують онтологічну структуру трасованості багатоцільових вимог MUPRET для опрацювання неоднорідності вимог до ПЗ на основі автоматичної генерації відношень трасованості. Точність відношень трасованості, що генеруються структурою MUPRET, перевіряються шляхом порівняння з набором відношень трасованості, які вручну ідентифіковані користувачами. В роботі [44] зважені онтології використовуються для опрацювання природної мови при переході від специфікації вимог, написаній природною мовою, до проєктування ПЗ. Саме онтології, як доводять автори [44], здатні проявити невідповідності вимог, представлених природною мовою. Представлені підходи спрямовані на усунення неоднорідності або невідповідності наявних вимог у специфікаціях, але жодним чином не перевіряють, чи всі потреби та вимоги користувача були відображені у специфікаціях. Отже, вони непридатні для оцінювання достатності інформації щодо нефункційних характеристик у специфікаціях вимог до ПЗ.

Онтологічну модель для опису та визначення предметних і операційних знань щодо забезпечення якості ПЗ розроблено у [45]. Автори [45] запропонували онтологію, яка об’єднує знання щодо термінології та семантичних відносин стандартів SWEBOK, IEEE, ISO, спрямованих на забезпечення якості ПЗ. Розроблена онтологія моделює процес життєвого циклу ПЗ – забезпечення якості ПЗ. Але забезпечення якості відбувається лише в кінці життєвого циклу ПЗ для готового програмного коду. Сьогодні якість ПЗ трактується як його здатність задовольнити потреби замовника при використанні за певних умов, тому вся необхідна інформація щодо потреб замовника повинна бути закладена вже у специфікації вимог до ПЗ, тобто вже на основі специфікації можна оцінити достатність інформації для подальшого досягнення якості ПЗ. Підхід, запропонований у [45], не передбачає забезпечення якості ПЗ на ранніх етапах життєвого циклу, зокрема, не передбачає оцінювання достатності інформації щодо нефункційних характеристик-складових якості ПЗ у специфікаціях вимог до ПЗ.

Ще однією перевагою використання онтологій є можливість доступу, розуміння та аналізу інформації інтелектуальними агентами [1].

Інтелектуальний агент (ІА) (англ. Intelligent agent) – система, котра спостерігає за навколишнім середовищем, взаємодіє з ним, а її по-ведінка інтелектуальна в тому сенсі, що агент розуміє суть власних дій, і ці дії скеровані на досягнення певної мети [124, 125]. Таким агентом може виступати програмна система, бот, сервіс. Інтелектуальний агент використовує під час свого функціонування інформацію, отриману з навколишнього середовища, аналізує її, зіставляючи з уже відомими йому фактами і, на основі результатів аналізу, приймає рішення про подальші дії.

Розв'язанню задачі розроблення інтелектуальних агентів на основі онтологічного підходу присвячено ряд досліджень – таблиця 1.3.

Таблиця 1.3

Відомі інтелектуальні агенти для галузі інженерії програмного забезпечення на базі онтологічного підходу

Інтелектуальні агенти (ІА) на основі онтологічного підходу для галузі інженерії ПЗ	**Автор(и)**
ІА на базі онтологічного підходу для усунення невизначеності у вимогах до ПЗ та покращення спілкування зацікавлених сторін	K.Ossowska та інші [48]
Уніфіковані методи побудови інтелектуальних агентів планування діяльності з використанням онтологічного підходу з метою підвищення ефективності процесів функціонування таких систем	В. Литвин та інші [126]
Багатоагентні системи, в яких агенти приймають рішення на основі знань, взятих з онтологій., що дозволяє інтегрувати платформи агентів з семантичними веб-даними та онтологіями	A. Freitas та інші [46]
Застосування онтологій для агентно-орієнтованої програмної інженерії, пропонують інструмент, який використовує існуючі онтологічні конструкції для створення програмного коду, а також експериментально підтверджують переваги застосування агентів на основі онтологій для галузі інженерії ПЗ	A. Freitas та інші [47]

ІА на базі онтологічного підходу для мінімізації існуючої семантичної невизначеності при розробленні специфікації вимог до ПЗ природньою (іспанською) мовою, для автоматичного отримання основних елементів специфікації та для автоматичної побудови діаграми цілей	L. A. Lezcano-Rodriguez та інші [127]
Методологічний підхід до інженерних систем, які базуються на інтелектуальних агентах на основі онтологічного підходу	V. Hilaire та інші [128]
Онтологічні агентно-орієнтовані моделі для формалізації первинних вимог до ПЗ з метою зниження витрат на прикладі розроблення додатків для Ambient Assisted Living для пацієнтів із хворобою Паркінсона	I. Garcia-Magarino та інші [49]
Завдання-орієнтована архітектура на основі агентно-орієнтованої парадигми та онтологічного дизайну для систем підтримки прийняття рішень (на прикладі клінічної системи для відділення невідкладної допомоги), яка дозволяє ітеративно переносити функційні вимоги в архітектурні компоненти	S. Wilk та інші [129]
Основа для формального подання та перевірки вимог та забезпечення функційної правильності обмежених ресурсами контекстних систем критичного застосування у вигляді інтелектуальних агентів на основі онтологічного підходу	A. Rakib та інші [50]
Концептуальний підхід програмної інженерії, орієнтований на підтримку цілей продуктивності та якості під час розробки програмних систем	O. Meyer та інші [130]
Поліпшення управління ризиками в програмних проєктах на базі інтелектуальних агентів та нечітких систем	C. De Olveira та інші [131]
Агентно-орієнтоване програмування та агентно-орієнтоване програмне забезпечення, яке має важливе значення для подолання кризи програмного забезпечення	H. Yan [132]

Агентно-орієнтована інтелектуальна система навчання, яка використовує функціональні точки як знання предметної галузі та виконує аналіз функціональних точок у середовищі, що забезпечує візуалізацію, негайний зворотний зв'язок, інтерактивну та керовану допомогу	A. Rahman та інші [133]
Інтерактивні, агентно-орієнтовані та орієнтовані на користувача підходи, що підтримують розробника на початку та під час концептуального проєктування програмного забезпечення (концептуальне проєктування програмного забезпечення є складним завданням для програмних інженерів)	C. Simons та інші [134]
Багатошаровий агентно-орієнтований підхід для вирішення питання впровадження правил дизайну бізнес-додатків протягом життєвого циклу програмного забезпечення. Структура програм, описана в цьому документі, охоплює та роз'яснює ключові питання проєктування бізнес-додатків шляхом розгортання аспектно-орієнтованих та інтелектуальних агентів, які можуть навчитися та адаптуватися до змін навколишнього середовища, щоб забезпечити правила проєктування бізнес-додатків протягом усього життєвого циклу програмного забезпечення.	F. Akkawi та інші [135]
Приклади застосування технологій інтелектуальних систем у сучасній системі управління життєвим циклом товару (PLM). Сучасне програмне забезпечення PLM включає інтелектуальні агенти, які автоматизують розробку продукту. Зокрема, автори пропонують новий підхід, що базується на онтології, семантичному вебі та концепції агента, для автоматизації розробки нового продукту	В. Карасев та інші [136], A. Abadi та інші. [137]
Основа для використання програмних агентів протягом життєвого циклу ПЗ. Серед інших завдань агенти могли б допомогти в діяльності із забезпечення якості ПЗ, управління проєктами та технічного обслуговування	T. Philip [138]

Для проведення оцінювання достатності інформації щодо якості у специфікації вимог до ПЗ слід зробити семантичний аналіз специфікації на предмет пошуку наявних атрибутів та/або показників якості. Існує чимало різних методів семантичного аналізу специфікацій вимог до ПЗ [56, 69] – таблиця 1.4.

Таблиця 1.4

Відомі методи семантичного аналізу вимог до програмного забезпечення

Метод (підхід) семантичного аналізу вимог до ПЗ	**Автор(и)**
Метод автоматичної допомоги розробникам шляхом трансформації вимог до природної мови за допомогою UML-діаграм діяльності та послідовності	S. Gulia та інші [139]
Метод перетворення специфікацій натуральної мови у формальні моделі, придатні для використання під час розробки інформаційних систем	M. Selway та інші [140]
Методологія, що складається з чотирьох процесів: аналіз вимог, складання карти вимог із використанням матриці, додавання вимог до шаблону специфікації та перевірка сторонніх розробників	S. W. Ali та інші [141]
Механізм автоматизації відображення функціональних вимог у формальних поданнях за допомогою позначення семантичної ролі	T. Diamantopoulos та інші [142]
Підхід до автоматичного видобування семантичної інформації із специфікацій вимог до програмного забезпечення шляхом поєднання методів позначення семантичної ролі та моделювання знань домену	Y. Wang [143]
Технологія, вхідними даними якої є артефакти (вимоги) природною мовою, і яка автоматично виявляє відповідні речення безпеки в артефактах та класифікує їх відповідно до цілей безпеки	M. Riaz та інші [144]

Підхід до підтримання процесу інженерії вимог у процесі розробки ПЗ за допомогою онтології, яка розроблена з урахуванням особливостей методології Scrum	M. Murtazina та інші [145]
Метод налаштування та створення комбінованого аналізатора для обробки та аналізу специфікацій природної мови, що поєднує переваги формальних аналізаторів, які використовуються для обробки описів, із жорстко визначеним синтаксисом (наприклад, вихідний код), та аналітиків, які розроблені для роботи з природними мовами, які добре розуміють вільний текст	F. Iwama та інші [146]
Методологія та інструмент QuARS Requirements Analysis Tool для систематичного та автоматичного аналізу природомовних вимог з метою автоматичного виявлення потенційних мовних дефектів	S. Gnesi та інші [147]
Онтологічний підхід до автоматизованого тестування та вимірювання вимог до програмного забезпечення, який використовується для виявлення невідповідностей, наслідків та недоліків вимог до ПЗ	K. Siegemund [148]
Прототип семантичної системи, який використовується для видобування вимог за допомогою напівформального подання	S. Farfeleder та інші [149]
Підхід, заснований на загальній моделі вимог користувачів до інтеграції різнорідних вимог за допомогою онтологій	A. Mustafa та інші [150]
Модель інтеграції веб-ресурсів передбачає декомпозицію процесу на інтеграцію інформації, синтаксис текстового вмісту, семантику та структуру для різнорідних веб-ресурсів в інтелектуальних інформаційних системах	J. Su та інші [151]

Проведений аналіз онтологічних моделей, методів та засобів для оцінювання початкових етапів життєвого циклу ПЗ показав, що вони не розв'язують задачу кількісного оцінювання рівня відпрацювання початкових етапів життєвого циклу ПЗ на основі аналізу специфікацій (зокрема, оцінювання достатності інформації у специфікації вимог до ПЗ). Невирішеність цієї задачі обумовлює необхідність розроблення методів та інформаційної технології оцінювання достатності інформації у специфікаціях вимог до ПЗ.

Всі розглянуті методи семантичного аналізу вимог до ПЗ не забезпечують автоматизованого пошуку у вимогах атрибутів та показників, необхідних для визначення нефункційних характеристик-складових якості та метрик ПЗ. Невирішеність цієї задачі обумовлює необхідність розроблення методу діяльності інтелектуального агента для семантичного аналізу природомовних вимог до ПЗ.

1.4. Висновки. Постановка задачі

Проведене дослідження показало, що предметні галузі, для яких розробляються інформаційні технології, суттєво впливають на методи опрацювання інформації, тому виправданим є підхід, що базується на дослідженні характеристик та особливостей предметних галузей та розроблення нових інформаційних технологій саме для конкретних галузей. Особливої уваги у напрямку розроблення та впровадження ефективних інформаційних технологій на сьогодні потребує галузь інженерії ПЗ

Проведене дослідження впливу інформації у специфікації вимог на успішність ПЗ показало, що чинники якості та успішності сучасних програмних систем є менш залежними від написання програмного коду, але суттєво залежать від формування вимог. Ризиками недостатньо відпрацьованого етапу формування вимог є недотримання термінів проєктів та фінансові перевитрати, що можуть призвести до закриття проєкту, а то й розпаду софтверної компанії внаслідок її фінансової нестабільності.

Як доведено вище, саме в кінці етапу проєктування можна й варто виявляти та усувати до 55% всіх помилок майбутнього ПЗ. Дефекти, внесені на етапах формування вимог і проєктування, варто виявляти та усувати до того, як вони почнуть впливати на результати більш пізніх етапів життєвого циклу. Якість та успішність реалізації програмного проєкту суттєво залежать від специфікації вимог до ПЗ.

На сьогодні, ключовими є 2 підходи до оцінювання ПЗ – за моделлю SQuaRE (стандарт ISO 25010:2011) та на основі результатів метричного аналізу.

Оцінювання ПЗ за стандартом ISO 25010:2011 відбувається так: на основі атрибутів якості, зазначених у ISO 25023:2016, оцінюються підхарактеристики та характеристики якості, які, в свою чергу, надають комплексну оцінку якості ПЗ.

Оцінювання ПЗ на основі використання результатів метричного аналізу відбувається так: на основі показників розраховуються значення метрик, які, в свою чергу, надають комплексну оцінку якості та складності програмного проєкту і прогноз якості та складності майбутнього ПЗ.

Аналіз відомих підходів, методів, інформаційних технологій, інструментів для аналізу вимог до програмного забезпечення та оцінки достатності інформації у специфікації вимог та інтелектуальних агентів на основі онтологічного підходу, показав, що вони не розв’язують проблему оцінювання ранніх етапів життєвого циклу ПЗ. Крім цього, всі вони належать до різних методологічних підходів і не інтегруються між собою, тобто наразі відсутні інтелектуальні інформаційно-аналітичні технології для оцінювання ранніх етапів життєвого циклу ПЗ, що є однією з причин проблем у галузі інженерії ПЗ.

Актуальність проблеми аналізу достатності інформації щодо якості у специфікаціях вимог до ПЗ, а також відсутність моделей, методів та засобів оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ обумовлює необхідність розроблення інтелектуальних інформаційно-аналітичних технологій для підвищення якості програмного забезпечення шляхом оцінювання достатності інформації на ранніх етапах життєвого циклу.

Потреба у автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення, а також відсутність інформаційної технології для оцінювання початкових етапів життєвого циклу ПЗ створюють *актуальну науково-прикладну задачу*, одним із шляхів розв'язання якої є розроблення агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу.

Розв'язання окресленої науково-прикладної задачі потребує комплексних досліджень за наступними напрямками:

1) моделювання діяльності інтелектуальних агентів на основі онтологічного підходу для автоматизованого аналізу специфікацій вимог до ПЗ, а також для оцінювання спецтфікацій вимог до ПЗ;

2) розроблення методів діяльності інтелектуальних агентів на основі онтологічного підходу для автоматизованого семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення, а також для оцінювання початкових етапів життєвого циклу програмного забезпечення;

3) проєктування та реалізація агентно-орієнтованої інформаційної технології для оцінювання початкових етапів життєвого циклу ПЗ;

4) виконаня практичного впровадження отриманих результатів при оцінюванні початкових етапів життєвого циклу ПЗ.

Метою дисертаційного дослідження є автоматизація оцінювання початкових етапів життєвого циклу програмного забезпечення шляхом розроблення агентно-орієнтованої інформаційної технології на основі онтологічного підходу.

Об'єкт дослідження – процеси оцінювання поачткових етапів життєвого циклу програмного забезпечення.

Предмет дослідження – моделі, методи та засоби агентно-орієнтованої інформаційної технології оцінювання поачткових етапів життєвого циклу програмного забезпечення.

РОЗДІЛ 2.
МОДЕЛЮВАННЯ ДІЯЛЬНОСТІ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ

2.1. Моделі нефункційних характеристик-складових якості програмного забезпечення

Найбільш використовуваною моделлю для оцінки якості ПЗ є модель якості зі стандарту ISO 25010 [98]. Модель ISO 25010 пропонує оцінити якість ПЗ як функцію від восьми характеристик, кожна з яких є функцією декількох підхарактеристик (всього 31 підхарактеристика), але підхарактеристики, в свою чергу, є функціями від декількох атрибутів. Аналіз стандарту ISO 25023 [102] дав можливість визначити залежність підхарактеристик якості від 203 атрибутів, в тому числі 138 різних атрибутів.

Відповідно до стандарту ISO 25010:2011, такі нефункційні характеристики ПЗ, як функційна придатність (F_s), ефективність (P_e), зручність використання (U_b), надійність (R_b), сумісність (C_b), захищеність (S_{cr}), супроводжуваність (M_b), можливість переносу (P_b) є функціями від декількох підхарактеристик, тоді представиимо кожну з вищевказаних характеристик як функцію від підхарактеристик [1]:

$$F_s = f_1(F_{Com}, F_{Cor}, F_{Appr}), \tag{2.1}$$

де F_{Com} – функційна повнота, F_{Cor} – функційна коректність, F_{Appr} – функційна доцільність;

$$P_e = f_2(T_b, R_u, C_c), \tag{2.2}$$

де T_b – поведінка у часі, R_u – поведінка ресурсів, C_c – ємність (місткість);

$$U_b = f_3(A_r, L_b, O_b, U_{ep}, U_{ia}, A_b), \tag{2.3}$$

де A_r – розпізнавання доцільності, L_b – можливість вивчення, O_b – керованість, U_{ep} – захист від помилок користувача, U_{ia} – естетичність інтерфейсу користувача, A_b – доступність;

$$R_b = f_4(M_{at}, A_{vb}, F_t, R_{cv}), \tag{2.4}$$

де M_{at} – зрілість, A_{vb} – наявність (доступність), F_t – відмовостійкість, R_{cv} – відновлюваність;

$$C_b = f_5(C_e, I_b), \tag{2.5}$$

де C_e – співіснування, I_b – взаємодія;

$$S_{cr} = f_6(C_{onf}, I_{nt}, N_r, A_{cb}, A_{uth}), \tag{2.6}$$

де C_{onf} – конфіденційність, I_{nt} – цілісність, N_r – невідхилюваність, A_{cb} – підзвітність, A_{uth} – ідентичність;

$$M_b = f_7(M_{od}, R_{ub}, A_{nb}, M_{dfb}, T_{sb}), \tag{2.7}$$

де M_{od} – модульність, R_{ub} – повторне використання, A_{nb} – аналізованість, M_{dfb} – модифікованість, T_{sb} – тестованість;

$$P_b = f_8(A_{db}, I_{nb}, R_{pb}), \tag{2.8}$$

де A_{db} – адаптованість, I_{nb} – можливість інсталяції, R_{pb} – можливість заміни.

Тоді *множини підхарактеристик нефункційних характеристик ПЗ* мають вигляд:

$A = \{F_{Com}, F_{Cor}, F_{Appr}\}$ – множина підхарактеристик функційної придатності;

$B = \{T_b, R_u, C_c\}$ – множина підхарактеристик ефективності;

$C = \{A_r, L_b, O_b, U_{ep}, U_{ia}, A_b\}$ – множина підхарактеристик зручності використання,

$D = \{M_{at}, A_{vb}, F_t, R_{cv}\}$ – множина підхарактеристик надійності;

$E = \{ C_e, I_b \}$ – множина підхарактеристик сумісності;

$F = \{ C_{onf}, I_{nt}, N_r, A_{cb}, A_{uth} \}$ – множина підхарактеристик захищеності;

$G = \{ M_{od}, R_{ub}, A_{nb}, M_{dfb}, T_{sb} \}$ – множина підхарактеристик супроводжуваності;

$H = \{ A_{db}, I_{nb}, R_{pb} \}$ – множина підхарактеристик можливості переносу.

Водночас, кожна підхарактеристика нефункційних характеристик ПЗ є функцією певних атрибутів, описаних у стандарті ISO 25023:2016. Залежності підхарактеристик нефункційних характеристик від атрибутів представимо в наступному вигляді:

$$F_{Com} = \varphi_1 (N_{of}, F_{icn}, F_{aq}, F_{ic}), \tag{2.9}$$

де N_{of} – кількість функцій, F_{icn} – повнота функційної реалізації, F_{aq} – функційна адекватність, F_{ic} – покриття функційної реалізації;

$$F_{Cor} = \varphi_2 (O_t, N_{ic}, N_{di}, C_a, P_c), \tag{2.10}$$

де O_t – час роботи, N_{ic} – кількість неточних обчислень, з якими стикаються користувачі, N_{di} – кількість елементів даних, C_a – обчислювальна точність, P_c – точність (відповідність);

$$F_{Appr} = \varphi_3 (O_t, N_{of}, F_{icn}, F_{aq}, F_{ic}, P_c); \tag{2.11}$$

$$T_b = \varphi_4 (O_t, N_{mot}, R_t, N_{oe}, T_{nt}, T_{skt}, M_{athr}), \tag{2.12}$$

де N_{mot} – кількість задач, R_t – час реакції, N_{oe} – кількість оцінок, T_{nt} – час обробки, T_{skt} – час задачі, M_{athr} – середнє значення пропускної здатності;

$$R_u = \varphi_5 \begin{pmatrix} O_t, N_{ofl}, N_{oe}, N_{iore}, U_{wt}, N_{mre}, N_{tre}, \\ T_{cc}, I_{ou}, N_{olcd}, I_{oll}, M_{mu}, M_{tu}, M_{ote} \end{pmatrix}, \tag{2.13}$$

де $N_{of}l$ – кількість відмов, N_{iore} – кількість помилок введення-виведення, U_{wt} – час очікування користувача під час використання пристрою введення-виведення, N_{mre} – кількість помилок пам’яті, N_{tre} – кількість помилок передачі даних, T_{cc} –

потужність (ємність) передачі даних, I_{ou} – кількість буферів при використанні введення-виведення, N_{olcd} – безпосередня кількість рядків коду, I_{oll} – ліміт завантаження пристроїв введення-виведення, M_{mu} – максимум використовуваної пам'яті, M_{tu} – максимум використовуваної передачі даних, M_{ote} – середня поява помилки передачі;

$$C_c = \varphi_6(N_{di}, N_{cu}, C_{bw}, M_{athr}, S_{db}), \tag{2.14}$$

де N_{cu} – кількість одночасних користувачів, C_{bw} – ширина смуги комунікації, S_{db} – розмір бази даних;

$$A_r = \varphi_7(N_{of}, N_{ott}, N_{iodi}, C_{nd}, F_{ua}, U_{aio}), \tag{2.15}$$

де N_{ott} – кількість посібників, N_{iodi} – кількість елементів даних введення-виведення, C_{nd} – повнота описів, F_{ua} – зрозумілість функціоналу, U_{aio} – зрозумілість входів та виходів;

$$L_b = \varphi_8(N_{of}, O_t, E_{fl}, N_{mot}, H_{fq}, E_{udhs}, H_{aa}, C_{udh}), \tag{2.16}$$

де E_{fl} – простота вивчення функціоналу, H_{fq} – частота звертань до довідки, E_{udhs} – ефективність документації користувача та системи допомоги, H_{aa} – доступність довідки, C_{udhf} – повнота документації користувача та довідкового фонду;

$$O_b = \varphi_9\begin{pmatrix} N_{of}, O_t, E_{cr}, N_{sf}, N_{uec}, N_{ac}, N_{iore}, \\ N_{op}, N_{iwccvd}, N_{mi}, N_{ie}, P_{ha}, N_{eum} \end{pmatrix}, \tag{2.17}$$

де E_{cr} – частота корекції помилок, N_{sf} – кількість екранів або форм, N_{uec} – кількість помилок або змін користувача, N_{ac} – кількість спроб налаштування, N_{op} – кількість операцій, N_{iwccd} – кількість елементів, які можна перевірити на наявність дійсних даних, N_{mi} – кількість виконаних повідомлень, N_{ie} – кількість елементів інтерфейсу, P_{ha} – фізична доступність, N_{eum} – кількість легко зрозумілих повідомлень;

$$U_{ep} = \varphi_{10}\begin{pmatrix} N_{urs}, N_{uec}, O_{tpdo}, N_{ouheo}, N_{iewusc}, \\ N_{acie}, N_{ecwusc}, T_{nect}, N_{fiuet}, T_{nfrtc}, T_{niop} \end{pmatrix}, \quad (2.18)$$

де N_{urs} – кількість невдало відновлених ситуацій, O_{ptdo} – час роботи під час спостережень, N_{ouheo} – кількість випадків операційних помилок користувача, N_{iewusc} – кількість вхідних помилок, які користувач успішно виправляє, N_{acie} – кількість спроб коригування вхідних помилок, N_{ecwusc} – клькість помилкових умов, які користувач успішно виправляє, T_{nect} – загальна кількість перевірених помилкових умов, N_{fiuet} – кількість функцій, виконаних з толерантністю до помилки користувача, T_{nfrtc} – загальна кількість функцій, що вимагають можливості толерантності, T_{niop} – загальна кількість некоректних шаблонів операцій;

$$U_{ia} = \varphi_{11}(N_{ie}, N_{ige}, D_{ipu}, D_{isu}, D_{ea}, D_{rwmu}), \quad (2.19)$$

де N_{ige} – кількість графічних елементів інтерфейсу, D_{ipu} – ступінь збільшення задоволення користувача, D_{isu} – ступінь збільшення задоволеності потреб користувача, D_{ea} – ступінь ергономічної привабливості, D_{rwmu} – ступінь використання метафор реального світу;

$$A_b = \varphi_{12}(E_{wscbuusd}, E_{wusd}, F_{frusd}, S_{usd}, P_{psa}), \quad (2.20)$$

де $E_{wscbuusd}$ – обсяг, до якого програмним забезпеченням можуть користуватись користувачі з обмеженими можливостями, E_{wusd} – ефективність роботи користувачів з обмеженими можливостями, F_{frusd} – свобода від ризику для користувачів із обмеженими можливостями, S_{usd} – задоволення потреб користувачів з обмеженими можливостями, P_{psa} – наявність властивостей, які підтримують доступність;

$$M_{at} = \varphi_{13}\begin{pmatrix} O_t, N_{oft}, N_{ofl}, P_s, N_{tc}, N_{rf}, N_{cf}, \\ F_{datc}, F_{rn}, F_{rl}, M_{tbf}, T_{my}, E_{lfd}, F_{dy} \end{pmatrix}, \quad (2.21)$$

де N_{oft} – кількість збоїв, P_s – розмір продукту, N_{tc} – кількість тестових випадків, N_{rf} – кількість фіксованих відмов, N_{cf} – кількість усунутих (виправлених) збоїв, F_{datc} – щільність відмов відносно до тестових випадків, F_{rn} – роздільна здатність відмов, F_{rl} – усунення збоїв, M_{tbf} – середній час між відмовами, T_{my} – випробувальний термін, E_{lfd} – орієнтовна щільність прихованих збоїв, F_{dy} – щільність збоїв;

$$A_{vb} = \varphi_{14}(O_t, T_{tdwsis}, N_{ob}, T_{dt}), \tag{2.22}$$

де T_{tdwsis} – загальний час, протягом якого програма перебуває у стані зростання, N_{ob} – кількість спостережуваних несправностей, T_{dt} – загальна тривалість простоїв;

$$F_t = \varphi_{15}(N_{ofl}, N_{tc}, N_{bd}, N_{of}, N_{io}), \tag{2.23}$$

де N_{bd} – кількість несправностей, N_{io} – кількість недозволених операцій;

$$R_{cv} = \varphi_{16}(O_t, N_{bd}, T_{tr}, D_t, N_{rs}, N_{rn}, R_{ay}), \tag{2.24}$$

де T_{tr} – час ремонту, D_t – тривалість простою, N_{rs} – кількість перезапусків, N_{rn} – кількість відновлень, R_{ay} – відновлюваність (можливість перезапуску);

$$C_e = \varphi_{17}(O_t, N_{ofl}, N_{of}, N_{di}); \tag{2.25}$$

$$I_b = \varphi_{18}(O_t, N_{dfrt}, N_{dfbe}, N_{ip}, D_{eay}), \tag{2.26}$$

де N_{dfrt} – кількість форматів даних, пов’язаних інструментом, N_{dfbe} – кількість форматів даних для обміну, N_{ip} – кількість протоколів інтерфейсу, D_{eay} – обмінність даних;

$$C_{onf} = \varphi_{19}(O_t, N_{io}, N_{tc}, N_{idc}, N_{di}, N_{at}, N_{cr}, A_{ca}, N_{diced}, N_{dibred}), \tag{2.27}$$

де N_{idc} – кількість випадків пошкодження даних, N_{at} – кількість типів доступу, N_{cr} – кількість контрольованих вимог, A_{ca} – керованість доступу, N_{diced} – кількість

елементів даних, які правильно зашифровані та розшифровані, N_{dibred} – кількість елементів даних, які потребують шифрування та розшифрування;

$$I_{nt} = \varphi_{20}(O_t, N_{io}, N_{tc}, N_{idc}, N_{di}, N_{at}, N_{cr}, A_{ca}); \quad (2.28)$$

$$N_r = \varphi_{21}(N_{epuds}, N_{ernrp}), \quad (2.29)$$

де N_{epuds} – кількість подій, оброблених за допомогою цифрового підпису, N_{ernrp} – кількість подій, що вимагають властивості невідхилення (неприпинення);

$$A_{cb} = \varphi_{22}(N_{asdrsl}, N_{aao}), \quad (2.30)$$

де N_{asdrsl} – кількість доступів до системи та даних, записаних у системний журнал, N_{aao} – кількість дійсно набутих доступів;

$$A_{uth} = \varphi_{23}(N_{pam}), \quad (2.31)$$

де N_{pam} – кількість наданих методів перевірки автентичності;

$$M_{od} = \varphi_{24}(O_t, N_{ofl}, N_{rf}, N_{mm}, N_v, N_{of}, N_m), \quad (2.32)$$

де N_{mm} – кількість внесених змін, N_v – кількість змінних, N_m – кількість модулів;

$$R_{ub} = \varphi_{25}(F_{cy}, N_{fcy}, V_{rn}, A_{ay}, T_{ay}, C_{ra}), \quad (2.33)$$

де F_{cy} – функційна спільність, N_{fcy} – нефункційна спільність, V_{rn} – міцініть варіабельності, A_{ay} – застосовність, T_{ay} – пристосованість, C_{ra} – заміна компонентів;

$$A_{nb} = \varphi_{26}(N_{ofl}, N_{di}, E_{rt}, N_{irbl}, N_{dfr}, A_{tc}), \quad (2.34)$$

де E_{rt} – час помилки, N_{irbl} – кількість елементів, необхідних для запису в журналі, N_{dfr} – кількість необхідних діагностичних функцій, A_{tc} – можливість аудиту сліду;

$$M_{dfb} = \varphi_{27}(O_t, N_{rv}, N_{rf}, E_{rt}, N_{of}, C_{cca}, N_{tcpbm}, N_{tspam}), \quad (2.35)$$

де N_{rv} – кількість переглянутих версій, C_{cca} – можливість управління змінами, N_{tcpbm} – гількість проблем протягом певного періоду до модифікації, N_{tspam} – гількість проблем за той же період після модифікації;

$$T_{sb} = \varphi_{28}(O_t, N_{tc}, N_{rf}, N_{btfr}, N_{tdos}, N_{cp}), \tag{2.36}$$

де N_{btfr} – кількість необхідних вбудованих тестових функцій, N_{tdos} – кількість тестових залежностей на інших системах, N_{cp} – кількість контрольних точок;

$$A_{db} = \varphi_{29}\begin{pmatrix} N_{of}, O_t, N_{oft,} P_{uf}, N_{di}, N_{ds}, \\ A_{ads}, H_{ea}, S_{ea}, N_{oftwnca}, T_{nfwtde} \end{pmatrix}, \tag{2.37}$$

де P_{uf} – портативна дружність користувача, N_{ds} – кількість структур даних, A_{ads} – адаптивність структур даних, H_{ea} – адаптивність апаратного оточення, S_{ea} – адаптивність програмного оточення, $N_{oftwnca}$ – кількість операційних функцій, завдання яких не були виконані або неадекватні, T_{nfwtde} – загальна кількість функцій, які були випробувані за різних умов;

$$I_{nb} = \varphi_{30}(N_{oft}, N_{so}, N_{is}, E_{oi}), \tag{2.38}$$

де N_{so} – кількість операцій налаштування, N_{is} – кількість кроків інсталяції, E_{oi} – простота інсталяції;

$$R_{pb} = \varphi_{31}(N_{of}, N_{di}, N_{et}), \tag{2.39}$$

де N_{et} – кількість сутностей.

Тоді *множини атрибутів для підхарактеристик нефункційних характеристик ПЗ* мають вигляд:

$I = \{N_{of}, F_{icn}, F_{aq}, F_{ic}\}$ – множина атрибутів для функційної повноти;

$J = \{O_t, N_{ic}, N_{di}, C_a, P_c\}$ – множина атрибутів для функційної коректності;

$K = \{O_t, N_{of}, F_{icn}, F_{aq}, F_{ic}, P_c\}$ – множина атрибутів для функційної доцільності;

$L = \{ O_t, N_{mot}, R_t, N_{oe}, T_{nt}, T_{skt}, M_{athr} \}$ – множина атрибутів для поведінки у часі;

$M = \{ O_t, N_{ofl}, N_{oe}, N_{iore}, U_{wt}, N_{mre}, N_{tre}, T_{cc}, I_{ou}, N_{olcd}, I_{oll}, M_{mu}, M_{tu}, M_{ote} \}$ – множина атрибутів для поведінки ресурсів;

$N = \{ N_{di}, N_{cu}, C_{bw}, M_{athr}, S_{db} \}$ – множина атрибутів для ємності (місткості);

$O = \{ N_{of}, N_{ott}, N_{iodi}, C_{nd}, F_{ua}, U_{aio} \}$ – множина атрибутів для розпізнавання доцільності;

$P = \{ N_{of}, O_t, E_{fl}, N_{mot}, H_{fq}, E_{udhs}, H_{aa}, C_{udhf} \}$ – множина атрибутів для можливості вивчення;

$Q = \{ N_{of}, O_t, E_{cr}, N_{sf}, N_{uec}, N_{ac}, N_{iore}, N_{op}, N_{iwccvd}, N_{mi}, N_{ie}, P_{ha}, N_{eum} \}$ – множина атрибутів для керованості;

$R = \{ N_{urs}, N_{uec}, O_{tpdo}, N_{ouheo}, N_{iewusc}, N_{acie,} N_{ecwusc}, T_{nect}, N_{fiuet}, T_{nfrtc}, T_{niop} \}$ – множина атрибутів для захисту від помилок користувача;

$S = \{ N_{ie}, N_{ige}, D_{ipu}, Di_{su}, D_{ea}, D_{rwmu} \}$ – множина атрибутів для естетичності інтерфейсу;

$T = \{ E_{wscbuusd}, E_{wusd}, F_{frusd}, S_{usd}, P_{psa} \}$ – множина атрибутів для доступності;

$U = \{ O_t, N_{oft}, N_{ofl}, P_s, N_{tc}, N_{rf}, N_{cf}, F_{datc}, F_{rn}, F_{rl}, M_{tbf}, T_{my}, E_{lfd}, F_{dy} \}$ – множина атрибутів для зрілості;

$V = \{ O_t, T_{tdwsis}, N_{ob}, T_{dt} \}$ – множина атрибутів для наявності (доступності);

$W = \{ N_{ofl}, N_{tc}, N_{bd}, N_{of}, N_{io} \}$ – множина атрибутів для відмовостійкості;

$X = \{ O_t, N_{bd}, T_{tr}, D_t, N_{rs}, N_{rn}, R_{ay} \}$ – множина атрибутів для відновлюваності;

$Y = \{ O_t, N_{ofl}, N_{of}, N_{di} \}$ – множина атрибутів для співіснування;

$Z = \{ O_t, N_{dfrt}, N_{dfbe}, N_{ip}, D_{eay} \}$ – множина атрибутів для взаємодії;

$A_A = \{ O_t, N_{io}, N_{tc}, N_{idc}, N_{di}, N_{at}, N_{cr}, A_{ca}, N_{diced}, N_{dibred} \}$ – множина атрибутів для конфіденційності;

$A_B = \{ O_t, N_{io}, N_{tc}, N_{idc}, N_{di}, N_{at}, N_{cr}, A_{ca} \}$ – множина атрибутів для цілісності;

$A_C = \{ N_{epuds}, N_{ernrp} \}$ – множина атрибутів для невідхилюваності;

$A_D = \{ N_{asdrsl}, N_{aao} \}$ – множина атрибутів для підзвітності;

$A_E = \{ N_{pam} \}$ – множина атрибутів для ідентичності;

$A_F = \{ O_t, N_{of} l, N_{rf}, N_{mm}, N_v, N_{of}, N_m \}$ – множина атрибутів для модульності;

$A_G = \{ F_{cy}, N_{fcy}, V_{rn}, A_{ay}, T_{ay}, C_{ra} \}$ – множина атрибутів для повторного використання;

$A_H = \{ N_{ofl}, N_{di}, E_{rt}, N_{irbl}, N_{dfr}, A_{tc} \}$ – множина атрибутів для аналізованості;

$A_I = \{ O_t, N_{rv}, N_{rf}, E_{rt}, N_{of}, C_{cca}, N_{tcpbm}, N_{tspam} \}$ – множина атрибутів для модифікованості;

$A_J = \{ O_t, N_{tc}, N_{rf}, N_{btfr}, N_{tdos}, N_{cp} \}$ – множина атрибутів для тестованості;

$A_K = \{ N_{of}, O_t, N_{oft}, P_{uf}, N_{di}, N_{ds}, A_{ads}, H_{ea}, S_{ea}, N_{oftwnca}, T_{nfwtde} \}$ – множина атрибутів для адаптованості;

$A_L = \{ N_{oft}, N_{so}, N_{is}, E_{oi} \}$ – множина атрибутів для можливості інсталяції;

$A_M = \{ N_{of}, N_{di}, N_{et} \}$ – множина атрибутів для можливості заміни.

Запропоновані теоретико-множинні моделі нефункційних характеристик-складових якості програмного забезпечення ґрунтуються на врахуванні вимог стандартів ISO 25010:2011 та ISO 25023:2016 і забезпечують підґрунтя для вибору достатнього обсягу інформації для оцінювання нефункційних характеристик ПЗ.

