МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Начально-науковий інститут інноваційних освітніх технологій

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Савченко А.С.
"____" _____2020 р.

ДИПЛОМНА РОБОТА (ПОЯСНЮВАЛЬНА ЗАПИСКА) ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТРА" ЗА СПЕЦІАЛІЗАЦІЄЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)"

Тема: «Модель якості програмних систем на основі стандарту ISO 25010»

Виконавець: Малиновський Владислав Володимирович
Керівник: к.т.н., доцент Харченко Олександр Григорович

Нормоконтролер: _____

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Начально-науковий інститут інноваційних освітніх технологій

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, спеціалізація: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології (за галузями)".

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Савченко А.С.
" " 2019 р.

З А В Д А Н Н Я
на виконання дипломної роботи студента

Малиновський Владислав Володимирович

1. Тема роботи: «Модель якості програмних систем на основі стандарту ISO 25010»
Затверджена наказом ректора від
2. Термін виконання роботи: з 25.11.2019р. до 29 лютого 2020р. №2701/ст.
3. Вихідні данні до роботи: аналіз та дослідження моделей щодо якості програмного забезпечення. Оцінювання достатності інформації щодо якості за стандартом ISO 25010:2011 у специфікаціях вимог до програмного забезпечення.

4. Зміст пояснювальної записки:
Вступ
Розділ 1 Дослідження моделей щодо якості програмного забезпечення
Розділ 2 Моделювання процесу оцінювання якості програмного забезпечення
Розділ 3 Оцінювання достатності інформації щодо якості за стандартом ISO 25010:2011 у специфікаціях вимог до програмного забезпечення
Висновки
Список використаних джерел
Додатки

5. Перелік обов'язкового ілюстративного матеріалу: рисунки, слайди презентації доповіді у PowerPoint

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1	Розроблення та затвердження календарного плану виконання дипломної роботи	27.09.2019р. – 05.10.2019р.	
2	Аналіз і опрацювання літератури	06.10.2019р. – 18.10.2019р.	
3	Проведення консультації з науковим керівником щодо розділів дипломної роботи	19.10.2019р. – 21.10.2019р.	
4	Підготовка та написання розділу 1	22.10.2019р. – 31.10.2019р.	
5	Підготовка та написання розділу 2	01.11.2019р. – 16.11.2019р	
6	Проведення досліджень та опрацювання їх результатів	17.12.2019р. – 08.01.2020р.	
7	Підготовка та написання розділу 3	09.01.2020р. – 16.01.2020р.	
8	Оформлення пояснювальної записки	17.01.2020р. – 21.01.2020р.	
9	Оформлення графічної частини роботи	22.01.2020р.	
10	Подати дипломний роботи керівнику	23.01.2020р.	
11	Підготовка до захисту та попередній захист дипломної роботи	24.01.2020р. – 03.02.2020р.	

7. Дата видачі завдання 21.10.2019 р.

Керівник дипломної роботи . Харченко О.Г.

Завдання прийняв до виконання Малиновський В.В.

(підпис випускника)
(ПІБ)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Модель якості програмних систем на основі стандарту ISO 25010»: обсяг роботи становить 99 сторінок, робота містить 11 рис., 72 літературних джерел.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПРОГРАМНА СИСТЕМА, СТАНДАРТ ISO 25010, ЯКІСТЬ, ДОСТАТНІСТЬ, ІНФОРМАЦІЯ.

Об’єкт дослідження – процеси оцінювання достатності інформації щодо якості ПЗ.

Предмет дослідження – моделі, методи та засоби інформаційної технології оцінювання достатності інформації щодо якості ПЗ.

Мета дипломної роботи – дослідження теоретичних та практичних засад оцінювання моделі якості програмних систем на основі стандарту ISO 25010.

Задачі дослідження: дослідження моделей щодо якості програмного забезпечення; моделювання процесу оцінювання якості програмного забезпечення; оцінювання достатності інформації щодо якості за стандартом ISO 25010:2011 у специфікаціях вимог до програмного забезпечення.

У процесі роботи над дипломною роботою, було зроблено аналіз та дослідження моделей щодо якості програмного забезпечення.Проведено оцінювання достатності інформації щодо якості за стандартом ISO 25010:2011 у специфікаціях вимог до програмного забезпечення.

Результат виконання над дипломної роботи можна використовувати для виявлення недостатності інформації щодо якості у специфікації вимог на ранніх етапах життєвого циклу ПЗ інформаційної системи підприємства чи установи.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ – програмне забезпечення

ПС – програмна система

ЖЦ – життєвий цикл

КВІ – комплекс вхідної інформації

ФП – функційна придатність

Н – надійність

ЗВ – зручність використання

Е – ефективність

С – супроводжуваність

МП – можливість переносу

СМ – сумісність

LOC – кількість рядків коду

ШНМ – штучна нейронна мережа

OSS – однокроковий алгоритм навчання ШНМ на основі методу січної

CGB – алгоритм навчання ШНМ на основі методу спряженого градієнта з оберненим поширенням і рестартами в модифікації Пауела-Біеле

SCG – алгоритм навчання ШНМ на основі методу спряжених градієнтів в поєднанні з квазіньютоновими методами

АС – автоматизована система

ІС – інформаційна система

ВСТУП

Практично всі сфери людської діяльності сьогодні пов'язані з комп'ютерними та прикладними інформаційними системами, які є основою програмного забезпечення (ПЗ). Однією з головних вимог користувачів до сучасного програмного забезпечення є якість. Якість розуміється як особливість програмного забезпечення, що відображає ступінь відповідності вимогам. Вимоги можна трактувати досить широко, що породжує низку незалежних визначень поняття якості. Відповідно до визначень ISO [1, 4], якість - це ступінь відповідності наявних характеристик вимогам.

Відповідно до [2], якість - це повнота властивостей та характеристик товару, процесу чи послуги, які забезпечують можливість задоволення заявлених чи призначених потреб. Згідно [3], якість програмного забезпечення - це ступінь, до якого він має правильне поєднання властивостей.

Сьогодні існує низка моделей, які дозволяють розрахувати якість програмного забезпечення, але неоднозначність тлумачення цих характеристик ускладнює такі розрахунки. Більшість моделей засновані на використанні різних програмних показників.

У той же час актуальною є задача оцінки адекватності програмної інформації (наприклад, можливість отримання достовірної інформації про метрики для обчислення метричних значень), на основі якої визначаються характеристики якості та складності.

Проблемі якості ПЗ присвячено ряд робіт українських та іноземних учених: О. Димо, Л. Омельчука, В. Ліпаєва, К. Вігерса, Н. Фентона, А. Чена, А. Фатванто, Т. Рехмана.

Питання оцінювання та забезпечення якості ПЗ одержали свій розвиток в роботах українських та іноземних учених: В. Сеньківського, В. Харченка, Б. Конорєва, Д. Маєвського, В. Яковини, В. Міщенка, О. Поморової, К. Лавріщевої, О. Харченка, Г. Майєрса, С. Макконнелла, Р. Фатрелла, І. Соммервилла, О. Медче, С. Джонса.

Метою роботи є дослідження теоретичних та практичних засад оцінювання моделі якості програмних систем на основі стандарту ISO 25010.

Виходячи з мети дослідження було виконано нас**тупні завдання:**

- проаналізовано вплив інформації на якість програмного забезпечення;
- обгрунтувано емерджентні властивості у специфікації вимог до програмного забезпечення;
- досліджено стандартів з оцінювання якості програмного забезпечення;
- описано онтології як засіб інтеграції вимог щодо якості програмного забезпечення;
- проаналізовано використання онтологій для оцінювання якості у специфікаціях вимог до ПЗ;
- описано модель процесу оцінювання якості програмного забезпечення на основі стандарту ISO 25010;
- сформувано логічні висновки щодо якості за стандартом ISO 25010:2011 у специфікаціях вимог до програмного забезпечення;
- обгрунтувано методи та технологія оцінювання достатності інформації щодо якості у специфікаціях вимог до програмного забезпечення;
- оцінено достатність інформації специфікацій вимог для визначення якості програмного забезпечення за стандартом ISO 25010:2011.

Об’єкт дослідження – процеси оцінювання достатності інформації щодо якості ПЗ.

Предмет дослідження – моделі, методи та засоби інформаційної технології оцінювання достатності інформації щодо якості ПЗ.

Методи дослідження. Методологія досліджень ґрунтується на принципах загальної теорії систем, системного аналізу, методах аналізу та моделюванні процесів. Під час дослідження використовувались методи теорії моделювання, теорії множин, евристичних оцінок, порівняльного аналізу онтологій та підсистеми оцінювання та прогнозування.

1 ДОСЛІДЖЕННЯ МОДЕЛЕЙ ЩОДО ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз впливу інформації на якість програмного забезпечення

Майже всі сфери людської діяльності сьогодні пов'язані з комп'ютерними системами, в основі яких лежить програмне забезпечення (ПС). Розробка програмного забезпечення - це наукоємна діяльність, яка вимагає детального вивчення предмета та повного розуміння цілей розробленого продукту.

Ключовим фактором забезпечення ефективного застосування та однією з основних вимог користувачів та зацікавлених сторін до сучасного є досягнення високих значень показників його якості. Якість - головний фактор його успішної реалізації та експлуатації. Необхідність забезпечення якості ПЗ випливає з того, що помилки та невдачі загрожують катастрофами, які спричиняють людські страждання, екологічні катастрофи, значні втрати у часі та фінансові втрати.

Відповідно до ISO 25010 [1] якість програмного забезпечення визначається як здатність програмного забезпечення задовольняти заявлені та неявні потреби при використанні у визначених умовах.

Аналіз ISO 25010 дав можливість визначити фактори, що впливають на якість ПЗ:

1) відповідальність керівництва;

2) якість та достатність інформаційних регламентів (насамперед, специфікацій вимог);

3) ефективність проектних технологій, методологій та середовищ розробки (головна вимога, яка висувається до сучасних технологій проектування, методологій та середовищ розробки, - це їх відповідність стандартам та нормативним документам [3], пов'язаних з процесами

життєвого циклу та оцінкою. технологічної зрілості організацій, розробників);

4) склад та якість інструментів;

5) планування забезпечення якості та регулярності та ефективності контролю якості;

6) візуалізація результатів контролю якості;

7) стимулювати створення якісної продукції;

8) кваліфікація розробників;

9) маркетинг.

Актуальність розробки якісного програмного забезпечення в Росії підтверджується також економічними чинниками. За даними Національного інституту стандартів та технологій США [4], сума економічних втрат через несправне ПО в США сягає мільярдів доларів на рік, що становить близько 1% національного валового внутрішнього продукту.

Хаотичний період розвитку, коли велика увага приділявся коду програмного забезпечення, а не його якості, відійшов у минуле. Останніми роками галузь програмного забезпечення досягла такого рівня розвитку, в якому вимоги щодо забезпечення якості для договорів, передбачених пунктом перед обслуговування, на розробку програмних систем, оскільки якість є його найважливішою характеристикою з позицій зацікавлених сторін [4].Відповідно до стандартів [3], забезпечення якості - це сукупність планових та систематичних заходів, необхідних для того, щоб продукція чи процеси відповідали певним вимогам якості. Система забезпечення якості програмного забезпечення - це сукупність методів і засобів управління організацією та виконавчими підрозділами підприємства, що займаються проектуванням, розробкою та обслуговуванням архітектури програмних систем з метою надання їм властивостей, що відповідають конкретним потребам замовники та споживачі з мінімальною або прийнятною вартістю ресурсів [13].

У сфері забезпечення якості програмного забезпечення і досі існують проблеми, які були помітні більше 50 років тому - великі проекти відстають від графіку або перевищують кошторис витрат, розроблений продукт має необхідну функціональність, його продуктивність часто низька, якість роботи програмне забезпечення не влаштовує споживачів [6]. Аналітичне дослідження та пов'язане з ним програмне забезпечення, яке проводилось за останні кілька років, провідні зарубіжні аналітики підтверджують ці, не дуже обнадійливі результати [4]. Таким чином, за наявності ряду методів та інструментів, залучайте найкращі таланти до розробки технологій та стандартів забезпечення якості та якості програмного забезпечення та все ще залежить від знань та досвіду розробників [3].

Згідно статистичних оцінок, наведених у [16], понад 50% від загальної кількості програмних проектів підприємств провалюються, а робота над 42% програмних проектів припиняється до їх завершення.

Щільність помилок, поява яких очікується відповідно до різного розміру, представлена в таблиці 1.1 [1]. З таблиці 1.1 випливає, що сучасний обсяг у мільйонах рядків коду в принципі не може бути непогрішним. Завдання розробників - забезпечити бажану якість з урахуванням того, що завжди залишається деяка невідома кількість помилок та дефектів, а їх негативні наслідки слід запобігати або зменшувати до прийнятного рівня.

Таблиця 1.1 -Типова щільність помилок для ПЗ різного розміру

Розмір ПЗ	**Типова щільність помилок**
Менше 2 К	0-25 помилок на 1000 рядків коду
2К-16К	0-40 помилок на 1000 рядків
16К-64К	0,5-50 помилок на 1000 рядків
64К-512К	2-70 помилок на 1000 рядків
Більше 512К	4-100 помилок на 1000 рядків

Взагалі природа програмного забезпечення істотно змінилась 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 [15]. Оскільки програмне забезпечення стає дедалі складнішим та масштабнішим, розроблення саме програмної системи буде відігравати все більш важливу роль в діяльності розробника [9].

На рис.1.1 представлено матрицю розміру/складності ПЗ, дані якої доводять, що чим складнішим є ПЗ та чим більший воно має розмір, тим вищим є ризик невдачі такого проекту [26].

		COMPLEXITY				
		C1	C2	C3	C4	C5
SIZE	S1	100	250	400	550	700
	S2	175	325	475	625	775
	S3	250	400	550	700	850
	S4	325	475	625	775	625
	S5	400	550	700	850	1000

Рисунок 1.1 - Матриця розміру/складності ПЗ [26]

Однією з найбільш переконливих причин незадовільної якості великих дослідників програмних проектів Standish Group International є збільшення кількості компонентів (підсистем) та інтерфейсів між ними, а також неконтрольована складність програмної системи. Research International The Standish Group (звіти CHAOS) доводять, що статистика успіху малих та великих програмних проектів істотно відрізняється [25].

Серед причин можливих збоїв називають:

1) нечітке та неповне формування та формулювання вимог до програмного забезпечення;

2) відсутність участі користувачів у роботі над проектом;

3) брак ресурсів;

4) погане планування;

5) часті зміни вимог та технічних характеристик;

6) новизна використовуваної технології;

7) відсутність грамотного управління проектом;

8) недостатня підтримка вищого керівництва;

9) неправильне розуміння або недостатній аналіз проекту [18].

У таблиці 1.2 наведено відсоткове число помилок, що виникають на етапах формування та формулювання вимог, проектування та реалізація архітектури. З таблиці 1.2 видно, що помилки формування та формулювання вимог та дизайну архітектури становлять 25-55% від усіх помилок, і чим більше обсяг, тим більше помилок вводиться на ранніх етапах [9].

Основне місце виникнення помилок - етап формування та формулювання вимог (специфікації), друге місце за помилками "займає етап архітектурного проектування. У більшості випадків помилки вказують на проблеми із специфікацією та дизайном архітектури, тобто фактично наприкінці етапу проектування архітектура може виявити та виправити основну частину програмних помилок [58].

Таблиця 1.2 - Розподіл помилок, припущених на різних етапах життєвого циклу

Етап ЖЦ	**Обсяг ПЗ**				
	2К	**8К**	**32К**	**128К**	**512К**
Формування та формулювання вимог	До 10%	До 15%	До 20%	До 22%	До 23%
Проектування архітектури	До 15%	До 19%	До 25%	До 28%	До 32%
Конструювання	До 75%	До 66%	До 55%	До 50%	До 45%

Є кілька причин, які роблять етап формування та формулювання вимог головним "постачальником" помилок:

1) відсутність специфікації;

2) неповні або суперечливі специфікації інформації;

3) часті зміни специфікацій;

4) щільність плану розробки, складеного на етапі проектування архітектури.

В роботі [27] підтверджується той факт, що причини майже всіх інцидентів та аварій, пов'язаних із програмним забезпеченням, полягають у конкретизації програмних вимог. Переважна більшість нещасних випадків, спричинених ВУ, внаслідок неправильних вимог, а не помилок кодування.

В [19] описані результати експерименту, проведеного для підтвердження або відхилення гіпотези про те, що збої та помилки ПЗ, написані різними розробниками в одній специфікації, статистично незалежні. Під час експерименту кілька незалежних груп розробників написали своє програмне забезпечення з однієї специфікації. Результатом цього експерименту було встановлено, що версії, написані різними розробниками з однаковими вимогами, містять ряд поширених помилок, пов’язаних з помилками або неточностями вимог (специфікацій).

Дефекти вимог бажано виявити та усунути до того, як вони почнуть впливати на пізніші стадії розвитку. Ранні етапи життєвого циклу впливають на якість більше, ніж пізні, тому час, витрачений на контроль якості на ранніх стадіях, дає можливість зменшити дефекти, скоротити час розробки та зменшити витрати на пізніх стадіях [7].

Зі збільшенням інтервалу між введенням та виявленням дефекту вартість ремонту значно збільшується. Чим довше помилка зберігається в ланцюжку розробки програмного забезпечення, тим більше вона проникає в інші частини, тим більше шкоди робиться на наступних етапах, і тим більше грошей доведеться витратити на її усунення [9].

Чим раніше буде виявлено дефект (помилка, порушення, несправність, несправність, тим дешевше буде коштувати його виправлення. Як показано на фіг.1.2, вартість виправлення дефектів після випуску виробу майже в 100

разів перевищує витрати на виправлення, якщо були виявлені недоліки в процесі формування та формулювання вимог [28].

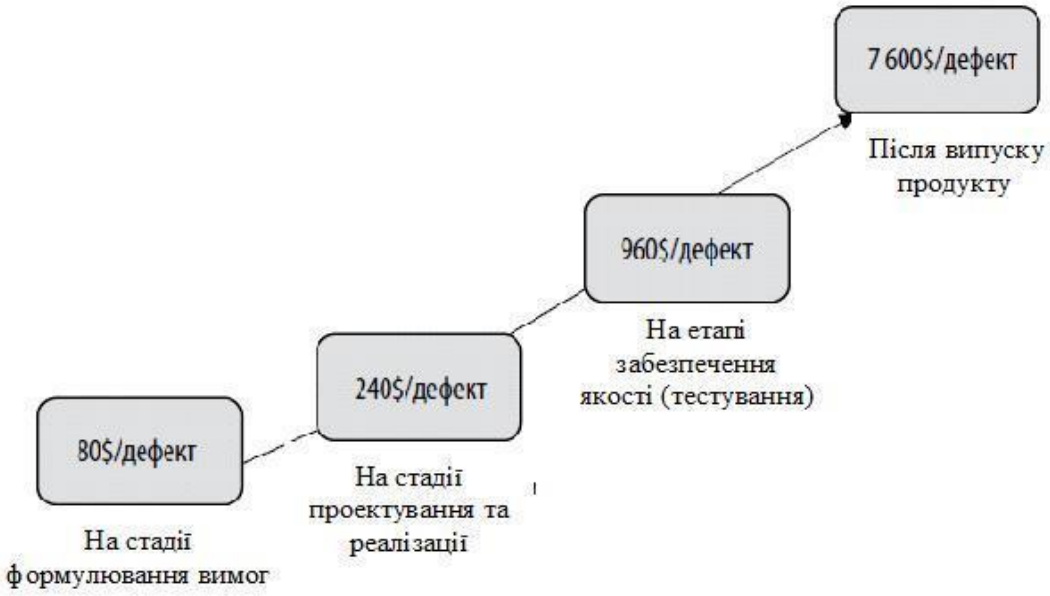

Рисунок 1.2 - Зростання вартості виправлення дефектів в процесі розроблення [28]

Десятки компаній виявили, що політика раннього виправлення дефекту може зменшити фінансові та часові витрати на розробку програмного забезпечення, що є вагомим аргументом на користь якнайшвидшого виявлення та усунення дефектів. Дослідження 50 проектів, розроблених лабораторією NASA, показали, що підвищена увага до раннього контролю якості може значно знизити рівень помилок, але не збільшує загальних витрат на розробку [18].

Вартість диференціалу якості - це вартість запобігання виникненню дефектів, кошторис витрат, вартість внутрішніх і зовнішніх відмов. Якість ПС зростає, організовуючи ітераційний процес постійного вдосконалення [14].

У процесі формування та формулювання вимог може статися втрата інформації при неповному та різному розумінні потреб та контексту

інформації - особливо така значна втрата для програмних проектів, які розробляються при перетині предметних полів (наприклад, медицини), коли необхідно враховувати, як розробляються стандарти та стандарти предметної області, для якої ви розробляєте ЗА. Цю кількість стандартів важко здійснити, а ще важче оцінити, наскільки рекомендації цих стандартів.

З визначення якості як задоволеності користувачів або ступеня відповідності потребам замовника випливає, що якщо цілі проекту, встановлені на ранній стадії їх життєвого циклу, не відповідають потребам користувачів, то визнати якість неможливо, навіть якщо вона була розроблена із застосуванням сучасних технологій та була залучена до найбільш кваліфікованих розробників. Тому якість та успішна реалізація програмного проекту значною мірою залежить від конкретизації вимог та адекватності наявної інформації щодо наявності всіх інформаційних елементів (даних), необхідних для визначення якості. Це експериментальне підтвердження безпосередньо призводить до необхідності поглиблення аналізу специфікацій [25].

Адекватність інформації щодо якості в специфікації вимог ДО наявності у специфікації всіх інформаційних елементів, необхідних для визначення якості [34].

Оцінка інформаційної специфікації вимог шляхом надання вибору програмного проекту з точки зору його передбачуваної якості підвищує ефективність управління проектами завдяки надійності рішень, скорочує їх розробку та прийняття, зменшує витрати на збір та обробку інформації. Відсутність інформаційних специфікацій вимог знижує продуктивність та надійність оцінки якості.Програмні вимоги визначають необхідні характеристики якості та впливають на методи кількісного оцінювання та формулюються для оцінки цих критеріїв прийняття. Тому вся необхідна інформація про якість вже включена в специфікацію вимог до програмного забезпечення, тобто вже на базі специфікації вимог до програмного забезпечення можна оцінити достатність інформації для подальшого

визначення якості програмного забезпечення. Якщо деякі інформаційні елементи якості відсутні, специфікації вимог є недостатніми для визначення якості програмного забезпечення, і розробники повинні внести необхідні доповнення в специфікацію. Але сьогодні оцінка інформації для визначення якості програмного забезпечення відбувається лише за готовим програмним кодом [46].

Таким чином, як показав аналіз впливу вимог специфікації інформації на якість програмного забезпечення, фактори якості сучасних програмних систем менш залежать від написання коду, але значно залежать від формування та формулювання вимог та архітектури дизайну. Дефекти, введені під час етапів формулювання вимог та етапів проектування, повинні бути визначені та виправлені до того, як вони вплинуть на результати подальшого життєвого циклу. Якість та успішність реалізації програмного проекту значною мірою залежать від конкретизації вимог до програмного забезпечення.

1.2 Емерджентні властивості у специфікації вимог до програмного забезпечення

У розробці програмного забезпечення останнім часом перемістився від виробництва автономних програмних продуктів до виробництва як системи систем з великої кількості компонентів з інтерфейсами між ними. Результатом виробництва як системи систем з великої кількості компонентів з інтерфейсами між ними є той факт, що інциденти та аварії, пов'язані з FOR сьогодні, включають новий тип аварій, що викликається взаємодією компонентів: кожен компонент не містить помилок (відповідає потребам), але неправильна взаємодія між компонентами призводить до проблем. Зрив взаємодії між компонентами спричиняє великі аварії через зростаючу складність програмних систем, що призводить до неможливості передбачити всі можливі наслідки взаємодії між компонентами [27].

Приклади катастроф взаємодії компонентів програмних систем наведені в таблиці 1.3 [34]. Просте підвищення якості та надійності окремих компонентів програмних систем не може запобігти подібним аваріям, оскільки вони є причиною виходу з ладу окремих компонентів.

Таблиця 1.3 - Приклади аварій взаємодії компонентів програмних систем

Подія	**Причина**	**Наслідки**
1	**2**	**2**
Вибух ракети-носія Ariane 5 у1996 році [166]	Протиріччя вимог щодо необхідності забезпечення надійності та максимально припустимого навантаження	Вартість обладнання та розроблення - 7,5 млрд. доларів, "упущена вигода" – 2 млрд. доларів
Аварія станції Mars Climate Orbiter та зникнення зв'язку із Mars Polar Lander у 1999 році [167]	Помилка в проекті через використання різних одиниць вимірювання	327,6 млн. доларів - апарати та 91,7 млн. доларів - запуски
Порушення польоту ракетоносія Titan IV у 1999 році [169]	Помилка в константі ПЗ системи керування двигуном	Втрата супутника Milstar
"Смертельні" сеанси радіаційної терапії із застосуванням Therac-25 у 1985-1987 рр. [170]	Неповнота специфікації; помилки у розробленні та постановці проекту; некоректні процедури оцінки та прогнозування ризиків	6 пацієнтів одержали смертельну дозу опромінення
Збій у мобільній системі протиракетної оборони Patriot у 1991 році [170]	Помилка заокруглення, яка не була критичною на рівні одного компоненту, але посилилась при його інтеграції у систему	Не перехоплено іракську ракету Scud, в результаті чого загинули 28 солдат та близько 100 чоловік одержали поранення
Аварія літака «Суперджет 100» [170]	Несумісність та неузгодженість ПЗ різних компаній	Загибель 48 чоловік
Падіння у Тихий океан трьох супутників у 2010 році [172]	Помилка у інтеграції програмної системи	Неможливість завершення навігаційної системи

Висока якість та надійність комплектуючих не запобігає подібним аваріям. Ці проблеми ускладнюються тим, що в даний час широко застосовуються такі методи інженерного аналізу надійності та безпеки, розроблені для виявлення несправностей компонентів, і їх не можна ефективно використовувати для запобігання взаємодії компонентів при аваріях. Проблема, з якою стикаються інженери, полягає в тому, що вони не мають інших, більш підходящих інструментів для визначення властивостей системи та прогнозування нещасних випадків взаємодії [27].