2.2. Онтологічні моделі нефункційних характеристик програмного забезпечення

Враховуючи отримані множини атрибутів та підхарактеристик нефункційних характеристик ПЗ, отримані функції залежностей характеристик від підхарактеристик і підхарактеристик від атрибутів, а також відношення між поняттями онтології «залежить від», розробимо *базові (універсальні) онтологічні моделі нефункційних характеристик-складових якості ПЗ* (принципи розроблення базових онтологічних моделей представлені у [1]). Тоді:

- базова онтологічна модель функційної придатності:

$$O_{F_s} = \{\{F_s, A, I, J, K\}, "depends\ on", \{f_1, \varphi_1, \varphi_2, \varphi_3\}\} = \\ = \{\{fsa_1, ..., fsa_{19}\}, "depends\ on", \{f_1, \varphi_1, \varphi_2, \varphi_3\}\}, \quad (2.40)$$

де $fsa_1 = F_s$, $\{fsa_2, ..., fsa_4\} \in A$, $\{fsa_5, ..., fsa_8\} \in I$, $\{fsa_9, ..., fsa_{13}\} \in J$, $\{fsa_{14}, ..., fsa_{19}\} \in K$;

- базова онтологічна модель ефективності:

$$O_{P_e} = \{\{P_e, B, L, M, N\}, "depends\ on", \{f_2, \varphi_4, \varphi_5, \varphi_6\}\} = \\ = \{\{pea_1, ..., pea_{30}\}, "depends\ on", \{f_2, \varphi_4, \varphi_5, \varphi_6\}\}; \quad (2.41)$$

- базова онтологічна модель зручності використання:

$$O_{U_b} = \{\{U_b, C, O, P, Q, R, S, T\}, "depends\ on", \{f_3, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}, \varphi_{11}, \varphi_{12}\}\} = \\ = \{\{uba_1, ...uba_{56}\}, "depends\ on", \{f_3, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}, \varphi_{11}, \varphi_{12}\}\}; \quad (2.42)$$

- базова онтологічна модель надійності:

$$O_{R_b} = \{\{R_b, D, U, V, W, X\}, "depends\ on", \{f_4, \varphi_{13}, \varphi_{14}, \varphi_{15}, \varphi_{16}\}\} = \\ = \{\{rba_1, ...rba_{35}\}, "depends\ on", \{f_4, \varphi_{13}, \varphi_{14}, \varphi_{15}, \varphi_{16}\}\}; \quad (2.43)$$

- базова онтологічна модель сумісності:

$$O_{C_b} = \{\{C_b, E, Y, Z\}, "depends\ on", \{f_5, \varphi_{17}, \varphi_{18}\}\} = \\ = \{\{cba_1, ..., cba_{12}\}, "depends\ on", \{f_5, \varphi_{17}, \varphi_{18}\}\}; \quad (2.44)$$

- базова онтологічна модель захищеності:

$$O_{S_{cr}} = \{\{ S_{cr}, F, A_A, A_B, A_C, A_D, A_E \}, "depends\ on", \{ f_6, \varphi_{19}, \varphi_{20}, \varphi_{21}, \varphi_{22}, \varphi_{23} \}\} = \\ = \{\{ scra_1, ..., scra_{29} \}, "depends\ on", \{ f_6, \varphi_{19}, \varphi_{20}, \varphi_{21}, \varphi_{22}, \varphi_{23} \}\};$$

(2.45)

- базова онтологічна модель супроводжуваності:

$$O_{M_b} = \{\{ M_b, G, A_F, A_G, A_H, A_I, A_J \}, "depends\ on", \{ f_7, \varphi_{24}, \varphi_{25}, \varphi_{26}, \varphi_{27}, \varphi_{28} \}\} = \\ = \{\{ mba_1, ..., mba_{39} \}, "depends\ on", \{ f_7, \varphi_{24}, \varphi_{25}, \varphi_{26}, \varphi_{27}, \varphi_{28} \}\};$$

(2.46)

- базова онтологічна модель можливості переносу:

$$O_{P_b} = \{\{ P_b, H, A_K, A_L, A_M \}, "depends\ on", \{ f_8, \varphi_{29}, \varphi_{30}, \varphi_{31} \}\} = \\ = \{\{ pba_1, ..., pba_{22} \}, "depends\ on", \{ f_8, \varphi_{29}, \varphi_{30}, \varphi_{31} \}\}. \quad (2.47)$$

Онтології (онтологічні база знань), які розробляються за моделями (2.40)-(2.47), наповнюються на основі інформації, взятої зі стандартів ISO 25010:2011 [98], ISO 25023:2016 [102]. Базові онтології розроблені у [1] та представлені на рис. 2.1-2.8.

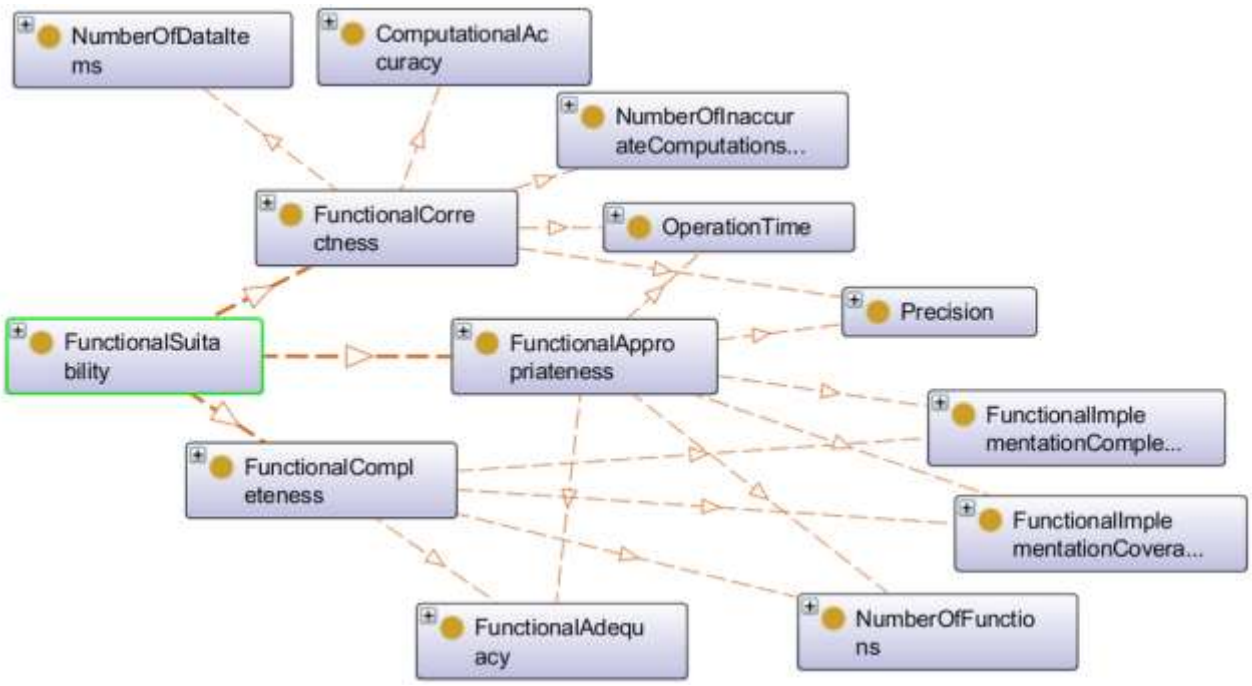

Рис 2.1 – Базова онтологія для Функційної придатності [1]

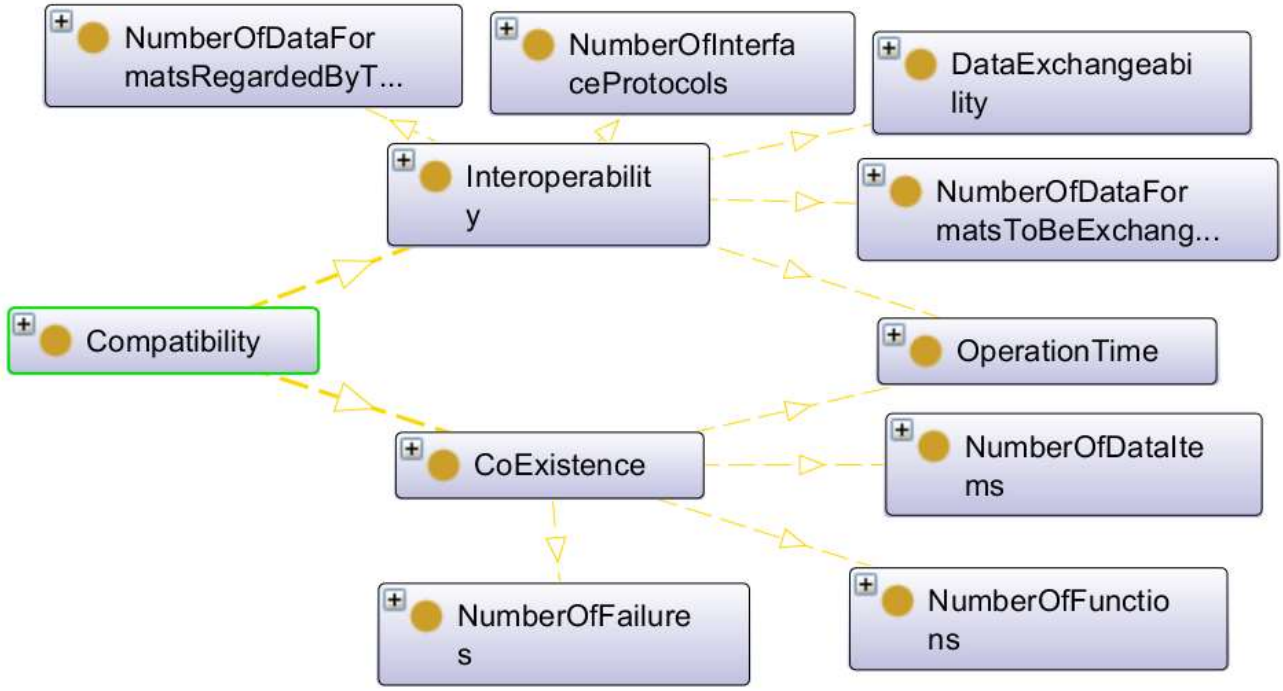

Рис. 2.2 – Базова онтологія для Сумісності [1]

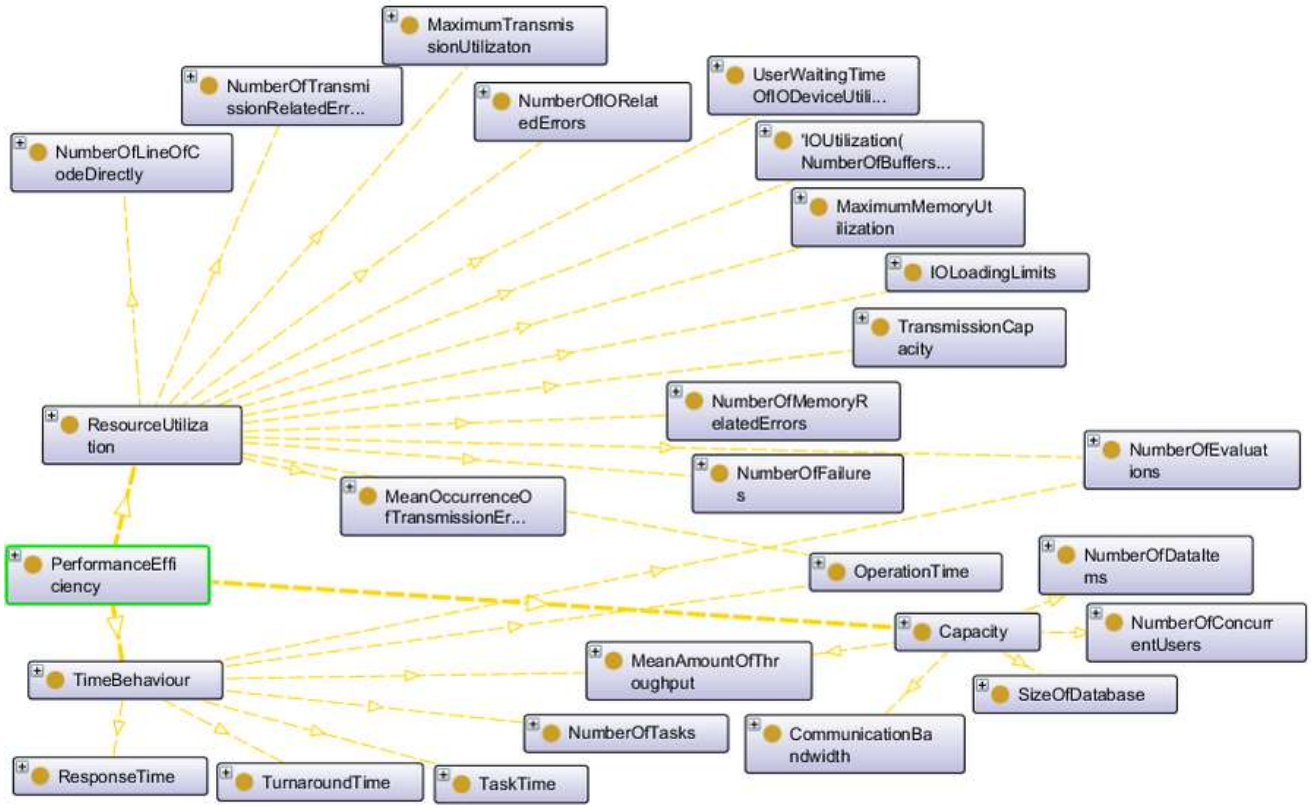

Рис. 2.3 – Базова онтологія для Ефективності [1]

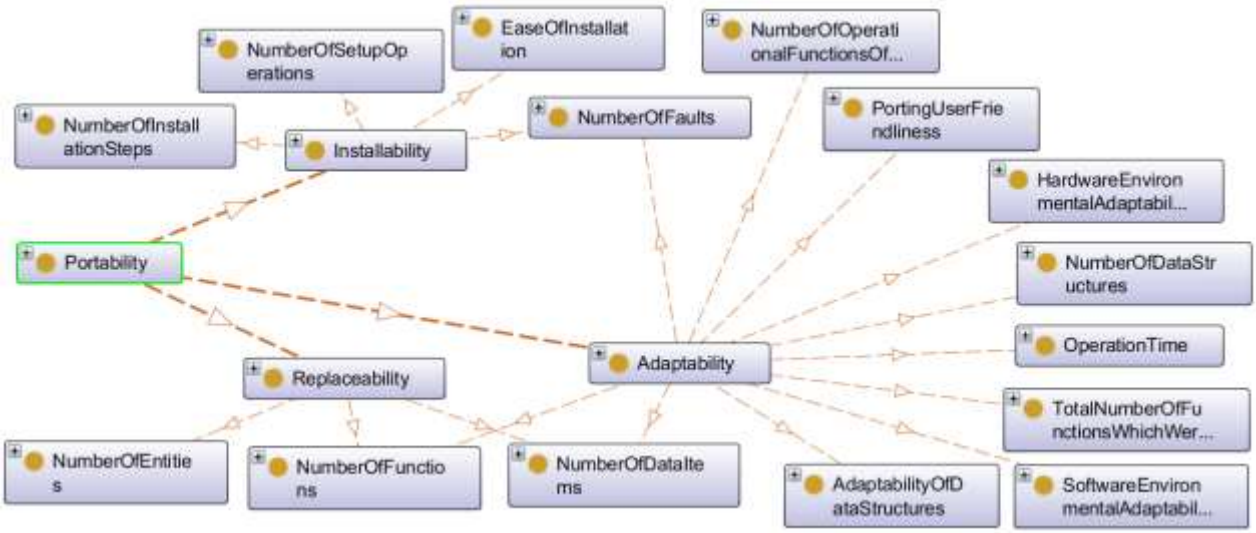

Рис. 2.4 – Базова онтологія для Можливості переносу [1]

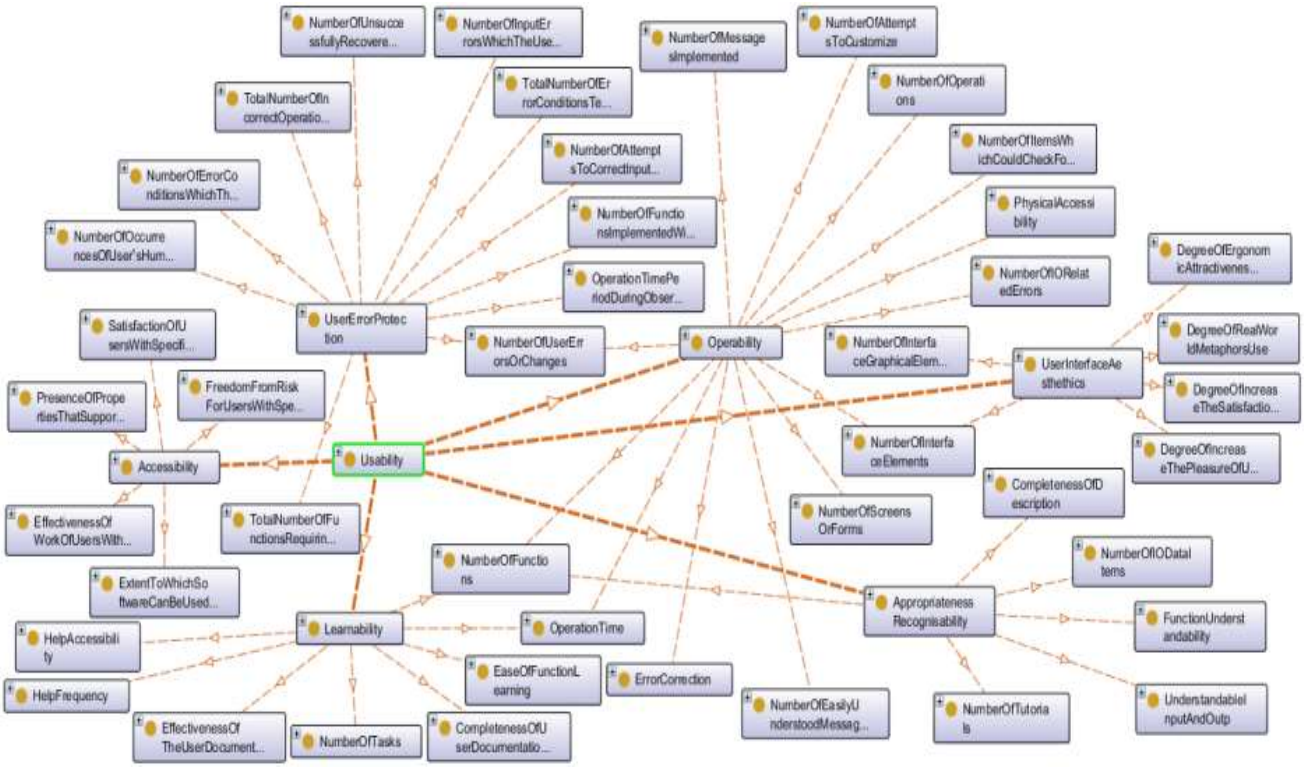

Рис. 2.5 – Базова онтологія для Зручності використання [1]

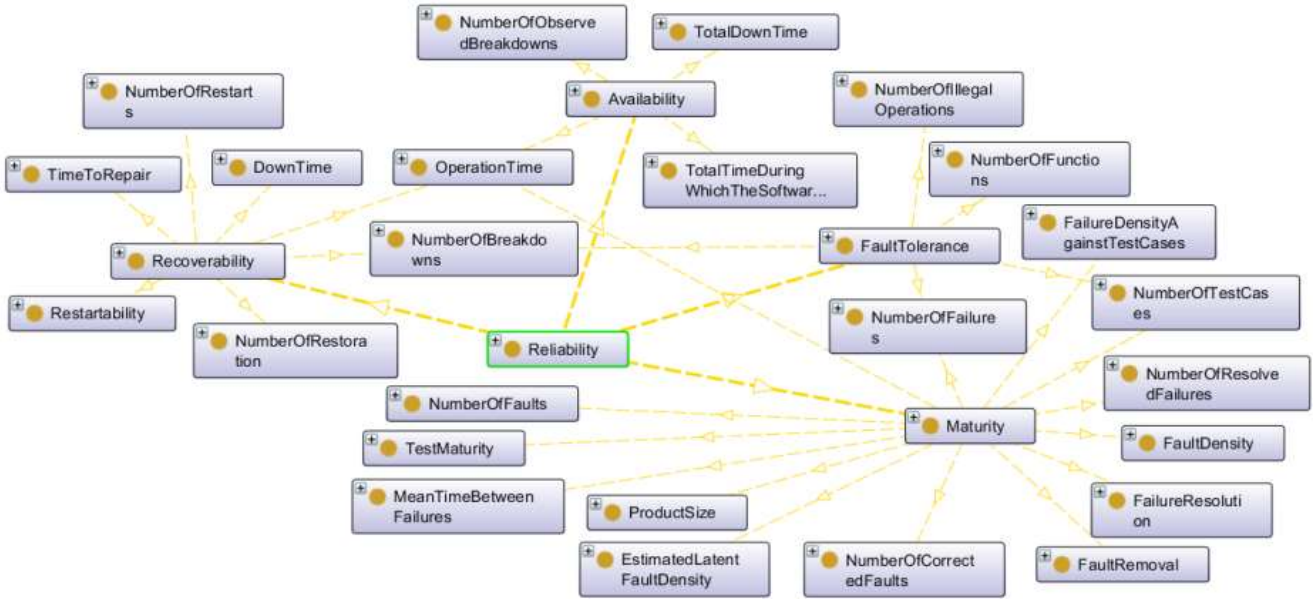

Рис 2.6 – Базова онтологія для Надійності [1]

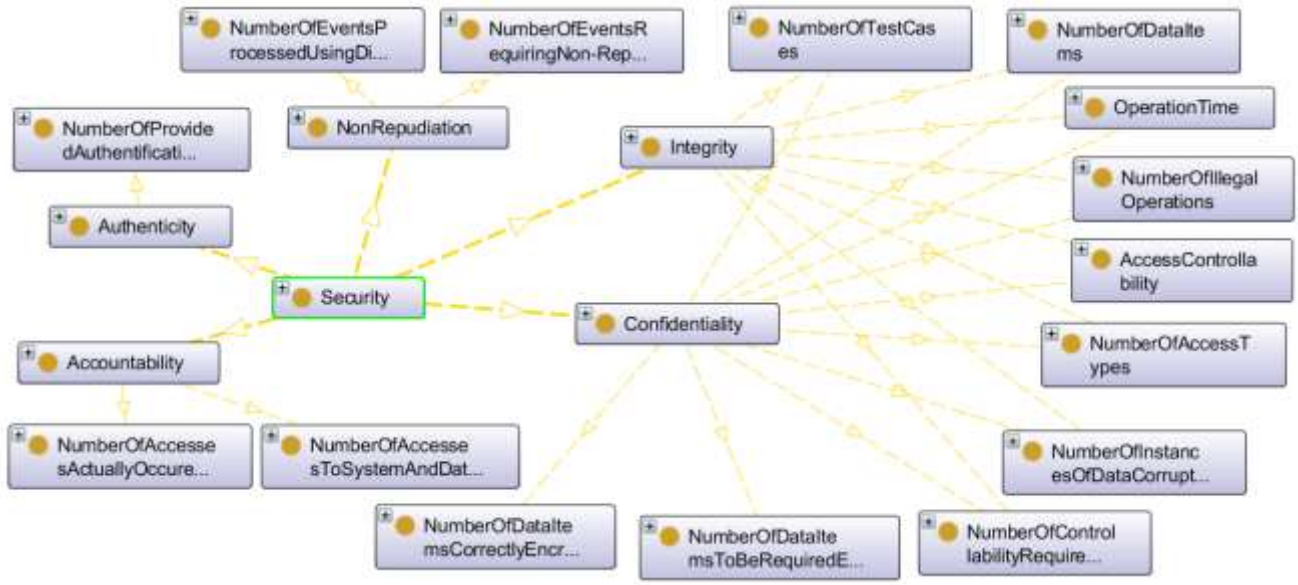

Рис. 2.7 – Базова онтологія для Захищеності [1]

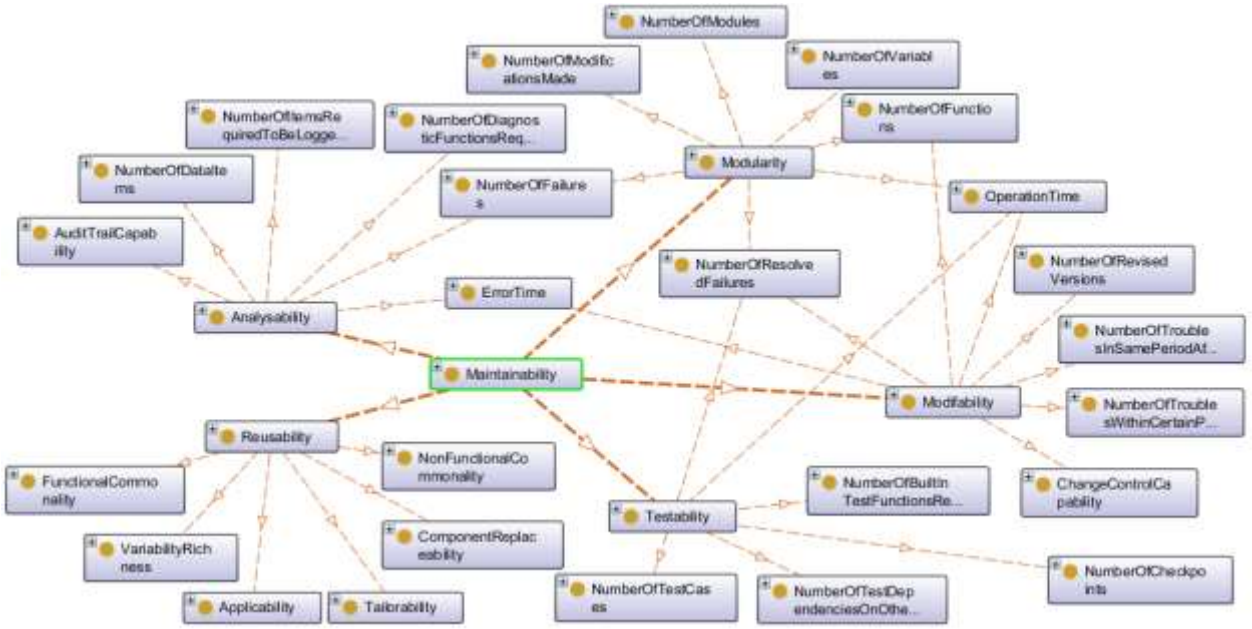

Рис. 2.8 – Базова онтологія для Супроводжуваності [1]

Онтологічні моделі нефункційних характеристик конкретного ПЗ мають вигляд:

- онтологічна модель функційної придатності:

$$O_{F_s}^{real} = \{\{ fsa_1,...,fsa_m \},"depends\ on",\{ f_1,\varphi_1,\varphi_2,\varphi_3 \}\}, \quad (2.48)$$

де $m \le 19$ – кількість атрибутів підхарактеристик функційної придатності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик функційної придатності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель ефективності:

$$O_{P_e}^{real} = \{\{ pea_1,...,pea_n \},"depends\ on",\{ f_2,\varphi_4,\varphi_5,\varphi_6 \}\}; \quad (2.49)$$

де $n \le 30$ – кількість атрибутів підхарактеристик ефективності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик ефективності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель зручності використання:

$$O_{U_b}^{real} = \{\{ uba_1,...uba_k \},"depends\ on",\{ f_3,\varphi_7,\varphi_8,\varphi_9,\varphi_{10},\varphi_{11},\varphi_{12} \}\}; \quad (2.50)$$

де $k \le 56$ – кількість атрибутів підхарактеристик зручності використання, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик зручності використання, які можна обчислити на основі наявних атрибутів;

- онтологічна модель надійності:

$$O_{R_b}^{real} = \{\{ rba_1, ... rba_o \}, "depends\ on", \{ f_4, \varphi_{13}, \varphi_{14}, \varphi_{15}, \varphi_{16} \}\}; \quad (2.51)$$

де $o \le 35$ – кількість атрибутів підхарактеристик надійності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик надійності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель сумісності:

$$O_{C_b}^{real} = \{\{ cba_1, ..., cba_p \}, "depends\ on", \{ f_5, \varphi_{17}, \varphi_{18} \}\}; \quad (2.52)$$

де $p \le 12$ – кількість атрибутів підхарактеристик сумісності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик сумісності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель захищеності:

$$O_{S_{cr}}^{real} = \{\{ scra_1, ..., scra_q \}, "depends\ on", \{ f_6, \varphi_{19}, \varphi_{20}, \varphi_{21}, \varphi_{22}, \varphi_{23} \}\}; \quad (2.53)$$

де $q \le 29$ – кількість атрибутів підхарактеристик захищеності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик захищеності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель супроводжуваності:

$$O_{M_b}^{real} = \{\{ mba_1, ..., mba_r \}, "depends\ on", \{ f_7, \varphi_{24}, \varphi_{25}, \varphi_{26}, \varphi_{27}, \varphi_{28} \}\}; \quad (2.54)$$

де $r \le 39$ – кількість атрибутів підхарактеристик супроводжуваності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик супроводжуваності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель можливості переносу:

$$O_{P_b}^{real} = \{\{ pba_1, ..., pba_l \}, "depends\ on", \{ f_8, \varphi_{29}, \varphi_{30}, \varphi_{31} \}\}. \quad (2.55)$$

де $l \le 22$ – кількість атрибутів підхарактеристик можливості переносу, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик можливості переносу, які можна обчислити на основі наявних атрибутів.

Онтології (онтологічні бази знань), які розробляються за моделями (2.48)-(2.55), наповнюються на основі інформації, взятої зі специфікації вимог до конкретного програмного забезпечення.

Розроблені онтологічні моделі нефункційних характеристик програмного забезпечення ґрунтуються на врахуванні вимог стандартів ISO 25010:2011 та ISO 25023:2016 і забезпечують підґрунтя для вибору достатнього обсягу інформації для оцінювання нефункційних характеристик ПЗ.

2.3. Моделювання діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до програмного забезпечення

Інтелектуальний агент на основі онтологічного підходу для оцінювання специфікацій вимог до програмного забезпечення використовує під час свого функціонування базові онтології нефункційних характеристик-складових якості ПЗ як відомі йому факти, з якими він зіставляє інформацію, отриману зі специфікації вимог до реального ПЗ, представлену у вигляді реальних онтологій, на основі чого оцінює інформацію у специфікації вимог до ПЗ та приймає рішення про подальші дії.

Отже, *процес оцінювання специфікації вимог інтелектуальним агентом* полягає у:

1) порівнянні онтологій нефункційних характеристик-складових якості конкретного ПЗ з базовими онтологіями нефункційних характеристик ПЗ з метою виявлення атрибутів, відсутніх у специфікації вимог до ПЗ, за якою реальні онтології були побудовані, а також виявлення підхрактеристик та нефункційних характеристик ПЗ, які неможливо обчислити на основі наявних у специфікації вимог до конкретного ПЗ атрибутів;

2) формуванні висновку про достатність або недостатність інформації у специфікації вимог для визначення кожної нефункційної характеристики ПЗ окремо та для визначення всіх нефункційних характеристик ПЗ разом;

3) розрахунку числових оцінок рівня достатності наявної у специфікації вимог інформації для визначення кожної нефункційної характеристики ПЗ;

4) розрахунку числової оцінки рівня достатності наявної у специфікації вимог інформації для визначення всіх нефункційних характеристик ПЗ.

Нехай $S_{F_s} = O_{F_s} \setminus (O_{F_s} \cap O_{F_s}^{real})$, де: $S_{F_s} = \{ fsa_1, ..., fsa_{(19-m)} \}$ – множина атрибутів підхарактеристик функційної придатності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик функційної придатності, які неможливо обчислити на основі наявних атрибутів;

$S_{P_e} = O_{P_e} \setminus (O_{P_e} \cap O_{P_e}^{real})$, де: $S_{P_e} = \{ pea_1, ..., pea_{(30-n)} \}$ – множина атрибутів підхарактеристик ефективності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик ефективності, які неможливо обчислити на основі наявних атрибутів;

$S_{U_b} = O_{U_b} \setminus (O_{U_b} \cap O_{U_b}^{real})$, де: $S_{U_b} = \{ uba_1, ..., uba_{(56-k)} \}$ – множина атрибутів підхарактеристик зручності використання, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик зручності використання, які неможливо обчислити на основі наявних атрибутів;

$S_{R_b} = O_{R_b} \setminus (O_{R_b} \cap O_{R_b}^{real})$, де: $S_{R_b} = \{ rba_1, ..., rba_{(35-o)} \}$ – множина атрибутів підхарактеристик надійності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик надійності, які неможливо обчислити на основі наявних атрибутів;

$S_{C_b} = O_{C_b} \setminus (O_{C_b} \cap O_{C_b}^{real})$, де: $S_{C_b} = \{ cba_1, ..., cba_{(12-p)} \}$ – множина атрибутів підхарактеристик суміності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик сумісності, які неможливо обчислити на основі наявних атрибутів;

$S_{S_{cr}} = O_{S_{cr}} \setminus (O_{S_{cr}} \cap O_{S_{cr}}^{real})$, де: $S_{S_{cr}} = \{scra_1,...,scra_{(29-q)}\}$ – множина атрибутів підхарактеристик захищеності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик захищеності, які неможливо обчислити на основі наявних атрибутів;

$S_{M_b} = O_{M_b} \setminus (O_{M_b} \cap O_{M_b}^{real})$, де: $S_{M_b} = \{mba_1,...,mba_{(39-r)}\}$ – множина атрибутів підхарактеристик супроводжуваності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик супроводжуваності, які неможливо обчислити на основі наявних атрибутів;

$S_{P_b} = O_{P_b} \setminus (O_{P_b} \cap O_{P_b}^{real})$, де: $S_{P_b} = \{pba_1,...,pba_{(22-l)}\}$ – множина атрибутів підхарактеристик можливості переносу, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик можливості переносу, які неможливо обчислити на основі наявних атрибутів.

Правила для формування висновку щодо достатності або недостатності інформації у специфікації вимог для визначення кожної нефункційної характеристики ПЗ:

- для функційної придатності:

$$\begin{aligned} &if\ S_{F_s} = \varnothing \\ &then\ "SRS\ information\ is\ sufficient\ for\ Functional\ Suitability"\ , \\ &else\ "SRS\ information\ is\ insufficient\ for\ Functional\ Suitability" \end{aligned} \quad (2.56)$$

- для ефективності:

$$\begin{aligned} &if\ S_{P_e} = \varnothing \\ &then\ "SRS\ information\ is\ sufficient\ for\ Perfromance\ Efficiency"\ , \\ &else\ "SRS\ information\ is\ insufficient\ for\ Perfromance\ Efficiency" \end{aligned} \quad (2.57)$$

- для зручності використання:

$$\begin{aligned} &if\ S_{U_b} = \varnothing \\ &then\ "SRS\ information\ is\ sufficient\ for\ Usability"\ , \\ &else\ "SRS\ information\ is\ insufficient\ for\ Usability" \end{aligned} \quad (2.58)$$

- для надійності:

$$\begin{array}{l} if\ S_{R_b} = \varnothing \\ then\ "SRS\ information\ is\ sufficient\ for\ Reliability"\ , \\ else\ "SRS\ information\ is\ insufficient\ for\ Reliability" \end{array} \quad (2.59)$$

- для сумісності:

$$\begin{array}{l} if\ S_{C_b} = \varnothing \\ then\ "SRS\ information\ is\ sufficient\ for\ Compatibility"\ , \\ else\ "SRS\ information\ is\ insufficient\ for\ Compatibility" \end{array} \quad (2.60)$$

- для захищеності:

$$\begin{array}{l} if\ S_{S_{cr}} = \varnothing \\ then\ "SRS\ information\ is\ sufficient\ for\ Security"\ , \\ else\ "SRS\ information\ is\ insufficient\ for\ Security" \end{array} \quad (2.61)$$

- для супроводжуваності:

$$\begin{array}{l} if\ S_{M_b} = \varnothing \\ then\ "SRS\ information\ is\ sufficient\ for\ Maintainability"\ , \\ else\ "SRS\ information\ is\ insufficient\ for\ Maintainability" \end{array} \quad (2.62)$$

- для можливості переносу:

$$\begin{array}{l} if\ S_{P_b} = \varnothing \\ then\ "SRS\ information\ is\ sufficient\ for\ Portability"\ . \\ else\ "SRS\ information\ is\ insufficient\ for\ Portability" \end{array} \quad (2.63)$$

Правило для формування висновку щодо достатності або недостатності інформації у специфікації вимог для визначення всіх нефункційних характеристик програмного забезпечення:

$$if\ (S_{F_s} \cup S_{P_e} \cup S_{U_b} \cup S_{R_b} \cup S_{C_b} \cup S_{S_{cr}} \cup S_{M_b} \cup S_{P_b}) = \varnothing$$
$$then\ "SRS\ information\ is\ sufficient" \quad .$$
$$else\ "SRS\ information\ is\ insufficient"$$

(2.64)

Розроблена модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ відображає особливості оцінювання достатності інформації для визначення нефункційних характеристик-складових якості ПЗ та є теоретичним підґрунтям для розроблення методу діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ.

2.4. Висновки

У розділі вперше запропоновані теоретико-множинні, базові (універсальні) онтологічні моделі нефункційних характеристик-складових якості ПЗ, а також онтологічні моделі нефункційних характеристик-складових якості конкретного ПЗ, які ґрунтуються на врахуванні вимог стандартів ISO 25010:2011 та ISO 25023:2016 і забезпечують підґрунтя для вибору достатнього обсягу інформації для оцінювання нефункційних характеристик-складових якості ПЗ. Адекватність розроблених моделей підтверджується збіжністю їх параметрів із стандартами ISO 25010:2011 та ISO 25023:2016.

Вперше розроблено модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ, яка ґрунтується на порівняльному аналізі онтологій та є теоретичним підґрунтям для розроблення методів діяльності інтелектуального агента на основі онтологічного підходу.

РОЗДІЛ 3.
МЕТОДИ ДІЯЛЬНОСТІ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ

3.1. Методи діяльності інтелектуальних агентів на основі онтологічного підходу для семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення

Для автоматизації семантичного розбору природомовної специфікації необхідно виконати її формалізацію. Формалізацію специфікації вимог до ПЗ потрібно виконувати, враховуючи, на видобування яких саме вимог буде спрямовуватись семантичний парсинг специфікації. Для парсингу специфікацій на предмет пошуку атрибутів для визначення нефункційних характеристик-складових якості ПЗ така формалізація буде виконана з використанням онтологій, оскільки саме онтології забезпечують виявлення дублювань та прогалин у знаннях на основі візуалізації відсутніх логічних зв'язків, а також можливість аналізу інформації інтелектуальними агентами. Для такої формалізації специфікації вимог може бути використана базова онтологія специфікації вимог до ПЗ (з точки зору наявності атрибутів) на основі стандарту ISO 29148:2018 [152]. В даній онтології атрибути, необхідні для визначення нефункційних характеристик-складових якості ПЗ, представлені з врахуванням розподілу за розділами специфікації, за рахунок чого розроблена онтологія є шаблоном специфікації вимог до ПЗ з точки зору наявності атрибутів та надає візуальні підказки користувачу про місце розташування тих чи інших атрибутів у специфікації вимог до ПЗ.

Інтелектуальний агент на основі онтологічного підходу для семантичного парсингу специфікацій вимог до ПЗ приймає на вхід специфікацію вимог до ПЗ та проводить автоматичний парсинг специфікації вимог до ПЗ на предмет пошуку атрибутів, необхідних для визначення нефункційних характеристик ПЗ.

Метод діяльності інтелектуального агента на основі онтологічного підходу для семантичного парсингу природомовних специфікацій вимог до ПЗ уf предмет пошуку атрибутів складається з наступних етапів:

1) пошук кожного атрибута з базової онтології специфікації вимог (така онтологія міститься у базі знань агента) у специфікації вимог до реального ПЗ (вважаємо, що специфікація формалізована та відповідає вимогам стандарту ISO 29148:2018);

2) якщо <атрибут$_i$> знайдено у специфікації вимог, то <атрибут$_i$> заноситься у множину наявних атрибутів, i=1..138 (оскільки, згідно зі стандартом ISO 25023:2016, є 138 різних атрибутів, від яких залежать нефункційні характеристики-складові якості ПЗ);

3) якщо <атрибут$_i$> не знайдено у специфікації вимог, то <атрибут$_i$> заноситься у множину відсутніх атрибутів, i=1..138;

4) з базової онтології для нефункційних характеристик-складових якості ПЗ, розробленої в [1], видаляються всі атрибути з множини відсутніх атрибутів;

5) відбувається перевірка, чи всі атрибути з множини наявних атрибутів залишились у онтології після її модифікації на попередньому кроці;

6) відбувається збереження внесених змін – створення реальної онтології для нефункційних характеристик-складових якості ПЗ.

Отже, результатом роботи такого інтелектуального агента є реальна онтологія для нефункційних характеристик-складових якості ПЗ, яка є вхідними даними для ІА на основі онтологічного підходу для оцінки початкових етапів життєвого циклу програмного забезпечення. Як додаткові результати функціювання агента для подальшої роботи можуть бути використані також множини наявних та відсутніх атрибутів у реальній специфікації вимог до ПЗ.

Структурна схема *інтелектуального агента на основі онтологічного підходу для семантичного парсингу природомовних специфікацій вимог до ПЗ* представлена на рис. 3.1.

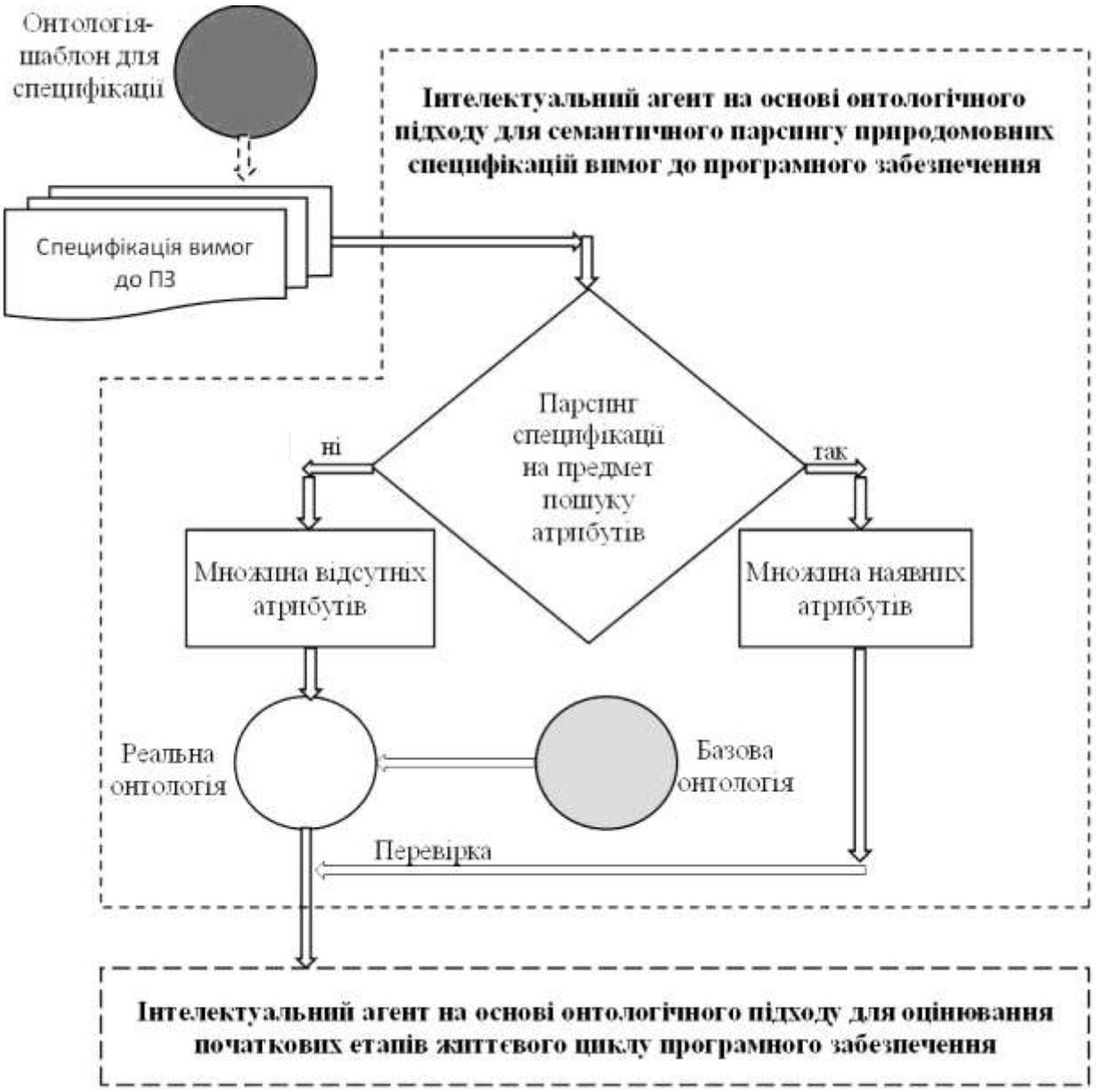

Рис. 3.1 – Структура інтелектуального агента на основі онтологічного підходу для семантичного розбору (парсингу) природомовних специфікацій вимог до ПЗ

За аналогією розробимо *метод діяльності інтелектуального агента для семантичного аналізу вимог до ПЗ на предмет пошуку показників для метричного аналізу* – рис. 3.2. Такий агент приймає на вхід природомовні вимоги до ПЗ та проводить автоматичний аналіз вимог на предмет пошуку показників, необхідних для визначення метрик ПЗ. Результатом роботи інтелектуального агента для семантичного аналізу вимог до ПЗ є реальна онтологія предметної галузі «Інженерія програмного забезпечення» (частина «Якість та складність ПЗ. Метричний аналіз»), розроблена у [1].

Інтелектуальний агент, який діє на основі методу з рис. 3.2 дає можливість виконувати семантичний аналіз природомовних специфікацій на предмет пошуку показників, необхідних для визначення метрик складності та якості ПЗ. Ці результати далі використовуються для оцінювання достатності інформації вимог для визначення метрик ПЗ, для чого потрібно лише встановити присутність або відсутність кожного показника у вимогах.

Рис. 3.2 – Метод діяльності інтелектуального агента для семантичного аналізу програмних вимог для пошуку показників для метричного аналізу

Розроблені інтелектуальні агенти дають можливість виконувати розбір природомовних специфікацій на предмет встановлення наявності чи відсутності атрибутів та/або показників, необхідних для визначення нефункційних характеристик-складових якості ПЗ та/або метрик ПЗ. Ці результати далі використовуються для оцінювання достатності інформації (атрибутів та/або показників) для визначення нефункційних характеристик-складових якості. Оскільки для визначення достатності необхідно лише знати, присутній або відсутній атрибут та/або показник у специфікації, або відсутній у ній, тому правила парсингу специфікацій, на основі яких працюють розроблені агенти, є простими та зрозумілими. Простота цих правил забезпечує високу швидкість та низьку ціну парсингу природомовних специфікацій.

3.2. Методи діяльності інтелектуальних агентів на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення

Інтелектуальний агент на основі онтологічного підходу використовує під час свого функціонування базові онтології нефункційних характеристик-складових якості ПЗ (які були розроблені у [1] і представлені в розділі 2) як відомі йому факти. Саме онтології, які відображають причинно-наслідкові зв’язки між поняттями, дозволяють виявити відсутні у специфікації атрибути і встановити, які нефункційні характеристики-складові якості ПЗ неможливо визначити без таких атрибутів. З цими базовими онтологіями агент зіставляє інформацію, отриману зі специфікації вимог до реального ПЗ, представлену у вигляді реальних онтологій для забезпечення можливості порівняння базових та реальних онтологій. На основі такого порівняння онтологій інтелектуальний агент отримує список відсутніх у специфікації атрибутів, оскільки саме відсутніми у специфікації атрибутами відрізнятимуться реальні онтології від базових. Інтелектуальний агент проводить аналіз отриманого списку відсутніх атрибутів та залежностей нефункційних характеристик від атрибутів (за базовими онтологіями) і

встановлює, які нефункційні характеристики-складові якості ПЗ неможливо визначити без відсутніх атрибутів. Крім цього, інтелектуальний агент підраховує кількості відсутніх атрибутів та нефункційних характеристик, які неможливо обчислити без певних атрибутів, для формування числової оцінки достатності інформації у специфікаціях вимог до ПЗ. Після цього інтелектуальний агент оцінює інформацію у специфікації вимог до ПЗ та приймає рішення про подальші дії, зокрема, надає висновки про достатність інформації та рекомендації щодо її підвищення.

Отже, *метод діяльності інтелектуального агента на основі онтологічного підходу для оцінки початкових етапів життєвого циклу програмного забезпечення на предмет оцінювання достатності інформації у специфікації для визначення нефункційних характеристик-складових якості ПЗ* складається з наступних етапів:

1) порівняння онтологій нефункційних характеристик-складових якості реального ПЗ з базовими онтологіями нефункційних характеристик-складових якості ПЗ з метою виявлення атрибутів, відсутніх у специфікації вимог до реального ПЗ, за якою реальні онтології були побудовані;

2) виявлення підхарактеристик та нефункційних характеристик-складових якості ПЗ, які неможливо обчислити на основі наявних у специфікації вимог до реального ПЗ атрибутів;

3) формування висновку про достатність або недостатність інформації у специфікації вимог для визначення кожної нефункційної характеристики ПЗ окремо та для визначення всіх нефункційних характеристик ПЗ разом;

4) розрахунок числових оцінок рівня достатності наявної у специфікації вимог інформації для визначення кожної нефункційної характеристики ПЗ за формулою (3.1):

$$D_j = \frac{(k_j - \sum_{i=1}^{k_j} \frac{qm_i}{qn_i})}{k_j}, \qquad (3.1)$$

де k_j – кількість підхарактеристик j-ї нефункційної характеристики ПЗ (j=1…8, оскільки у стандарті ISO 25010 визначено саме 8 нефункційних характеристик-складових якості ПЗ), qm_i – кількість відсутніх у специфікації вимог до реального ПЗ атрибутів для i-ї підхарактеристики j-ї нефункційної характеристики ПЗ, qn_i – кількість необхідних атрибутів для i-ї підхарактеристики j-ї нефункційної характеристики ПЗ (визначається базовими онтологіями для кожної нефункційної характеристики-складової якості ПЗ);

5) розрахунок числової оцінки рівня достатності наявної у специфікації вимог інформації для визначення всіх нефункційних характеристик-складових якості ПЗ за формулою (3.2):

$$D = \frac{(k - \sum_{j=1}^{k} \frac{qmc_j}{qnc_j})}{k}, \qquad (3.2)$$

де k – кількість нефункційних характеристик–складових якості ПЗ (k=8 відповідно до ISO 25010), qmc_j – кількість відсутніх у специфікації вимог до реального ПЗ атрибутів для j-ї нефункційної характеристики ПЗ, qnc_j – кількість необхідних атрибутів для j-ї нефункційної характеристики ПЗ (визначається базовими онтологіями для кожної нефункційної характеристики-складової якості ПЗ);

6) візуалізація прогалин у знаннях щодо нефункційних характеристик-складових якості ПЗ.

Пропонований агент є інтелектуальним, тому що він автоматично опрацьовує наявні знання (вимоги щодо нефункційних характеристик, представлені у вигляді онтологій) та формує нові знання (онтології для реального ПЗ, висновки про рівень достатності інформації, рекомендації щодо підвищення рівня достатності інформації у специфікації вимог).

Даний інтелектуальний агент не працює з нечіткими даними, оскільки задача оцінювання достатності інформації не передбачає нечіткості. Необхідний атрибут або присутній у специфікації, або відсутній у ній і тоді відбувається падіння рівня достатності інформації на величину, яка залежить від того, скільки нефункційних характеристик залежать від даного атрибуту (корелюють за ним).

За аналогією розробимо *метод діяльності інтелектуального агента на основі онтологічного підходу для оцінки початкових етапів життєвого циклу програмного забезпечення на предмет оцінювання достатності інформації у специфікації для визначення метрик складності та якості ПЗ*. Базовою для такого агента виступатиме онтологія предметної галузі «Інженерія програмного забезпечення» (частина «Якість та складність ПЗ. Метричний аналіз»), розроблена у розділі 2. Крім цього, числова оцінка рівня достатності метричної інформації у специфікації обчислюватиметься за формулою:

$$D = \frac{(l - \sum_{j=1}^{l} \frac{qmi_j}{qni_j})}{l}, \qquad (3.3)$$

де l – кількість метрик ($l=24$, оскільки було обрано саме 24 метрики складності та якості, доступних на ранніх етапах життєвого циклу), qmi_j – кількість відсутніх у реальній специфікації показників для j-ї метрики, qni_j – кількість необхідних показників для j-ї метрики (визначається базовою онтологією предметної галузі «Інженерія програмного забезпечення» (частина «Якість та складність ПЗ. Метричний аналіз»)), $j=1..k$, $\Sigma(qni_j)=72$ – загальна кількість показників (в т.ч. й тих, що повторюються), від яких залежать метрики ПЗ.

Тоді метод діяльності інтелектуального агента на основі онтологічного підходу для оцінки початкових етапів життєвого циклу програмного забезпечення на предмет оцінювання достатності інформації у специфікації для визначення метрик складності та якості ПЗ представлений на рис. 3.3.

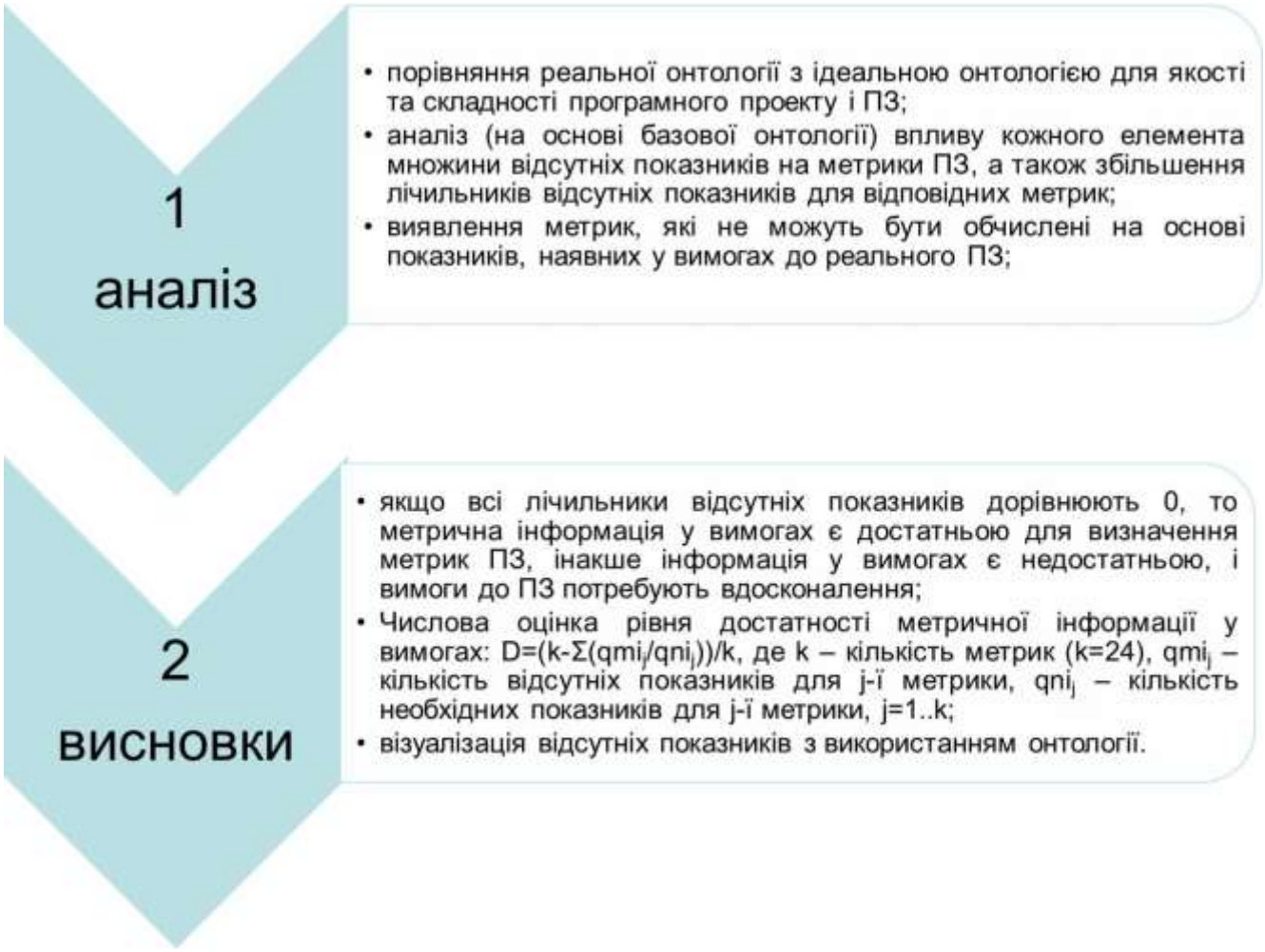

Рис. 3.3 – Метод діяльності ІА на основі онтологічного підходу для оцінки початкових етапів життєвого циклу програмного забезпечення на предмет оцінювання достатності інформації у специфікації для визначення метрик складності та якості ПЗ

Розроблені інтелектуальні агенти на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення на предмет оцінювання достатності інформації у специфікації для оцінювання нефункційних характеристик-складових якості ПЗ та для визначення метрик ПЗ формує висновок про достатність або недостатність інформації (атрибутів та/або показників) у специфікації вимог до ПЗ, обчислюють числову оцінку рівня достатності інформації, а також візуалізовано надають відсутні атрибути та/або показники з розподілом за характеристиками та/або метриками, для яких вони використовуються.

3.3. Експерименти: аналіз нефункційних характеристик у специфікаціях вимог до програмного забезпечення за допомогою інтелектуальних агентів

На основі розроблених методів діяльності інтелектуальних агентів на основі онтологічного підходу для семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення та для оцінювання початкових етапів життєвого циклу програмного забезпечення були розроблені відповідні інтелектуальні агенти, які були апробовані на реальних специфікаціях вимог до ПЗ.

Для проведення *першого експерименту* використовувалась специфікація вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос». Для оцінювання інформації щодо нефункційних характеристик у цій специфікації вимог до ПЗ, розроблений інтелектуальний агент сформував реальні онтології нефункційних характеристик за специфікацією вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос», після чого виконав співставлення та порівняння реальних і базових онтологій нефункційних характеристик ПЗ, внаслідок якого було сформовано множини відсутніх атрибутів для кожної нефункційної характеристики:

1) відсутні атрибути для характеристики «Надійність»: Кількість відмов, Кількість тестових випадків, Кількість фіксованих відмов, Щільність відмов відносно до тестових випадків, Кількість спостережуваних несправностей, Кількість функцій, Кількість несправностей, Можливість відновлюваності; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» недостатньо для обчислення «Надійності»;

2) відсутні атрибути для характеристики «Функційна придатність»: Кількість функцій, Функційна адекватність, Кількість елементів даних, Обчислювальна точність; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» недостатньо для обчислення «Функційної придатності»;

3) відсутні атрибути для характеристики «Ефективність»: Кількість задач, Кількість оцінок, Кількість відмов, Кількість помилок введення-виведення, Кількість елементів даних, Кількість помилок пам’яті, Максимум використовуваної пам’яті, Розмір бази даних; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» недостатньо для обчислення «Ефективності»;

4) відсутні атрибути для характеристики «Сумісність»: Кількість відмов, Кількість функцій, Кількість елементів даних, Кількість форматів даних для обміну, Кількість протоколів інтерфейсу; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» недостатньо для обчислення «Сумісності»;

5) відсутні атрибути для характеристики «Супроводжуваність»: Кількість відмов, Кількість фіксованих відмов, Кількість функцій, Кількість модулів, Міцність варіабельності, Можливість заміни компонентів, Кількість елементів даних, Кількість необхідних діагностичних функцій, Кількість тестових випадків, Кількість контрольних точок; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» недостатньо для обчислення «Супроводжуваності»;

6) відсутні атрибути для характеристики «Можливість переносу»: Кількість функцій, Кількість елементів даних, Кількість структур даних, Кількість операцій налаштування, Кількість кроків інсталяції; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» недостатньо для обчислення «Можливості переносу»;

7) відсутні атрибути для характеристики «Захищеність»: Кількість тестових випадків, Кількість елементів даних, Кількість типів доступу, Кількість елементів даних, які потребують шифрування та розшифрування, Кількість подій, що

обробляються за допомогою цифрового підпису, Кількість доступів до системи та даних, що повинні бути записані у системний журнал, Кількість методів перевірки автентичності; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» недостатньо для обчислення «Захищеності»;

8) відсутні атрибути для характеристики «Зручність використання»: Кількість функцій, Кількість елементів даних введення-виведення, Кількість посібників, Кількість задач, Повнота документації користувача та довідкового фонду, Кількість екранів або форм, Кількість помилок введення-виведення, Кількість елементів інтерфейсу, Кількість легко зрозумілих повідомлень, Загальна кількість перевірених помилкових умов, Загальна кількість некоректних шаблонів операцій, Кількість графічних елементів інтерфейсу, Ступінь ергономічної привабливості, Задоволення потреб користувачів з обмеженими можливостями; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» недостатньо для обчислення «Зручності використання».

Аналіз впливу кожного елемента множин відсутніх атрибутів на нефункційні характеристики ПЗ та їх підхарактеристики, проведений розробленим інтелектуальним агентом на основі базових онтологій нефункційних характеристик, дав можливість обчислити кількості відсутніх атрибутів для підхарактеристик нефункційних характеристик. Крім цього, на основі такого аналізу розроблений інтелектуальний агент сформував висновок, що на основі наявних атрибутів у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» неможливо обчислити такі підхарактеристики нефункційних характеристик:

1) підхарактеристики «Надійності»: Зрілість, Наявність, Відмовостійкість, Відновлюваність, тобто недостатньо інформації для визначення всіх 4-х підхарактеристик «Надійності» (згідно стандартів ISO 25010:2011 [98] та ISO

25023:2016 [102], які регламентують залежність нефункційних характеристик від підхарактеристик, а також підхарактеристик від атрибутів; саме на основі цих стандартів були розроблені базові онтології нефункційних характеристик, представлені в [1] та в розділі 2);

2) підхарактеристики «Функційної придатності»: Функційна повнота, Функційна коректність, Функційна доцільність, тобто недостатньо інформації для визначення всіх 3-х підхарактеристик «Функційної придатності»;

3) підхарактеристики «Ефективності»: Поведінка у часі, Поведінка ресурсів, Ємність, тобто недостатньо інформації для визначення всіх 3-х підхарактеристик «Ефективності»;

4) підхарактеристики «Сумісності»: Співіснування, Взаємодія, тобто недостатньо інформації для визначення обох підхарактеристик «Сумісності»;

5) підхарактеристики «Супроводжуваності»: Модульність, Повторне використання, Аналізованість, Модифікованість, Тестованість, тобто недостатньо інформації для визначення всіх 5-х підхарактеристик «Супроводжуваності»;

6) підхарактеристики «Можливості переносу»: Адаптованість, Можливість інсталяції, Можливість заміни, тобто недостатньо інформації для визначення всіх 3-х підхарактеристик «Можливості переносу»;

7) підхарактеристики «Захищеності»: Конфіденційність, Цілісність, Невідхилюваність, Підзвітність, Ідентичність, тобто недостатньо інформації для визначення всіх 5-х підхарактеристик «Захищеності»;

8) підхарактеристики «Зручності використання»: Розпізнавання доцільності, Можливість вивчення, Керованість, Захист від помилок користувача, Естетичність інтерфейсу користувача, Доступність, тобто недостатньо інформації для визначення всіх 6-х підхарактеристик «Зручності використання».

Отже, розроблений інтелектуальний агент надав висновок, що атрибутів, наявних у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос», недостатньо для обчислення всіх підхарактеристик та всіх нефункційних характеристик ПЗ. Для забезпечення достатності інформації щодо нефункційних характеристик у специфікації вимог

до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» слід виконати доповнення специфікації вищезазначеними відсутніми атрибутами.

Візуалізацію прогалин у знаннях щодо нефункційних характеристик ПЗ, надана розробленим інтелектуальним агентом, представлено на рис. 3.4-3.11, на яких відсутні атрибути представлені як закреслені, а підхарактеристики, для визначення яких наявних у специфікації атрибутів недостатньо, представлені як окреселні колом у відповідних базових онтологіях.

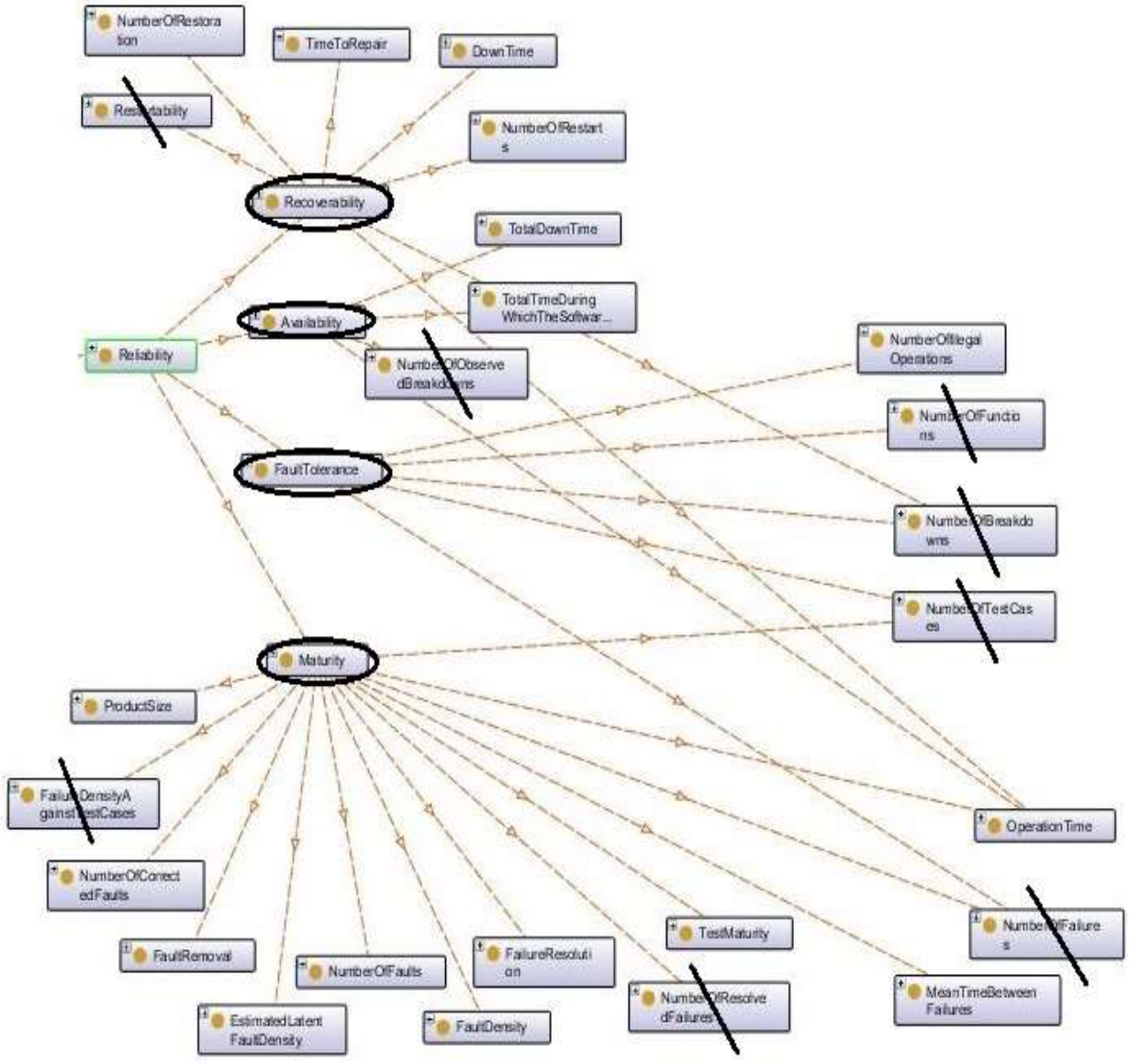

Рис. 3.4 – Візуалізація прогалин у знаннях щодо «Надійності»

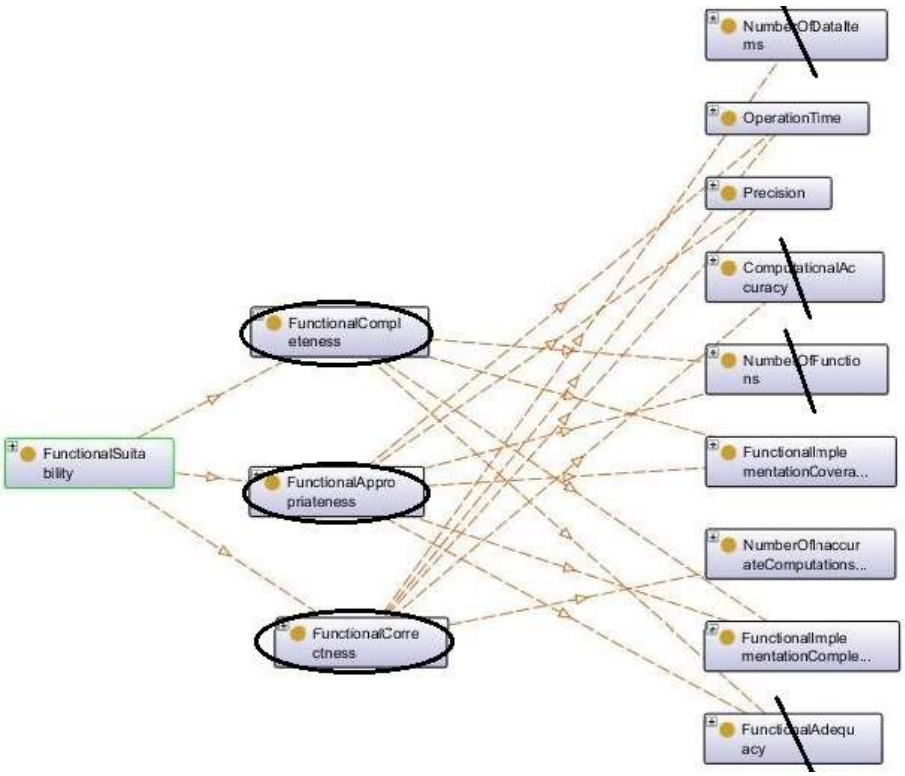

Рис. 3.5 – Візуалізація прогалин у знаннях щодо «Функційної придатності»

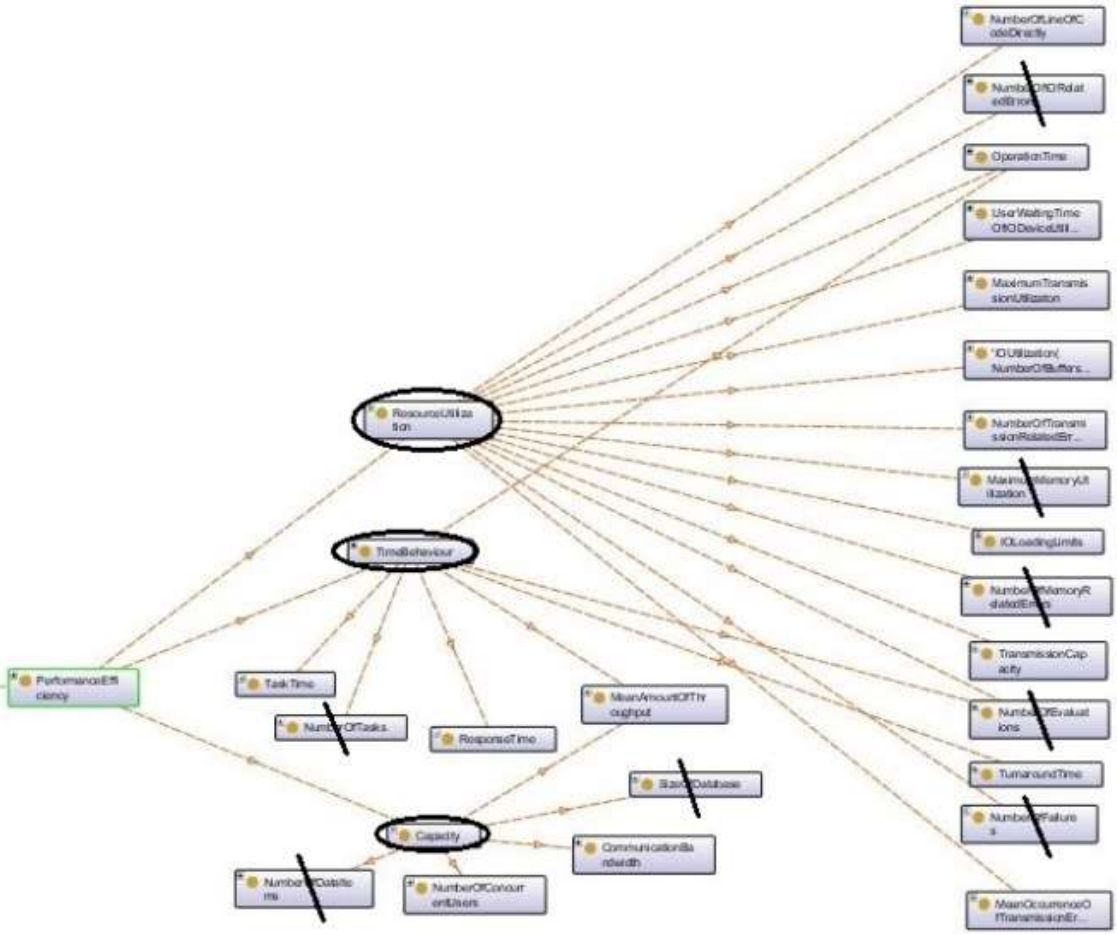

Рис. 3.6 – Візуалізація прогалин у знаннях щодо «Ефективності»