Якість може бути низькою, оскільки недостатньо уваги приділяється інформаційній тематиці на різних етапах життєвого циклу, її достатності, надійності, вишуканості. Частина інформації аналізується занадто прискіпливою, а частина - не враховується. Часто відкидається інформація предметної області з невеликою ймовірністю, а іноді ймовірність її насправді не оцінюється. Нова інформація може надходити на різних етапах життєвого циклу - як на стадії формування та формулювання вимог та архітектурного проектування, так і на етапах впровадження та експлуатації, однак часто нехтують. Таке нехтування інформацією про предметну область на всіх етапах життєвого циклу є одним з найважливіших факторів розвитку програмного забезпечення [17].

У процесі роботи над програмним проектом важливо оцінити частку інформаційної невизначеності проекту. Причиною невизначеності проекту є низький рівень документальності знань, особливо на системному рівні.

Той факт, що домен інформації частково не враховується на різних етапах життєвого циклу програмного забезпечення, свідчить про те, що розмір розриву знань не є постійним для програмного проекту - він може збільшуватися та зменшуватися в життєвому циклі, коли з’являється нова інформація, яка потребує враховувати. Немає гарантії того, що інформація про домени на різних етапах життєвого циклу повністю враховується в моделях та стандартах якості програмного забезпечення, про що свідчить

огляд відомих моделей та стандартів якості програмного забезпечення в [179].

Тому ми представляємо всі наявні знання та інформацію про програмну систему у вигляді діаграми, в якій є сектор, що показує кількість недостатньої (невідомої) інформації (розрив знань) - рис. 1.3.

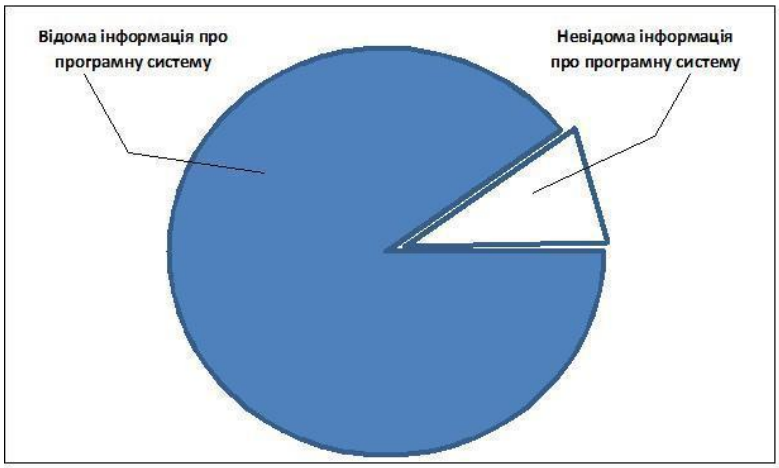

Рисунок 1.3 - Поле знань про програмну систему із сектором невідомої інформації

Сектор є незареєстрованою інформаційною сферою інформації (включаючи інформацію, відсутню у специфікації вимог). Розмір сектору з невідомою інформацією не визначений, оскільки незрозуміло, що і скільки інформації залишається невідомим. Сектор з невідомою інформацією повинен бути вузьким завдяки більш повному запису області інформації, починаючи з ранніх стадій життєвого циклу. Чим менший розмір сектору невідома інформація, тим краще буде програмне забезпечення та безпечніше воно працюватиме. Отже, власне підхід полягає у зменшенні частки невідомої інформації про програмну систему.

Однак важко визначити інформацію, яка з’явиться в ході взаємодії «розробляються підсистеми - інтерфейси - дані - зовнішні впливи», а також майбутні характеристики, за допомогою яких ця інформація може з’являтися (нові властивості, що входять до складу розрив знань) [10].

Відомі методи оцінки якості системи ПЗ не враховують аспекту програмного забезпечення, а саме, недостатньо уваги приділяється появі системи. Емерджент - це нова властивість, яка є результатом поєднання елементів у системі. Поява системи породжується її нелінійністю, новими елементами системи або новими відносинами. Причинами властивостей Емерджентном є: різкий стрибок властивостей (присутній, але не видно раніше); біфуркація (втрата стійкості, період нестабільності, точка, в якій трапляється катастрофа) підсистема; рекомбінація зв'язків. Такі різкі переходи та якісні зміни досліджують теорію катастроф, яка вивчає та прогнозує нестабільність різних систем, а також корисна при аналізі великої кількості інформації. Чим вище інтенсивність взаємодії між елементами системи, тим більша відмінність властивостей системи від властивостей її елементів, тим вище системний ефект. Чим вище рівень системи, тим більше переваг у досягненні цілей - здатність системи до вибору прямо пропорційна ступеню сформованості [63].

У ряді визначень виникаючі властивості розглядаються як властивості, які неможливо локалізувати в одному компоненті, і є результатом взаємодії компонентів. Деякі визначення вказують на труднощі прогнозування властивостей Emergenti або розуміють під цим терміном різницю між наміченою та реалізованою конструкцією програмної системи, не розкриваючи природу аварійних властивостей та їх сутність.Отже, властивостями емергентизму ми припускаємо наступні властивості програмної системи, які з'являються випадковим чином (випали з фокусу) і проявляються в процесі функціонування через взаємодію підсистем через інтерфейси та за наявності певних даних та зовнішніх впливів. поведінка розробниками програмної системи вважається непередбачуваною поведінкою, яка не властива окремій підсистемі окремо, але відбувається лише в процесі взаємодії підсистеми [54].

Екстрені властивості можуть як позитивно, так і негативно впливати на якість програмних систем.

В [15] представлено результати детального аналізу сорока дев'яти звітів про комерційні аерокосмічні проблеми, розбиті на наступні категорії: порушення властивостей, нові властивості (проблеми) та проблеми відповідності. Аварійні проблеми описані авторами [15], коли фактична поведінка програмної системи не відповідала намірам проектувальника, хоча жодна вимога не була порушена. Проблеми, спричинені виникаючими властивостями, становлять 16,3% від загальної кількості проблем.

Інженери не мають належних інструментів для ідентифікації та прогнозування властивостей системного програмного забезпечення (включаючи виникаючі властивості) та збоїв взаємодії. Екстрені властивості не враховуються в сучасних підходах до розробки та оцінки програмного забезпечення, тому не розроблено ефективних механізмів виявлення проблем інтеграції, які б пояснювали великий відсоток неякісних (несправних та проблемних) програмних проектів [16].

Причини ігнорування виникаючих властивостей не технологічні, а методологічні. Екстрені властивості бажано виявити та усунути на ранніх стадіях життєвого циклу програмного забезпечення, що було б способом поліпшити якість програмних систем.

1.3. Дослідження стандартів з оцінювання якості програмного забезпечення

При розробці програмного забезпечення програмні організації повинні керуватися стандартами як в процесі розробки, так і в процесах оцінки та забезпечення якості. Самі стандарти є зв'язком між розробниками та користувачами і повинні включати попередній досвід розробника. Але довгострокова розробка стандартів, їхнє пізнє оновлення та вдосконалення призводять до уповільнення розвитку технологій, відставання від практичних потреб та перешкоджання впровадженню інновацій.

Для оцінки розробки програмного забезпечення та стандартів якості можна виділити наступні основні критерії:

1) довільність стандартного тлумачення для оцінки відповідності;

2) важливість аналізу вимог до продукції;

3) можливість багаторазового використання процесу;

4) важливість управління;

5) можливість змін та розвитку технологій;

6) узгодженість теорії систем та інженерії систем;

7) дотримання теорії безпеки;

8) відповідність людському фактору;

9) розділення рівнів абстракції в еталоні;

10) цілісність та критичність рівнів абстракції;

11) встановлення цілей та стандартів для зацікавлених сторін; властивості (функції) компонентів і систем;

12) підтримка незалежного оцінювання та сертифікації;

13) значення етапу операції [26].

Існує десятки різних підходів до забезпечення якості програмного забезпечення, і кожен має свої переваги та недоліки.Загальне керування якістю (TQM – Total Quality Management) – це філософія софтверної організації, яка заснована на прагненні до якості ПЗ та практиці керування, що призводить до загальної якості ПЗ. Це підхід до керування будь-яким процесом життєвого циклу ПЗ, націлений на якість, заснований на участі всіх учасників процесу та спрямований на досягненні довготривалого успіху завдяки задоволенню потреб користувача і вигоди для суспільства в цілому [22].

Взагалі, виділяється 7 факторів загальної якості: 1) орієнтація на споживача; 2) орієнтація на процес та його результати; 3) керування участю в роботі та відповідальністю; 4) неперервне вдосконалення; 5) проблеми, які залежать від працівників, повинні складати не більше 20%; 6) проведення вимірювань; 7) командна організація робіт з покращення якості. Ці 7

факторів є абсолютно правильними і для програмного забезпечення як продукту, і для розроблення ПЗ як процесу[22].

Рушійною силою програмних проектів є бажання створити ПЗ, яке має певну цінність, тобто є значущим для розв'язання певних задач або досягнення цілей [45]. Замовник має своє уявлення про максимальні вартісні вкладення, також певні очікування по відношенню до якості ПЗ. Більшість рішень щодо якості приймаються у процесі роботи з вимогами, однак ці питання варто підіймати протягом всього життєвого циклу ПЗ. Не існує стандартизованих правил, за якими слід приймати рішення. Зазвичай існує ряд альтернатив досягнення різного рівня якості з різною вартістю, тобто має місце суб'єктивізм та значний вплив людського фактору [21].

Деякі з сьогоднішніх стандартів не повністю відповідають вимогам до сучасного ПЗ, а також всім потребам користувача. Створено велику кількість стандартів, які по-різному стандартизують та регламентують одні й ті ж процеси (як правило, рутинні масові процеси), внаслідок чого виникає неповне покриття об'єктів стандартизації, несумісність нормативних документів різних організацій, лобіювання інтересів окремих софтверних організацій та пристосування стандартів розробниками до своїх потреб.

Перераховані проблеми в галузі оцінювання та забезпечення якості ПЗ змушують експертів постійно розробляти рішення в цій галузі, що призвело до створення цілої низки стандартів та методологій: Capability Maturity Model (CMM), ISO/IEC 33002 (SPICE), ISO 12207). Також створено серію стандартів ISO25000 – нове покоління нормативних документів, пов'язаних із нормуванням та оцінюванням якості ПЗ [64].

Для програмного забезпечення розроблено ряд спеціалізованих моделей якості. Модель якості ПЗ – це набір характеристик і відношення між ними, що забезпечують основу для визначення вимог та оцінки якості [12]. Структура моделі якості описується ієрархією, елементами якої є множини характеристик, підхарактеристик, атрибутів та зв'язків між ними.

Згідно зі стандартом [13], характеристика – це набір властивостей ПЗ, за допомогою яких описується та оцінюється його якість. Характеристики якості ПЗ можуть бути уточнені на основі комплексних показників (підхарактеристик), які ґрунтуються на властивості задовольняти заявлені або виникаючі потреби. Підхарактеристика якості ПЗ виражається середнім зваженим арифметичним показником з урахуванням значень атрибутів, що оцінюють цю підхарактеристику, та коефіцієнтів їхньої вагомості. В свою чергу, атрибут – це фізична або абстрактна властивість ПЗ, яка може бути виміряною.

Характеристики, які увійшли в модель, є основою при формуванні вимог до ПЗ. Кожна модель містить різну кількість рівнів ієрархії та різну загальну кількість характеристик якості. На сьогодні створено різні моделі якості з унікальним набором характеристик, підхарактеристик та атрибутів. Ці моделі можуть бути корисні для обговорення, планування та оцінювання якості ПЗ.

Класифікують моделі за характеристиками користувача. Вони виділяють три категорії моделей, які відповідають:

1) рівню загального використання або конкретної галузі;

2) організаційному рівню, що фокусується на задоволенні інтересів конкретної організації;

3) проектному рівню, який застосовується до конкретного проекту, щоб гарантувати якість [25].

З 2000 року розроблення ПЗ почало залежати від створених або виготовлених компонентів, що призвело до нових проблем в галузі оцінки якості.

Наслідком став розподіл моделей на базові, що були розроблені до 2000 року, і (адаптовані) моделі – рис.1.4 [64]. Аспекти комунікації були і залишаються важливим фактором впливу на якість ПЗ.

Базові моделі (Mc Call, Boehm, FURPS, Dromey, ISO 9126, ISO 25010) мають ієрархічну структуру; вони можуть бути адаптовані до програмного продукту будь-якого типу та орієнтовані на оцінки і вдосконалення.

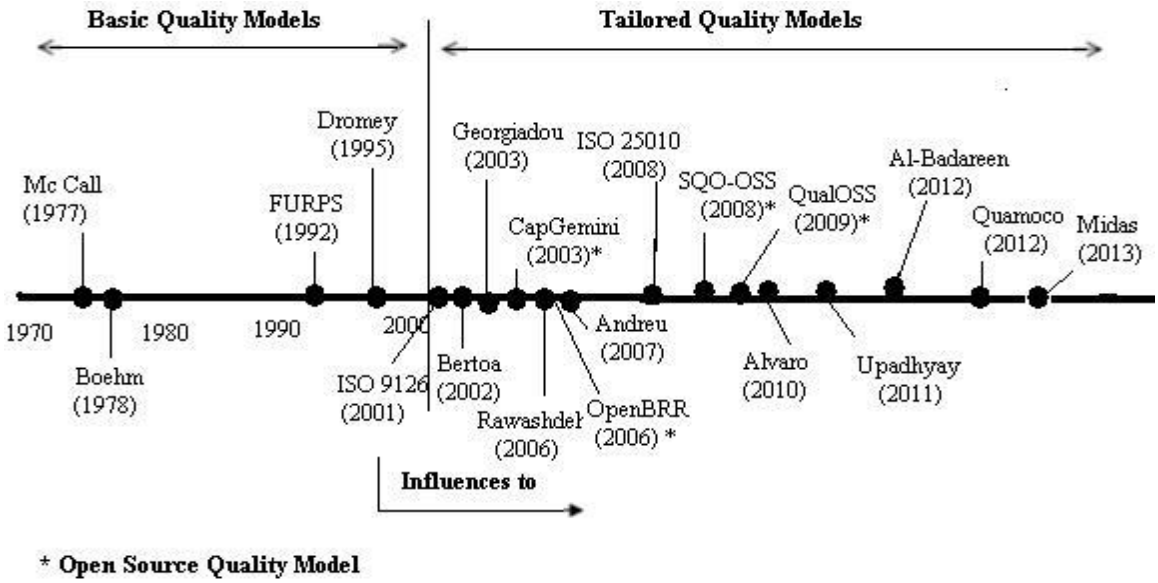

Рисунок 1.4 -Моделі якості програмного забезпечення

Основним внеском моделі Мак-Кола (McCall) було встановлення зв'язків між якісними характеристиками і показниками. Основним її недоліком є низька точність вимірювання якості та неврахування функціональності. Боем (Boehm) встановлює великомасштабні характеристики, які надають покращення в порівнянні з моделлю Мак-Кола, оскільки додають фактори на різних рівнях.

Модель Дромі (Dromey) базується на точці зору якості продукції. Вона стверджує, що для високої якості продукту всі елементи, які його складають, повинні бути високоякісними. Оцінки якості для кожного продукту різні, і використовується більш динамічна оцінка.

Модель FURPS класифікує характеристики на функціональні та нефункціональні. Модель 9126 має дві основні частини, що складаються з: 1) атрибутів внутрішньої (системні властивості, які можуть бути оцінені без

виконання) та зовнішньої (системні властивості, які можуть бути визначені спостереженням під час виконання) якості; 2) якості в атрибутах використання. *Модель ISO 25010* виділяє 8 ключових характеристик. Однією з її головних цілей є спрямування розроблення ПЗ на специфікацію і оцінювання вимог [19].

Основною характеристикою адаптованих моделей якості (Bertoa, GEQUAMO, Alvaro, Rawashdeh та інших) є те, що вони є специфічними для конкретної галузі, і важливість функцій може бути змінною по відношенню до загальної моделі. Вони виникають з організаційної необхідності та індустрії ПЗ, здатні робити спеціалізовану оцінку за окремими компонентами. Відповідно до таких моделей, успіх програмного продукту в значній мірі залежить від якості компонентів. Модель Бертоа (Bertoa) базується на моделі ISO 9126. Вона визначає набір атрибутів якості для ефективної оцінки ПЗ. Розрізняє ті функції, які мають сенс для окремих компонентів.

Модель GEQUAMO (загальна, багатошарова і налагоджувальна модель) автора Georgiadou складається з підшарів функцій і характеристик і призначена для інкапсуляції різних потреб користувачів в динамічній та гнучкій формі. У цій формі користувач (кінцевий користувач, розробник, і менеджер) може побудувати свою власну модель, яка відображатиме акцент (вагу) для кожного атрибуту та/або вимоги. Модель Альваро (Alvaro) вважається основою для сертифікації програмних компонентів – в порядку встановлення елементів якісних компонентів. Модель Равашдеха (Rawashdeh) має якості основної мети потреби різних типів користувачів. Адаптовані моделі можуть бути орієнтованими на продукт (GECUAMO), на певні галузі (Bertoa) або адаптованими з точки зору користувача (Rawashdeh) [180].

Моделі для оцінки якості безкоштовних програмних продуктів адаптують моделі, додаючи деякі конкретні аспекти вільно поширюваного ПЗ. Варто відзначити, що ідеальна модель, яка охоплює всі аспекти якості

вільно поширюваного програмного продукту, поки не визначена. CapGemini Open Source Maturity Model заснована на зрілості продукту і встановлюється відповідно до показників зрілості. OpenBRR Model була розроблена під впливом CapGemini і ISO 9126. У цьому контексті визначає категорії, важливі для оцінки вільно поширюваного ПЗ. Вона має сім категорій, тим самим прискорює процес оцінки, забезпечує кращий вибір з невеликого набору. SQO-OSS модель – це ієрархічна модель, яка оцінює вихідний (сирцевий) код і спільні процеси, що забезпечує автоматичний розрахунок показників. Він відрізняється від інших такими аспектами: зосередженість на автоматизації, є ядром системи постійного контролю якості та забезпечує автоматичний збір метрик; оцінює функціональність; фокусується на вихідному коді, враховуючи лише соціальні фактори, які можна виміряти автоматично. Модель QualOSS стверджує, що якість сильно залежить від контексту, в якому вона використовується в цілях, що переслідуються компанією або особою щодо цього товару [17].

Вивчення моделей якості ПЗ показало, що вони погано враховують системний аспект сучасного програмного забезпечення (особливо з урахуванням моделей якості), що недостатня увага приділяється властивостям емергентизму, які є невід'ємною характеристикою системи. Якість моделі (особливо адаптована), спрямована на визначення та поліпшення якості окремих компонентів, але систематичний аналіз доводить, що функції системи - це не просто сума функцій компонентів системи, а наявність аварійних ситуацій (інтегративні) властивості вважаються однією з найважливіших особливостей системи [74].

Деякі енергетичні властивості програмних систем виявляються нефункціональними вимогами, оскільки нефункціональні вимоги містять певну ступінь внутрішньої невизначеності (ступінь суб'єктивності), сформульовані на системному рівні і частково відображають виникаючі властивості програмних систем. Якість ПЗ повинно забезпечувати можливість прогнозувати зростаючу кількість нефункціональних вимог, що

зменшить негативний вплив властивостей Emergenti. Індивідуальні моделі якості не враховують усіх нефункціональних вимог. Проаналізуйте відомі моделі для розпізнавання та вимірювання нефункціональних вимог [18].

Відповідно до цих правил, ми отримуємо кількість нефункціональних вимог, які можна вирішити та оцінити за допомогою моделі таблиці програмного забезпечення 1.4.3 таблиці 1.5 видно, що жодна модель не враховує такої важливої нефункціональної вимоги, як гарантоздатність (dependability). Маємо три моделі якості ПЗ, які враховують та оцінюють найбільшу кількість нефункціональних вимог. Це моделі: ISO 9126, ISO 25010, модель Равашдеха (Rawashdeh). Модель Rawashdeh є адаптованою моделлю якості ПЗ, тобто спрямована на визначення та вдосконалення якості окремих компонентів і слабко враховує системний аспект ПЗ.

Модель ISO 9126 є дещо застарілою, модель ISO 25010 (2011 року) є її оновленою версією, тому *для подальшої роботи використовуватимемо базову модель якості ISO 25010 (SQuaRE),* враховуючи ще й той факт, що однією з її головних цілей є спрямування розроблення ПЗ на специфікацію і оцінювання вимог. Але ця модель враховує та оцінює лише 6 нефункціональних вимог з 9 можливих.

Таблиця 1.4 - Підрахунок кількості нефункціональних вимог, врахованих та оцінюваних моделями якості програмного забезпечення [18]

Нефункціональні вимоги	**Моделі якості ПЗ (позначені номерами згідно зтаблицею А.1 додатку А)**													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Якість (Quality)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Складність(Complexity)										+				
Продуктивність(Performance)				+		+					+			
Супроводжуваність, ремонтопридатність (Maintainability)	+		+	+	+	+	+		+	+			+	+
Безпека (Safety)					+									
Надійність (Reliability)	+	+	+	+	+	+	+		+	+	+		+	

Захищеність (Security)				+	+	+	+		+	+	+	+	+	+
Можливість взаємодії	+				+	+	+		+	+				
(Interoperability)														
Гарантоздатність														
(Dependability)														
Загальна кількість	4	2	3	5	6	6	5	1	5	6	4	2	4	3

Загальний підхід до моделювання якості ПЗ полягає в тому, щоб спочатку ідентифікувати невеликий набір характеристик якості найвищого рівня абстракції, потім в напрямку "згори-донизу" розбити ці характеристики на підхарактеристики та набори підлеглих атрибутів.

Стандарт ISO 25010 [12], який описує *модель якості ISO 25010 (SQuaRE),* типовим прикладом такого загального підходу та найбільш використовуваною моделлю для оцінки якості ПЗ. Модель якості, створена в межах стандарту [132], визначається 8-ма загальними характеристиками якості продукту: функційна придатність (Functional Suitability), ефективність (Performance Efficiency), сумісність (Compatibility), зручність використання (Usability), надійність (Reliability), захищеність (Security), супроводжуваність або ремонтопридатність (Maintainability), можливість переносу (Portability). Кожна характеристика якості ПЗ є функцією від декількох підхарактеристик якості (всього 31 підхарактеристика, згідно зі стандартом [12]). Нижній рівень ієрархії представляють атрибути (міри) якості ПЗ, які підлягають точному опису та вимірюванню. Атрибути якості ПЗ визначені та описані у стандарті ISO 25023.

Викладена концепція оцінювання якості ПЗ представлена на рис. 1.5 [64].

Рисунок 1.5 - Сучасна концепція оцінювання якості програмного забезпечення в рамках моделі ISO 25010 (SQuaRE)

Сьогодні оцінка характеристик якості IN відповідно до ISO 25010 [13] полягає в наступному. Спочатку оцінюються атрибути якості ПЗ - калібрована оцінка шкали залежно від можливих ступенів відповідності накладеного обмеження атрибуту. На основі багатьох ознак оцінюється якість характеристики і на основі множинної характеристики оцінюють характеристики якості програмного забезпечення. Але оцінка атрибутів якості ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ суб'єктивна, оскільки не існує єдиних стандартів їх оцінки. Інтерпретація значень атрибутів також здійснюється суб'єктивно, оскільки немає стандартизованих значень атрибутів «еталон». Позначення в рейтинговій шкалі, знову ж таки, суб'єктивні, оскільки це залежить від можливих ступенів відповідності атрибуту, накладеного обмеженням, та ступеня відповідності не стандартизовані і визначаються програмною організацією [18].

Тому оцінка якості ПЗ як функцій основних восьми характеристик є суб'єктивною, оскільки інтерпретатор програмної організації отримав значення атрибутів як максимальні, щоб поступово оцінювати шкалу оцінки кожної характеристики на основі їх інтерпретації значень атрибутів та можливі ступені відповідності атрибутів обмеження, результат, який отримує максимальне значення кожної характеристики, а отже, і максимальне

значення якості ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ [18]. Фактично існує лише формальна зустріч якості через неповне покриття стандартів об'єктів стандартизації, а також завдяки вибору розробника сприятливих стандартів та адаптації стандартів до їх потреб.