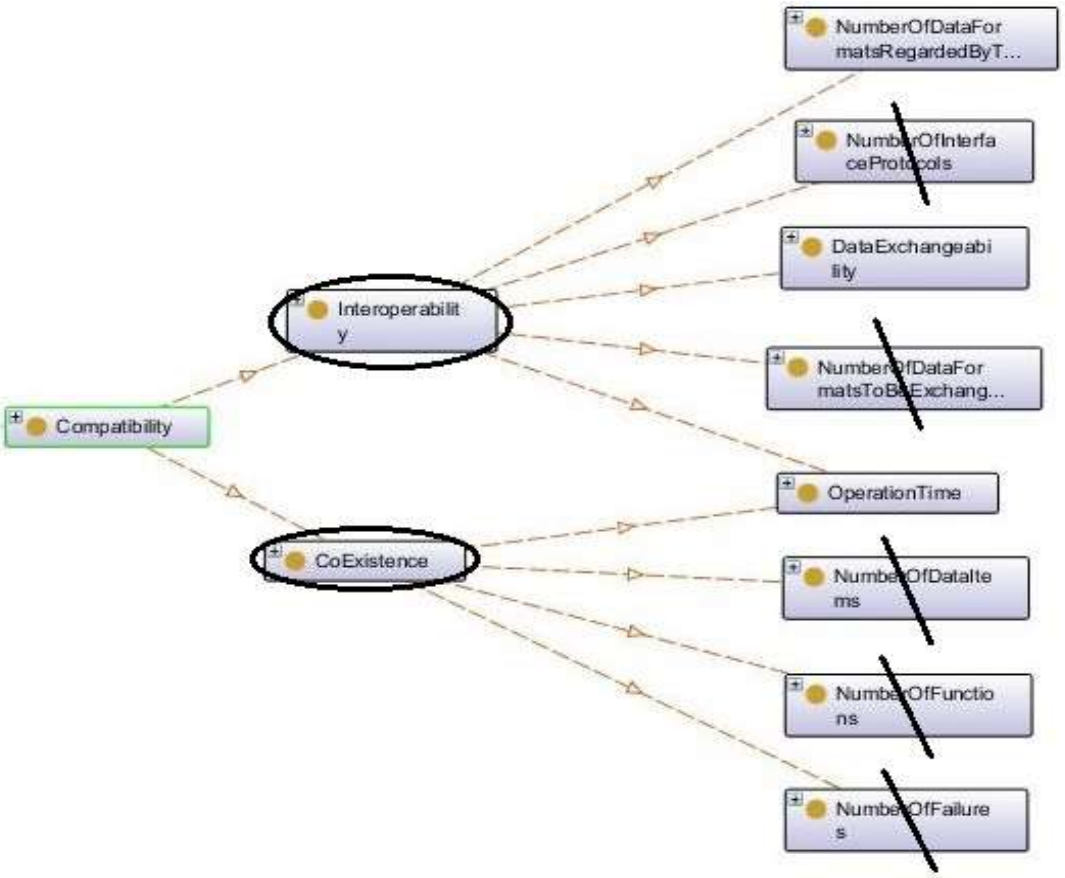

Рис. 3.7 – Візуалізація прогалин у знаннях щодо «Сумісності»

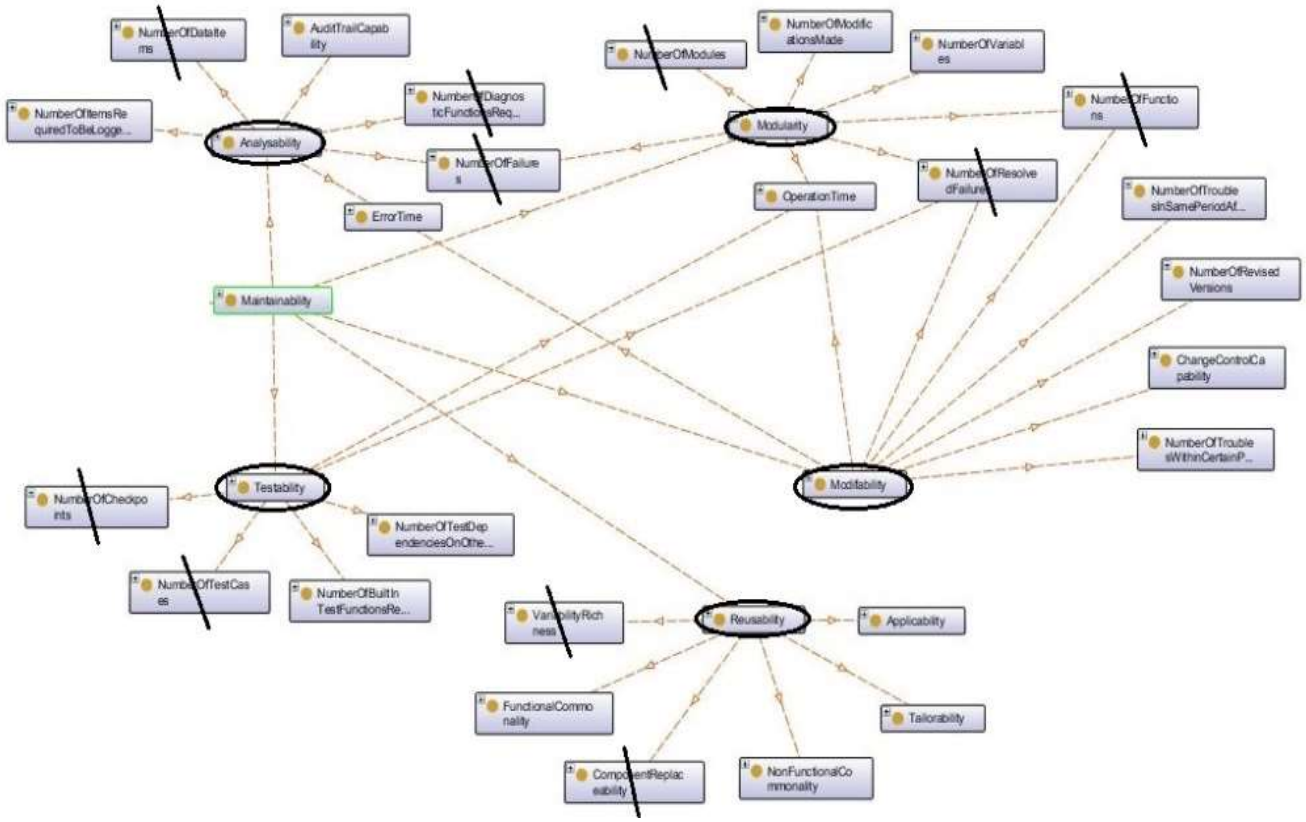

Рис. 3.8 – Візуалізація прогалин у знаннях щодо «Супроводжуваності»

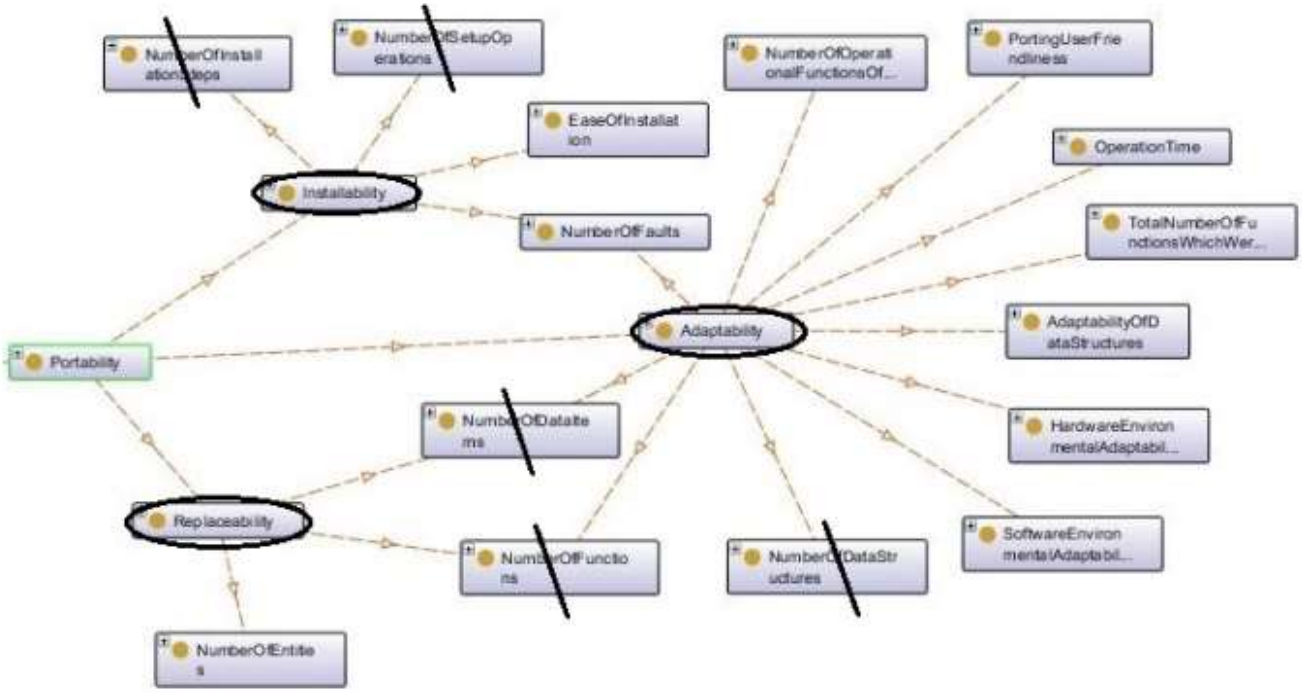

Рис. 3.9 – Візуалізація прогалин у знаннях щодо «Можливості переносу»

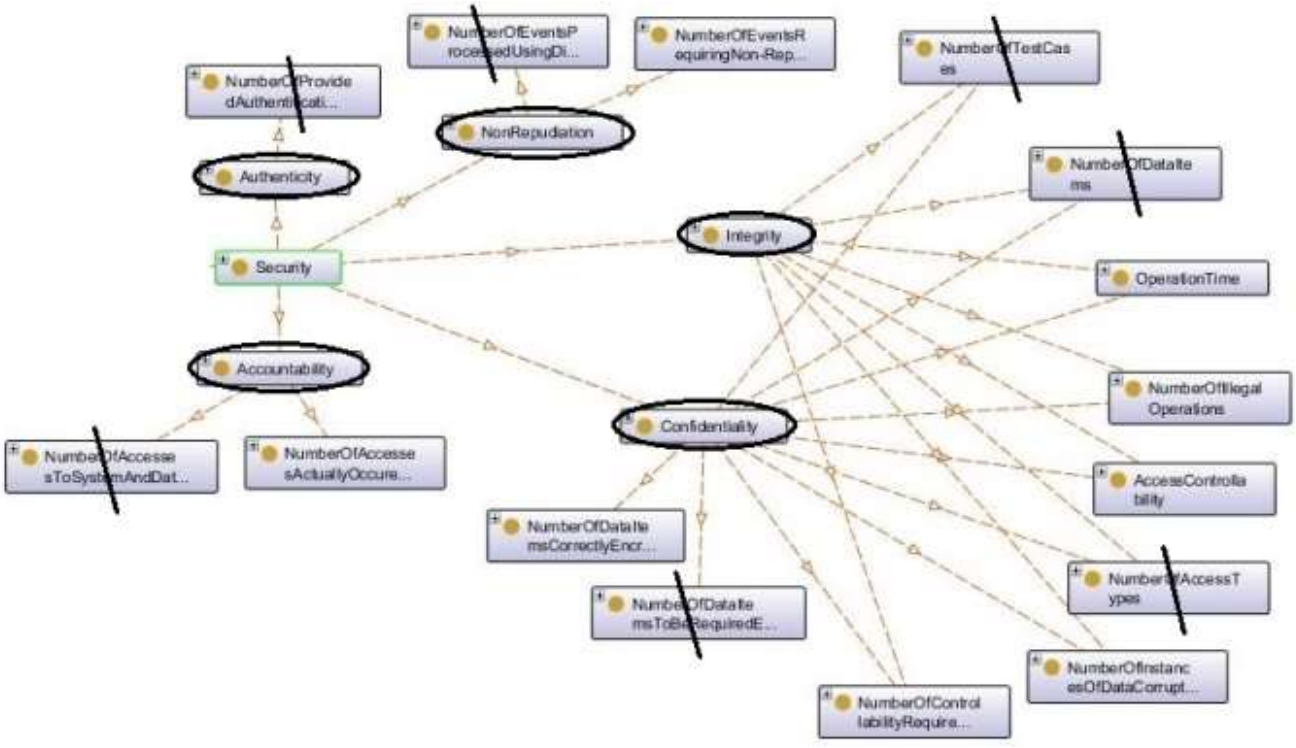

Рис. 3.10 – Візуалізація прогалин у знаннях щодо «Захищеності»

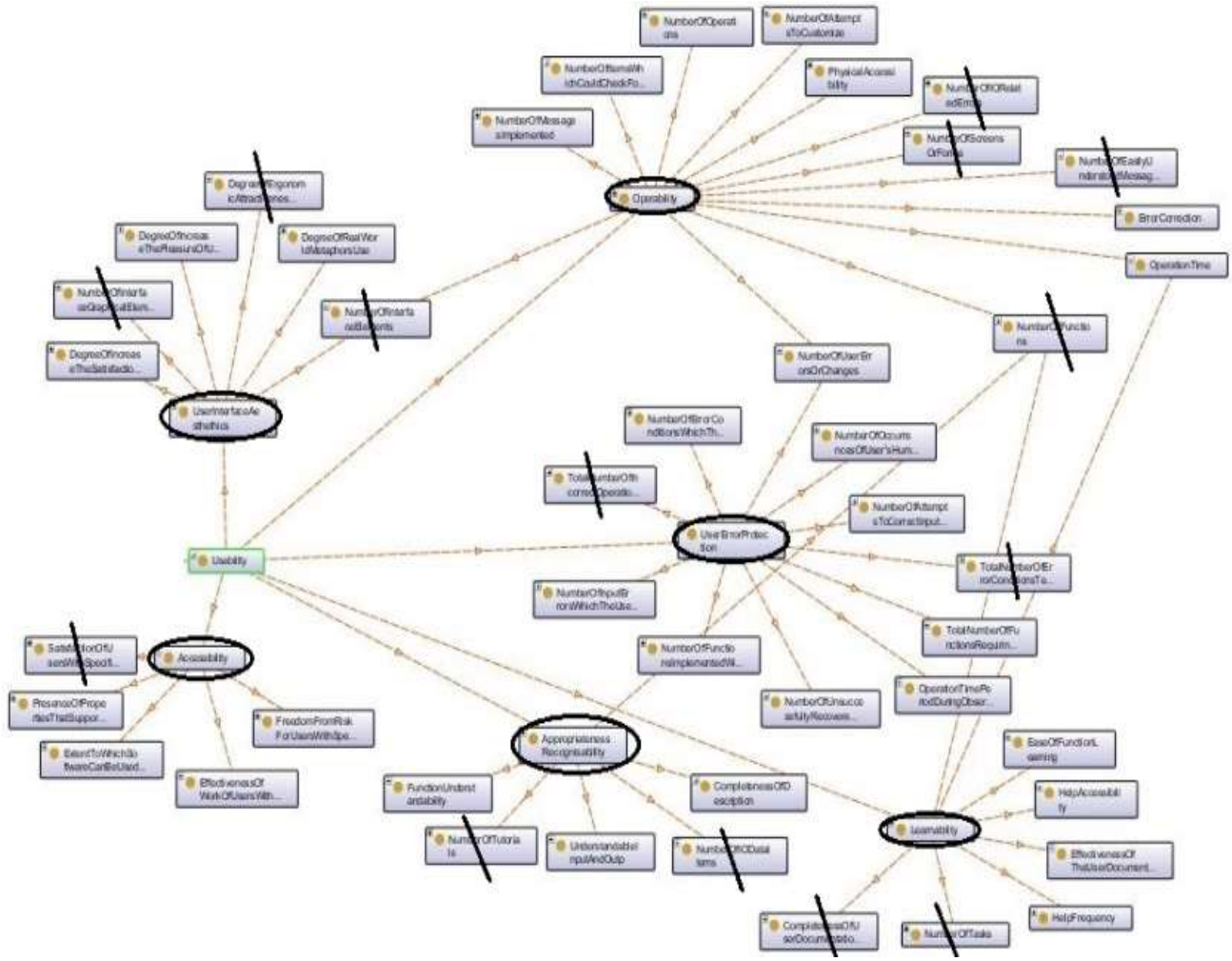

Рис. 3.11 – Візуалізація прогалин у знаннях щодо «Зручності використання»

Числова оцінка рівня достатності наявної у специфікації вимог інформації для визначення нефункційних характеристик становить:

1) для визначення «Надійності»:

$$D_{Rb} = \frac{(4-(\frac{4}{14}+\frac{1}{4}+\frac{4}{5}+\frac{2}{7}))}{4} = 0{,}60 = 60\%$$

2) для визначення «Функційної придатності»:

$$D_{Fs} = \frac{(3-(\frac{2}{4}+\frac{2}{5}+\frac{2}{6}))}{3} = 0{,}59 - 59\%$$

3) для визначення «Ефективності»:

$$D_{Pe} = \frac{(3-(\frac{2}{7}+\frac{5}{14}+\frac{2}{5}))}{3} = 0{,}65 = 65\%$$

4) для визначення «Сумісності»:

$$D_{Cb} = \frac{(2 - (\frac{3}{4} + \frac{2}{5}))}{2} = 0{,}43 = 43\%$$

5) для визначення «Супроводжуваності»:

$$D_{Mb} = \frac{(5 - (\frac{4}{7} + \frac{2}{6} + \frac{3}{6} + \frac{2}{8} + \frac{3}{6}))}{5} = 0{,}57 = 57\%$$

6) для визначення «Можливості переносу»:

$$D_{Pb} = \frac{(3 - (\frac{3}{11} + \frac{2}{4} + \frac{2}{3}))}{3} = 0{,}52 = 52\%$$

7) для визначення «Захищеності»:

$$D_{Scr} = \frac{(5 - (\frac{4}{10} + \frac{3}{8} + \frac{1}{2} + \frac{1}{2} + \frac{1}{1}))}{5} = 0{,}45 = 45\%$$

8) для визначення «Зручності використання»:

$$D_{Ub} = \frac{(6 - (\frac{3}{6} + \frac{3}{8} + \frac{5}{13} + \frac{2}{11} + \frac{3}{6} + \frac{1}{5}))}{6} = 0{,}64 = 64\%$$

Числова оцінка рівня достатності наявної у специфікації вимог інформації для визначення всіх нефункційних характеристик:

$$D = \frac{(8 - (\frac{17}{49} + \frac{6}{15} + \frac{9}{26} + \frac{11}{30} + \frac{5}{9} + \frac{10}{23} + \frac{14}{33} + \frac{7}{18}))}{8} = 0{,}59 = 59\%$$

Отже, розроблений інтелектуальний агент надає висновок: «У аналізованій специфікації вимог до ПЗ недостатньо атрибутів для визначення всіх нефункційних характеристик. Рівень достатності наявної у специфікації вимог інформації для визначення «Надійності» становить 60%. Рівень достатності наявної у специфікації вимог інформації для визначення «Функційної придатності» становить 59%. Рівень достатності наявної у специфікації вимог інформації для визначення «Ефективності» становить 65%. Рівень достатності наявної у специфікації вимог інформації для визначення «Сумісності» становить

43%. Рівень достатності наявної у специфікації вимог інформації для визначення «Супроводжуваності» становить 57%. Рівень достатності наявної у специфікації вимог інформації для визначення «Можливості переносу» становить 52%. Рівень достатності наявної у специфікації вимог інформації для визначення «Захищеності» становить 45%. Рівень достатності наявної у специфікації вимог інформації для визначення «Зручності використання» становить 64%. Рівень достатності наявної у специфікації вимог інформації для визначення всіх нефункційних характеристик становить 59%. Є потреба доповнення цієї специфікації атрибутами, необхідними для обчислення всіх нефункційних характеристик».

Ми рекомендуємо прийняти рівень достатності, який дорівнює 85%, але остаточне рішення щодо необхідного рівня достатності приймається замовником програмного забезпечення. Клієнт може встановити поріг цього рівня, наприклад, 90%, 95% або 100% (навіть для некритичного програмного забезпечення). В результаті аналізу висновків інтелектуального агента замовник вирішив, що рівень достатності інформації вимог розглядуваної специфікації не є допустимим для подальшої роботи над програмним проєктом.

Розробники специфікації внесли зміни у вимоги, враховуючи візуалізовані рекомендації, надані інтелектуальним агентом на основі онтологічного підходу. Після доопрацювання, специфікація знов була проаналізована розробленими інтелектуальними агентами, внаслідок якого було сформовано множини відсутніх атрибутів для кожної нефункційної характеристики:

1) відсутні атрибути для характеристики «Надійність»: Щільність відмов відносно до тестових випадків, Можливість відновлюваності; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» все ще недостатньо для обчислення «Надійності»;

2) відсутній атрибут для характеристики «Функційна придатність»: Функційна адекватність; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої

системи управління виробничими процесами ТОВ «Деймос» все ще недостатньо для обчислення «Функційної придатності»;

3) відсутні атрибути для характеристики «Ефективність»: Кількість оцінок, Розмір бази даних; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» все ще недостатньо для обчислення «Ефективності»;

4) відсутні атрибути для характеристики «Сумісність»: Кількість форматів даних для обміну, Кількість протоколів інтерфейсу; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» все ще недостатньо для обчислення «Сумісності»;

5) відсутні атрибути для характеристики «Супроводжуваність»: Міцність варіабельності, Можливість заміни компонентів, Кількість необхідних діагностичних функцій; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» все ще недостатньо для обчислення «Супроводжуваності»;

6) відсутніх атрибутів для характеристики «Можливість переносу» не виявлено; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» вже достатньо для обчислення «Можливості переносу»;

7) відсутні атрибути для характеристики «Захищеність»: Кількість елементів даних, які потребують шифрування та розшифрування, Кількість подій, що обробляються за допомогою цифрового підпису; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» все ще недостатньо для обчислення «Захищеності»;

8) відсутні атрибути для характеристики «Зручність використання» не виявлені; отже, розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» вже достатньо для обчислення «Зручності використання».

Аналіз впливу кожного елемента множин відсутніх атрибутів на нефункційні характеристики ПЗ та їх підхарактеристики, проведений розробленим інтелектуальним агентом на основі базових онтологій нефункційних характеристик, дав можливість обчислити кількості відсутніх атрибутів для підхарактеристик нефункційних характеристик. Крім цього, на основі такого аналізу розроблений інтелектуальний агент сформував висновок, що на основі наявних атрибутів у специфікації вимог до реального ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос» неможливо обчислити такі підхарактеристики нефункційних характеристик:

1) підхарактеристики «Надійності»: Зрілість, Відновлюваність, тобто недостатньо інформації для визначення 2-х з 4-х підхарактеристик «Надійності» (згідно стандартів ISO 25010:2011 [98] та ISO 25023:2016 [102]);

2) підхарактеристики «Функційної придатності»: Функційна повнота, Функційна доцільність, тобто недостатньо інформації для визначення 2х з 3-х підхарактеристик «Функційної придатності»;

3) підхарактеристики «Ефективності»: Поведінка у часі, Поведінка ресурсів, Ємність, тобто недостатньо інформації для визначення всіх 3-х підхарактеристик «Ефективності»;

4) підхарактеристики «Сумісності»: Взаємодія, тобто недостатньо інформації для визначення 1 з 2-ох підхарактеристик «Сумісності»;

5) підхарактеристики «Супроводжуваності»: Повторне використання, Аналізованість, тобто недостатньо інформації для визначення 2-х з 5-и підхарактеристик «Супроводжуваності»;

6) підхарактеристики «Захищеності»: Конфіденційність, Невідхилюваність, тобто недостатньо інформації для визначення 2-х з 5-и підхарактеристик.

Числова оцінка рівня достатності наявної у специфікації вимог інформації для визначення нефункційних характеристик становить:

1) для визначення «Надійності»:

$$D_{Rb} = \frac{(4-(\frac{1}{14}+\frac{0}{4}+\frac{0}{5}+\frac{1}{7}))}{4} = 0{,}95 = 95\%$$

2) для визначення «Функційної придатності»:

$$D_{Fs} = \frac{(3-(\frac{1}{4}+\frac{0}{5}+\frac{1}{6}))}{3} = 0{,}86 = 86\%$$

3) для визначення «Ефективності»:

$$D_{Pe} = \frac{(3-(\frac{1}{7}+\frac{1}{14}+\frac{1}{5}))}{3} = 0{,}86 = 86\%$$

4) для визначення «Сумісності»:

$$D_{Cb} = \frac{(2-(\frac{0}{4}+\frac{2}{5}))}{2} = 0{,}80 = 80\%$$

5) для визначення «Супроводжуваності»:

$$D_{Mb} = \frac{(5-(\frac{0}{7}+\frac{2}{6}+\frac{1}{6}+\frac{0}{8}+\frac{0}{6}))}{5} = 0{,}90 = 90\%$$

6) для визначення «Можливості переносу»:

$$D_{Pb} = \frac{(3-(\frac{0}{11}+\frac{0}{4}+\frac{0}{3}))}{3} = 1{,}00 = 100\%$$

7) для визначення «Захищеності»:

$$D_{Scr} = \frac{(5-(\frac{1}{10}+\frac{0}{8}+\frac{1}{2}+\frac{0}{2}+\frac{0}{1}))}{5} = 0{,}88 = 88\%$$

8) для визначення «Зручності використання»:

$$D_{Ub} = \frac{(6-(\frac{0}{6}+\frac{0}{8}+\frac{0}{13}+\frac{0}{11}+\frac{0}{6}+\frac{0}{5}))}{6} = 1{,}00 = 100\%$$

Числова оцінка рівня достатності наявної у специфікації вимог інформації для визначення всіх нефункційних характеристик:

$$D = \frac{(8 - (\frac{0}{49} + \frac{2}{15} + \frac{3}{26} + \frac{2}{30} + \frac{2}{9} + \frac{2}{23} + \frac{2}{33} + \frac{0}{18}))}{8} = 0{,}91 = 91\%$$

Отже, розроблений інтелектуальний агент надає висновок: «У аналізованій специфікації вимог до ПЗ недостатньо атрибутів для визначення всіх нефункційних характеристик. Рівень достатності наявної у специфікації вимог інформації для визначення «Надійності» становить 95%. Рівень достатності наявної у специфікації вимог інформації для визначення «Функційної придатності» становить 86%. Рівень достатності наявної у специфікації вимог інформації для визначення «Ефективності» становить 86%. Рівень достатності наявної у специфікації вимог інформації для визначення «Сумісності» становить 80%. Рівень достатності наявної у специфікації вимог інформації для визначення «Супроводжуваності» становить 90%. Рівень достатності наявної у специфікації вимог інформації для визначення «Можливості переносу» становить 100%. Рівень достатності наявної у специфікації вимог інформації для визначення «Захищеності» становить 88%. Рівень достатності наявної у специфікації вимог інформації для визначення «Зручності використання» становить 100%. Рівень достатності наявної у специфікації вимог інформації для визначення всіх нефункційних характеристик становить 91%. Є потреба доповнення цієї специфікації атрибутами, необхідними для обчислення всіх нефункційних характеристик».

В результаті аналізу висновків інтелектуального агента замовник вирішив, що рівень достатності інформації вимог розглядуваної специфікації є допустимим для подальшої роботи над програмним проєктом.

Як показав проведений перший експеримент, доповнення вимог до програмної системи підвищення безпеки програмного забезпечення комп'ютерних систем необхідними атрибутами підвищило: рівень достатності інформації для визначення «Надійності» на 35% – з 60% до 95% (рис. 3.12); рівень достатності

інформації для визначення «Функційної придатності» на 27% – з 59% до 86% (рис. 3.12); рівень достатності інформації для визначення «Ефективності» на 21% – з 65% до 86% (рис. 3.12); рівень достатності інформації для визначення «Сумісності» на 37% – з 43% до 80% (рис. 3.12); рівень достатності інформації для визначення «Супроводжуваності» на 33% – з 57% до 90% (рис. 3.12); рівень достатності інформації для визначення «Можливості переносу» на 48% – з 52% до 100% (рис. 3.12); рівень достатності інформації для визначення «Захищеності» на 43% – з 45% до 88% (рис. 3.12); рівень достатності інформації для визначення «Зручності використання» на 36% – з 64% до 100% (рис. 3.12); рівень достатності інформації для визначення всіх нефункційних характеристик у специфікації вимог на 32% – з 59% до 91%.

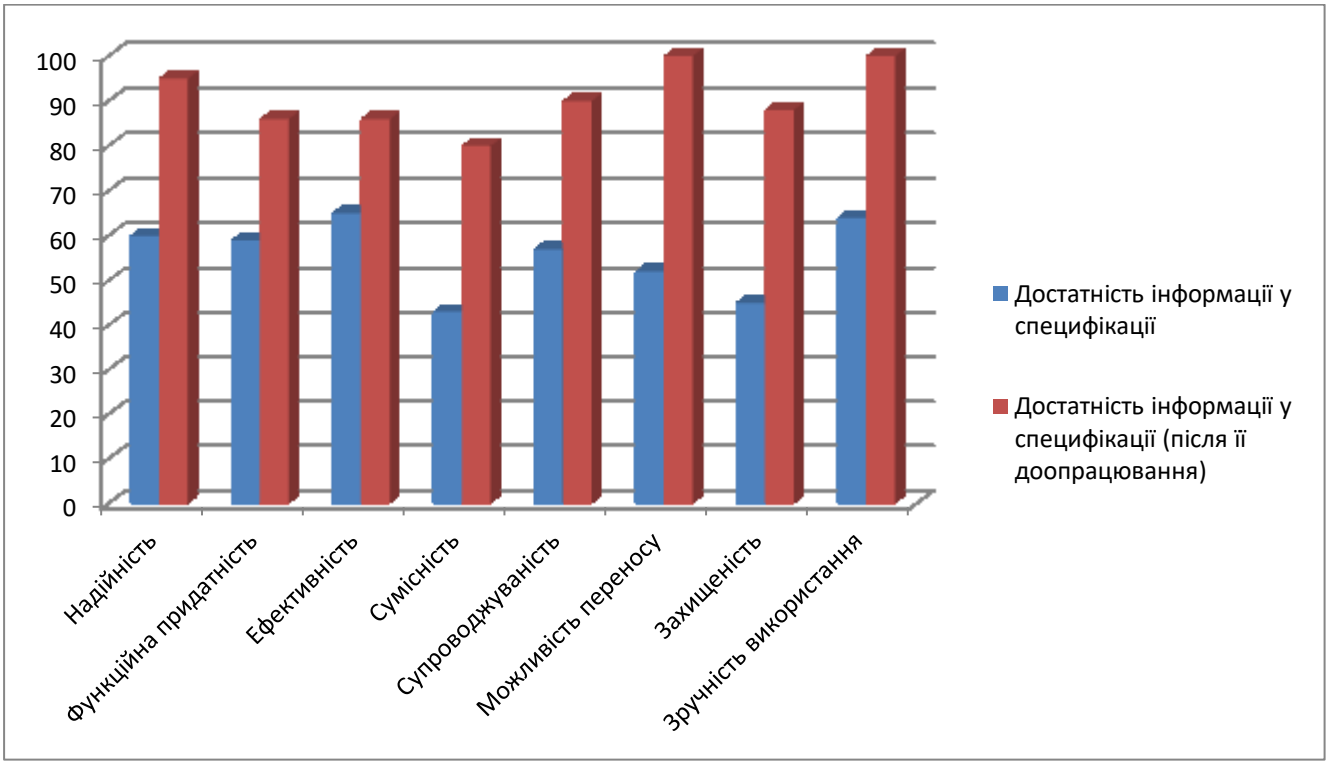

Рис. 3.12 – Приріст достатності інформації у специфікаціях вимог після врахування розробниками специфікацій висновків, наданих розробленими інтелектуальним агентами (експеримент 1)

Для проведення *другого експерименту* був виконаний автоматичний аналіз (парсинг) вимог до програмної системи підвищення безпеки програмного забезпечення комп’ютерних систем.

Відповідно до розробленого методу діяльності, розроблені інтелектуальні агенти надали множину відсутніх показників у вимогах до ПЗ системи підвищення безпеки програмного забезпечення комп'ютерних систем:

– для метрик складності програмного проєкту: Quantity Of Modules, Quantity Of Procedures To Read From Data Structure;

– для метрик якості програмного проєкту: Quantity Of Modules, Type Of Module Input Data, Type Of Module Output Data, Quantity Of Code Lines, Quantity Of Potential Access To Global Variables, Project Duration;

– для метрик складності ПЗ: Quantity Of Modules, Quantity Of Code Lines, Total Quantity Of Operators, Total Quantity Of Operands;

– для метрик якості ПЗ: Quantity Of Code Lines, Project Duration, Cost Of One Line.

Інтелектуальний агент виявляє метрики, які не можуть бути обчислені на основі показників, наявних у специфікації вимог до програмної системи підвищення безпеки програмного забезпечення комп'ютерних систем: 4 (з 4-х) метрики складності програмного проєкту, 4 (з 5-и) метрики якості програмного проєкту, 6 (з 6-и) метрик складності ПЗ, 8 (з 9-и) метрик якості ПЗ не можуть бути обчислені на основі наявних показників.

Числова оцінка рівня достатності метричної інформації у специфікації вимог становить:

$$D = \frac{1}{24} \cdot \left(\begin{array}{c} 24 - (\frac{1}{5} + \frac{1}{2} + \frac{1}{5} + \frac{2}{4} + \frac{0}{2} + \frac{2}{5} + \frac{1}{2} + \frac{2}{3} + \frac{1}{2} + \frac{1}{1} + \frac{3}{5} + \frac{1}{3} + \\ + \frac{1}{3} + \frac{2}{2} + \frac{2}{3} + \frac{2}{2} + \frac{2}{3} + \frac{2}{2} + \frac{2}{4} + \frac{2}{2} + \frac{2}{3} + \frac{0}{5} + \frac{1}{2} + \frac{1}{2}) \end{array} \right) = 0{,}4486 = 44{,}86\%$$

Так, інтелектуальний агент надає наступний висновок: «Наявних показників в аналізованих вимогах недостатньо для визначення 22 метрик (з 24). Рівень достатності метричної інформації аналізованих вимог до ПЗ становить 44,86%. Є потреба у вдосконаленні вимог шляхом доповнення показників».

ІА надає також візуалізацію відсутніх показників. Така візуалізація надає користувачу список відсутніх показників; відображає, на які метрики впливає той чи інший показник; показує, які показники першочергово слід внести до вимог з

метою збільшення рівня достатності їх інформації. Використані онтології проявляють кореляцію метрик за показниками, тобто підказують користувачу, які показники потрібно внести у вимоги першочергово для більш стрімкого збільшення достатності метричної інформації.

Рішення про потребу у вдосконаленні вимог приймає замовник на основі наданого висновку про рівень достатності метричної інформації. Замовник системи підвищення безпеки програмного забезпечення комп'ютерних систем прийняв рішення, що, враховуючи низький рівень достатності метричної інформації, вимоги потребують вдосконалення (доповнення показників).

Після вдосконалення вимог до програмної системи підвищення безпеки програмного забезпечення комп'ютерних систем був виконаний повторний автоматичний аналіз (парсинг) вимог.

Відповідно до розробленого методу діяльності, реалізований інтелектуальний агент надава множину відсутніх показників у вдосконалених вимогах до ПЗ системи підвищення безпеки програмного забезпечення комп'ютерних систем:

– для метрик складності програмного проєкту: Quantity Of Procedures To Read From Data Structure;

– для метрик якості програмного проєкту: Type Of Module Input Data, Quantity Of Potential Access To Global Variables;

– для метрик складності ПЗ: Total Quantity Of Operands;

– для метрик якості ПЗ: Cost Of One Line.

Інтелектуальний агент на основі онтологічного підходу виявляє метрики, які не можуть бути обчислені на основі наявних показників у специфікації вимог до програмної системи підвищення безпеки програмного забезпечення комп'ютерних систем: 1 (з 4-х) метрики складності програмного проєкту, 2 (з 5-и) метрики якості програмного проєкту, 2 (з 6-и) метрик складності ПЗ, 3 (з 9-и) метрик якості ПЗ не можуть бути обчислені на основі наявних показників.

Числова оцінка рівня достатності метричної інформації у специфікації вимог становить:

$$D = \frac{1}{24} \cdot \left(24 - (\frac{0}{5} + \frac{0}{2} + \frac{0}{5} + \frac{1}{4} + \frac{0}{2} + \frac{1}{5} + \frac{1}{2} + \frac{0}{3} + \frac{0}{2} + \frac{0}{1} + \frac{1}{5} + \frac{0}{3} + \frac{0}{3} + \frac{0}{2} + \frac{1}{3} + \frac{0}{2} + \frac{0}{3} + \frac{1}{2} + \frac{1}{4} + \frac{0}{2} + \frac{1}{3} + \frac{0}{5} + \frac{0}{2} + \frac{0}{2}) \right) = 0{,}8931 = 89{,}31\%$$

Розроблений інтелектуальний агент на основі онтологічного підходу надає наступний висновок: «Наявних показників в аналізованих вимогах недостатньо для визначення 8 метрик (з 24). Рівень достатності метричної інформації аналізованих вимог до ПЗ становить 89,31%. Є потреба в доповненні вимог показниками для визначення метрик». ІА надає також візуалізацію відсутніх показників.

Замовник програмної системи підвищення безпеки програмного забезпечення комп'ютерних систем прийняв рішення, що, враховуючи рівень достатності метричної інформації більше 85%, вимоги не потребують подальшого вдосконалення.

Як показав проведений другий експеримент, доповнення вимог до програмної системи підвищення безпеки програмного забезпечення комп'ютерних систем п'ятьма показниками підвищило рівень достатності метричної інформації на 44,45% – з 44,86% до 89,31% .

Для проведення *третього експерименту* використовувалась специфікація вимог до реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, яку розробляла одна з софтверних компаній м.Хмельницького. В результаті роботи запропонованих інтелектуальних агентів було сформовано множину відсутніх у зазначеній специфікації атрибутів якості: Number of Functions, Number of Data Items, Number of Tasks, Number of Evaluations, Number of Failures, Number of IO Related Errors, Mean Amount of Throughput, Number of Tutorials, Number of IO Data Items, Completeness of User Documentation and/or Help Facility, Number of Screens or Forms, Number of Interface Elements, Number of User Errors or Changes, Number of Interface Graphical Elements, Degree of Ergonomic Attractiveness, Number of Faults, Product Size, Number of Interface Protocols, Number of Access Types, Number of Controllability Requirements, Number of Modules, Number of Variables, Number of Checkpoints, Number of Installation Steps, Ease of Installation. Отже, оскільки

множина відсутніх атрибутів не порожня, то розроблений інтелектуальний агент надає висновок, що інформації у специфікації вимог до реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, недостатньо.

Аналіз впливу кожного елемента множин відсутніх атрибутів на підхарактеристики та характеристики якості ПЗ, проведений розробленим інтелектуальним агентом, дав можливість обчислити кількості відсутніх атрибутів для підхарактеристик якості ПЗ. Крім цього, на основі такого аналізу розроблений інтелектуальний агент сформував висновок, що на основі наявних атрибутів у специфікації вимог до реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, неможливо обчислити 24 з 31 підхарактеристики нефункційних характеристик.