Основна ідея моделі SQuaRE [13] полягає в тому, що оцінку якості та її характеристик та характеристик слід проводити всебічно з урахуванням усіх цих характеристик, характеристик та ознак відповідно. Але не існує комплексної методології, яка надасть можливість оцінити не тільки вплив кожної характеристики на якість (це питання присвячено серії робіт), а й надасть можливість оцінити всі атрибути, необхідні для визначення всіх характеристик і якісні характеристики (достатня інформація), а також для оцінки характеристик перешкод.

Сьогодні оцінка характеристик якості ПС відповідно до ISO 25010 [13] полягає в наступному. Спочатку оцінюються атрибути якості ПС - калібрована оцінка шкали залежно від можливих ступенів відповідності накладеного обмеження атрибуту. На основі багатьох ознак оцінюється якість характеристики і на основі множинної характеристики оцінюють характеристики якості програмного забезпечення. Але оцінка атрибутів якості ПС суб'єктивна, оскільки не існує єдиних стандартів їх оцінки. Інтерпретація значень атрибутів також здійснюється суб'єктивно, оскільки немає стандартизованих значень атрибутів «еталон». Позначення в рейтинговій шкалі, знову ж таки, суб'єктивні, оскільки це залежить від можливих ступенів відповідності атрибуту, накладеного обмеженням, та ступеня відповідності не стандартизовані і визначаються програмною організацією [18].

Тому оцінка якості ПС як функцій основних восьми характеристик є суб'єктивною, оскільки інтерпретатор програмної організації отримав значення атрибутів як максимальні, щоб поступово оцінювати шкалу оцінки кожної характеристики на основі їх інтерпретації значень атрибутів та можливі ступені відповідності атрибутів обмеження, результат, який отримує

максимальне значення кожної характеристики, а отже, і максимальне значення якості ПС [18]. Фактично існує лише формальна зустріч якості через неповне покриття стандартів об'єктів стандартизації, а також завдяки вибору розробника сприятливих стандартів та адаптації стандартів до їх потреб.

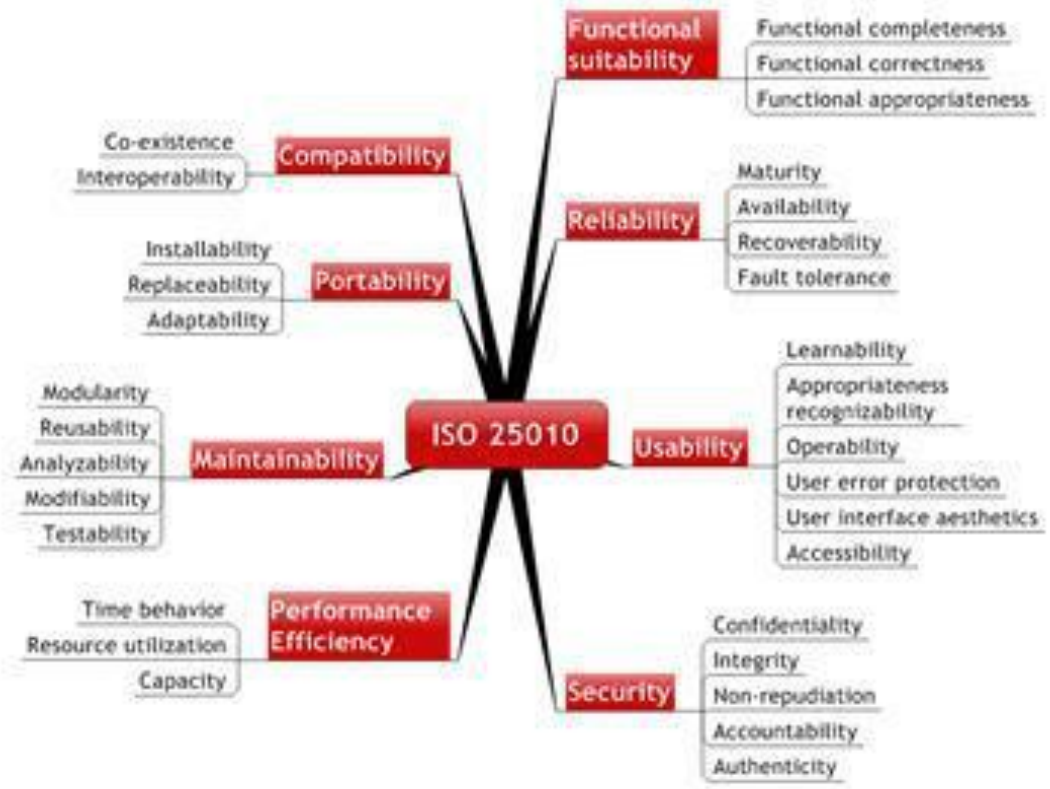

Рисунок 1.6 - Модель якості ПЗ за стандартом ISO/IEC 25010:2011 [12]

Основна ідея моделі SQuaRE [13] полягає в тому, що оцінку якості та її характеристик та характеристик слід проводити всебічно з урахуванням усіх цих характеристик, характеристик та ознак відповідно. Але не існує комплексної методології, яка надасть можливість оцінити не тільки вплив кожної характеристики на якість (це питання присвячено серії робіт), а й надасть можливість оцінити всі атрибути, необхідні для визначення всіх характеристик і якісні характеристики (достатня інформація), а також для оцінки характеристик перешкод.

Якість моделі SQuaRE показано на рис.1.6 [12]. У таблиці А. 2 у додатку А представлено зміст характеристик та характеристичну якість відповідно до стандарту [132].

Аналіз робіт та ISO 25010, ISO 25023 дав можливість зробити висновок про наявність атрибутів, які залежать від декількох характеристик та характеристик якості ПС, тобто існує кореляційна характеристика та характеристики для певних ознак (наприклад, відповідно до стандарту [19] якість характеристики залежить від 203 атрибутів, але лише від 138 різних ознак).

Аналіз робіт та ISO 25010, ISO 25023 дав можливість зробити висновок про наявність атрибутів, які залежать від декількох характеристик та характеристик якості ПЗ, тобто існує кореляційна характеристика та характеристики для певних ознак (наприклад, відповідно до стандарту [139] якість характеристики залежить від 203 атрибутів, але лише від 138 різних ознак).

При оцінюванні якості ПЗ для усунення проблеми суб’єктивного оцінювання та формального задоволення якості варто враховувати як можливість обчислення та ступінь вираженості характеристик, підхарактеристик та тарибутів якості, так і їх значущість. Якщо атрибути, які входять до складу декількох підхарактеристик та (або) характеристик якості, визначені неточно або відсутні (має місце недостатність інформації), то одночасне використання цих атрибутів суттєво вплине на достовірність отриманих оцінок якості ПЗ. За такої ситуації важливою є умова пом’якшення впливу взаємної кореляції таких характеристик та підхарактеристик при використанні їх у моделі якості. Таке пом’якшення здійснюється шляхом виявлення спільних атрибутів, забезпечення їх наявності, підвищення точності їх значень, або, за можливості, обмеження одночасного задіювання наборів підхарактеристик, що містять однакові атрибути.

Висновки до розділу 1

Враховуючи той факт, що саме модель якості ISO 25010 (SQuaRE) оцінює найбільшу кількість нефункціональних вимог (які відображають деякі з емерджентних властивостей програмних систем) та спрямовує розроблення ПЗ на специфікацію і оцінювання вимог, для подальшої роботи обрано дану модель якості ПЗ. Проведений аналіз стандартів ISO 25010, ISO 25023 показав, що вони представлені природньою мовою у текстовій формі, тому відсутній механізм верифікації результатів імплементації цих стандартів у процес розроблення ПЗ. Сьогодні оцінювання якості ПЗ за стандартом ISO 25010 [13] відбувається так: на основі атрибутів якості, зазначених у ISO 25023 [19], оцінюються підхарактеристики та характеристики якості, які, в свою чергу, надають комплексну оцінку якості ПЗ. Атрибути якості, визначені у специфікації вимог, входять до інформації щодо якості специфікації вимог до ПЗ. Однією з проблем моделей якості є саме визначення значущості атрибутів якості, на яку впливає існування кореляції характеристик та підхарактеристик за атрибутами.

2 МОДЕЛЮВАННЯ ПРОЦЕСУ ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Онтології як засіб інтеграції вимог щодо якості програмного забезпечення

Для підвищення надійності оцінки якості ПЗ важливе значення мають професіонали, які вже мають досвід оцінки якості та складності для різних типів програмного забезпечення. Дуже цінними є знання характеристик кореляції перешкод та характерних атрибутів якості та показників складності та показників якості, як частина параметрів або показників, щодо яких існує кореляція, можливо взагалі апостатизація, що може погіршити точність та достовірність оцінок.

Інформація про оцінку якості відповідно до ISO 25010: 2011 (наприклад, характеристик та характеристик для атрибутів), а також для визначення якості та складності програмного проекту та на основі результатів метричного аналізу (наприклад, взаємозв'язки, метрики, показники) можна зручно представити у вигляді онтологій, що відображають причинно-наслідкові зв’язки між поняттями.

Концепцію онтології в галузі інформаційних технологій застосував Том Грубер [64]. Онтологія - це специфікація концептуалізації, де концептуалізація - це опис понять, а також усієї інформації, що стосується понять, необхідних для опису та вирішення проблем предметної області. Онтології використовуються для відображення відомих знань та набуття, структурування знань та створення нових знань домену.

Онтологія - це формальне поняття конкретних предметних областей. Вони дозволяють концептуалізувати домен шляхом захоплення сутностей та відносин у домені. Визначення, яке пов’язує суб'єкт, що бере участь, частково дозволяє нам зрозуміти його (зміст), оскільки воно дає можливість побачити, де суб'єкт відноситься до іншого домену [12].

Формально онтологія визначається як: $O = X , RX , F$, де X – скінченна множина понять (концептів) предметної галузі, RX – скінченна множина відношень між поняттями, F – скінченна множина функцій інтерпретації, заданих на концептах чи відношеннях.

Для представлення взаємозв'язів онтологій різних рівнів використовуються моделі онтологічних систем: $Z = O, PO, M$, де O – онтологія верхнього рівня, що містить загальні поняття і відношення, які не залежать від предметної галузі; PO – множина предметних онтологій предметної галузі (відповідно до задач, що розв’язуються); MO – модель машини виведення даної онтологічної системи.

У найпростішому випадку процес побудови онтологій складається з двох фаз:

1) виділення понять - основні поняття предметної області;

2) побудова зв’язків між поняттями - визначення відносин та взаємодій основних понять.

Перевага використання онтологій - це системний підхід до вивчення предметної області, можливість цілісного перегляду відомої інформації предметної області, ідентифікація дублювання та прогалини в знаннях на основі візуалізації відсутніх логічних зв’язків, доступу, розуміння та інформаційний аналіз інтелектуальних та неінтелектуальних агентів (що дуже актуально в сучасному переході до епохи семантичної павутини (Semantic Web), коли ресурси повинні бути зрозумілими не лише людині, а й агентам) [60].

Онтологія визначається як критична технологія розвитку семантичного Інтернету і відіграє вирішальну роль в організації обробки інформації на основі Веб, її обміну та обміну між додатками [12].

Ідея використання онтологій як інструментів інтеграції даних, знань та вимог самого програмного забезпечення не нова. Українські вчені використали онтології в розробці та розробці. Так, є. Бур запропонував методи та інструменти для побудови програмних систем на основі

онтологічних моделей завдань [60]. І. Шостак та Ю. Бутенко розробили онтологічні моделі та методи формування нормативної профільної сертифікації [62]. Л. Бабенко запропонував онтологічний підхід до специфічних властивостей програмних систем та їх компонентів [63]. В. Литвин розробив онтологічні моделі інтелектуальних систем підтримки прийняття рішень [61].

У галузі інформаційних технологій поводження з інформацією та даними як українські, так і зарубіжні вчені також зважили онтологію: на реалізацію семантичних пошукових, класифікаційних та текстових документів квасена, на моделювання інтелектуальних систем підтримки прийняття рішень, на розвиток комунікацій мереж, заснованих на агентах, для організації динамічних зв'язків та динамічних каталогів для вимог до сліду IN [65].

Онтологія також використовувалася для формалізації стандартів [70]. Таким чином, автори [69] досліджували створення онтологічної інфраструктури як єдиного когерентного стандарту, заснованого на стандартах ISO / IEC JTC1 / SC7. Автори роботи [70] висловились за використання мови мовної онтологічної моделі (OPL) як основного компонента онтологічної інфраструктури, створення спільної концептуалізації ISO / IEC JTC1 / SC7.Мета [51] - створити онтологію домену для ISO / IEC 24744, яка послужить смисловим зв'язком для допомоги та кращої взаємодії між різними користувачами стандарту (людиною, програмним забезпеченням, машиною). Стаття 52] стосується використання онтологій домену для аналізу програмного забезпечення та інструментів реінжинірингу. У роботі [53] представлена онтологічна модель опису та визначення предметних та експлуатаційних знань щодо забезпечення якості програмного забезпечення.