Після цього розроблений інтелектуальний агент розрахував числові оцінки рівня достатності інформації у специфікації вимог до ПЗ для визначення нефункційних характеристик:

1) для визначення «Надійності»:

$$D_{Rb} = \frac{(4 - (\frac{3}{14} + \frac{0}{4} + \frac{2}{5} + \frac{0}{7}))}{4} = 0{,}85 = 85\%$$

2) для визначення «Функційної придатності»:

$$D_{Fs} = \frac{(3 - (\frac{1}{4} + \frac{1}{5} + \frac{1}{6}))}{3} = 0{,}79 = 79\%$$

3) для визначення «Ефективності»:

$$D_{Pe} = \frac{(3 - (\frac{3}{7} + \frac{3}{14} + \frac{2}{5}))}{3} = 0{,}65 = 65\%$$

4) для визначення «Сумісності»:

$$D_{Cb} = \frac{(2 - (\frac{3}{4} + \frac{1}{5}))}{2} = 0{,}68 = 68\%$$

5) для визначення «Супроводжуваності»:

$$D_{Mb} = \frac{(5 - (\frac{4}{7} + \frac{0}{6} + \frac{2}{6} + \frac{1}{8} + \frac{1}{6}))}{5} = 0{,}76 = 76\%$$

6) для визначення «Можливості переносу»:

$$D_{Pb} = \frac{(3 - (\frac{3}{11} + \frac{3}{4} + \frac{2}{3}))}{3} = 0{,}44 = 44\%$$

7) для визначення «Захищеності»:

$$D_{Scr} = \frac{(5 - (\frac{3}{10} + \frac{3}{8} + \frac{0}{2} + \frac{0}{2} + \frac{0}{1}))}{5} = 0{,}87 = 87\%$$

8) для визначення «Зручності використання»:

$$D_{Ub} = \frac{(6 - (\frac{3}{6} + \frac{3}{8} + \frac{5}{13} + \frac{1}{11} + \frac{3}{6} + \frac{0}{5}))}{6} = 0{,}69 = 69\%$$

Розроблений інтелектуальний агент на основі онтологічного підходу також проводить обчислення числової оцінки рівня достатності інформації у специфікації вимог для визначення всіх нефункційних характеристик-складових якості ПЗ:

$$D = \frac{(8 - (\frac{15}{49} + \frac{3}{15} + \frac{8}{26} + \frac{5}{30} + \frac{4}{9} + \frac{6}{23} + \frac{8}{33} + \frac{8}{18}))}{8} = 0{,}70 = 70\%$$

Отже, розроблений інтелектуальний агент надає висновок: «У аналізованій специфікації вимог до медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, недостатньо атрибутів для визначення всіх нефункційних характеристик. Рівень достатності наявної інформації для визначення «Функційної придатності» становить 79%. Рівень достатності наявної інформації для визначення «Ефективності» становить 65%. Рівень достатності наявної інформації для визначення «Зручності використання» становить 69%. Рівень достатності наявної інформації для визначення «Надійності» становить 85%. Рівень достатності наявної інформації для визначення «Сумісності» становить 68%. Рівень достатності наявної

інформації для визначення «Захищеності» становить 87%. Рівень достатності наявної інформації для визначення «Супроводжуваності» становить 76%. Рівень достатності наявної інформації для визначення «Можливості переносу» становить 44%. Рівень достатності наявної нформації у специфікації вимог для визначення нефункційних характеристик-складових якості ПЗ становить 70%. Є потреба доповнення вхідної інформації (зокрема, бізнес-вимог), що регламентує атрибути якості, а також доповнення цієї специфікації атрибутами, необхідними для обчислення всіх нефункційних характеристик».

Для забезпечення достатності інформації у специфікації вимог до реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, для визначення характеристик якості слід виконати доповнення специфікації вищезазначеними відсутніми атрибутами, відповідно для цього слід виконати доповнення вхідної інформації (зокрема, бізнес-вимог), що регламентує такі атрибути якості. Крім списку відсутніх атрибутів, розроблений інтелектуальний агент надає також візуалізацію прогалин у знаннях щодо характеристик якості ПЗ.

Відбулось доповнення інформації у специфікації вимог до реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, наступними атрибутами: Number of Functions, Number of Data Items, Mean Amount of Throughput, Number of Interface Elements, Number of Tutorials, Number of Interface Graphical Elements, Number of Failures, Number of Faults, Number of Interface Protocols, Number of Access Types, Number of Modules, Number of Installation Steps, Ease of Installation.

Після цього агент знову обчислив кількості відсутніх атрибутів для підхарактеристик нефункційних характеристик (після доповнення), а також сформував висновок, що на основі наявних після доповнення атрибутів у специфікації вимог до реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, неможливо обчислити 12 з 31 підхарактеристик якості.

Розроблений агент надав наступні числові оцінки рівня достатності інформації у специфікації вимог (після доповнення) для визначення нефункційних характеристик-складових якості реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування:

1) для визначення «Надійності»:

$$D_{Rb} = \frac{(4-(1\frac{3}{14}+\frac{0}{4}+\frac{0}{5}+\frac{0}{7}))}{4} = 0{,}98 = 98\%$$

2) для визначення «Функційної придатності»:

$$D_{Fs} = \frac{(3-(\frac{0}{4}+\frac{0}{5}+\frac{0}{6}))}{3} = 1{,}00 = 100\%$$

3) для визначення «Ефективності»:

$$D_{Pe} = \frac{(3-(\frac{2}{7}+\frac{2}{14}+\frac{0}{5}))}{3} = 0{,}86 = 86\%$$

4) для визначення «Сумісності»:

$$D_{Cb} = \frac{(2-(\frac{0}{4}+\frac{0}{5}))}{2} = 1{,}00 = 100\%$$

5) для визначення «Супроводжуваності»:

$$D_{Mb} = \frac{(5-(\frac{1}{7}+\frac{0}{6}+\frac{0}{6}+\frac{0}{8}+\frac{1}{6}))}{5} = 0{,}94 = 94\%$$

6) для визначення «Можливості переносу»:

$$D_{Pb} = \frac{(3-(\frac{0}{11}+\frac{0}{4}+\frac{0}{3}))}{3} = 1{,}00 = 100\%$$

7) для визначення «Захищеності»:

$$D_{Scr} = \frac{(5-(\frac{1}{10}+\frac{1}{8}+\frac{0}{2}+\frac{0}{2}+\frac{0}{1}))}{5} = 0{,}96 = 96\%$$

8) для визначення «Зручності використання»:

$$D_{Ub} = \frac{(6 - (\frac{1}{6} + \frac{2}{8} + \frac{3}{13} + \frac{1}{11} + \frac{1}{6} + \frac{0}{5}))}{6} = 0{,}85 = 85\%$$

Числова оцінка рівня достатності інформації у специфікації вимог (після доповнення) до реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності для визначення всіх нефункційних характеристик, наданої пораненому під час транспортування, надана розробленим інтелектуальним агентом, становить:

$$D = \frac{(8 - (\frac{8}{49} + \frac{0}{15} + \frac{4}{26} + \frac{1}{30} + \frac{0}{9} + \frac{2}{23} + \frac{2}{33} + \frac{0}{18}))}{8} = 0{,}94 = 94\%$$

Отже, розроблений інтелектуальний агент надає висновок: «У аналізованій специфікації вимог до медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, все ще недостатньо атрибутів для визначення 5 нефункційних характеристик. Рівень достатності наявної після доповнення інформації для визначення «Функційної придатності» становить 100%. Рівень достатності наявної після доповнення інформації для визначення «Ефективності» становить 86%. Рівень достатності наявної після доповнення інформації для визначення «Зручності використання» становить 85%. Рівень достатності наявної після доповнення інформації для визначення «Надійності» становить 98%. Рівень достатності наявної після доповнення інформації для визначення «Сумісності» становить 100%. Рівень достатності наявної після доповнення інформації для визначення «Захищеності» становить 96%. Рівень достатності наявної після доповнення інформації для визначення «Супроводжуваності» становить 94%. Рівень достатності наявної після доповнення інформації для визначення «Можливості переносу» становить 100%. Рівень достатності наявної після доповнення вхідної інформації та інформації у специфікації вимог для визначення всіх нефункційних характеристик становить 94%. Є потреба доповнення вхідної інформації (зокрема, бізнес-вимог), що регламентує атрибути якості, а також

доповнення цієї специфікації атрибутами, необхідними для обчислення всіх характеристик якості».

Замовника влаштував такий рівень достатності інформації у специфікації вимог, тому додавання атрибутів у вхідну інформацію та у специфікацію вимог більше не відбувалось.

Діаграма, яка відображає рівень достатності інформації у специфікації вимог для визначення нефункційних характеристик реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, до та після доповнення, представлена на рис. 3.13.

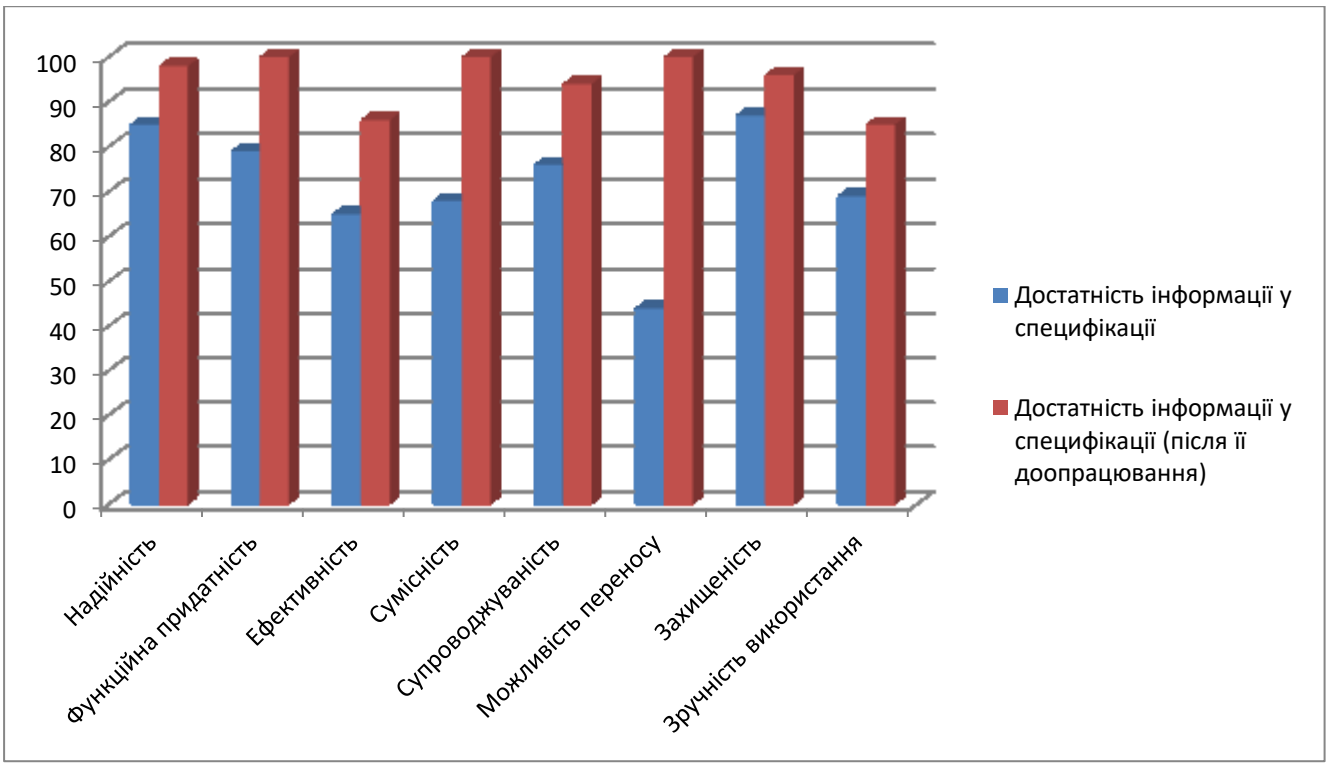

Рис. 3.13 – Приріст достатності інформації у специфікаціях вимог після врахування розробниками специфікацій висновків, наданих розробленими інтелектуальним агентами (експеримент 3)

Приріст рівня достатності інформації у специфікації вимог для визначення характеристик якості реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, складає від 9% (для Захищеності) до 56% (для Можливості

переносу). Приріст рівня достатності інформації у специфікації вимог для визначення якості реального медичного ПЗ інформаційно-аналітичної системи для обліку лікувально-діагностичної діяльності, наданої пораненому під час транспортування, складає 24%. Отже, розроблений інтелектуальний агент дав змогу вдосконалити інформацію специфікації вимог до медичного ПЗ, а саме дозволив підвищити рівень її достатності для визначення якості ПЗ на 24%.

Прирости достатності інформації у специфікаціях вимог для визначення всіх нефункційних характеристик-складових якості ПЗ після врахування розробниками специфікацій висновків, наданих розробленими інтелектуальним агентами, отримані в результаті проведених експериментів 1-3, представлені на рис. 3.14.

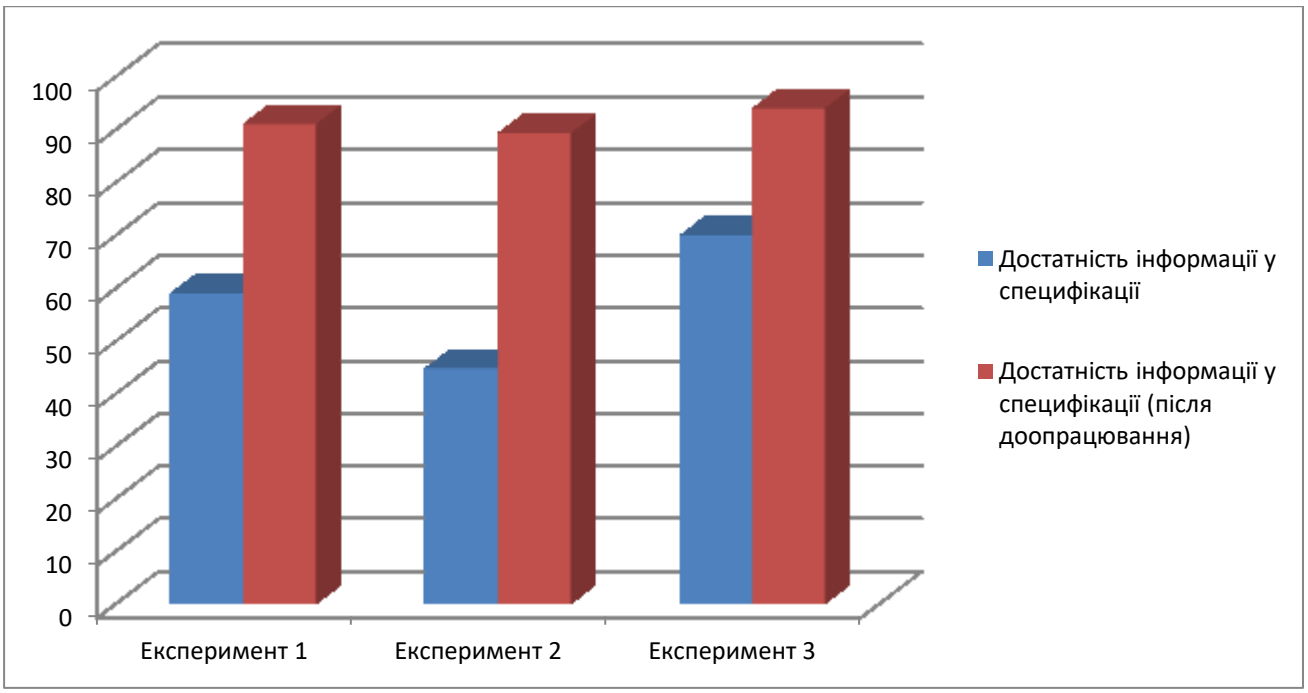

Рис. 3.14 – Приріст достатності інформації у специфікаціях вимог для визначення всіх нефункційних характеристик-складових якості ПЗ після врахування розробниками специфікацій висновків, наданих розробленими інтелектуальним агентами (експерименти 1-3)

Під час експериментів інтелектуальними агентами було встановлено, в яких специфікаціях є недостатність інформації для визначення всіх нефункційних характеристик та/або метрик ПЗ. Якщо загальний рівень достатності інформації у специфікації для визначення всіх нефункційних характеристик та/або метрик ПЗ

не задовольняв замовника ПЗ, то було рекомендовано доповнити цю специфікацію атрибутами та/або показниками, необхідними для обчислення нефункційних характеристик та/або метрик ПЗ (із наданням списку та візуалізацією відсутніх атрибутів та/або показників).

3.4. Висновки

Для автоматизації семантичного розбору природомовної специфікації з метою перевірки відповідності її нефункційних вимог потребам замовника необхідно виконати її формалізацію, наприклад, з використанням онтологій. Для такої формалізації було запропоновано використати розроблені у [1] онтологію для специфікацій вимог до ПЗ, онтологію для нефункційних характеристик-складових якості ПЗ, а також онтологію для метричного аналізу, які стали основою (відомими фактами) інтелектуальних агентів для семантичного парсингу природомовних специфікацій на предмет пошуку атрибутів та/або показників, необхідних для визначення нефункційних характеристик-складових якості ПЗ та метрик.

У даному розділі удосконалено метод діяльності інтелектуального агента на основі онтологічного підходу для оцінювання інформації щодо нефункційних характеристик у специфікаціях вимог до ПЗ, який дає можливість: сформувати висновок про достатність або недостатність інформації щодо нефункційних характеристик у специфікаціях вимог до реального ПЗ і про необхідність доповнення специфікації атрибутами; оцінити рівень достатності інформації у специфікації вимог до реального ПЗ для визначення кожної нефункційної характеристики ПЗ та для визначення всіх нефункційних характеристик.

Розроблено інтелектуальні агенти для семантичного парсингу природомовних специфікацій, які виконують парсинг специфікації, визначають кількість та відсоток відсутніх атрибутів та/або показників, відображають, яких атрибутів та/або показників не вистачає для тієї чи іншої підхарактеристики нефункційної характеристики та/або метрики, а також формують реальну онтологію для

нефункційних характеристик та/або метрик, яка може бути використана ОВІА для оцінки початкових етапів життєвого циклу ПЗ шляхом оцінювання достатності інформації для визначення нефункційних характеристик та метрик.

Крім того, розроблені ІА для семантичного парсингу природомовних специфікацій забезпечують: автоматизацію та прискорення семантичного аналізу специфікацій; зазначення «вузьких місць» (прогалин знань) специфікацій та демонстрація, які вимоги потребують переробки для подальшої оцінки нефункційних характеристик-складових якості ПЗ; навчання для розробників специфікацій та інженерів вимог (вони можуть бачити свої помилки та прогалини у вимогах); безкоштовний доступ в будь-який час без будь-якої реєстрації. Обмеженням розроблених ІА є пошук лише атрибутів, визначених стандартом ISO 25023:2016 як необхідні для обчислення нефункційних характеристик-складових якості ПЗ, а також показників, визначених галузевими публікаціями як необхідні для обчислення метрик складності та якості ПЗ.

Реалізовано інтелектуальні агенти на основі онтологічного підходу для оцінювання початкових етапів життєогов циклу ПЗ, які здійснюють оцінювання достатності інформації у специфікації вимог для визначення всіх нефункційних характеристик-складових якості ПЗ, а також для розрахунку метрик якості та складності ПЗ. Реалізовані інтелектуальні агенти забезпечують висновок про достатність або недостатність інформації у специфікації. Крім цього, вони надають числові оцінки рівня достатності інформації для визначення кожної нефункційної характеристики ПЗ та/або метрики та для визначення всіх нефункційних характеристик-складових якості ПЗ та/або метрик разом. Агентами також формується список атрибутів та показників, якими варто доповнити специфікацію вимог для підвищення рівня достатності її інформації, та візуалізація прогалин у знаннях про всі нефункційні характеристики-складові якості ПЗ та метрики якості і складності ПЗ. Отже, реалізовані інтелектуальні агенти забезпечують автоматизацію аналізу специфікацій вимог до ПЗ на предмет достатності їх інформації. Таким чином, представлені агенти дозволяють частково усунути людину з процесів опрацювання інформації та здобуття знань.

Під час експериментів інтелектуальними агентами було встановлено, в яких специфікаціях є недостатність інформації для визначення всіх нефункційних характеристик та/або метрик ПЗ. Якщо загальний рівень достатності інформації у специфікації для визначення всіх нефункційних характеристик та/або метрик ПЗ не задовольняв замовника ПЗ, то було рекомендовано доповнити цю специфікацію атрибутами та/або показниками, необхідними для обчислення нефункційних характеристик та/або метрик ПЗ (із наданням списку та візуалізацією відсутніх атрибутів та/або показників).

Всі результати функціонування реалізованих інтелектуальних агентів на основі онтологічного підходу в комплексі забезпечують підвищення рівня достатності інформації у специфікації вимог до ПЗ для визначення нефункційних характеристик-складових якості ПЗ та метрик складності і якості ПЗ. Крім цього, результати функціонування реалізованих інтелектуальних агентів спрямовані на уникнення втрат істотної інформації і мінімізацію виникнення помилок на ранніх етапах життєвого циклу ПЗ.

РОЗДІЛ 4.
АГЕНТНО-ОРІЄНТОВАНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОЦІНЮВАННЯ ПОЧАТКОВИХ ЕТАПІВ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ

4.1. Структура агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу

Агентно-орієнтована інформаційна технологія (АОІТ) для оцінювання початкових етапів життєвого циклу ПЗ розв’язує задачу оцінювання достатності інформації вимог для визначення нефункційних характеристик-складових якості ПЗ. Достатність інформації вимог для визначення нефункційних характеристик визначається наявністю у специфікації всіх атрибутів, необхідних для визначення нефункційних характеристик.

Агентно-орієнтована інформаційна технологія (АОІТ) для оцінювання початкових етапів життєвого циклу ПЗ може бути застосована як розробниками вимог, так і замовниками програмних проєктів одразу, щойно буде готова перша версія специфікації вимог, в якій міститься інформація про нефункційні характеристики, і доти, доки є потреба перевіряти вимоги на предмет їх достатності (враховуючи сьогоднішню мінливість вимог, такі перевірки можуть виконуватись протягом практично всього життєвого циклу проєкту).

Враховуючи проведене моделювання діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ, розроблені методи діяльності інтелектуальних агентів на основі онтологічного підходу для семантичного аналізу (парсингу) специфікацій вимог до ПЗ та для оцінювання початкових етапів життєвого циклу ПЗ, розроблено *агентно-орієнтовану інформаційну технологію для оцінювання початкових етапів життєвого циклу ПЗ* (як сукупність процесів, що використовує засоби та методи накопичення, обробки і передачі первинної інформації для отримання інформації нової якості про стан об’єкту, процесу або явища) – рис. 4.1.

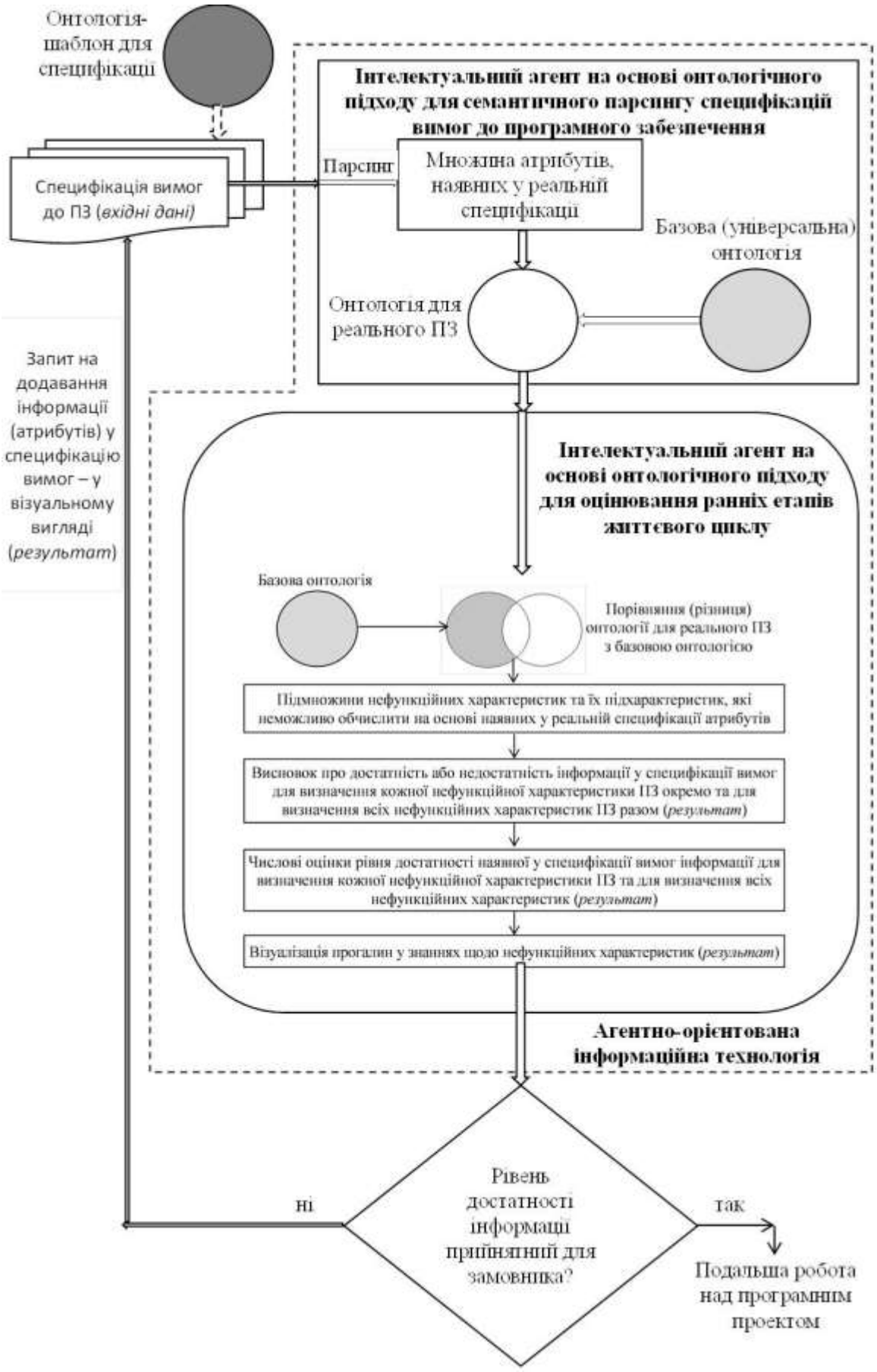

Рис. 4.1 – Агентно-орієнтована інформаційна технологія для оцінювання початкових етапів життєвого циклу ПЗ

Мета агентно-орієнтованої інформаційної технології – автоматизація кількісного оцінювання рівня достатності інформації вимог для визначення нефункційних характеристик з метою мінімізації впливу людського фактору та з метою спрощення виконання зазначеного оцінювання як розробником, так і замовником. Інформаційна технологія дозволяє виявити необхідність формування повторного запиту на додавання атрибутів, необхідних для визначення нефункційних характеристик, та, за необхідності, сформувати та візуалізувати його вміст.

Пропонована інформаційна технологія автоматично опрацьовує наявні знання (нефункційні вимоги специфікації) та формує нові знання (висновки про достатність інформації, про рівень достатності інформації, рекомендації щодо підвищення рівня достатності інформації у специфікації вимог).

Етапи процесу опрацювання інформації розробленою агентно-орієнтованою інформаційною технологією:

1) автоматичний парсинг специфікації вимог до ПЗ (вхідні дані) на предмет пошуку атрибутів, необхідних для визначення нефункційних характеристик ПЗ – за методом діяльності інтелектуального агента на основі онтологічного підходу для семантичного аналізу (парсингу) специфікацій вимог до ПЗ (підрозділ 3.1);

2) формування множини атрибутів, наявних у реальній специфікації вимог;

3) формування онтології для реального ПЗ (формули (2.48)-(2.55), підрозділ 2.2) – з використанням розроблених моделей нефункційних характеристик-складових якості ПЗ (формули (2.1)-(2.39), підрозділ 2.1) та онтологічних моделей нефункційних характеристик ПЗ (формули (2.40)-(2.47), підрозділ 2.2), які використовуються інтелектуальним агентом як відомі факти;

4) порівняння базової онтології для нефункційних характеристик (формули (2.40)-(2.47), підрозділ 2.2) із отриманою онтологією для реального ПЗ (формули (2.48)-(2.55), підрозділ 2.2) – з використанням правил (формули (2.56)-(2.64), підрозділ 2.3), моделі діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до програмного забезпечення (підрозділ 2.3) та за методом діяльності інтелектуальних агентів на основі

онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення (підрозділ 3.2);

5) формування підмножини нефункційних характеристик та їх підхарактеристик, які неможливо обчислити на основі наявних у реальній специфікації атрибутів;

6) формування висновку про достатність або недостатність інформації у специфікації вимог для визначення кожної нефункційної характеристики ПЗ окремо та для визначення всіх нефункційних характеристик ПЗ разом (результат функціювання AOIT);

7) обчислення числових оцінок рівня достатності наявної у специфікації вимог інформації для визначення кожної нефункційної характеристики ПЗ та для визначення всіх нефункційних характеристик (результат функціювання AOIT);

8) візуалізація прогалин у знаннях щодо нефункційних характеристик (результат функціювання AOIT);

9) формування запиту на додавання необхідних атрибутів у специфікацію (за умови формування висновку про недостатність інформації у специфікації) з візуалізованими підказками, які саме атрибути необхідно додати (результат функціювання AOIT).

Як видно з рис. 4.1, агентно-орієнтована інформаційна технологія грунтується на: інтелектуальному агенті на основі онтологічного підходу для семантичного парсингу специфікацій вимог до ПЗ та інтелектуальному агенті на основі онтологічного підходу для оцінювання ранніх етапів життєвого циклу.

Інтелектуальний агент на основі онтологічного підходу для семантичного парсингу специфікацій вимог до ПЗ приймає на вхід специфікацію вимог до ПЗ та проводить автоматичний парсинг специфікації вимог до ПЗ на предмет пошуку атрибутів, необхідних для визначення нефункційних характеристик ПЗ. Шаблон специфікації вимог до ПЗ, який демонструє всі необхідні атрибути для визначення нефункційних характеристик, а також їх місце розташування у специфікації, пропонується користувачу у вигляді базової онтології предметної галузі «Інженерія програмного забезпечення» (частина «Специфікація вимог до ПЗ

(атрибути)»), розробленої у [1] на основі стандарту ISO 29148. Після проведення парсингу специфікації формується множина атрибутів, наявних у реальній специфікації. Використовуючи розроблену у [1] базову (універсальну) онтологію для нефункційних характеристик (відомі факти) та множину наявних атрибутів, інтелектуальний агент формує онтологію для реального ПЗ, яку передає як вхідні дані до інтелектуального агента на основі онтологічного підходу для оцінювання ранніх етапів життєвого циклу.

Інтелектуальний агент на основі онтологічного підходу для оцінювання ранніх етапів життєвого циклу проводить порівняння базової онтології для нефункційних характеристик (відомі факти) із отриманою онтологією для реального ПЗ. В результаті такого порівняння агент, згідно з розробленим методом діяльності, формує підмножини нефункційних характеристик та їх підхарактеристик, які неможливо обчислити на основі наявних у реальній специфікації атрибутів; надає висновок про достатність або недостатність інформації у специфікації вимог для визначення кожної нефункційної характеристики ПЗ окремо та для визначення всіх нефункційних характеристик ПЗ разом; обчислює числові оцінки рівня достатності наявної у специфікації вимог інформації для визначення кожної нефункційної характеристики ПЗ та для визначення всіх нефункційних характеристик; надає візуалізацію прогалин у знаннях щодо нефункційних характеристик.

Якщо рівень достатності інформації складає 100% або є прийнятним для замовника, то виконується подальша робота над програмним проєктом, в іншому випадку розробникам специфікації надається запит на додавання необхідних атрибутів у специфікацію (з візуалізованими підказками, які саме атрибути необхідно додати), після чого специфікація може бути знову опрацьована розробленою агентно-орієнтованою інформаційною технологією.

Розроблена агентно-орієнтована інформаційна технологія дає можливість порівняти між собою різні специфікації вимог для програмних проєктів зі схожими або однаковими вартістю і тривалістю, гарантувати внесення у вимоги інформації, необхідної для подальшого визначення нефункційних характеристик,

за рахунок чого зменшується розрив у знаннях про нефункційні характеристики для програмних проєктів. Основною перевагою розробленої інформаційної технології є автоматизація процесів парсингу специфікації та оцінювання достатності інформації вимог для визначення нефункційних характеристик, за рахунок чого досягається усунення суб'єктивного впливу людини, а також збережуваність інформації у софтверній компанії у випадку звільнення фахівця.

4.2. Реалізація та результати функціювання агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу

Агентно-орієнтовану інформаційну технологію (АОІТ) для оцінювання початкових етапів життєвого циклу ПЗ реалізовано мовою PHP в формі вільно поширюваного програмного забезпечення, доступного за посиланням – https://olp-project.herokuapp.com. Для реалізації агентно-орієнтованої інформаційної технології у вигляді веб-сервісу була використана мова програмування PHP та фреймворк Symfony з архітектурою MVC.

Програмний код складається з двох великих модулів – модуль для аналізу специфікації вимог на предмет достатності за моделлю SQuaRE (стандарт ISO 25010:2011) та модуль аналізу специфікації на основі результатів метричного аналізу. Вони доступні користувачу у вигляді окремих пунктів меню у спадаючому списку на головній сторінці веб-сервісу. Користувачу також доступний наступний функціонал: перегляд шаблону специфікації у вигляді онтології, завантаження власної специфікації вимог до ПЗ в pdf-форматі та розрахунок достатності інформації у специфікації для визначення нефункційних характеристик, а також розрахунок достатності метричної інформації.

Для збереження даних щодо залежності характеристик, підхарактеристик та атрибутів була використана СУБД MySQL. Для інтерфейсу користувача була застосована бібліотека Bootstrap. Код програми інтерфейсу користувача міститься у файлі index.html, який і є точкою входу на даний веб-сервіс. Функції для роботи із

специфікацією (парсинг специфікації та робота із формулами для обчислення числової оцінки достатності інформації та відсутніх атрибутів, підхарактеритсик та характеристик) містяться у файлі SrsParserTest.php. Для можливості розпізнавання онтологій та роботи з форматом owl була використана бібліотека ARC2. Основний код (ядро) програми міститься у файлі AppKernel.php (Додаток В).

Перш ніж завантажувати специфікацію вимог для її опрацювання, користувач інформаційної технології може ознайомитись з шаблоном специфікації вимог до ПЗ, який демонструє всі необхідні атрибути для визначення нефункційних характеристик, а також їх місце розташування у специфікації, представлений у вигляді онтології.

Для проведення аналізу, користувач агентно-орієнтованої інформаційної технології для оцінювання початкових етапів життєвого циклу ПЗ повинен завантажити специфікацію вимог до ПЗ в pdf-форматі. Після цього інформаційна технологія проводить парсинг завантаженої специфікації та формує онтологію для реального ПЗ в owl-форматі, яку користувач може завантажити на свій компютер для подальшої роботи.

Після порівняння онтології для реального програмного забезпечення з базовою онтологією для нефункційних характеристик-складових якості ПЗ, AOIT дає висновок про достатність інформації щодо вимог, який складається з: кількості та відсотка відсутніх атрибутів (під час цього підрахунку AOIT дає два числа – без і з урахуванням того, скільки разів використовується відсутній атрибут при визначенні підхарактеристик нефункційних характеристик (без і з повтореннями)); кількісні оцінки достатності інформації щодо вимог для визначення кожної нефункційної характеристики та всіх нефункційних характеристик разом. Крім того, розроблена AOIT пропонує перелік відсутніх атрибутів у вигляді списку, який розділений за підхарактеристиками нефункційних характеристик, що забезпечує візуалізацію відсутніх атрибутів для визначення тих чи інших підхарактеристик нефункційної характеристики.

Для експерименту запропоновано три специфікації вимог до програмного агента для підвищення безпеки програмного забезпечення комп’ютерних систем,

які розроблені різними IT-компаніями Хмельницького. Ці специфікації мають приблизно однакову вартість та тривалість, тому вибір специфікації відповідно до цих критеріїв є складним.

В результаті аналізу специфікації №1 розроблена AOIT дала такі висновки - рис. 4.2, рис. 4.3. В результаті аналізу специфікації №2 розроблена AOIT надала висновки, представлені на рис. 4.4. Після аналізу специфікації № 3 розроблена AOIT надала наступні висновки – рис. 4.5.