Таким чином, дослідження відомих онтологічних моделей показало, що в даний час не існує онтологічних моделей предметної області «Програмна інженерія» (частина «Якість програмного забезпечення»).

Сьогодні розроблено ряд програмних продуктів, включаючи універсальні, для створення та візуалізації онтологій, що дозволяє вам працювати з різними предметними областями: Ontolingua Server, SMART, Protégé, OntoEdit, WebOnto, ODE (Ontological Design Environment), DOE (Диференціальний редактор онтології), CONE, OntoEditor+. Для подальшої роботи використовується безкоштовне програмне забезпечення Protégé 4.2 [57], яке дає можливість працювати (створювати, редагувати, переглядати та порівнювати) з онтологіями різних предметних областей.

2.2 Використання онтологій для оцінювання якості у специфікаціях вимог до ПЗ

Основними джерелами інформації на етапі формування та формулювання вимог до програмного забезпечення є бізнес-вимоги замовника, доменні вимоги, стандарти, описи процесу розробки та впровадження такого програмного забезпечення тощо, які складають Комплекс вхідного інформації (CVI) що описує функції майбутнього забезпечення програмного забезпечення, а також його властивості та обмеження. Вимоги бізнесу описують ціль системи, критерії її досягнення, ключові вимоги до результатів та їх пріоритети та обмеження.

Розглянемо більш детально інформацію, яка може бути включена в CVI, наприклад, вимоги до бухгалтерського обліку Enterprise-системи [58]. Вся документація на цей продукт складалася із загальної частини, вимог, опису впровадження, тестування, посібників та управління.

Загальна частина документації складалася з двох розділів: перелік термінів, їх визначення та опис ділових ролей користувачів. Будь-яка системна документація, включаючи, наприклад, тестові сценарії, заснована на визначеннях у цьому розділі. Близько 200 визначень бізнесу та системи з’явились у переліку термінів у системі бухгалтерського обліку підприємства. Перелік ділових ролей використовується для реалізації груп і

ролей користувачів, присвоєння функціональних прав, тестерам необхідно протестувати сценарій під необхідними ролями.

Опис ролей було надано на якісному рівні у текстовій формі на основі аналізу основних функцій працівників [58].

Розділ вимог до документації щодо Системи бухгалтерського обліку підприємства включав вимоги бізнесу (загальні сценарії, сценарії використання, алгоритми та аудити), системні вимоги, нефункціональні вимоги, вимоги інтеграції та вимоги до інтерфейсу. Бізнес-вимоги описували, що потрібно діловим користувачам. Наприклад, їм взагалі не потрібен об’єкт користувача, але їм потрібно мати можливість змінити значення елемента у рахунку-фактурі та роздрукувати його. Бізнес-вимоги складалися із загальних сценаріїв, випадків використання та описів алгоритмів обробки даних. Вимоги були представлені у вигляді дерева (з циклами). Тобто загальні сценарії були уточнені сценаріями використання, які, в свою чергу, мали посилання на перевірки та алгоритми [58].

Коренева сторінка дерева вимог до системи Enterprise була в загальних сценаріях, кожен з яких описував один з 24 бізнес-процесів, які потрібно впровадити. Поширений сценарій - це послідовність кроків користувача та системи для досягнення конкретної мети, головна мета - узагальнити сценарії використання та побачити, що він хоче зробити і як система допомагає йому у цьому. Поширені сценарії містять кроки, які користувач здійснив систему, оскільки необхідно було показати його роботу в повному обсязі, з усіма кроками, необхідними для досягнення цілей бізнесу [58].

Сценарій містив пронумеровані кроки. Кожен крок - це, як правило, просте речення в теперішньому часі [58].

При написанні алгоритмів аналітик намагався описати їх якнайповніше. Однак текст, що виходить, погано читабельний, і, як правило, все ж були втрачені деякі деталі. Тому аналітик повинен описати алгоритм максимально повно, оскільки це важливо з точки зору ділової логіки, фонових перевірок, програміст зобов'язаний включити до коду 258].

Загальна частина та частина вимог до документації на товар включені до набору вхідної інформації. Вся ця інформація, як видно з вищенаведеного прикладу, представлена у вигляді словесних описів бізнес-процесів на рівнях бізнес-функцій та операцій для різних функціональних частин, наборах бізнес-правил 99 та наборі доменних моделей, наприклад, концептуальні моделі, сценарії, схеми тощо.

Крім того, UII необхідно враховувати, подані у текстовій формі розроблення стандартів та стандартів предметної області, для якої ви розробляєтесь (рис. 2.1).

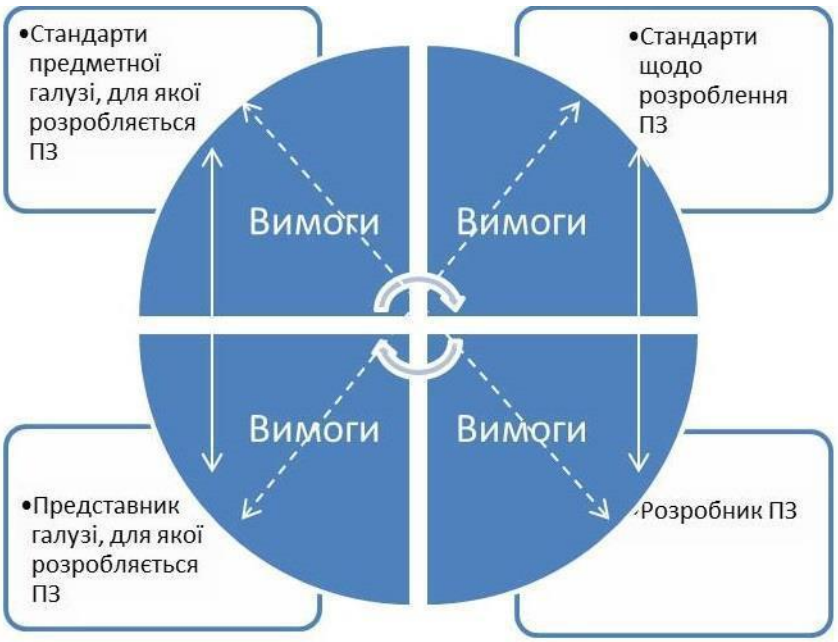

Рисунок 2.1 - Врахування вимог стандартів щодо розроблення ПЗ та стандартів предметної галузі, для якої розробляється ПЗ

На основі вищевказаного комплексу вхідної інформації (КВІ) формується специфікація вимог до ПЗ (рис. 2.2).

Отже, характеристики специфікації вимог до ПЗ значною мірою визначаються характеристиками комплексу вхідної інформації. Якщо у КВІ недостатньо інформації, або ж вона є неточною, неоднозначною чи

суперечливою, то існує висока імовірність, що усі ці недоліки проявляться і у специфікації вимог до ПЗ. А програмні проекти, специфікація вимог яких містять недостатню, неточну, неповну, суперечливу інформацію, не можуть мати успішної реалізації.

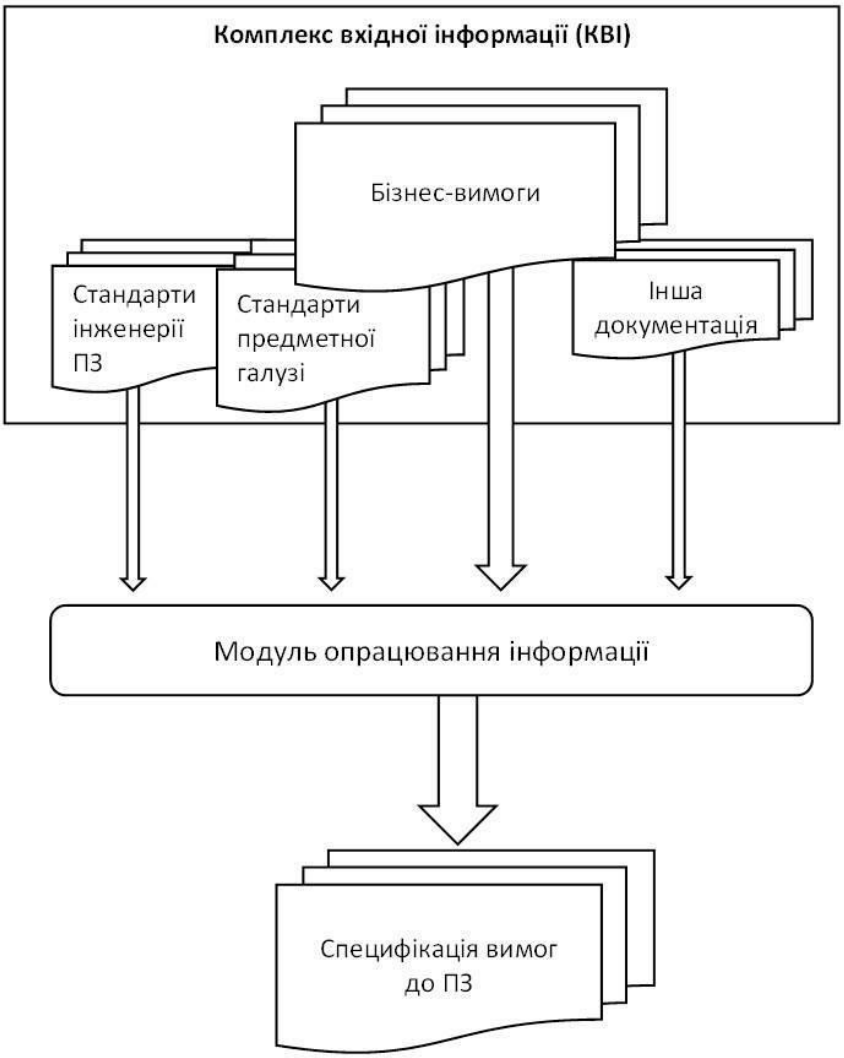

Рисунок 2.2 - Структура інформаційних потоків при формуванні специфікації вимог до програмного забезпечення

Тому, для забезпечення якості ПЗ необхідно здійснити дослідження характеристик КВІ з метою виявлення та усунення проблем і недоліків на початкових етапах життєвого циклу ПЗ. В процесі такого дослідження необхідно оцінити, наскільки повно у КВІ, зокрема, у бізнес-вимогах,

відображена інформація, що описує функції, властивості та обмеження майбутнього програмного забезпечення, особливо ті, що характеризують його нефункційні характеристики, такі як якість, надійність, гарантоздатність та ін., і виявити факти недостатності інформації, котра має до них відношення (рис.2.3). За таких умов актуальною проблемою є оцінювання достатності інформації щодо якості у специфікації вимог до розроблюваного ПЗ.

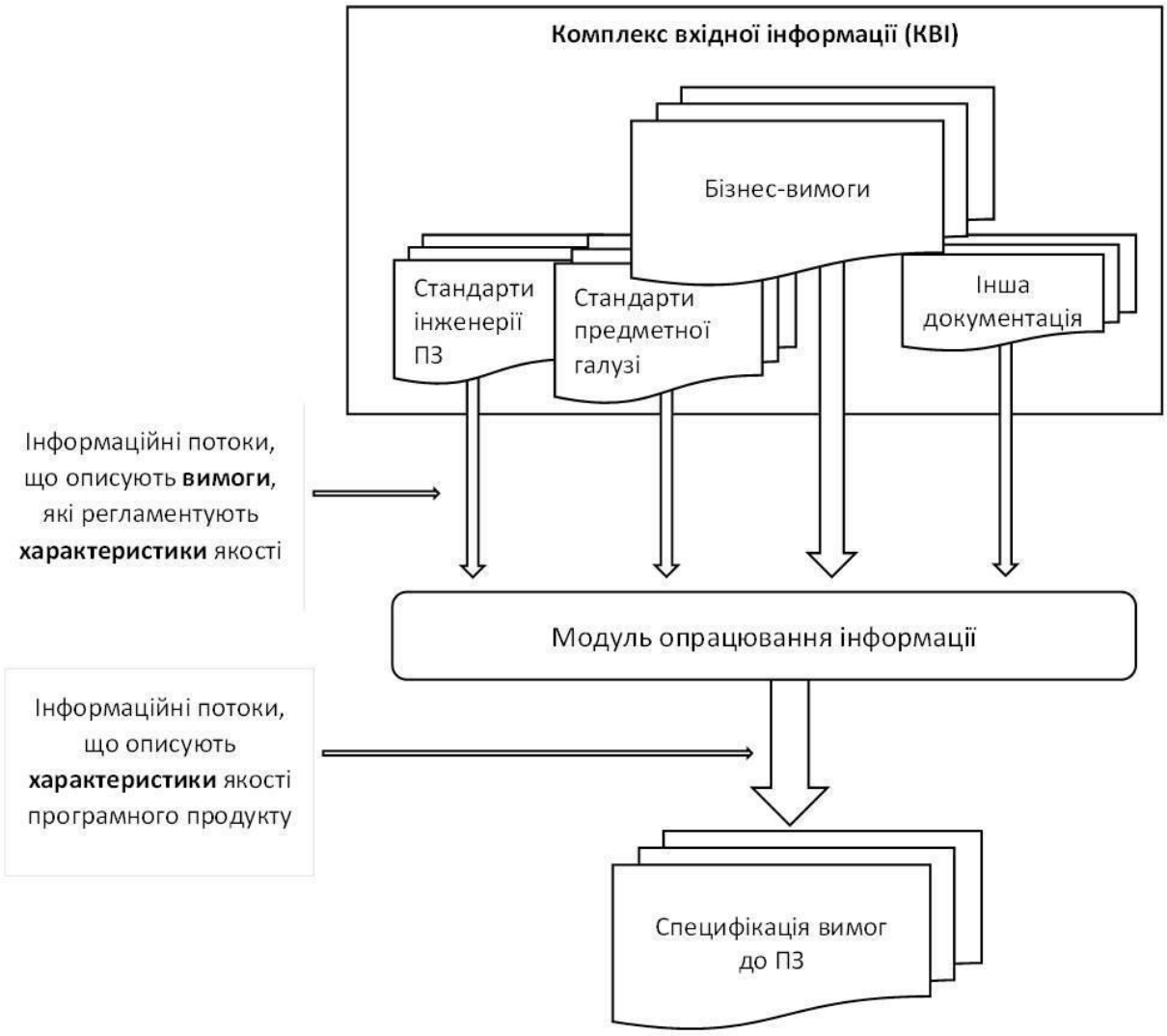

Рисунок 2.3 - Зміст інформаційних потоків при розробленні специфікації вимог до програмного забезпечення на основі КВІ

Інформаційні потоки, що описують вимоги, що регулюють характеристики якості, формуються на основі цілей та завдань організації високого рівня, а також на ділових правилах, що включають вимоги щодо корпоративної політики та галузевих стандартів, представлених у формі юридичного документа та проекту межі, обмеження в дизайні та реалізації. Такі інформаційні потоки складаються з вимог КВІ, які визначають якісні характеристики - наприклад, вимога "Принаймні 25% наявної пам'яті процесора пам'яті, що використовується, не повинні використовуватися під запланованим піковим навантаженням", регулює таку якість характеристики як ефективність; вимога "Тільки користувачі, які мають привілеї рівня аудиту, повинні мати змогу переглядати клієнтські транзакції" регулює таку характеристику якості, як безпека.

Модуль обробки інформації (автоматизована підсистема людина-машина) виконує побудову та заповнення моделей якості програмного забезпечення на основі інформації. Моделі якості програмного забезпечення включають інформацію про характеристики якості, підхарактеристики, атрибути якості, взяті зі стандартів [32], а також інформацію про метрики та показники якості, взяті з галузевих публікацій [16]. Оскільки моделі якості програмного забезпечення використовуються для систематизації інформації про якість, отриманої від КВІ, і для приведення її до загального (рівномірного) вигляду відповідно до стандартів та публікацій, доцільно розробляти такі моделі в статичній множинній формі.

Інформаційні потоки, що описують характеристики якості програмного продукту, формуються модулем обробки інформації на основі моделей якості програмного забезпечення, і тому містять систематизовану інформацію про якість, що входить до специфікації вимог до програмного забезпечення. Наприклад, вимога КВІ "Принаймні 25% пропускної здатності процесора та пам'яті, доступної для використання, не повинні використовуватися при запланованому піковому навантаженні" має таке подання в специфікації

програмних вимог: "Середня пропускна здатність становить 75%", "Максимальне використання пам'яті" - 75% "," Межа навантаження - максимальна навантаження ", яка регулює атрибути якості, від яких залежить атрибут ефективності" Ефективність ". Вимоги КВІ "Тільки користувачі, які мають привілеї рівня аудиту, повинні мати змогу переглядати клієнтські транзакції", містяться у специфікації вимог до програмного забезпечення: "Управління доступністю - рівні привілеїв", а також вимоги цього типу такі вимоги специфікації розраховуються як "Кількість подій, які потребують певного типу властивості", "Кількість наданих методів аутентифікації", які керують атрибутами якості, від яких залежить характеристика якості "Захищеність".

Отже, моделювання руху інформаційних потоків при формуванні специфікації вимог до ПЗ показало, що для визначення складу та наповнення інформаційних потоків необхідно розглянути теоретико-множинні моделі представлення інформації щодо якості у специфікації вимог до ПЗ, які вирішать задачу систематизації всієї наявної інформації щодо якості ПЗ та приведення її до єдиної уніфікованої форми.

Для вирішення проблеми оцінювання достатності об’єму інформації щодо якості у специфікаціях вимог до ПЗ розроблено *базові (універсальні) онтології* предметної галузі «Інженерія програмного забезпечення» (частини «Якість ПЗ») на основі стандартів ISO 25010:2011 [132](для забезпечення та оцінювання якості за моделлю SQuaRE) та галузевих публікацій [16] (для забезпечення та оцінювання якості на основі опрацювання метричної інформації). Базові онтології відображають необхідну інформацію щодо якості (атрибути та показники), яка повинна бути наявною у специфікації вимог до ПЗ для забезпечення достатності її інформації щодо якості.

В процесі розроблення конкретного ПЗ, окрім базових онтологій, необхідно мати *онтологію цього ПЗ*, яка відображає наявну інформацію щодо якості (атрибути та показники) у специфікації вимог до конкретного ПЗ.

Критерій достатності інформації щодо якості у специфікаціях вимог до ПЗ. Нехай *SAMI* – множина відсутніх атрибутів та показників, і

$$SAMI = Базова\ онтол./(Базова\ онтол. \cap Онтол.\ конкретного\ ПЗ). \quad (2.1)$$

Тоді:

- якщо $SAMI = \varnothing$, то інформації щодо якості у специфікації вимог до ПЗ достатньо;

- якщо $SAMI \neq \varnothing$, то інформації щодо якості у специфікації вимог до ПЗ недостатньо, і специфікація вимог потребує доповнення інформацією щодо якості (атрибутами та (або) показниками).

Для визначення структури та наповнення базових (універсальних) онтологій предметної галузі «Інженерія програмного забезпечення» (частини «Якість ПЗ») необхідні теоретико-множинні та онтологічні моделі якості ПЗ (за стандартом ISO 25010). Крім цього, враховуючи представлений критерій достатності інформації щодо якості у специфікаціях вимог до ПЗ, необхідно розробити моделі процесу оцінювання достатності інформації для визначення якості програмного забезпечення на основі стандарту ISO 25010:2011 та з використанням результатів метричного аналізу.

2.3 Модель процесу оцінювання якості програмного забезпечення на основі стандарту ISO 25010

Якість ПЗ (Q), відповідно до стандарту [132], є функцією від восьми основних характеристик якості

$$(QCH = \{qch_1,..., qch_8\}).$$

Множину характеристик якості ПЗ запишемо у вигляді

$$QCH = \{Fs, Pe, Ub, Rb, Cb, Scr, Mb, Pb\},$$

де: *Fs* – функційна придатність,

Pe – ефективність,

Ub – зручність використання,

Rb – надійність,

Cb – сумісність,

Scr – захищеність,

Mb – супроводжуваність,

Pb – можливість переносу.

Вказані характеристики якості можуть приймати значення з певного діапазону. Тоді якість ПЗ є функцією від цих характеристик:

$$Q = f\,(Fs,\ Pe, Ub,\ Rb,\ Cb,\ Scr,\ Mb,\ Pb)\ . \qquad (2.2)$$

Кожна з вищевказаних характеристик якості є функцією від декількох підхарактеристик якості:

$$Fs = f_1(qsch_1,\ qsch_2\ ,\ qsch_3\ ) = f_1(FCom,\ FCor,\ FAppr), \qquad (2.3)$$

Де *FCom* – функційна повнота,

FCor – функційна коректність,

FAppr – функційна доцільність;

$$Pe = f_{\,2}\,(qsch_4\ ,\ qsch_5\ ,\ qsch_6\ ) = f_{\,2}\,(Tb,\ Ru,\ Cc)\ , \qquad (2.4)$$

Де *Tb* – поведінка у часі,

Ru – поведінка ресурсів,

Cc – ємність (місткість);

$$Ub = f_3\,(qsch_7\ ,...,\ qsch_{12}\ ) = f_3\,(\ Ar,\ Lb,\ Ob, Uep, Uia,\ Ab)\ , \qquad (2.5)$$

де *Ar* – розпізнавання доцільності,

Lb – можливість вивчення,

Ob – керованість,

Uep – захист від помилок користувача,

Uia – естетичність інтерфейсу користувача,

Ab – доступність;

$$Rb = f_4\,(qsch_{13}\,,\ldots,\,qsch_{16}\,) = f_{\,4}\,(Mat,\,Avb,\,Ft,\,Rcv)\,, \qquad (2.6)$$

Де *Mat* – зрілість,

Avb – наявність (доступність),

Ft – відмовостійкість,

Rcv – відновлюваність;

$$Cb = f_5\,(qsch_{17}\,,\,qsch_{18}\,) = f_5\,(Ce,\,Ib)\,, \quad (2.7$$

Де *Ce* – співіснування,

Ib – взаємодія;

$$Scr = f_6\,(qsch_{19}\,,\ldots,\,qsch_{23}\,) = f_6\,(Conf\,,\,Int,\,Nr,\,Acb,\,Auth)\,, \qquad (2.8)$$

Де *Conf* - конфіденційність,

Int – цілісність,

Nr – невідхилюваність,

Acb – підзвітність,

Auth – ідентичність;

$$Mb = f_7\,(qsch_{24}\,,\ldots,\,qsch_{28}\,) = f_7\,(Mod\,,\,Rub,\,Anb,\,Mdfb,Tsb)\,, \qquad (2.9)$$

Де *Mod* – модульність,

Rub – повторне використання,

Anb – аналізованість,

Mdfb – модифікованість,

Tsb – тестованість.

$$Pb = f_8\,(qsch_{29}\,,\, qsch_{30}\,,\, qsch_{31}) = f_8\,(\,Adb,\, Inb,\, Rpb)\,, \qquad (2.10)$$

де *Adb* – адаптованість,

Inb – можливість інсталяції,

Rpb – можливість заміни.

Таким чином, множина основних підхарактеристик якості ПЗ має вигляд:

$$QSCH = \{qsch1, \dots, qsch31\} =$$

$$= \begin{Bmatrix} FCom, FCor, FAppr, Tb, Ru, Cc, Ar, Lb, Ob, Uep, Uia, \\ Avb, Ft, Rcv, Ce, Ib, Conf\ , Int, Nr, Acb, Auth, \\ Mod\ , Rub, Anb, Mdfb, Tsb, Adb, Inb, Rpb \end{Bmatrix},$$

Кожна підхарактеристика якості ПЗ є функцією певних атрибутів якості ПЗ, описаних у стандарті [139]. Підхарактеристики якості залежать від 138 різних атрибутів. Множину атрибутів якості ПЗ представимо як:

$$QMS = \{qms_1, \dots, qms_{138}\}\ .$$

Однією з найважливіших задач при оцінюванні якості ПЗ є забезпечення цілісності, повноти та несуперечливості характеристик, підхарактеристик та атрибутів якості. Коректність та достовірність результату оцінювання якості ПЗ не може бути забезпечена, якщо у характеристиках, підхарактеристиках чи атрибутах якості наявна неповнота або суперечливість. Також надлишковість характеристик, підхарактеристик або атрибутів якості ПЗ може призвести до зростання складності обчислення

якості програмного забезпечення. На сьогодні не існує єдиного підходу до оцінювання цілісності, повноти, суперечливості та надлишковості характеристик, підхарактеристик та атрибутів якості.

Введемо визначення понять цілісності, повноти, несуперечливості та надлишковості характеристик, підхарактеристик та атрибутів якості ПЗ на основі відповідних визначень теорії інформації [259-261]. *Несуперечливість* означає, що однакові характеристики, підхарактеристики та атрибути повинні позначати одні ті ж властивості ПЗ. *Повнота* передбачає, що після надання кількісної оцінки всіх атрибутів можна одержати коректні кількісні оцінки підхарактеристик та характеристик якості ПЗ, які, в свою чергу, повинні забезпечити коректну та достовірну кількісну оцінку якості ПЗ. *Цілісність* передбачає узгоджене представлення інформації для зв'язаних характеристик та підхарактеристик.