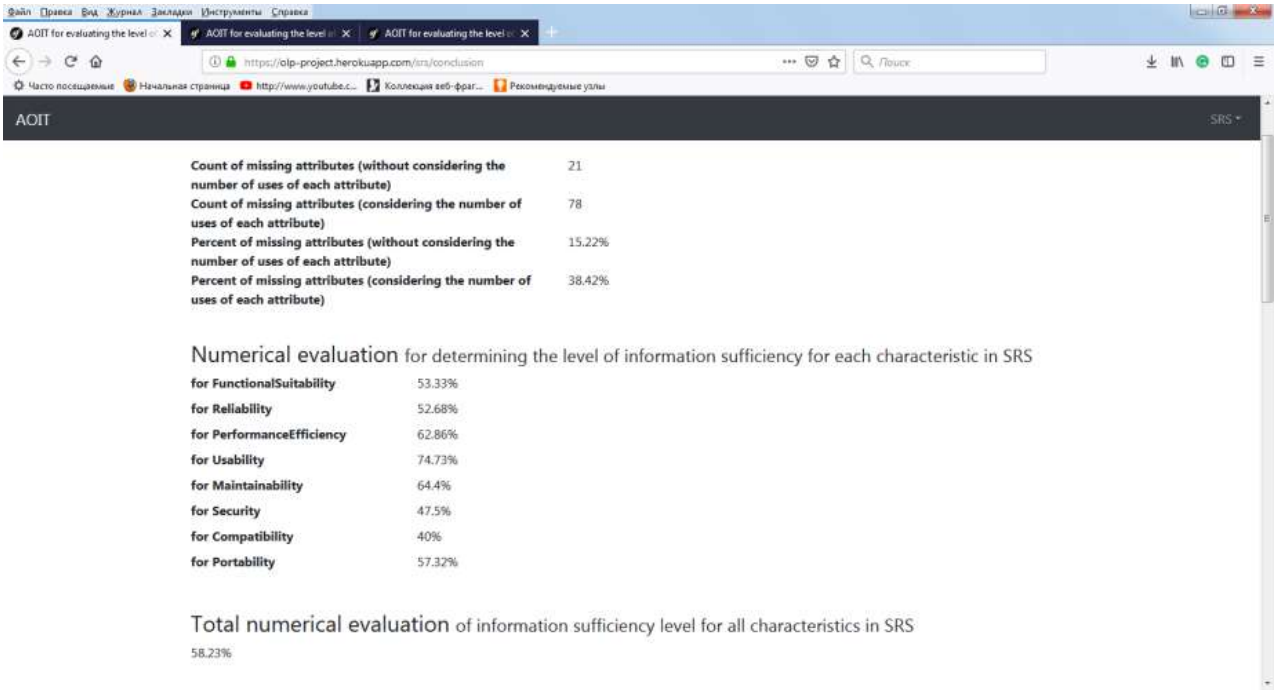

Рис. 4.2 – Кількісні оцінки щодо достатності інформації у вимогах (специфікація №1) для визначення нефункційних характеристик ПЗ (надані розробленою AOIT)

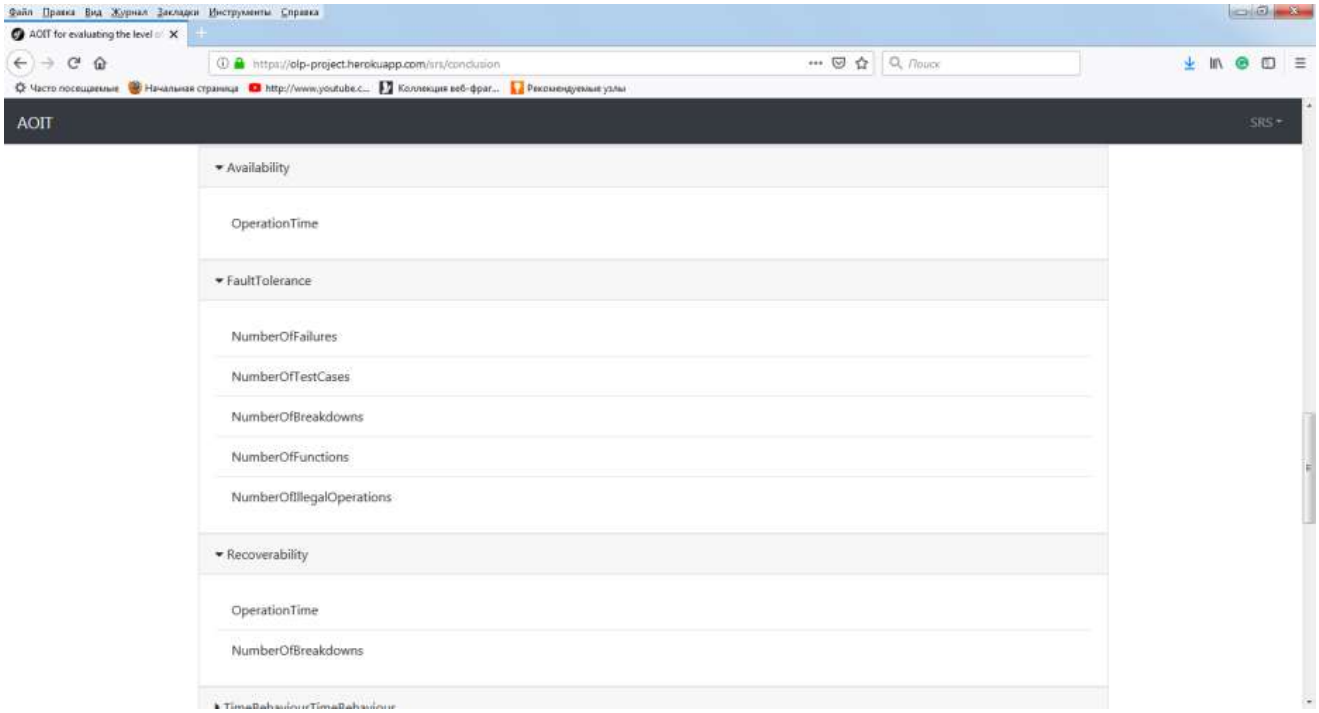

Рис. 4.3 – Візуалізація відсутніх атрибутів (у специфікації №1) для визначення трьох підхарактеристик нефункційної характеристики "Надійність" (надана розробленою AOIT)

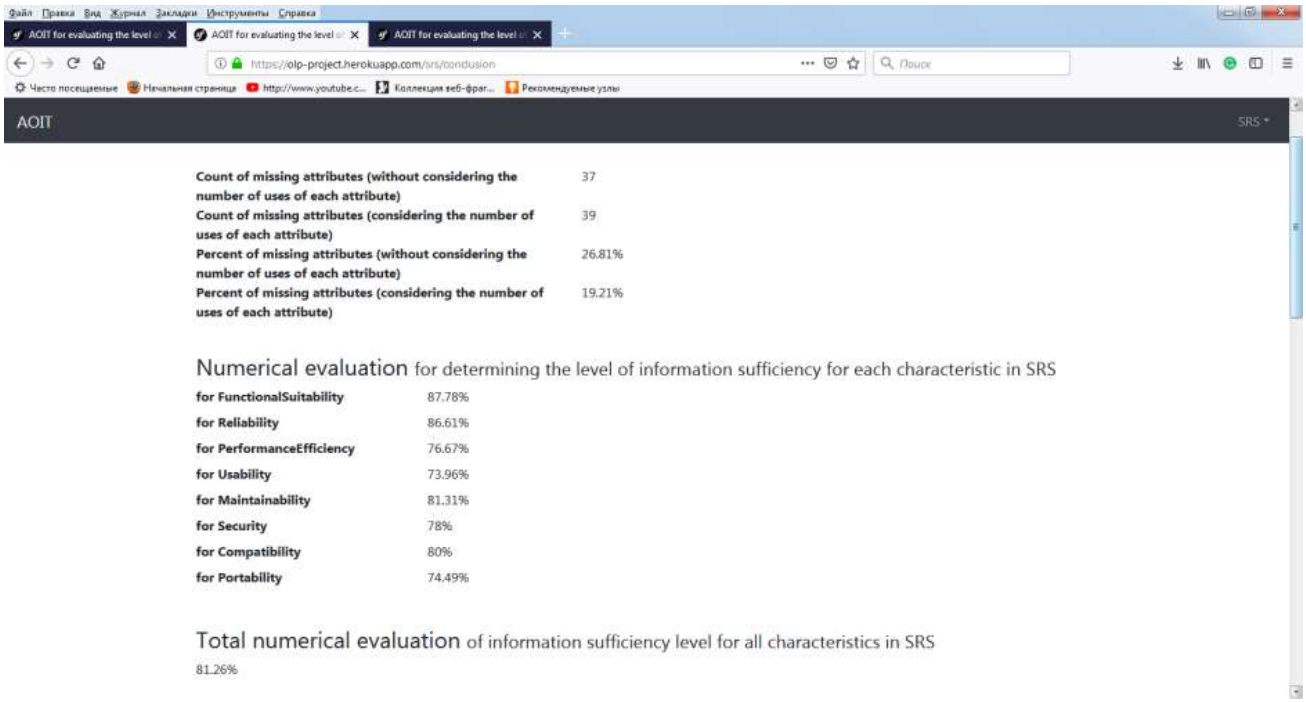

Рис. 4.4 – Кількісні оцінки щодо достатності інформації у вимогах (специфікація №2) для визначення нефункційних характеристик ПЗ (надані розробленою AOIT)

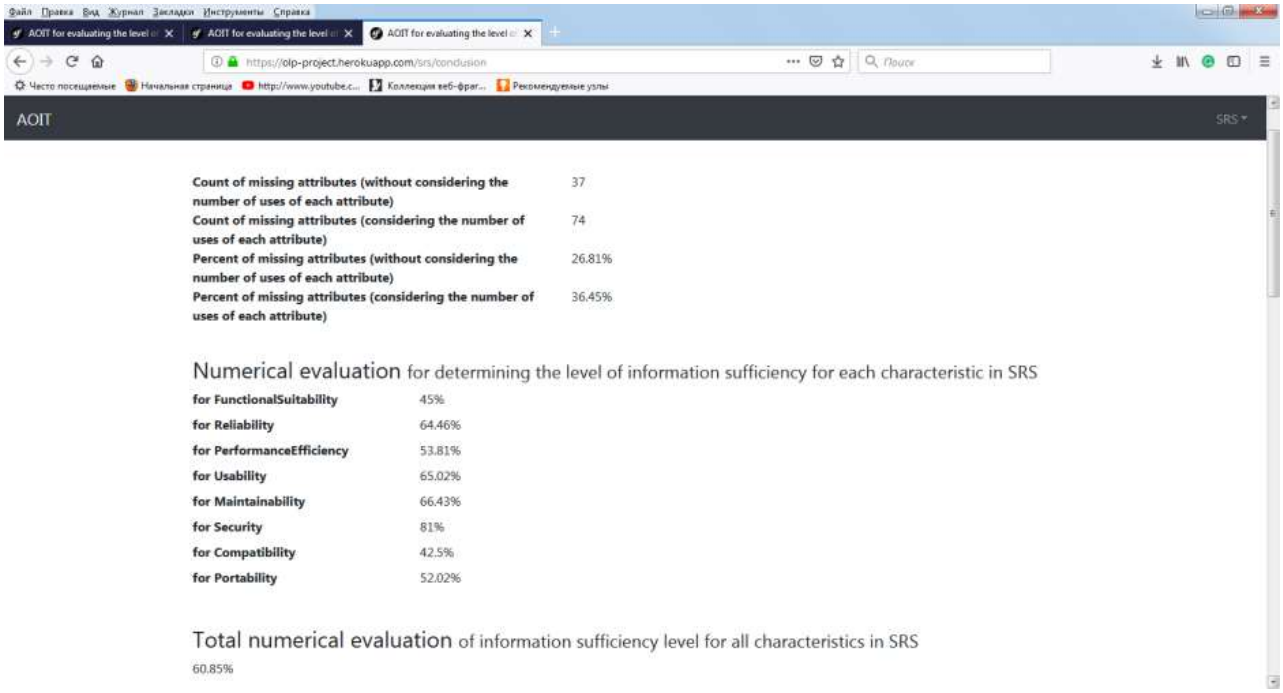

Рис. 4.5 – Кількісні оцінки щодо достатності інформації у вимогах (специфікація №3) для визначення нефункційних характеристик ПЗ (надані розробленою AOIT)

Аналіз результатів роботи розробленої AOIT підтвердив закономірність, що більш важливими та пріоритетними є атрибути, які впливають на більш ніж одну підхарактеристику нефункційної характеристики. Отже, в специфікації №1 відсутній 21 атрибут без урахування кількості застосувань кожного атрибута при визначенні підхарактеристик нефункціних характеристик (але відсутні 78 атрибутів, враховуючи кількість застосувань кожного атрибуту). У специфікації №2 відсутні 37 атрибутів без урахування кількості використання кожного атрибута (але відсутні 39 атрибутів, враховуючи кількість використання кожного атрибута). У специфікації №3 відсутні 37 атрибутів без урахування кількості використання кожного атрибута (але відсутні 74 атрибутів, враховуючи кількість використання кожного атрибута). У той же час рівень достатності інформації у специфікації №1 для визначення всіх нефункційних характеристик становить 58,23%, рівень достатності інформації у специфікації №2 – 81,26%, рівень достатності інформації у специфікації №3 – 60,85%.

Таким чином, при меншій кількості відсутніх атрибутів (без урахування кількості застосувань кожного атрибута), специфікація №1 має нижчий рівень

достатності інформації, ніж специфікація №2, незважаючи на більшу кількість відсутніх атрибутів (без урахування кількості використання кожного атрибута). Ця ситуація пояснюється тим, що в специфікації №1 відсутня більша кількість атрибутів (у порівнянні з специфікацією №2), які впливають на більш ніж одну підхарактеристику нефункційних характеристик. Цей факт підтверджується кількістю відсутніх атрибутів, які надаються AOIT (враховуючи кількість використання кожного атрибута).

В результаті аналізу висновків, наданих розробленою AOIT, замовник програмного агента для підвищення безпеки програмного забезпечення комп'ютерних систем вирішив, що рівень достатності інформації вимог у всіх трьох специфікаціях не є прийнятним для переходу до подальшої роботи над програмним проєктом. Тому всі три специфікації були відправлені розробникам на доопрацювання (в частині доповнення атрибутів для визначення нефункційних характеристик). Доопрацьовані специфікації також були проаналізовані розробленою AOIT. Висновки AOIT після аналізу доопрацьованих специфікацій представлені на рис. 4.6-4.8.

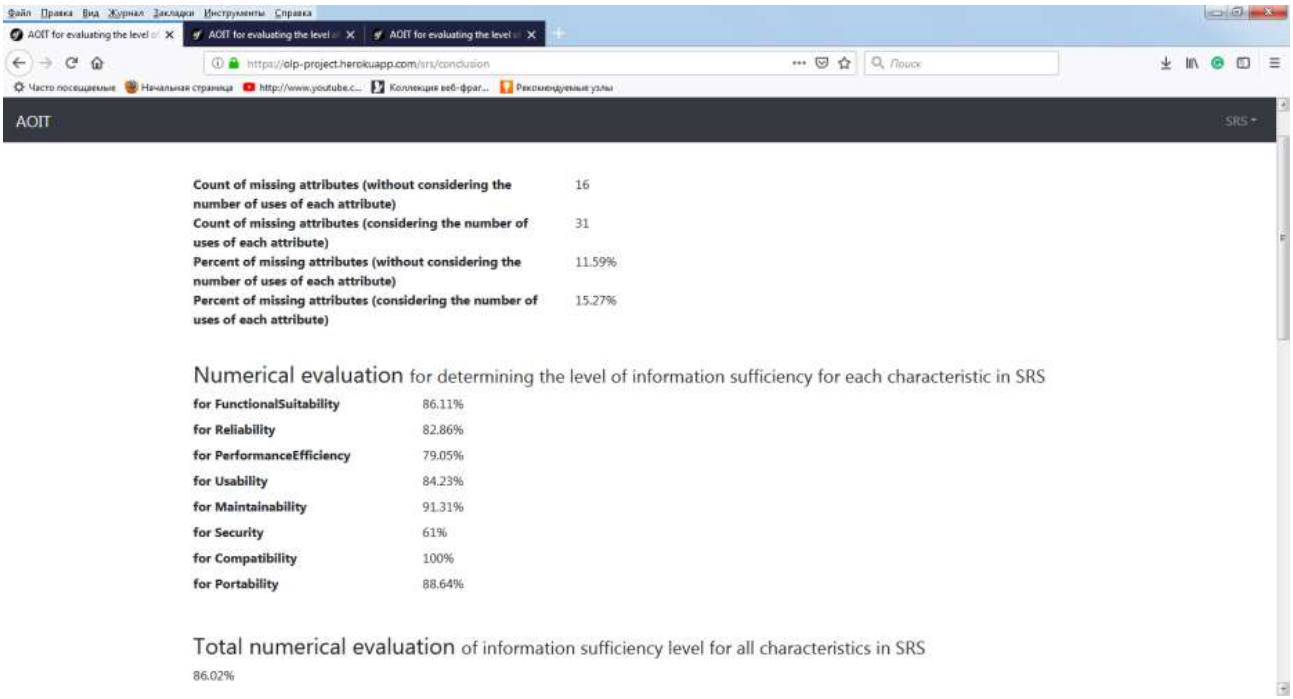

Рис. 4.6 – Кількісні оцінки достатності інформації у специфікації вимог (специфікація №1, після доопрацювання), які надаються розробленою AOIT

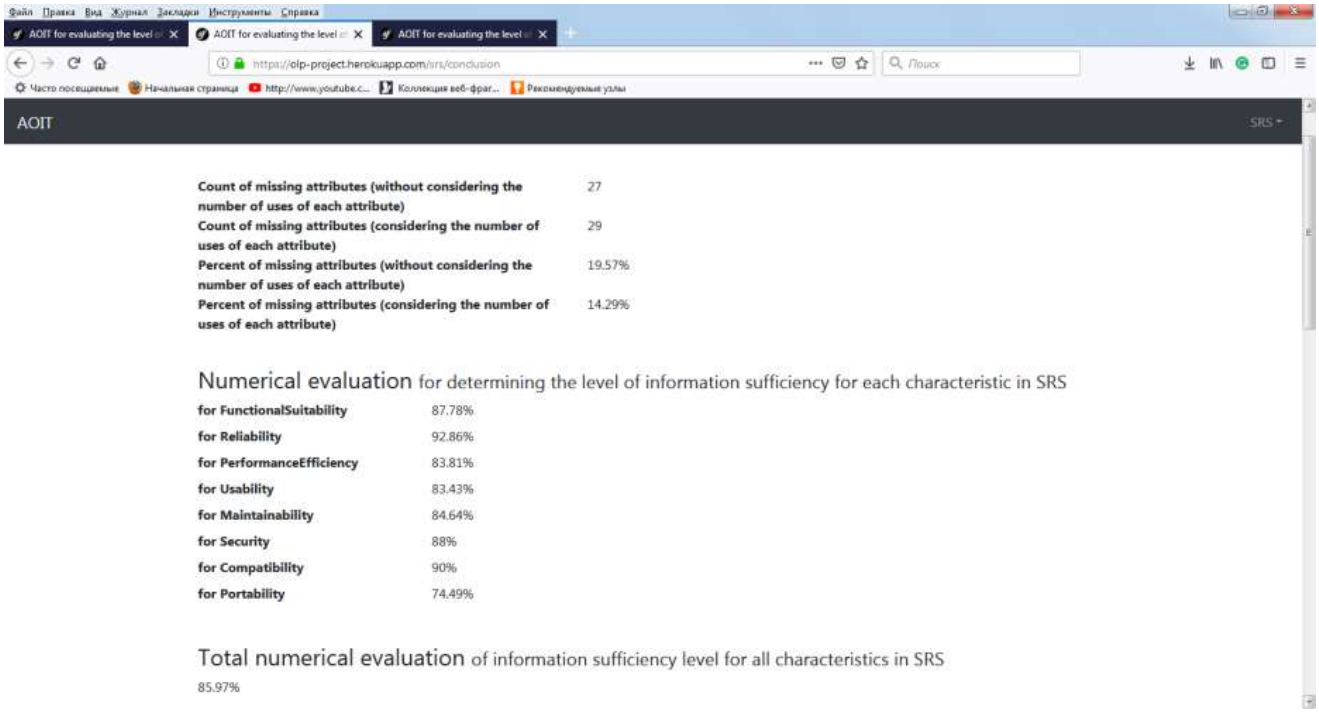

Рис. 4.7 – Кількісні оцінки достатності інформації у специфікації вимог (специфікація №2, після доопрацювання), які надаються розробленою AOIT

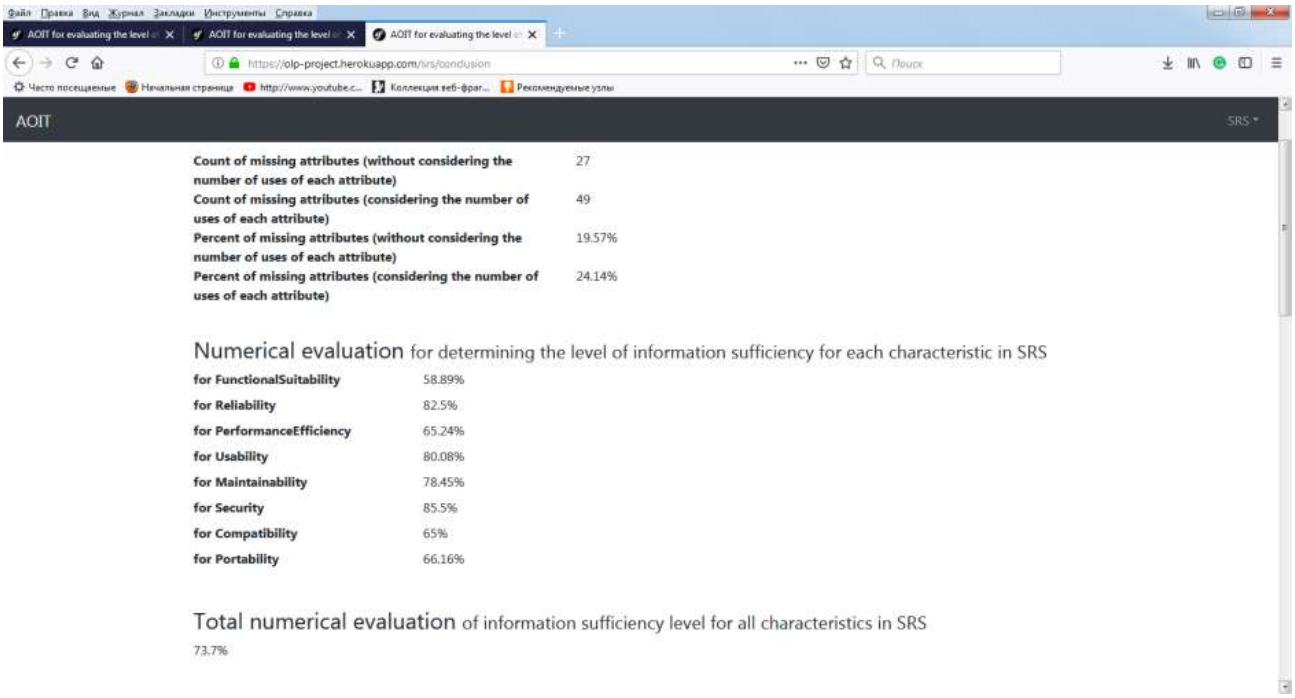

Рис. 4.8 – Кількісні оцінки достатності інформації у специфікації вимог (специфікація №3, після доопрацювання), які надаються розробленою AOIT

У специфікацію №1 було додано 5 атрибутів без урахування кількості застосувань кожного атрибута для визначення підхарактеристик нефункційних

характеристик (та 47 атрибутів, враховуючи кількість застосувань кожного атрибута). У специфікацію №2 було додано 10 атрибутів, не враховуючи кількість використання кожного атрибута (і 10 атрибутів, враховуючи кількість використань кожного атрибута). У специфікацію №3 також було додано 10 атрибутів, не враховуючи кількість використання кожного атрибута (і 25 атрибутів, враховуючи кількість використань кожного атрибута). У той же час рівень достатності інформації у специфікації вимог №1 для визначення всіх нефункційних характеристик вже становить 86,02%, рівень достатності інформації у специфікації №2 – 85,97%, рівень достатності інформації у специфікації №3 – 73,7%. Отже, висновки розробленої АОІТ для оцінювання початкових етапів життєвого циклу програмного забезпечення забезпечують підвищення рівня достатності інформації у специфікаціях для визначення нефункційних характеристик, відповідно, на 27,79%, 4, 71% та 12,85% для специфікацій № 1-3.

Ми (розробники АОІТ) рекомендуємо прийняти рівень достатності, який дорівнює 85%, але остаточне рішення щодо необхідного рівня достатності приймається замовником програмного забезпечення. Клієнт може встановити поріг цього рівня, наприклад, 90%, 95% або 100% (навіть для некритичного програмного забезпечення). Отже, в результаті аналізу висновків розробленої АОІТ замовник вирішив, що рівень достатності інформації вимог специфікацій №1 та №2 вже є допустимим для подальшої роботи над програмним проєктом. Замовник обрав специфікацію №1 для подальшої роботи. Отже, замовник зміг зробити вибір специфікації з точки зору рівня достатності інформації у ній, а не лише з точки зору вартості та тривалості програмного проєкту.

Окремою підсистемою запропонованої АОІТ є *інтелектуальна підсистема для визначення достатності метричної інформації в специфікаціях вимог до програмного забезпечення*. Користувач інтелектуальної підсистеми для визначення достатності метричної інформації при завантажує специфікацію вимог до ПЗ у pdf-форматі. Агент системи, призначений для парсингу специфікації вимог до ПЗ на предмет пошуку метричної інформації (показників для

обчислення метрик), виконує парсинг специфікації та формує онтологію для реального ПЗ у вигляді файлу з розширенням .owl, котрий користувач може завантажити собі на комп’ютер для подальшої роботи.

Далі агент системи, призначений для оцінювання достатності метричної інформації у специфікаціях, виконує порівняння онтології для реального ПЗ з базовою онтологією та видає висновок про достатність метричної інформації, а саме: Кількість відсутніх показників (без урахування кількості використань кожного показника), Кількість відсутніх показників (враховуючи кількість використань кожного показника), Відсоток відсутніх показників (без урахування кількості використань кожного показника), Відсоток відсутніх показники (враховуючи кількість застосувань кожного показника), Загальна кількісна оцінка рівня достатності інформації для всіх показників у специфікації, а також візуалізований список відсутніх показників, розділених за метриками складності та якості ПЗ.

Для проведення експерименту було проаналізовано дві специфікації вимог до програмного агента для підвищення безпеки програмного забезпечення комп’ютерних систем, розроблені двома різними софтверними компаніями м. Хмельницького.

Рівень достатності метричної інформації специфікації №1 склав 49,72% за відсутніх 7 показників без урахування кількості використання кожного показника та 31 показника, враховуючи кількість використань кожного показника – рис. 4.9.

Рівень достатності метричної інформації специфікації №2 склав 75,28% за відсутніх 14 показників без врахування кількості використань кожного показника та 20 показників із врахуванням кількості використань кожного показника – рис. 4.10.

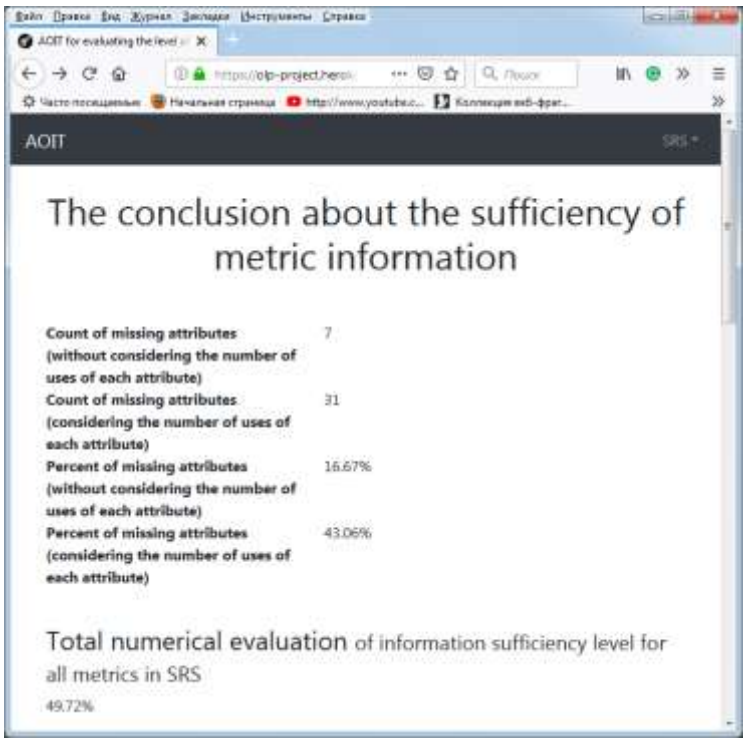

Рис. 4.9 – Інтерфейс інтелектуальної підсистеми для визначення достатності метричної інформації в специфікації – висновок про достатність метричної інформації у специфікації №1

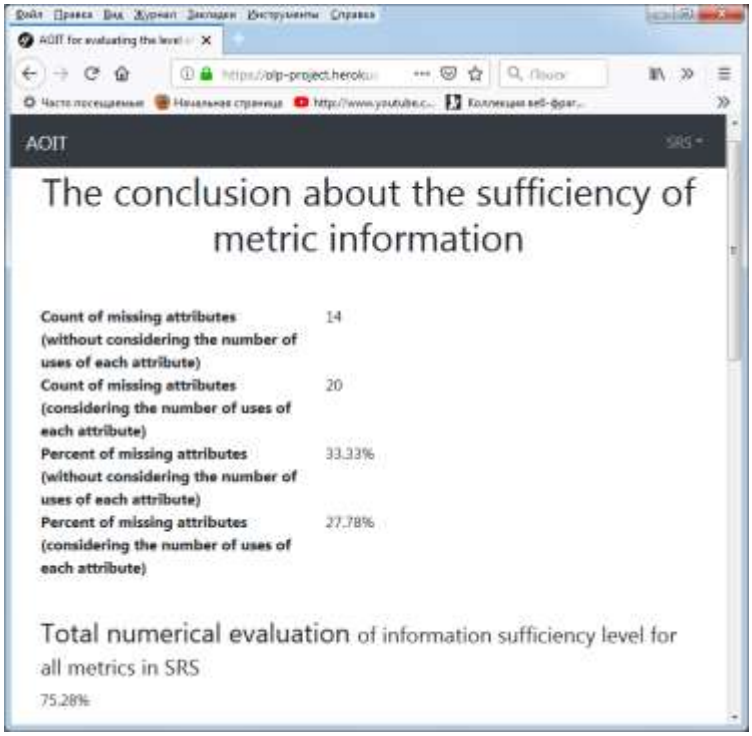

Рис. 4.10 – Інтерфейс інтелектуальної підсистеми для визначення достатності метричної інформації в специфікації вимог – висновок про достатність метричної інформації у специфікації №2

Очевидно, що достатність метричної інформації у специфікації №2 вища, ніж у специфікації №1, а кількість відсутніх показників із врахуванням кількості використань кожного показника у специфікації №2 менша, ніж у специфікації №1 (в той час, як кількість відсутніх показників без врахування кількості використань кожного показника у специфікації №2 вдвічі більша, ніж у специфікації №1). Такі результати пояснюються тим фактом, що більш важливими і пріоритетними є показники, від яких залежать більше однієї метрики.

Замовник програмного агента для підвищення безпеки програмного забезпечення комп'ютерних систем вважає прийнятним рівень достатності, який дорівнює 95% (оскільки це ПЗ критичного застосування). Відтак замовник затребував доопрацювання обох специфікацій вимог їх розробниками.

Після доопрацювання, специфікації знову були проаналізовані інтелектуальною підсистемою для визначення достатності метричної інформації у специфікації вимог до ПЗ – рис. 4.11, 4.12.

Рівень достатності метричної інформації специфікації 1 після доопрацювання склав 100% (в специфікації є всі необхідні показники для обчислення метрик), а рівень достатності метричної інформації специфікації 2 після доопрацювання склав 87.99% за відсутніх 8 показників без врахування кількості використань кожного показника та 10 показників враховуючи кількість використань кожного показника.

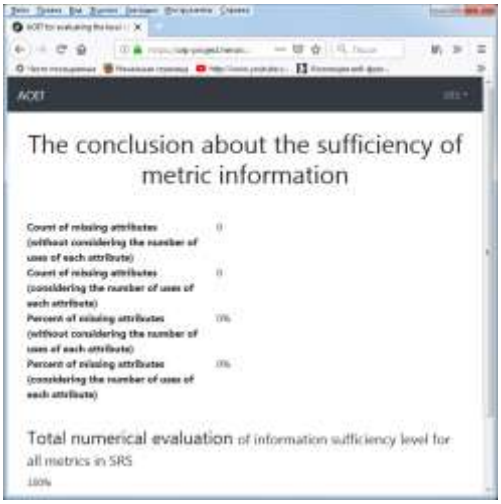

Рис. 4.11 – Інтерфейс інтелектуальної підсистеми для визначення достатності метричної інформації у специфікації вимог – висновок про достатність метричної інформації у специфікації №1 (після доопрацювання)

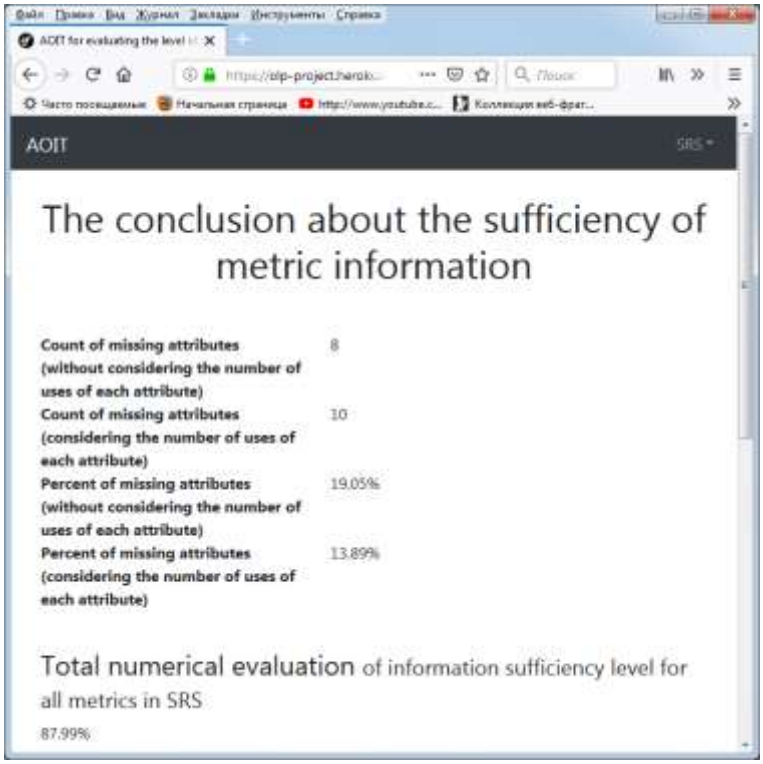

Рис. 4.12 – Інтерфейс інтелектуальної підсистеми для визначення достатності метричної інформації у специфікації вимог – висновок про достатність метричної інформації у специфікації №2 (після доопрацювання)

Замовник програмного агента для підвищення безпеки програмного забезпечення комп'ютерних систем обрав специфікацію №1 з рівнем достатності метричної інформації 100% для подальшої роботи над програмним проєктом.

Використання реалізованої агентно-орієнтованої інформаційної технології дозволило замовнику виконати вибір специфікації з точки зору рівня достатності інформації у ній, а не лише з точки зору вартості та тривалості програмного проєкту. Висновки розробленої AOIT для оцінювання початкових етапів життєвого циклу програмного забезпечення забезпечують підвищення рівня достатності інформації у специфікаціях для визначення нефункційних характеристик, відповідно, на 27,79%, 4, 71% та 12,85% для специфікацій № 1-3. Розроблена інтелектуальна система визначення достатності метричної інформації у специфікації вимог до ПЗ забезпечила приріст достатності інформації на 50.28% для специфікації №1 та на 12.71% для специфікації №2.

4.3. Переваги агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу

Як було доведено у розділі 1, на зміст та методи опрацювання інформації істотно впливають особливості предметних галузей, для яких розробляються інформаційні технології, причому особливої уваги щодо розроблення та впровадження інформаційних технологій на сьогодні потребує галузь інженерії програмного забезпечення (ПЗ) в частині оцінювання початкових етапів життєвого циклу.

Недостатня увага до вимог призводить до недотримання термінів проєктів та фінансових перевитрат (проблемні проєкти), що можуть призвести до закриття проєкту (провальний проєкт), а то й розпаду софтверної компанії внаслідок її фінансової нестабільності. Витрати на виправлення некоректних вимог специфікації, виявлених після випуску продукту, в сотні разів перевищують витрати на виправлення недоліків специфікації, які були виявлені в процесі формування та формулювання вимог.

Проведений у підрозділі 1.2 огляд відомих підходів до оцінки достатності інформації вимог, а також інформаційних технологій та інструментів для аналізу специфікацій показав, що існує ряд ефективних рішень, але всі вони не розв’язують задачі оцінювання початкових етапів життєвого циклу ПЗ. Крім того, вони належать до різних методологічних підходів, розроблені для різних завдань і не поєднуються одне з одним, а також передбачають людинно-машинну взаємодію на всіх етапах опрацювання інформації, під час якої всю інформацію інтерпретує людина, що часто призводить до втрат істотної інформації.

У [1, 104] вперше розроблено теоретичні та прикладні засади інформаційної технології оцінювання достатності інформації щодо якості у специфікаціях вимог до програмного забезпечення, в якій людина частково усувається з процесів опрацювання інформації та здобуття знань, проте ця ІТ також передбачає аналіз специфікації людиною. Отже, в даний час розроблення потребує інформаційна

технологія для оцінювання початкових етапів життєвого циклу ПЗ, яка повністю усуватиме людину з процесів опрацювання інформації та здобуття знань.

Порівняльна характеристика розробленої агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу із відомими інформаційними технологіями та інструментами представлена в таблиці 4.1.

Таблиця 4.1

Порівняльна характеристика розробленої агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу із відомими інформаційними технологіями та інструментами

Інформаційна технологія, інструмент	**Критерії**				**Виконуваність критеріїв**
	Оцінювання достатності (кількісні оцінки)	**Автоматизація всіх обчислень**	**Валідація вимог (перевірка відповідності вимог потребам замовника)**	**Безкоштовний доступ**	
Модель для перевірки достатності вимог безпеки [107, 108]	Так (кількість вимог безпеки)	Ні	Так (тільки вимоги безпеки)	Так	75%
Оцінка достатності як досягнення рівня покриття тестами [109]	Ні	Так	Ні	Так	50%

CORE [110]	Hi	Так	Hi	Так	50%
Visure [111]	Hi	Так	Hi	Так	50%
Accompa [111]	Hi	Так	Hi	Hi	25%
Innoslate [111]	Hi	Так	Так	Hi	50%
ReqView [111]	Hi	Так	Hi	Так	50%
Modern Requirements 4TFS[111]	Hi	Так	Hi	Так	50%
Natural Language Processing Requirements Analysis Tools (наприклад, QVscribe) [8]	Hi	Так	Hi	Так	50%
Requirements Analysis Tool [113]	Hi	Так	Так	Так	75%
QARCC (Quality Attribute Risk and Conflict Consultant) [114]	Hi	Так	Так	Так	75%
QuARS Requirements Analysis Tool [114, 116]	Hi	Так	Так	Hi	50%
DESIRe [117]	Hi	Так	Так	Hi	50%

Requirements Assistant [115]	Ні	Так	Так	Ні	50%
RQV Tool (Requirement Quality Verification Tool) [117]	Ні	Так	Так	Так	75%
Інформаційна технологія оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ [1, 104]	Так	Ні	Так	Так	75%
Розроблена агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу	*Так*	*Так*	*Так*	*Так*	*100%*

Результати порівняння розробленої агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу із відомими інформаційними технологіями та інструментами, представлені в таблиці 4.1, показали, що жодна з відомих інформаційних технологій та інструментів не забезпечує одночасного кількісного оцінювання достатності інформації вимог, повної автоматизації всіх обчислень, виконання валідації вимог потребам замовника, а також безкоштовного доступу. Єдиним рішенням, яке забезпечує виконання одразу всіх 4-х необхідних критеріїв, є розроблена в дисертаційній роботі агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу.

Як показали результати функціювання агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу, представлені в підрозділі 4.2, рівень достатності інформації у специфікації №1 вимог до програмного агента для підвищення безпеки ПЗ комп'ютерних систем становив 58,23% після початкового аналізу розробленою АОІТ, рівень достатності інформації у специфікації №2 – 81,26%, рівень достатності інформації у специфікації №3 – 60,85%. Інформаційна технологія сформувала висновок про недостатнсть інформації у всіх 3-х специфікаціях вимог, надала замовнику кількісні оцінки достатності інформації, замовник прийняв рішення про необхідність доопрацювання всіх специфікацій вимог, інформаційною технологією було надано запит на додавання необхідних атрибутів у специфікації з візуалізованими підказками, які саме атрибути необхідно додати. Розробники всіх специфікацій внесли доповнення до вимог, після чого специфікації були піддані повторному аналізу розробленою АОІТ. Після доопрацювання рівень достатності інформації у специфікації вимог №1 вже став 86,02%, рівень достатності інформації у специфікації №2 – 85,97%, рівень достатності інформації у специфікації №3 – 73,7%. Отже, висновки розробленої АОІТ для оцінювання початкових етапів життєвого циклу ПЗ забезпечили підвищення рівня достатності інформації у специфікаціях, відповідно, на 27,79%, 4, 71% та 12,85% – рис. 4.13.

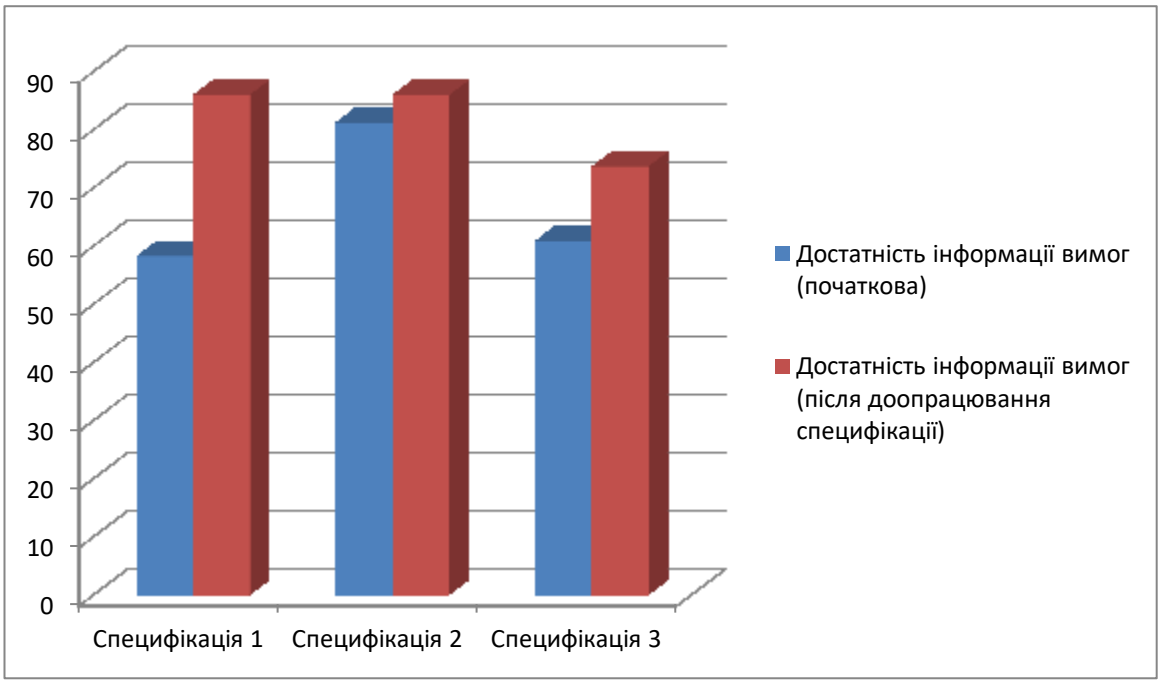

Рис. 4.13 – Діаграма приросту достатності інформації у специфікаціях вимог до програмного агента для підвищення безпеки ПЗ комп'ютерних систем внаслідок врахування рекомендацій розробленої агентно-орієнтованої інформаційної технології

Як показали результати функціювання інтелектуальної підсистеми визначення достатності метричної інформації у специфікації вимог до ПЗ, представлені в підрозділі 4.2, рівень достатності метричної інформації у специфікації №1 вимог до програмного агента для підвищення безпеки ПЗ комп'ютерних систем становив 49,72% після початкового аналізу, рівень достатності метричної інформації у специфікації №2 – 75,28%. Інтелектуальна підсистема AOIT сформувала висновок про недостатнсть метричної інформації у обох специфікаціях вимог, надала замовнику кількісні оцінки достатності інформації, замовник прийняв рішення про необхідність доопрацювання специфікацій вимог, інтелектуальною підсистемою було надано запит на додавання необхідних показників у специфікації з візуалізованими підказками, які саме показники необхідно додати. Розробники обох специфікацій внесли доповнення до вимог, після чого специфікації були піддані повторному аналізу розробленою інтелектуальною підсистемою AOIT. Після доопрацювання рівень достатності метричної інформації у специфікації вимог №1 вже став 100%, рівень достатності

інформації у специфікації №2 – 87,99%. Отже, висновки розробленої інтелектуальної підсистеми АОІТ для визначення достатності метричної інформації у специфікації вимог до ПЗ забезпечили приріст достатності інформації на 50.28% для специфікації №1 та на 12.71% для специфікації №2 – рис. 4.14.

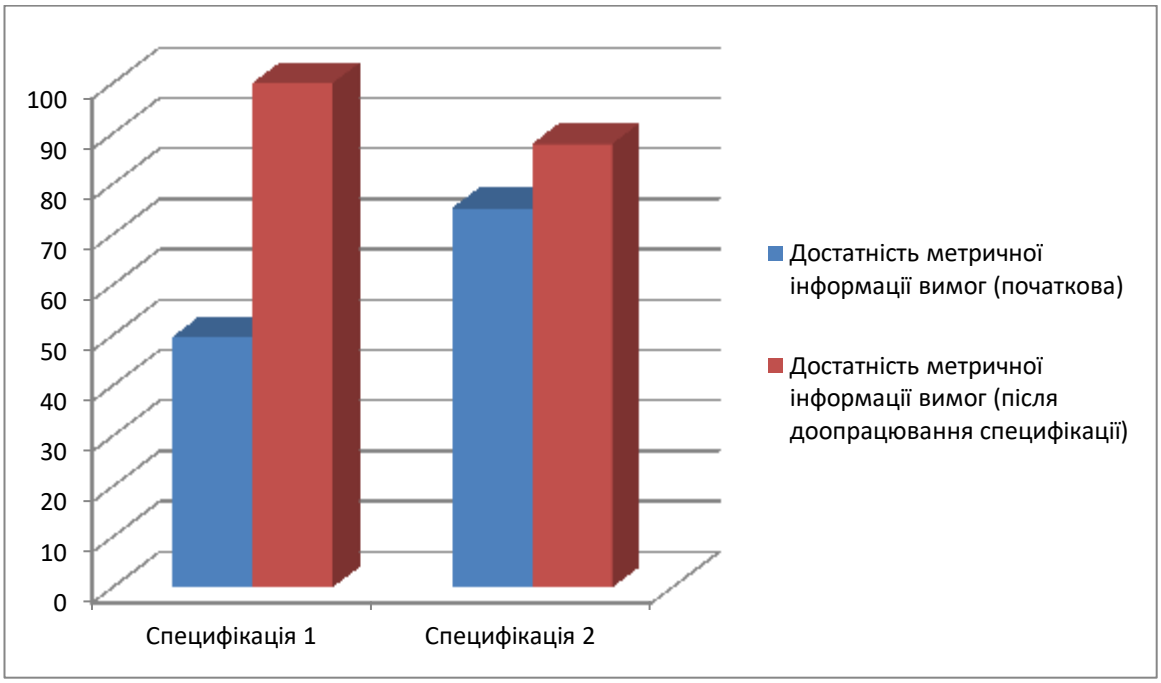

Рис. 4.14 – Діаграма приросту достатності метричної інформації у специфікаціях вимог до програмного агента для підвищення безпеки ПЗ комп'ютерних систем внаслідок врахування рекомендацій розробленої інтелектуальної підсистеми агентно-орієнтованої інформаційної технології

Крім того, як відомо [9], всі ітерації в ході програмного проєкту розділяються на дві групи:

1) плановані – викликані першою потребою; заздалегідь відомо, коли і де вони відбуватимуться; повинні сприяти вибору оптимальних методів та інструментів проектування та координації;

2) неплановані – викликані помилками та/або непередбаченими проблемами; їх час і місце не піддаються передбаченню; призводять до часовитратного перероблення (rework); повинні бути зведені до мінімуму.

Розроблена агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу дозволяє мінімізувати кількість непланованих ітерацій, а відтак і скоротити витрати часу на перероблення на пізніх етапах життєвого циклу ПЗ, а також суттєво скоротити перероблення пля завершення початкової роботи – за рахунок раннього виявлення дефектів вимог (зокрема, недостатності їх інформації) та раннього усунення виявлених дефектів. На рис. 4.15 суцільною чорною лінією показано витрати часу та людських ресурсів на початкову роботу (initial work) та суцільною червоною лінією показано витрати часу і людських ресурсів на перероблення (rework), що відповідають сучасному стану речей, пунктирною червоною лінією показано витрати часу і людських ресурсів на перероблення (rework), які стають можливими внаслідок використання розробленої в дисертаційній роботі агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу.

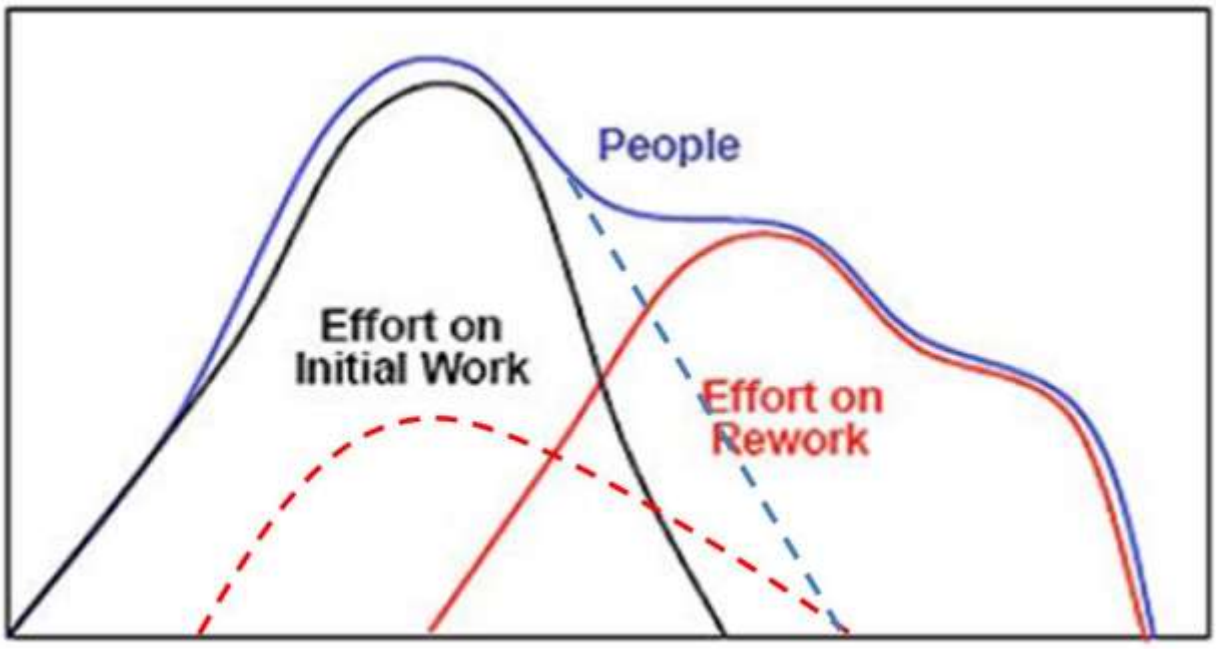

Рис. 4.15 – Порівняльна характеристика витрат часу та людських ресурсів на перероблення без та з використанням розробленої агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу

Крім цього, розроблена агентно-орієнтована інформаційна технологія підходить для роботи зі специфікаціями вимог в сучасних умовах, коли вимоги до ПЗ часто змінюються, – за рахунок швидкості та легкості аналізу специфікації з використанням АОІТ. Користувач може щоразу перевіряти на достатність оновлену специфікацію вимог, не витрачаючи на це багато часу та зусиль.

Отже, розроблена агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу, завдяки опрацюванню інформації специфікації інтелектуальними агентами, без участі людини, забезпечує можливість повної автоматизації таких процесів та усунення суб'єктивного впливу людини.

Впровадження агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу у софтверних компаніях автоматизує процес верифікації достатності інформації у специфікаціях вимог до ПЗ, дозволить використання програмних агентів для аналізу та доповнення специфікації, для оцінювання початкових етапів життєвого циклу з метою мінімізації інформаційних та фінансових втрат.

4.4. Висновки

У даному розділі отримала подальшого розвитку агентно-орієнтована інформаційна технологія для оцінювання початкових етапів життєого циклу ПЗ шляхом оцінювання достатності інформації на початкових етапах життєвого циклу програмного забезпечення. Дана АОІТ оцінює та забезпечує приріст рівня достатності інформації у специфікації вимог для визначення нефункційних характеристик програмного забезпечення – приріст рівня достатності становить від 4,71% до 27,79% (наприклад, з 58,23% до 86,02% для специфікації вимог №1 до програмного агента для підвищення безпеки програмного забезпечення комп'ютерних систем, від 81,26% до 85,97% для специфікації №2, з 60,85% до 73,7% для специфікації №3).

На додаток до вищезазначеного, *перевагами розробленої АОІТ* є:

1) автоматизація трудомісткого, рутинного та схильного до помилок завдання розбору специфікацій вимог та майже миттєве його виконання;

2) підказка, де потрібна повторна робота над специфікацією вимог (користувач може переглядати відсутні атрибути та бачити області специфікації, яким потрібна додаткова увага, а також які вимоги потребують переробки);

3) забезпечення навчання для нових розробників специфікацій, системних інженерів та менеджерів проєктів (використання цієї АОІТ допомагає їм побачити помилки, які вони можуть допускати, та допомагає їм розпізнати ці помилки в роботі інших);

4) допомога у розробці вимог високої якості;

5) допомога у виправленні та усуненні помилок у вимогах там, де вони виникають – на ранніх етапах життєвого циклу програмного забезпечення – до того, як вони стануть дорожчими для виправлення;

6) надання інструменту для вибору більш якісних специфікацій програмних вимог;

7) скорочення витрат часу на перероблення на пізніх етапах життєвого циклу ПЗ, а також суттєве скорочення перероблення псля завершення початкової роботи – за рахунок раннього виявлення дефектів вимог (зокрема, недостатності їх інформації) та раннього усунення виявлених дефектів;

8) прийнятність для роботи з мінливими вимогами – за рахунок швидкості та легкості аналізу специфікації з використанням розробленої АОІТ;

9) безкоштовний доступ через Інтернет в будь-який час без будь-якої реєстрації.

Економічним ефектом від використання розробленої АОІТ є можливість економії бюджету програмних проєктів на обробку та виправлення (протягом життєвого циклу) дефектів та помилок, які утворюються на ранніх етапах життєвого циклу – завдяки демонстрації слабких сторін специфікацій вимог до ПЗ, які потрібно доопрацювати або переробити в той момент, коли вони виникають.

Обмеженнями розробленої АОІТ є:

1) оцінка достатності інформації лише щодо вимог для визначення нефункційних характеристик-складових якості ПЗ та для визначення метрик ПЗ;

2) врахування лише нефункційних характеристик, які регламентуються стандартом ISO 25010 як характеристики якості програмного забезпечення;

3) неврахування інших SQUARE стандартів серії 25000 (ISO 25011, ISO 25012);

4) під час аналізу специфікацій здійснюється пошук лише атрибутів, які визначені ISO 25023 як необхідні для нефункційних характеристик-компонентів якості програмного забезпечення, а також показників, необхідних для визначення обраних метрик складності та якості ПЗ.

ВИСНОВКИ

У дисертації розв’язана актуальна науково-прикладна задача автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення шляхом розроблення агентно-орієнтованої інформаційної технології на основі онтологічного підходу, яка забезпечує: автоматизацію трудомісткого, рутинного та схильного до помилок завдання розбору специфікацій вимог та майже миттєве його виконання; підказку, де потрібна повторна робота над специфікацією вимог (користувач може переглядати відсутні атрибути та бачити області специфікації, яким потрібна додаткова увага, а також які вимоги потребують переробки); забезпечення навчання для нових розробників специфікацій, системних інженерів та менеджерів проєктів (використання цієї АОІТ допомагає їм побачити помилки, які вони можуть допускати, та допомагає їм розпізнати ці помилки в роботі інших); допомогу в розробці вимог високої якості; допомогу у виправленні та усуненні помилок у вимогах там, де вони виникають – на ранніх етапах життєвого циклу програмного забезпечення – до того, як вони стануть дорожчими для виправлення; надання інструменту для вибору більш якісних специфікацій програмних вимог; скорочення витрат часу на перероблення на пізніх етапах життєвого циклу ПЗ, а також суттєве скорочення перероблення пслся завершення початкової роботи; прийнятність для роботи з мінливими вимогами; безкоштовний доступ через Інтернет в будь-який час без будь-якої реєстрації.

У роботі отримано наступні наукові та практичні результати:

1. Аналіз відомих підходів, методів, інформаційних технологій, інструментів для аналізу вимог до програмного забезпечення та оцінки достатності інформації у специфікації вимог та інтелектуальних агентів на основі онтологічного підходу, показав, що вони не розв’язують проблему оцінювання ранніх етапів життєвого циклу ПЗ. Крім цього, всі вони належать до різних методологічних підходів і не інтегруються між собою, тобто наразі відсутні інтелектуальні інформаційно-аналітичні технології для оцінювання ранніх етапів життєвого циклу ПЗ, що є однією з причин проблем у галузі інженерії ПЗ.

Актуальність проблеми аналізу достатності інформації щодо якості у специфікаціях вимог до ПЗ, а також відсутність моделей, методів та засобів оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ обумовлює необхідність розроблення інтелектуальних інформаційно-аналітичних технологій для підвищення якості програмного забезпечення шляхом оцінювання достатності інформації на ранніх етапах життєвого циклу.

2. У дисертаційній роботі запропоновані базові (універсальні) онтологічні моделі нефункційних характеристик-складових якості ПЗ, а також онтологічні моделі нефункційних характеристик-складових якості конкретного ПЗ, які ґрунтуються на врахуванні вимог стандартів ISO 25010:2011 та ISO 25023:2016 і забезпечують підґрунтя для вибору достатнього обсягу інформації для оцінювання нефункційних характеристик ПЗ. Також розроблено модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ, яка ґрунтується на порівняльному аналізі онтологій та є теоретичним підґрунтям для розроблення методів діяльності інтелектуального агента на основі онтологічного підходу.

3. Вперше розроблено метод діяльності інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення, який працює на основі розробленої моделі та здійснює оцінювання достатності інформації у специфікації вимог для визначення всіх нефункційних характеристик-складових якості ПЗ, забезпечує висновок про достатність або недостатність інформації у специфікації, надає числові оцінки рівня достатності інформації для визначення кожної нефункційної характеристики ПЗ та для визначення всіх нефункційних характеристик-складових якості ПЗ разом, формує список атрибутів, якими варто доповнити специфікацію вимог для підвищення рівня достатності її інформації, тобто в комплексі дозволяє частково усунути людину з процесів опрацювання інформації та здобуття знань.

4. Реалізовано інтелектуальні агенти на основі онтологічного підходу для оцінювання ранніх етапів життєвого циклу ПЗ, які здійснюють оцінювання

достатності інформації у специфікації вимог для визначення всіх нефункційних характеристик-складових якості ПЗ, а також для розрахунку метрик якості та складності ПЗ. Реалізовані інтелектуальні агенти забезпечують висновок про достатність або недостатність інформації у специфікації. Крім цього, вони надають числові оцінки рівня достатності інформації для визначення кожної нефункційної характеристики ПЗ та/або метрики та для визначення всіх нефункційних характеристик-складових якості ПЗ та/або метрик разом. Агентами також формується список атрибутів та показників, якими варто доповнити специфікацію вимог для підвищення рівня достатності її інформації, та візуалізація прогалин у знаннях про всі нефункційні характеристики-складові якості ПЗ та метрики якості і складності ПЗ. Отже, реалізовані інтелектуальні агенти забезпечують автоматизацію аналізу специфікацій вимог до ПЗ на предмет достатності їх інформації. Таким чином, представлені агенти дозволяють частково усунути людину з процесів опрацювання інформації та здобуття знань.

5. Удосконалено метод діяльності інтелектуального агента для автоматизованого семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення, який, на відміну від відомих, грунтується на врахуванні вимог стандарту ISO 25010 і на обраній номенклатурі метрик та виконує парсинг специфікації, визначає кількість та відсоток відсутніх атрибутів, відображає, яких атрибутів не вистачає для тієї чи іншої підхарактеристики нефункційної характеристики, а також формує реальну онтологію для нефункційних характеристик, яка може бути використана інтелектуальним агентом на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення.

6. Розроблено інтелектуальні агенти для семантичного парсингу природомовних специфікацій, які виконують парсинг специфікації, визначають кількість та відсоток відсутніх атрибутів та/або показників, відображають, яких атрибутів та/або показників не вистачає для тієї чи іншої підхарактеристики нефункційної характеристики та/або метрики, а також формують реальну онтологію для нефункційних характеристик та/або метрик, яка може бути

використана інтелектуальним агентом на основі онтологічного підходу для оцінювання ранніх етапів життєвого циклу ПЗ шляхом оцінювання достатності інформації для визначення нефункційних характеристик та метрик.

7. Отримала подальшого розвитку агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу в частині автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення, яка, на відміну від відомих, виконує оцінювання та забезпечує підвищення рівня достатності інформації вимог для визначення кожної нефункційної характеристики окремо та всіх нефункційних характеристик разом. Дана АОІТ оцінює та забезпечує приріст рівня достатності інформації у специфікації вимог для визначення нефункційних характеристик програмного забезпечення – приріст рівня достатності становить від 4,71% до 27,79% (наприклад, з 58,23% до 86,02% для специфікації вимог №1 до програмного агента для підвищення безпеки програмного забезпечення комп'ютерних систем, від 81,26% до 85,97% для специфікації №2, з 60,85% до 73,7% для специфікації №3). Перевагами розробленої АОІТ є: автоматизація трудомісткого, рутинного та схильного до помилок завдання розбору специфікацій вимог та майже миттєве його виконання; підказка, де потрібна повторна робота над специфікацією вимог (користувач може переглядати відсутні атрибути та бачити області специфікації, яким потрібна додаткова увага, а також які вимоги потребують переробки); забезпечення навчання для нових розробників специфікацій, системних інженерів та менеджерів проєктів (використання цієї АОІТ допомагає їм побачити помилки, які вони можуть допускати, та допомагає їм розпізнати ці помилки в роботі інших); допомога у розробці вимог високої якості; допомога у виправленні та усуненні помилок у вимогах там, де вони виникають – на ранніх етапах життєвого циклу програмного забезпечення – до того, як вони стануть дорожчими для виправлення; надання інструменту для вибору більш якісних специфікацій програмних вимог; безкоштовний доступ через Інтернет в будь-який час без будь-якої реєстрації. Економічним ефектом від використання розробленої АОІТ є можливість економії бюджету програмних

проєктів на обробку та виправлення (протягом життєвого циклу) дефектів та помилок, які утворюються на ранніх етапах життєвого циклу – завдяки демонстрації слабких сторін специфікацій вимог до ПЗ, які потрібно доопрацювати або переробити в той момент, коли вони виникають.

8. Практичне значення отриманих результатів полягає у розробленні агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу, яка: автоматизує трудомістку та схильну до помилок задачу розбору специфікації вимог; вказує на необхідність доопрацювання специфікації із зазначенням вимог, які потребують доопрацювання; забезпечує швидке навчання нових системних інженерів та керівників проєктів (використання розробленої інформаційної технології під час створення або аналізу вимог допомагає їм швидко бачити помилки, яких вони можуть припуститись, і допомагає розпізнавати ці помилки в роботі інших); допомагає виправити та усунути помилки та неточності у вимогах на ранніх етапах життєвого циклу, є інструментом для вибору більш якісної специфікації, доступна онлайн в будь-який час без реєстрації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 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.

2. Hastie Shane, Wojewoda Stéphane. Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch. Web-site. URL: http://www.infoq.com/articles/standish-chaos-2015 (Last accessed: August 20, 2020).

3. The Standish Group Report CHAOS. Web-site. URL: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf (Last accessed: August 20, 2020).

4. A Look at 25 Years of Software Projects. What Can We Learn? Web-site. URL: https://speedandfunction.com/look-25-years-software-projects-can-learn/ (Last accessed: August 20, 2020).

5. Макконнелл С. Совершенный код. Мастер-класс. Москва: Изд. «Русская редакция», 2013. 896 с.

6. Levenson N. G. Engineering a safer world: systems thinking applied to safety. MIT Press, 2012. 560 p.

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

8. Leveraging Natural Language Processing in Requirements Analysis: How to eliminate over half of all design errors before they occur. Web-site. URL: http://qracorp.com/wp-content/uploads/2017/03/Leveraging-NLP-in-Requirements-Analysis.pdf (Last accessed: August 20, 2020).

9. Шамие К. Системная инженерия для «чайников». New York: John Wiley & Sons, 2014. 76 с.

10. Hovorushchenko T., Krasiy A. Method of Evaluating the Success of Software Project Implementation Based on Analysis of Specification Using Neuronet Information Technologies. *CEUR-WS*. 2015. Vol. 1356. Pp. 100-107.

11. Krasiy A., Hovorushchenko T. Information Technology of Predicting the Characteristics and Evaluating the Success of Software Projects Implementation. *CEUR-WS*. 2016. Vol.1614. Pp.87-102.

12. Димо О. Б. Підвищення ефективності управління проєктами розробки програмного забезпечення з відкритим вихідним кодом: дис. ... канд. техн. наук: 05.13.22. Миколаїв, 2007. 120 с.

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

14. Харченко О., Яцишин В. Розробка та керування вимогами до програмного забезпечення на основі моделі якості. *Вісник Тернопільського державного технічного університету*. 2009. Т. 14. № 1. С. 201-207.

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

16. Липаев В. В. Проєктирование и производство сложных заказных программных продуктов. Москва-Берлин: Директ-Медиа, 2015. 537 с.

17. Kharchenko V., Gordieiev O., Fedoseeva A. Profiling of software requirements for the pharmaceutical enterprise manufacturing execution system. *Studies in Computational Intelligence*. 2015. Vol. 606. Pp. 67-92.

18. Оценка и обеспечение качества программных средств космических систем: монография / Харченко В. С. и др. ; под ред. В. С. Харченка, Б. М. Конорева. Харьков: Нац. косм. агентство Украины, Гос. центр регулирования качества, НАУ «ХАИ», 2007. 244 с.

19. Gordieiev O., Kharchenko V., Fominykh N., Sklyar V. Evolution of software quality models in context of the standard ISO 25010. *Advances in Intelligent Systems and Computing*. 2014. Vol. 286. Pp. 223-232.

20. Gordieiev O., Kharchenko V., Leontiiev K. Usability, security and safety interaction: Profile and metrics based analysis. *Advances in Intelligent Systems and Computing*. 2019. Vol. 761. Pp. 238-247.

21. Gordieiev O., Kharchenko V., Vereshchak K. Usable security versus secure usability: An assessment of attributes interaction. *CEUR WS*. 2017. Vol. 1844. Pp. 727-740.

22. Gordieiev O., Gordieieva D., Tryfonov A., Dokukin V., Odarushchenko E. Method and tool for support of software requirements profile quality assessment. *Proceedings of 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies* (Kyiv, Ukraine, May 14-16, 2020). Pp. 72-79.

23. Мищенко В. О. Компьютерное моделирование характеристик схем программных систем. *Радіоелектронні і комп'ютерні системи*. 2010. № 5. С. 158-164.

24. 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.

25. Яковина В., Федасюк Д., Мамроха Н. Якість програмного забезпечення. *Інженерія програмного забезпечення*. 2010. № 2. С. 24-29.

26. Яковина В. С. Вплив функції активації RBF нейронної мережі на ефективність прогнозування кількості відмов програмного забезпечення. *Вісник Національного університету «Львівська політехніка». Серія «Комп'ютерні науки та інформаційні технології»*. 2012. № 732. С. 36-39.

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

28. Gordieiev O., Kharchenko V., Fusani M. Software quality standards and models evolution: Greenness and reliability issues. *Communications in Computer and Information Science*. 2016. Vol. 594. Pp. 38-55.

29. Levenson N. G. Systemic factors in software-related spacecraft accidents. *AIAA Space 2001 Conference and Exposition*: Proceedings (Albuquerque, August 28-30, 2001). Albuquerque (USA), 2001. Pp. 1-11.

30. Levenson N. Software challenges in achieving space safety. *Journal of the British Interplanetary Society*. 2009. Vol. 62. Pp. 265-272.

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

32. Jones C. Applied software measurement: global analysis of productivity and quality. McGraw-Hill Education, 2008. 662 p.

33. Jones C. Estimating software costs: bringing realism to estimating. McGraw-Hill Education, 2007. 644 p.

34. Chen A., Beatty J. Visual models for software requirements. Washington: MS Press, 2012. 444 p.

35. Fatwanto A. Software requirements specification analysis using natural language processing technique. *The 13-th IEEE International Conference on Quality in Research*: Proceedings (Yogyakarta, June 25-28, 2013). Yogyakarta (Indonesia), 2013. Pp. 105-110.

36. Rehman T., Khan M. N. A., Riaz N. Analysis of requirement engineering processes, tools/techniques and methodologies. *International Journal of Information Technology and Computer Science*. 2013. Vol. 5. No. 3. Pp. 40-48.

37. Bass L., Bergey J., Clements P., Merson P., Ozkaya I., Sangwan R. A comparison of requirements specification methods from a software architecture perspective. Web-site. URL: http://resources.sei.cmu.edu/asset_files/TechnicalReport/2006_005_001_14786.pdf (Last accessed: August 20, 2020).

38. Wiegers K., Beatty J. Software requirements: 3rd edition. Washington: MS Press, 2013. 640 p.

39. Sommerville I. Software Engineering. London: Pearson, 2015. 816 p.

40. Андон Ф. И., Гришанова И. Ю., Резниченко В. А. Semantic Web как новая модель информационного пространства Интернет. *Проблеми програмування*. 2008. №2-3. Спеціальний випуск. С. 417-430.

41. Бабенко Л. Онтологический подход к спецификации свойств программных систем и их компонентов. *Кибернетика и системный анализ*. 2009. № 1. С. 180-187.

42. Li Y., Cleland-Huang J. Ontology-based trace retrieval. *The 7-th International Workshop on Traceability in Emerging Forms of Software Engineering:* Proceedings (Passau, May 19, 2013). Passau (Germany), 2013. Pp. 30-36.

43. Assawamekin N., Namfon A., Sunetnanta T., Pluempitiwiriyawej C. Ontology-based multiperspective requirements traceability framework. *Knowledge Information Systems*. 2010. No. 3 Pp. 493-522.

44. Leonid K., Gacitua R., Rouncefield M., Sawyer P. Ontology and model alignment as a means for requirements validation. *The 4-th IEEE International Conference on Semantic Computing*: Proceedings (Pittsburgh, September 22-24, 2010). Pittsburgh (USA), 2010. Pp. 46-51.

45. Bajnaid N. O., Benlamri R., Pakstas A., Salekzamankhani Sh. An ontological approach to model software quality assurance knowledge domain. *Lecture Notes on Software Engineering*. 2016. Vol. 4. No. 3. Pp. 193-198.

46. Freitas A., Panisson A. R., Hilgert L., Meneguzzi F., Vieira R., Bordini R. H. Applying ontologies to the development and execution of Multi-Agent Systems. *Web Inteliigence*. 2017. Vol. 15. Issue 4. Pp. 291-302.

47. Freitas A., Bordini R. H., Vieira R. Model-driven engineering of multi-agent systems based on ontologies. *Applied Ontology*. 2017. Vol. 12. Issue 2. Pp. 157-188.

48. Ossowska K., Szewc L., Weichbroth P., Garnik I., Sikorski M. Exploring an Ontological Approach for User Requirements Elicitation in the Design of Online Virtual Agents. *Information Systems: Development, Research, Applications, Education*. 2017. Vol. 264. Pp. 40-55.

49. Garcia-Magarino I., Gomez-Sanz J. J. An ontological and agent-oriented modeling approach for the specification of intelligent Ambient Assisted Living systems for Parkinson patients. *Hybrid Artificial Intelligent Systems*. 2013. Vol. 8073. Pp. 11-20.

50. Rakib A., Faruqui R. U. A formal approach to modelling and verifying resource-bounded context-aware agents. *Lecture Notes of the Institute for Computer Sciences Social Informatics and Telecommunications Engineering*. 2013. Vol. 109. Pp. 86-96.

51. Hovorushchenko T., Pavlova O., Fedula M. Improving the input information for medical software requirements specifications using ontology-based intelligent agent. *CEUR-WS*. 2018. Vol. 2255. Pp.113-125.

52. Hovorushchenko T., Pavlova O. Method of Activity of Ontology-Based Intelligent Agent for Evaluating the Initial Stages of the Software Lifecycle. *Advances in Intelligent Systems and Computing*. 2019. Vol. 836. Pp. 169-178.

53. Hovorushchenko T., Pavlova O., Bodnar M. Development of an Intelligent Agent for Analysis of Nonfunctional Characteristics in Specifications of Software Requirements. *Eastern-European Journal of Enterprise Technologies*. 2019. Vol. 1. No. 2 (97). Pp. 6-17.

54. Hovorushchenko T., Pavlova O. Intelligent System for Determining the Sufficiency of Metric Information in the Software Requirements Specifications. *CEUR-WS*. 2019. Vol. 2353. Pp.253-266.

55. Hovorushchenko T., Boyarchuk A., Pavlova O., Bobrovnikova K. Agent-Oriented Information Technology for Assessing the Initial Stages of the Software Life Cycle. *CEUR-WS*. 2019. Vol. 2393. Pp.617-632.

56. Hovorushchenko T., Boyarchuk A., Pavlova O. Ontology-Based Intelligent Agent for Semantic Parsing the Software Requirements Specifications. *International Journal on Information Technologies and Security*. 2019. No. 2. Vol. 11. Pp.59-70.

57. Hovorushchenko T., Pavlova O., Boyarchuk A. Modelling of non-functional characteristics of the software for selection of accurate scope of information for their evaluation. *CEUR-WS*. 2019. Vol. 2533. Pp. 206-216.

58. Hovorushchenko T., Pavlova O., Medzatyi D. Ontology-Based Intelligent Agent for Determination of Sufficiency of Metric Information in the Software Requirements. *Advances in Intelligent Systems and Computing*. 2020. Vol. 1020. Pp. 447-460.

59. Boyarchuk A., Pavlova O., Bodnar M., Lopatto I. Approach to the Analysis of Software Requirements Specification on Its Structure Correctness. *CEUR-WS*. 2020. Vol. 2623. Pp. 85-95.

60. Говорущенко Т.О., Іванов О.В., Павлова О.О. Метод оцінювання достатності інформації для визначення якості програмного забезпечення на основі зваженої онтології. *Вісник Хмельницького національного університету. Серія «Технічні науки»*. Хмельницький: ХНУ. 2016. №5. С.146-156.

61. Говорущенко Т. О., Павлова О. О. Сучасні проблеми оцінювання початкових етапів життєвого циклу програмного забезпечення. *Електротехнічні та комп'ютерні системи*. 2018. №27 (103). С. 165-175.

62. Говорущенко Т. О., Поморова О. В., Павлова О. О. Моделювання діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до програмного забезпечення. *Вісник Хмельницького національного університету. Серія «Технічні науки»*. 2018. № 4. С. 128-137.

63. Павлова О. О., Говорущенко Т. О., Іванов О. В. Діяльність інтелектуального агента для оцінювання інформації у специфікаціях вимог до програмного забезпечення. *Штучний інтелект*. 2018. №2. С. 66-75.

64. Говорущенко Т. О., Павлова О.О., Боднар М. А. Сучасні проблеми семантичного аналізу специфікацій вимог до програмного забезпечення. *Вчені записки Таврійського національного університету ім. В. І. Вернадського. Серія «Технічні науки»*. 2019. Том 30 (69). №1. Частина 1. С. 38-43.

65. Говорущенко Т.О., Павлова О.О., Тоненька М.М. Структура агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу. *Вісник*

Хмельницького національного університету. Серія «Технічні науки». 2020. №1. С. 77-81.

66. Павлова О.О., Боднар М.А., Гнатчук Є.Г. Метод діяльності та реалізація інтелектуального агента на основі онтологічного підходу для парсингу природомовних специфікацій вимог до програмного забезпечення. *Вісник Хмельницького національного університету. Серія «Технічні науки».* 2020. №2. С.171-175.

67. Павлова О.О., Лопатто І.Ю., Говорущенко Т.О. Метод діяльності та структура інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення. *Вісник Хмельницького національного університету. Серія «Технічні науки»*. 2020. №3. С. 61-64.

68. Hovorushchenko T., Pavlova O. Evaluating the Software Requirements Specifications Using Ontology-Based Intelligent Agent. *Proceedings of 2018 IEEE International Scientific and Technical Conference "Computer Science and Information Technologies"* (Lviv, Ukraine, September 11-14, 2018). Vol.1. Pp.215-218.

69. Pavlova O., Hovorushchenko T., Boyarchuk A. Method of activity of intelligent agent for semantic analysis of software requirements. *Proceedings of the 2019 IEEE 10-th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (Mez, France, September 18-21, 2019). Vol.2. Pp. 902-906.

70. Hovorushchenko T., Lopatto I., Pavlova O. Concept of Intelligent Agent for Verification of Considering the Subject Area Information. *Proceedings of 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies* (Kyiv, Ukraine, May 14-16, 2020). Pp. 465-469.

71. Говорущенко Т.О., Павлова О.О. Аналіз сучасного стану інформаційних технологій для галузі інженерії програмного забезпечення. *Матеріали Всеукраїнської науково-практичної конференції «Сучасні тенденції розвитку додрукарських систем»* (Львів, Україна, 19 квітня 2018 р.). С. 32-33.

72. Говорущенко Т. О., Павлова О. О., Медзатий Д. М. Інтелектуальний агент на основі онтологічного підходу для визначення достатності метричної інформації у вимогах до програмного забезпечення. *Матеріали Міжнародної наукової конференції "Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту"* (Херсон, Україна, травень 2019 р.). С. 38-40.

73. Павлова О.О. Інтелектуальна система для визначення достатності метричної інформації у вимогах до програмного забезпечення. Збірник наукових праць молодих науковців і студентів "Інтелектуальний потенціал-2019" (Хмельницький, Україна, листопад 2019 р.). С. 59-62.

74. А. с. 80645 Україна. Метод діяльності інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення / Т. О. Говорущенко, О. О. Павлова. 2018.

75. А. с. 89841 Україна. Інтелектуальна система для визначення достатності метричної інформації у специфікаціях вимог до програмного забезпечення / Т. О. Говорущенко, О. О. Павлова. 2019.

76. А. с. 89840 Україна. Метод діяльності інтелектуального агента на основі онтологічного підходу для семантичного парсингу природомовних специфікацій вимог до програмного забезпечення / Т. О. Говорущенко, О. О. Павлова. 2019.

77. А. с. 97014 Україна. Інтелектуальна інформаційно-аналітична технологія для підвищення якості програмного забезпечення шляхом оцінювання достатності інформації на ранніх етапах життєвого циклу / Т. О. Говорущенко, О. О. Павлова. 2020.

78. А. с. 97015 Україна. Комп'ютерна програма «Веб-орієнтована інформаційно-аналітична система оцінювання достатності інформації у специфікаціях вимог до програмного забезпечення» / О. О. Павлова, Т. О. Говорущенко. 2020.

79. Maedche A., Botzenhardt A., Neer L. Software for people: fundamentals, trends and best practices (Management for professionals). Springer-Verlag Berlin Heidelberg, 2012. 293 p.

80. Latest study shows rise in project failures. Web-site. URL: http://kinzz.com/resources/articles/91-project-failures-rise-study-shows (Last accessed: August 20, 2020).

81. CHAOS Summary 2009. The 10 laws of CHAOS. Web-site. URL: https://www.classes.cs.uchicago.edu/archive/2014/fall/51210~1/required.reading/Standish.Group.Chaos.2009.pdf (Last accessed: August 20, 2020).

82. The Standish Group International: CHAOS Manifesto – Think big, act small. Technical report,CHAOS Knowledge Center (2013). Web-site. URL: http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf (Last accessed: August 20, 2020).

83. PMI's Pulse of the Profession 9-th Global Project Management Survey, 2017. Web-site. URL: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf (Last accessed: August 20, 2020).

84. Leonard J. CSCE 606: Introduction. Web-site. URL: https://slideplayer.com/slide/15545897/ (Last accessed: August 20, 2020).

85. Mersino A. Agile Project Success Rates are 2X Higher than Traditional Projects. Web-site. URL: https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/ (Last accessed: August 20, 2020).

86. 4-Step Project Plan for 2020. Web-site. URL: http://projexs.io/project-plan-made-easy/ (Last accessed: August 20, 2020).

87. Cost of a bug within a software lifecycle. Web-site. URL: http://www.testically.org/2012/02/09/cost-of-a-bug-within-a-software-lifecycle/ (Last accessed: August 20, 2020).

88. Delivering large-scale IT projects on time, on budget, and on value. Web-site. URL: https://www.mckinsey.com/~/media/McKinsey/dotcom/client_service/BTO/PDF/MOBT_27_Delivering_large-scale_IT_projects_on_time_budget_and_value.ashx (Last accessed: August 21, 2020).

89. A Systemic approach for assessing software supply-chain risk / A.Dorofee and others. *The 44-th Hawaii International Conference on System Sciences*: Proceedings (Honolulu, January 04-07, 2011). Honolulu (USA), 2011. Pp. 1-8.

90. Ghadge A., Dani S., Chester M., Kalawsky R. A systems approach for modelling supply chain risks. *Supply Chain Management: An International Journal*. 2013. Vol. 18 (5). Pp. 523-538.

91. Lynch J. Project Resolution Benchmark Report. Web-site. URL: https://www.standishgroup.com/sample_research_files/DemoPRBR.pdf (Last accessed: August 21, 2020).

92. Johnson J. CHAOS Report: Decision Latency Theory: It Is All About the Interval. The Standish Group, 2018. 68 p.

93. Sarif Sh., Ramly S., Yusof R., Fadzillah N. A. A., Sulaiman N. Y. Investigation of Success and Failure Factors in IT Project Management. *Advances in Intelligent Systems and Computing*. 2018. Vol. 739. Pp. 671-682.

94. Rosato M. Go Small for Project Success. *PM World Journal*. 2018. Vol. 7. Issue 5. Pp. 1-10.

95. Meziane F., Vadera S. Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects. *Advances in Intelligent Information Technologies*. 2010. Pp. 278-299.

96. Maier R. Knowledge management systems. Information and communication technologies for knowledge management. *Springer Science & Business Media*, 2013. 635 p.

97. Hazard analysis of complex spacecraft using systems-theoretic process analysis / T. Ishimatsu and others. *Journal of Spacecraft and Rockets*. 2014. Vol. 51. No. 2. Pp. 509-522.

98. 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).

99. 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).

100. 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).

101. Куликов С. Тестирование программного обеспечения. Базовый курс. 3-е издание. Веб-сайт. URL: https://svyatoslav.biz/software_testing_book_download/ (дата обращения 21.08.2020).

102. 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).

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

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

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

106. Kan S. Metrics and models in software quality engineering. Addison-Wesley Professional, 2002. 560 p.

107. Cruickshank K. J. A validation metrics framework for safety-critical software-intensive systems. Monterey: Naval Postraduate School, 2009. 144 p.

108. Michael J. B., Shing M. T., Cruickshank K. J., Redmond P. J. Hazard analysis and validation metrics framework for system of systems software safety. *IEEE Systems Journal*. 2010. Vol. 4. Issue 2. Pp. 186-197.

109. Baker R., Habli I. An empirical evaluation of mutation testing for improving the test quality of safety-critical software. *IEEE Transactions on Software Engineering*. 2013. Vol. 39. Issue 6. Pp. 787-805.

110. Software Requirements Engineering Tools. Web-site. URL: http://ecomputernotes.com/software-engineering/softwarerequirements engineeringtools (Last accessed: August 21, 2020).

111. 30 Best Requirements Management Tools in 2019. Web-site. URL: https://www.guru99.com/requirement-management-tools.html (Last accessed: August 21, 2020).

112. Top 20+ Best Requirements Management Tools (The Complete List). Web-site. URL: https://www.softwaretestinghelp.com/requirements-management-tools/ (Last accessed: August 21, 2020).

113. Verma K., Kass A. Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents. *Lecture Notes in Computer Science*. 2008. Vol. 5318. Pp. 751-763.

114. Mahmood H., Rehman M. S. Tools for software engineers. *International Journal of Research in Engineering & Technology*. 2015. Vol. 3. Issue 10. Pp. 75-86.

115. Jones V., Murray J. Evaluation of current requirements analysis tools capabilities for IV&V in the requirements analysis phase. Web-site. URL: https://www.slideserve.com/shlomo/evaluation-of-current-requirements-analysis-tools-capabilities-for-ivv-in-the-requirements-analysis-phase (Last accessed: August 21, 2020).

116. Raffo D. M., Ferguson R. Evaluating the Impact of The QuARS Requirements Analysis Tool Using Simulation. Web-site. URL: https://pdfs.semanticscholar.org/7e7d/4e6f5ab13d00ca57c87711e30cd080730f34.pdf (Last accessed: August 21, 2020).

117. Konig F., Ballejos L., Ale M. A semi-automatic verification tool for software requirements specification documents. *Simposio Argentino de Ingeniería de Software*: Proceedings. (Cordoba, September, 2017). Cordoba, 2017. Pp.75-83.

118. Wu W., Li H., Wang H., Zhu K. Q. Probase: a probabilistic taxonomy for text understanding. *The 2012 ACM International Conference on Management of Data (SIGMOD)*: Proceedings (Scottsdale, May 20-24, 2012). Scottsdale (USA), 2012. Pp.13-24.

119. Roussey C., Pinet F., Kang M. A., Corcho O. An Introduction to Ontologies and Ontology Engineering. *Ontologies in Urban Development Projects, Advanced Information and Knowledge Processing.* 2011. Vol.1. Pp. 9-38.

120. Burov E. Complex ontology management using task models. *International Journal of Knowledge-Based and Intelligent Engineering Systems*. 2014. Vol. 18. No. 2. Pp. 111-120.

121. Burov E., Pasitchnyk V., Gritsyk V. Modeling software testing processes with task ontologies. *British Journal of Education and Science*. 2014. No. 2(6). Pp. 256-263.

122. Zhang Y. G., Witte R., Rilling J., Haarslev V. Ontological approach for the semantic recovery of traceability links between software artefacts. *IET Software*. 2008. Vol. 2. Issue 3. Pp. 185-203.

123. Jin D., Cordy J. R. Ontology-based software analysis and reengineering tool integration: The OASIS service-sharing methodology. *The 21-st IEEE International Conference on Software Maintenance*: Proceedings (Budapest, September 25-30, 2005). Budapest (Hungary), 2005. Pp. 613-616.

124. Wooldridge M., Jennings N. R. Intelligent agents – theory and practice. Knowledge Engineering Review. 1995. Vol. 10. Issue 2. Pp. 115-152.

125. Weiss G. Multiagent systems. Cambridge: The MIT Press. 2013. 920 p.

126. Lytvyn V., Oborska O., Vovnjanka R. Approach to decision support intelligent systems development based on ontologies. *Econtechmod*. 2015. Vol. 4. No 4. Pp. 29-35.

127. Lezcano-Rodriguez L. A., Guzman-Luna J. A. Ontological characterization of basics of KAOS chart from natural language. *Revista Iteckne*. 2016. Vol. 13. Issue 2. Pp. 157-168.

128. Hilaire V., Cossentino M., Gechter F., Rodriguez S., Koukam A. An approach for the integration of swarm intelligence in MAS: An engineering perspective. *Expert Systems with Applications*. 2013. Vol. 40. Issue 4. Pp. 1323-1332.

129. Wilk S., Michalowski W., O'Sullivan D., Farion K., Sayyad-Shirabad J., Kuziemsky C., Kukawka B. A task-based support architecture for developing point-of-care clinical decision support systems for the emergency department. *Methods of Information in Medicine*. 2013. Vol. 52. Issue 1. Pp. 18-32.

130. Meyer O., Gruhn V. Towards Concept based Software Engineering for Intelligent Agents. *2019 IEEE/ACM 7-th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*: Proceedings. 2019.

131. De Olveira C. D. C., Cintra M. E., Neto F. M. M. Learning Risk Management in Software Projects with a Serious Game Based on Intelligent Agents and Fuzzy Systems. *The 8-th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)*: Proceedings. 2013.

132. Yan H. Related Discussion on Agent-oriented Programming. *AER-Advances in Engineering Research*. 2016. Vol. 67. Pp. 1315-1317.

133. Rahman A. A., Abdullah M., Alias S. H. The Architecture of Agent-Based Intelligent Tutoring System for the Learning of Software Engineering Function Point Metrics. *2016 2-nd International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR)*: Proceedings. 2016.

134. Simons C. L., Parmee I. C. User-centered, Evolutionary Search in Conceptual Software Design. *IEEE Congress on Evolutionary Computation*. 2008. Pp. 869-876.

135. Akkawi F., Bader A., Elrad T. The multi-layered approach to building intelligent systems. *International Conference on Artificial Intelligence*: Proceedings. 2001.

136. Karasev V. O., Sukhanov V. A. Product lifecycle management using multi-agent systems models. *Procedia Computer Science*. 2017. Vol. 103. Pp. 142-147.

137. Abadi A., Sekkat S., Zemmouri E., Benazza H. Using ontologies for the integration of information systems dedicated to product (CFAO, PLM...) and those of

systems monitoring (ERP, MES…). *10-th International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA)*: Proceedings. 2017.

138. Philip T., Konda R. A model to use software agents during software life-cycle. *Computers and Their Applications*. 2001. Pp. 530-536.

139. Gulia S., Choudhury T. An Efficient Automated Design to Generate UML Diagram From Natural Language Specifications. *The 6-th International Conference on Cloud System and Big Data Engineering*: Proceedings. 2016. Pp. 641-648.

140. Selway M., Grossman G., Mayer W., Stumptner M. Formalising natural language specifications using a cognitive linguistic/configuration based approach. *Information Systems*. 2015. Vol. 54. Pp. 191-208.

141. Ali S. W., Ahmed Q. A., Shafi I. Process to Enhance the Quality of Software Requirement Specification Document. *The International Conference on Engineering and Emerging Technologies*: Proceedings. 2018. Pp. 113-118.

142. Diamantopoulos T., Roth M., Symeonidis A., Klein E. Software requirements as an application domain for natural language processing. *Language Resources and Evaluation*. 2017. Vol. 51 (2). Pp. 495-524.

143. Wang Y. Semantic Information Extraction for Software Requirements using Semantic Role Labeling. *The IEEE International Conference on Progress in Informatics and Computing*: Proceedings. 2015. Pp. 332-337.

144. Riaz M., King J., Slankas J., Williams L. Hidden in Plain Sight: Automatically Identifying Security Requirements from Natural Language Artifacts. *The 2-nd IEEE International Requirements Engineering Conference*: Proceedings. 2014. Pp. 183-192.

145. Murtazina M., Avdeenko T. The ontology-driven approach to support the requirements engineering process in Scrum framework. *CEUR-WS*. 2018. Vol. 2212. Pp. 287-295.

146. Iwama F., Nakamura T., Takeuchi H. Constructing Parser for Industrial Software Specifications Containing Formal and Natural Language Description. *The 34-th International Conference on Software Engineering*: Proceedings. 2012. Pp. 1012-1021.

147. Gnesi S., Lami G., Trentanni G., Fabbrini F., Fusani M. An automatic tool for the analysis of natural language requirements. *Computer Systems Science and Engineering.* 2005. Vol. 20 (1). Pp. 53-62.

148. Siegemund K. Contributions to Ontology-Driven Requirements Engineering: Dissertation. Dresden: Technischen Universität Dresden, 2014.

149. Farfeleder S., Moser T., Krall A., Stalhane T., Omoroniya I., Zojer H. Ontology-Driven Guidance for Requirements Elicitation. *Lecture Notes in Computer Science*. 2011. Vol. 6644. Pp. 212-226.

150. Mustafa A., Wan-Kadir W. M. N., Ibrahim N., Shah A., Younas M. Integration of Heterogeneous Requirements using Ontologies. *International Journal of Advanced Computer Science and Applications*. 2018. Vol. 9 (5). Pp. 213-218.

151. Su J., Sachenko A., Lytvyn V., Vysotska V., Dosyn D. Model of Touristic Information Resources Integration According to User Needs. The 2018 IEEE 13-th International Scientific and Technical Conference on Computer Sciences and Information Technologies: Proceedings. 2018. Vol. 2. Pp. 113-116.

152. ISO/IEC/IEEE 29148:2018. Systems and software engineering. Life cycle processes. Requirements engineering. [Introduced 01.12.2018]. Geneva (Switzerland), 2018. 28 p. (International standard).

ДОДАТОК А.
СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Основні наукові публікації у періодичних виданнях, що індексуються в наукометричних базах SCOPUS, Web of Science:

1. Hovorushchenko T., Pavlova O., Bodnar M. Development of an Intelligent Agent for Analysis of Nonfunctional Characteristics in Specifications of Software Requirements. *Eastern-European Journal of Enterprise Technologies*. 2019. Vol. 1. No. 2 (97). Pp. 6-17. *(Scopus, Q2)*

2. Hovorushchenko T., Boyarchuk A., Pavlova O. Ontology-Based Intelligent Agent for Semantic Parsing the Software Requirements Specifications. *International Journal on Information Technologies and Security*. 2019. No. 2. Vol. 11. Pp.59-70. *(WoS, Bulgaria)*

Публікації у періодичних виданнях, що індексуються в наукометричних базах SCOPUS, Web of Science:

3. Hovorushchenko T., Pavlova O., Fedula M. Improving the input information for medical software requirements specifications using ontology-based intelligent agent. *CEUR-WS*. 2018. Vol. 2255. Pp.113-125. *(Scopus, WoS)*

4. Hovorushchenko T., Pavlova O. Method of Activity of Ontology-Based Intelligent Agent for Evaluating the Initial Stages of the Software Lifecycle. *Advances in Intelligent Systems and Computing*. 2019. Vol. 836. Pp. 169-178. *(Scopus)*

5. Hovorushchenko T., Pavlova O. Intelligent System for Determining the Sufficiency of Metric Information in the Software Requirements Specifications. *CEUR-WS*. 2019. Vol. 2353. Pp.253-266. *(Scopus)*

6. Hovorushchenko T., Boyarchuk A., Pavlova O., Bobrovnikova K. Agent-Oriented Information Technology for Assessing the Initial Stages of the Software Life Cycle. *CEUR-WS*. 2019. Vol. 2393. Pp.617-632. *(Scopus)*

7. Hovorushchenko T., Pavlova O., Boyarchuk A. Modelling of non-functional characteristics of the software for selection of accurate scope of information for their evaluation. *CEUR-WS*. 2019. Vol. 2533. Pp. 206-216. *(Scopus, WoS)*

8. Hovorushchenko T., Pavlova O., Medzatyi D. Ontology-Based Intelligent Agent for Determination of Sufficiency of Metric Information in the Software Requirements. *Advances in Intelligent Systems and Computing*. 2020. Vol. 1020. Pp. 447-460. *(Scopus)*

9. Boyarchuk A., Pavlova O., Bodnar M., Lopatto I. Approach to the Analysis of Software Requirements Specification on Its Structure Correctness. *CEUR-WS*. 2020. Vol. 2623. Pp. 85-95. *(Scopus)*

Статті у фахових наукових виданнях України:

10. Говорущенко Т.О., Іванов О.В., Павлова О.О. Метод оцінювання достатності інформації для визначення якості програмного забезпечення на основі зваженої онтології. *Вісник Хмельницького національного університету. Серія «Технічні науки»*. Хмельницький: ХНУ. 2016. №5. С.146-156.

11. Говорущенко Т. О., Павлова О. О. Сучасні проблеми оцінювання початкових етапів життєвого циклу програмного забезпечення. *Електротехнічні та комп'ютерні системи*. 2018. №27 (103). С. 165-175.

12. Говорущенко Т. О., Поморова О. В., Павлова О. О. Моделювання діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до програмного забезпечення. *Вісник Хмельницького національного університету. Серія «Технічні науки»*. 2018. № 4. С. 128-137.

13. Павлова О. О., Говорущенко Т. О., Іванов О. В. Діяльність інтелектуального агента для оцінювання інформації у специфікаціях вимог до програмного забезпечення. *Штучний інтелект*. 2018. №2. С. 66-75.

14. Говорущенко Т. О., Павлова О.О., Боднар М. А. Сучасні проблеми семантичного аналізу специфікацій вимог до програмного забезпечення. *Вчені записки Таврійського національного університету ім. В. І. Вернадського. Серія «Технічні науки»*. 2019. Том 30 (69). №1. Частина 1. С. 38-43.

15. Говорущенко Т.О., Павлова О.О., Тоненька М.М. Структура агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу. *Вісник*

Хмельницького національного університету. Серія «Технічні науки». 2020. №1. С. 77-81.

16. Павлова О.О., Боднар М.А., Гнатчук Є.Г. Метод діяльності та реалізація інтелектуального агента на основі онтологічного підходу для парсингу природомовних специфікацій вимог до програмного забезпечення. *Вісник Хмельницького національного університету. Серія «Технічні науки».* 2020. №2. С.171-175.

17. Павлова О.О., Лопатто І.Ю., Говорущенко Т.О. Метод діяльності та структура інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2020. №3. С. 61-64.

Статті в матеріалах конференцій, що індексуються в наукометричних базах SCOPUS, Web of Science:

18. Hovorushchenko T., Pavlova O. Evaluating the Software Requirements Specifications Using Ontology-Based Intelligent Agent. *Proceedings of 2018 IEEE International Scientific and Technical Conference "Computer Science and Information Technologies"* (Lviv, Ukraine, September 11-14, 2018). Vol.1. Pp.215-218. *(Scopus, WoS)*

19. Pavlova O., Hovorushchenko T., Boyarchuk A. Method of activity of intelligent agent for semantic analysis of software requirements. *Proceedings of the 2019 IEEE 10-th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (Mez, France, September 18-21, 2019). Vol.2. Pp. 902-906. *(Scopus, WoS)*

20. Hovorushchenko T., Lopatto I., Pavlova O. Concept of Intelligent Agent for Verification of Considering the Subject Area Information. *Proceedings of 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies* (Kyiv, Ukraine, May 14-16, 2020). Pp. 465-469. *(Scopus)*

Публікації у матеріалах конференцій (тези доповідей):

21. Говорущенко Т.О., Павлова О.О. Аналіз сучасного стану інформаційних технологій для галузі інженерії програмного забезпечення. *Матеріали*

Всеукраїнської науково-практичної конференції «Сучасні тенденції розвитку додрукарських систем» (Львів, Україна, 19 квітня 2018 р.). С. 32-33.

22. Говорущенко Т. О., Павлова О. О., Медзатий Д. М. Інтелектуальний агент на основі онтологічного підходу для визначення достатності метричної інформації у вимогах до програмного забезпечення. *Матеріали Міжнародної наукової конференції "Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту"* (Херсон, Україна, травень 2019 р.). С. 38-40.

23. Павлова О.О. Інтелектуальна система для визначення достатності метричної інформації у вимогах до програмного забезпечення. Збірник наукових праць молодих науковців і студентів "Інтелектуальний потенціал-2019" (Хмельницький, Україна, листопад 2019 р.). С. 59-62.

Свідоцтва про реєстрацію авторського права на твір

24. А. с. 80645 Україна. Метод діяльності інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення / Т. О. Говорущенко, О. О. Павлова. 2018.

25. А. с. 89841 Україна. Інтелектуальна система для визначення достатності метричної інформації у специфікаціях вимог до програмного забезпечення / Т. О. Говорущенко, О. О. Павлова. 2019.

26. А. с. 89840 Україна. Метод діяльності інтелектуального агента на основі онтологічного підходу для семантичного парсингу природомовних специфікацій вимог до програмного забезпечення / Т. О. Говорущенко, О. О. Павлова. 2019.

27. А. с. 97014 Україна. Інтелектуальна інформаційно-аналітична технологія для підвищення якості програмного забезпечення шляхом оцінювання достатності інформації на ранніх етапах життєвого циклу / Т. О. Говорущенко, О. О. Павлова. 2020.

28. А. с. 97015 Україна. Комп'ютерна програма «Веб-орієнтована інформаційно-аналітична система оцінювання достатності інформації у специфікаціях вимог до програмного забезпечення» / О. О. Павлова, Т. О. Говорущенко. 2020.

ДОДАТОК Б.
АКТИ ВПРОВАДЖЕННЯ

«Затверджую»

Директор ТОВ «ІТТ»

_____ Сімогук В.С.

«10» березня 2020 р.

АКТ

впровадження результатів дисертаційної роботи
«Агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу»
Павлової Ольги Олександрівни

Комісія у складі: системного адміністратора Веремеєнко В.А., інженера Вінтера Ю.Г., інженера Іванової М.В. склала цього акта про впровадження результатів дисертаційної роботи здобувача наукового ступеня доктора філософії О. О. Павлової на підприємстві ТОВ «ІТТ» у тому, що вона проводила роботи з впровадження агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення для оцінювання специфікацій вимог до програмного забезпечення (ПЗ) обліку надання послуг доступу до мережі Інтернет, розроблених для ТОВ «ІТТ» різними софтверними компаніями м. Хмельницького.

В процесі розв'язання здобувачем науково-прикладної задачі автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення, Павловою О. О. були одержані особисто та використані на підприємстві ТОВ «ІТТ» такі наукові та практичні результати:

1) агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу, яка виконує оцінювання та забезпечує підвищення рівня достатності інформації вимог для визначення кожної нефункційної характеристики окремо та всіх нефункційних характеристик разом;

2) розроблена та впроваджена агентно-орієнтована інформаційна технологія дозволила автоматизувати виконання трудомісткого, рутинного та схильного до помилок завдання розбору специфікацій вимог та майже миттєве його виконання; надала підказки, де потрібна повторна робота над специфікацією вимог (за рахунок перегляду відсутніх атрибутів та областей специфікації, яким потрібна додаткова увага); надала можливість вибору більш якісних специфікацій програмних вимог; забезпечила безкоштовний доступ через Інтернет в будь-який час без будь-якої реєстрації.

Веремеєнко В.А.

Вінтер Ю.Г.

Іванова М.В.

«Затверджую»
Голова
ГО «ІТ-КЛАСТЕР міста Хмельницького»
_____ Яцишен С.О.
«__» _____ 2020 р.

АКТ
впровадження результатів дисертаційної роботи
Павлової Ольги Олександрівни

Голова ГО «ІТ-КЛАСТЕР міста Хмельницького» Яцишен С.О. склав цього акта про впровадження результатів дисертаційної роботи «Агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу» здобувача наукового ступеня доктора філософії О. О. Павлової у ГО «ІТ-КЛАСТЕР м. Хмельницького» у тому, що вона проводила роботи з впровадження агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу програмного забезпечення для оцінювання специфікацій вимог до програмного агента для підвищення безпеки програмного забезпечення комп'ютерних систем.

В процесі розв'язання здобувачем науково-прикладної задачі автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення, Павловою О. О. були одержані особисто та використані у ГО «ІТ-КЛАСТЕР міста Хмельницького» такі наукові та практичні результати:

- агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу, яка виконує оцінювання та забезпечує підвищення рівня достатності інформації вимог для визначення кожної нефункційної характеристики окремо та всіх нефункційних характеристик разом.
- розроблена та впроваджена агентно-орієнтована інформаційна технологія оцінює та забезпечує приріст рівня достатності інформації у специфікації вимог для визначення нефункційних характеристик програмного забезпечення – приріст рівня достатності становить від 4,71% до 27,79% (наприклад, з 58,23% до 86,02% для специфікації вимог №1 до програмного агента для підвищення безпеки програмного забезпечення комп'ютерних систем, від 81,26% до 85,97% для специфікації №2, з 60,85% до 73,7% для специфікації №3);
- розроблена та впроваджена агентно-орієнтована інформаційна технологія забезпечує навчання для нових розробників специфікацій, системних інженерів та менеджерів проєктів (використання цієї технології допомагає їм побачити помилки, які вони можуть допускати, та допомагає їм розпізнати ці помилки в роботі інших); допомога у розробці вимог високої якості; допомога у виправленні та усуненні помилок у вимогах там, де вони виникають – на ранніх етапах життєвого циклу програмного забезпечення – до того, як вони стануть дорожчими для виправлення.

«Затверджую»
Директор ТОВ «Деймос»
Пантелєєв В.І.
«20» 02 2020 р.

АКТ

впровадження результатів дисертаційної роботи
Павлової Ольги Олександрівни

Комісія в складі: головного інженера Пантелєєва Г.І., системного адміністратора Артеменка В.О., менеджера з інформації та програмного забезпечення Шимко І.О. склала цього акта про впровадження результатів дисертаційної роботи «Агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу» здобувача наукового ступеня доктора філософії О. О. Павлової на ТОВ «Деймос» у тому, що вона проводила роботи з впровадження інтелектуального агента на основі онтологічного підходу для оцінки початкових етапів життєвого циклу програмного забезпечення на предмет оцінювання достатності інформації у специфікації.

В процесі розв'язання здобувачем науково-прикладної задачі автоматизації оцінювання початкових етапів життєвого циклу програмного забезпечення, Павловою О. О. були одержані особисто і використані на ТОВ «Деймос» такі результати наукових досліджень та практичні результати:

1) розроблено метод діяльності та реалізовано інтелектуальний агент на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення, який здійснює оцінювання достатності інформації у специфікації вимог для визначення всіх нефункційних характеристик-складових якості ПЗ, забезпечує висновок про достатність або недостатність інформації у специфікації, надає числові оцінки рівня достатності інформації для визначення кожної нефункційної характеристики ПЗ та для визначення всіх нефункційних

характеристик-складових якості ПЗ разом, формує список атрибутів, якими варто доповнити специфікацію вимог для підвищення рівня достатності її інформації, тобто в комплексі дозволяє частково усунути людину з процесів опрацювання інформації та здобуття знань;

2) аналізу за допомогою розробленого інтелектуального агента піддавалась специфікація вимог до ПЗ автоматизованої системи управління виробничими процесами ТОВ «Деймос», в результаті якого розроблений інтелектуальний агент надав висновок, що в аналізованій специфікації вимог до ПЗ недостатньо атрибутів для визначення всіх нефункційних характеристик. Рівень достатності наявної у специфікації вимог інформації для визначення «Надійності» склав 60%. Рівень достатності наявної у специфікації вимог інформації для визначення «Функційної придатності» склав 59%. Рівень достатності наявної у специфікації вимог інформації для визначення «Ефективності» склав 65%. Рівень достатності наявної у специфікації вимог інформації для визначення «Сумісності» склав 43%. Рівень достатності наявної у специфікації вимог інформації для визначення «Супроводжуваності» склав 57%. Рівень достатності наявної у специфікації вимог інформації для визначення «Можливості переносу» склав 52%. Рівень достатності наявної у специфікації вимог інформації для визначення «Захищеності» склав 45%. Рівень достатності наявної у специфікації вимог інформації для визначення «Зручності використання» склав 64%. Рівень достатності наявної у специфікації вимог інформації для визначення всіх нефункційних характеристик склав 59%. Інтелектуальний агент надав також висновок про необхідність доповнення цієї специфікації атрибутами, необхідними для обчислення всіх нефункційних характеристик».

Пантелєєв Г.І.

Артеменко В.О.

Шимко І.О.

«Затверджую»
Проректор з наукової роботи
Хмельницького національного університету
д.т.н., професор
_____ Синюк О.М.
«11» 05 2020 р.

АКТ

впровадження результатів дисертаційної роботи
«Агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу»
Павлової Ольги Олександрівни

Комісія Хмельницького національного університету в складі: завідувача кафедри комп'ютерної інженерії та системного програмування, д.т.н., професора Говорущенко Т.О.; декана Факультету програмування та комп'ютерних і телекомунікаційних систем, професора кафедри комп'ютерної інженерії та системного програмування д.т.н., професора Савенка О.С., доцента кафедри комп'ютерної інженерії та системного програмування, к.т.н., доцента Медзатого Д.М. склали цього акта в тому, що результати дисертаційної роботи здобувача наукового ступеня доктора філософії Павлової О.О. впроваджені у навчальний процес на кафедрі комп'ютерної інженерії та системного програмування, зокрема, в навчальних дисциплінах: «Інженерія програмного забезпечення» (2018-2020 рр), «Технології проєктування програмних систем» (2018-2020 рр), «Проєктування інтерфейсів користувача програмного забезпечення комп'ютерних систем» (2017-2020 рр), «Системна інженерія програмного забезпечення комп'ютерних систем» (2019-2020 рр).

При викладанні цих навчальних дисциплін використовувались і використовуються такі матеріали досліджень, проведених автором:

1) модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ, яка грунтується на

порівняльному аналізі онтологій та є теоретичним підґрунтям для реалізації інтелектуального агента на основі онтологічного підходу;

2) метод діяльності інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення, який працює на основі розробленої моделі та здійснює оцінювання достатності інформації у специфікації вимог для визначення всіх нефункційних характеристик-складових якості ПЗ, забезпечує висновок про достатність або недостатність інформації у специфікації, надає числові оцінки рівня достатності інформації для визначення кожної нефункційної характеристики ПЗ та для визначення всіх нефункційних характеристик-складових якості ПЗ разом, формує список атрибутів, якими варто доповнити специфікацію вимог для підвищення рівня достатності її інформації, тобто в комплексі дозволяє частково усунути людину з процесів опрацювання інформації та здобуття знань;

3) метод діяльності інтелектуального агента для автоматизованого семантичного аналізу (парсингу) специфікацій вимог до програмного забезпечення, який виконує парсинг специфікації, визначає кількість та відсоток відсутніх атрибутів, відображає, яких атрибутів не вистачає для тієї чи іншої підхарактеристики нефункційної характеристики, а також формує реальну онтологію для нефункційних характеристик, яка може бути використана інтелектуальним агентом на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення.

Використання зазначених результатів дозволили підвищити якість викладання зазначених навчальних дисциплін.

Говорущенко Т.О.

Савенко О.С.

Медзатий Д.М.

«Затверджую»
Ректор Закладу вищої освіти
«Міжнародний науково-технічний університет
імені академіка Юрія Бугая»
д.е.н., професор
_____ В. Ю. Худолей
«08» жовтня 2020 р.

АКТ

впровадження результатів дисертаційної роботи
Павлової Ольги Олександрівни

Комісія Закладу вищої освіти «Міжнародний науково-технічний університет імені академіка Юрія Бугая» у складі: Москаленка А. О., кандидата технічних наук, завідувача кафедри комп'ютерних наук та інженерії програмного забезпечення; Коротун Т. М., кандидата фізико-математичних наук, доцента, доцента кафедри комп'ютерних наук та інженерії програмного забезпечення; Слабоспицької О. О., кандидата фізико-математичних наук, старшого наукового співробітника, доцента кафедри комп'ютерних наук та інженерії програмного забезпечення, – склали цього акта в тому, що результати дисертаційної роботи «Агентно-орієнтована інформаційна технологія оцінювання початкових етапів життєвого циклу програмного забезпечення на основі онтологічного підходу» здобувача наукового ступеня доктора філософії Павлової О. О. впроваджені в освітній процес на кафедрі комп'ютерних наук та інженерії програмного забезпечення, зокрема, в навчальних дисциплінах: «Безпека інформаційних систем» (2018–2020 рр.), «Методи та системи підтримки прийняття рішень» (2019–2020 рр.) у вигляді конспектів лекцій, рекомендацій із вирішення задач, навчально-методичних матеріалів щодо дипломного проектування, тестових завдань.

Використання зазначених результатів дозволили підвищити якість викладання наведених навчальних дисциплін.

Т. М. Коротун
А. О. Москаленко
О. О. Слабоспицька

ДОДАТОК В

Лістинг модуля ядра програми AppKernel.php комп’ютерної програми «Інтелектуальний агент на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного забезпечення»

```
<?php
    use Symfony\Component\HttpKernel\Kernel;
    use Symfony\Component\Config\Loader\LoaderInterface;

    class AppKernel extends Kernel
    {
        public function registerBundles()
        {
            $bundles = [
                new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
                new Symfony\Bundle\SecurityBundle\SecurityBundle(),
                new Symfony\Bundle\TwigBundle\TwigBundle(),
                new Symfony\Bundle\MonologBundle\MonologBundle(),
                new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
                new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),
                new
Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
                new AppBundle\AppBundle(),
                new Doctrine\Bundle\FixturesBundle\DoctrineFixturesBundle(),
                new Symfony\WebpackEncoreBundle\WebpackEncoreBundle()
            ];

            if (in_array($this->getEnvironment(), ['dev', 'test'], true)) {
                $bundles[] = new Symfony\Bundle\DebugBundle\DebugBundle();
```

```
            $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();
            $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();
            $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();
        }

        return $bundles;
    }

    public function getRootDir()
    {
        return __DIR__;
    }

    public function getCacheDir()
    {
        return dirname(__DIR__).'/var/cache/'.$this->getEnvironment();
    }

    public function getLogDir()
    {
        return dirname(__DIR__).'/var/logs';
    }

    public function registerContainerConfiguration(LoaderInterface $loader)
    {
        $loader->load($this->getRootDir().'/config/config_'.$this->getEnvironment().'.yml');
    }
}
```