Надлишковість означає відсутність новизни в інформації, яку вони несуть [74]. Для визначення повноти, цілісності, надлишковості та несуперечливості характеристик та підхарактеристик якості ПЗ розглянемо сутність всіх характеристик та підхарактеристик. З моделі якості ПЗ, а також з аналізу змісту характеристик та підхарактеристик слідує, що якість не залежить від однакових характеристик, а жодна характеристика не залежить від однакових підхарактеристик. Аналіз робіт [225] та стандарту ISO 25023 [139] показав, що є атрибути, від яких залежать більше однієї підхарактеристики та характеристики, але такі атрибути позначають однакові властивості ПЗ. Тому характеристики, підхарактеристики та атрибути якості ПЗ *є несуперечливими*.

Характеристики, підхарактеристики та атрибути якості ПЗ *є повними,* оскільки кількісні оцінки всіх атрибутів надають коректні кількісні оцінки всіх підхарактеристик якості ПЗ, які, в свою чергу, забезпечують коректну та достовірну кількісну оцінку характеристик якості та, власне, якості ПЗ. Але на практиці існують певні труднощі, зокрема, визначення функції f, яка за кількісними оцінками підхарактеристик якості ПЗ (функції f_1... f_8),

враховуючи взаємовпливи атрибутів (функції $\varphi_1 ... \varphi_{31}$), дозволяє отримати кількісну оцінку характеристик якості.

Щодо *цілісності* характеристик, підхарактеристик та атрибутів якості, то для зв'язаних характеристик, підхарактеристик та атрибутів необхідне узгоджене представлення інформації з врахуванням взаємовпливів атрибутів при оцінюванні підхарактеристик, взаємовпливів підхарактеристик при оцінюванні характеристик взаємовпливів характеристик при оцінюванні якості ПЗ, що лежить в основі стандарту ISO 25010:2011.

Якщо атрибути якості мають рівну ймовірність, то інформаційна ентропія атрибутів складає 916,32 біт, тобто відмінна від 0, тому інформація, яку вони несуть, містить новизну, тобто *надлишковості атрибутів якості ПЗ немає*.

Оскільки має місце кореляція підхарактеристик та характеристик за певними атрибутами (підхарактеристики якості залежать від 203 атрибутів, частина з яких повторюється, тому є всього 138 різних атрибутів), то є атрибути, від яких залежать більше однієї підхарактеристики та характеристики якості ПЗ, а існування взаємозв’язків між характеристиками та підхарактеристиками впливає на значущість та вагу характеристик і підхарактеристик якості ПЗ [229], тому для підвищення достатності об’єму інформації щодо якості у специфікації вимог необхідно виявити спільні атрибути для характеристик і підхарактеристик якості ПЗ та визначити значущість атрибутів якості ПЗ. При виявленні спільних атрибутів цінною інформацією є знання досвідчених фахівців щодо взаємовпливу та кореляції характеристик і підхарактеристик за атрибутами якості ПЗ, тому їх в арто зберігати та використовувати. У якості засобу відображення цих знань обрано онтології.

Процес оцінювання достатності інформації для визначення якості ПЗ полягає у:

- порівнянні онтології для визначення якості конкретного ПЗ з базовою (універсальною) онтологією з метою виявлення атрибутів, відсутніх в

онтології для визначення якості конкретного ПЗ, тобто відсутніх у специфікації вимог до ПЗ, за якою ця онтологія була побудована, а також виявлення характеристик та підхрактеристик якості ПЗ, які неможливо обчислити на основі наявних у специфікації вимог до конкретного ПЗ атрибутів;

- формуванні висновку про недостатність інформації щодо якості у специфікації вимог до конкретного ПЗ, якщо встановлено атрибути, відсутні у специфікації вимог до ПЗ, а також характеристики та підхарактеристики якості ПЗ, які неможливо обчислити на основі наявних атрибутів у специфікації вимог до конкретного ПЗ;

- порівнянні онтології для визначення якості конкретного ПЗ зі зваженою базовою (універсальною) онтологією, якщо було сформовано висновок про недостатність інформації, – з метою встановлення вагових коефіцієнтів атрибутів, відсутніх у специфікації вимог до конкретного ПЗ, для подальшого визначення пріоритетності доповнення атрибутів у специфікацію вимог до ПЗ.

Висновок до розділу 2

Для визначення структури та наповнення базових онтологій предметної галузі «Інженерія програмного забезпечення» (частини «Якість ПЗ») необхідні теоретико-множинні та онтологічні моделі якості ПЗ (за стандартом ISO 25010).

Враховуючи представлений критерій достатності інформації щодо якості у специфікаціях вимог до ПЗ, необхідно розробити моделі процесу оцінювання достатності інформації для визначення якості програмного забезпечення на основі стандарту ISO 25010:2011 та з використанням результатів метричного аналізу.

Розглянута модель процесу оцінювання достатності інформації для визначення якості ПЗ за стандартом ISO 25010:2011 відображає особливості оцінювання достатності інформації для визначення якості ПЗ за стандартом

ISO 25010:2011, забезпечує адаптацію до особливостей предметної галузі та надає можливість доступу, аналізу і розуміння інформації не лише людиною, але і віртуальними агентами (ботами), а також є теоретичним підґрунтям для розроблення методів та засобів оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ.

3 ОЦІНЮВАННЯ ДОСТАТНОСТІ ІНФОРМАЦІЇ ЩОДО ЯКОСТІ ЗА СТАНДАРТОМ ISO 25010:2011 У СПЕЦИФІКАЦІЯХ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Формування логічного висновку щодо якості за стандартом ISO 25010:2011 у специфікаціях вимог до програмного забезпечення

Для оцінювання достатності інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ на основі онтології за методами оцінювання достатності інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ на основі онтології та на основі зваженої онтології, необхідно виконати генерування та наповнення шаблону онтології для визначення якості конкретного програмного забезпечення. Процес генерування та наповнення шаблону онтології для визначення якості конкретного ПЗ слід максимально автоматизувати та спростити з метою мінімізації впливу людського фактору та з метою спрощення оцінювання достатності інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ як розробником, так і замовником.

Спосіб генерування та наповнення шаблону онтології для визначення якості конкретного ПЗ складається з таких етапів [74]:

- відкрити базову онтологію предметної галузі «Інженерія програмного забезпечення» (частина «Якість ПЗ»);

- використовуючи результати аналізу специфікації вимог до ПЗ конкретного ПЗ на предмет наявності атрибутів, необхідних для визначення підхарактеристик та характеристик якості ПЗ, проведеного на кроці №1 Методу оцінювання достатності інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ на основі онтології, видалити з базової онтології всі атрибути, яких не було виявлено у специфікації вимог до ПЗ конкретного ПЗ;

- зберегти внесені зміни, створюючи тим самим онтологію для визначення якості конкретного програмного забезпечення.

Проаналізуємо продукційні правила формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ (множина PR = { pr_1,.., pr_{140}}) на основі базової та зваженої базової онтологій предметної галузі «Інженерія програмного забезпечення (частина «Якість ПЗ») [77]:

1) якщо у специфікації вимог до ПЗ відсутній атрибут «Number Of Functions (Кількість функцій)» або користувач його видалив з базової онтології), то: fc := fc +1, тобто недостатньо інформації для визначення підхарактеристики Functional Completeness (Функційна повнота). Тоді лічильник відсутніх атрибутів для цієї підхарактеристики збільшується на 1, 2 або 3.

fa := *fa* +1	лічильник відсутніх атрибутів для підхарактеристики Functional Appropriateness (Функційна доцільність)
ft := *ft* + 1	лічильник відсутніх атрибутів для Fault Tolerance (Відмовостійкість)
ar := *ar* +1	лічильник відсутніх атрибутів для Appropriateness Recognisability (Розпізнавання доцільності),
lb := *lb* +1	лічильник відсутніх атрибутів для Learnability (Можливість вивчення)
ob := *ob* +1	лічильник відсутніх атрибутів для Operability (Керованість)
md:= *md* +1	лічильник відсутніх атрибутів для Modularity (Модульність)
mfb:= *mfb* +1	лічильник відсутніх атрибутів для Modifability (Модифікованість)
cex := *cex* +1	лічильник відсутніх атрибутів для CoExistence (Співіснування)
ab := *ab* +1	лічильник відсутніх атрибутів для Adaptability (Адаптованість)
rb := *rb* +1	лічильник відсутніх атрибутів для Replaceability (Можливість заміни)
fy := *fy* + 2	недостатньо інформації для визначення двох підхарактеристик характеристики Functional Suitability (Функційна придатність)
ry := *ry* + 1	лічильник відсутніх атрибутів для характеристики Reliability (Надійність)

uy := *uy* + 3 лічильник відсутніх атрибутів для Usability (Зручність використання)

my := *my* + 2 лічильник відсутніх атрибутів для Maintainability (Супроводжуваність)

cy := *cy* + 1 лічильник відсутніх атрибутів для Compatibility (Сумісність)

py := *py* + 2 лічильник відсутніх атрибутів для Portability (Можливість переносу)

Отже, недостатньо інформації для визначення 11 з 31 підхарактеристики, а також 6 з 8 характеристик якості ПЗ; mas [Number Of Functions]:=11/138 (у відповідний елемент масиву mas записується ваговий коефіцієнт відсутнього атрибуту, визначений за зваженою базовою онтологією предметної галузі «Інженерія програмного забезпечення (частина «Якість ПЗ»)»);

2.) Якщо у специфікації вимог до ПЗ відсутній атрибут «Operation Time (Час роботи)», то:

fcr := *fcr* + 1 лічильник відсутніх атрибутів для підхарактеристики Functional Correctness (Функційна коректність)

fa := *fa* +1 , лічильник відсутніх атрибутів для Maturity (Зрілість)

ma := *ma* +1

av : = *av* +1 лічильник відсутніх атрибутів для Availability (Наявність)

rvb := *rvb* +1 лічильник відсутніх атрибутів для Recoverability (Відновлюваність)

(*tb* := *tb* +1 лічильник відсутніх атрибутів для Time Behaviour (Поведінка у часі)

ru := *ru* +1 лічильник відсутніх атрибутів для Resource Utilization (Поведінка ресурсів)

lb := *lb* +1, лічильник відсутніх атрибутів для Testability (Тестованість)

ob := *ob* +1,

md := *md* +1,

mfb:=*mfb* +1,

tst := *tst* +1

cf := *cf*+1 лічильник відсутніх атрибутів для Confidentiality (Конфіденційність)

ig := *ig*+1 лічильник відсутніх атрибутів для Integrity (Цілісність)

cex := *cex* +1, *io* := *io* +1 лічильник відсутніх атрибутів для Interoperability (Взаємодія)

ab := *ab* +1 лічильник відсутніх атрибутів для характеристики

fy := *fy* + 2 *ry* := *ry* + 3 Performance Efficiency (Ефективність)

ey := *ey* + 2

uy := *uy* + 2 лічильник відсутніх атрибутів для Security (Захищеність)

my := *my* + 3

sy := *sy* + 2

cy := *cy* + 2

py := *py* + 1,

Тобто недостатньо інформації для визначення 17 з 31 підхарактеристики, а також 8 з 8 характеристик якості ПЗ; mas[Operation Time]:=17/138;

3) аналогічно сформовано правила для решти 136 атрибутів у Додатку В;

139) якщо *fc, fcr,fa, ma,av,ft,vb,tb,ru,ccy,ar,lb,ob, uepr, uiae, acsb,md, rusb, anb, mfb, tsb, cf, ig, nrpd, accb, athc, cex, io, ab, ib* та *rb* дорівнюють нулю, то у КВІ та специфікації вимог до ПЗ достатньо інформації для визначення всіх підхарактеристик, інакше: у КВІ та специфікації вимог до ПЗ недостатньо інформації щодо якості (атрибутів) для визначення певних підхарактеристик якості ПЗ:

$0 < fc \leq 4$ у КВІ та специфікації вимог до ПЗ недостатньо інформації для визначення підхарактеристики Functional

	Completeness (Функційна повнота)
$fc = 4$	у специфікації вимог до ПЗ взагалі відсутня інформація для визначення підхарактеристики Functional Completeness
$0 < fcr \leq 5$	недостатньо інформації для Functional Correctness
$fcr = 5$	відсутня інформація для Functional Correctness
$0 < fa \leq 6$,	недостатньо інформації для Functional Appropriateness (Функційна доцільність);
$fa = 6$	відсутня інформація для Functional Appropriateness
$0 < ma \leq 14$	недостатньо інформації для Maturity (Зрілість)
$ma = 14$	відсутня інформація для Maturity
$0 < av \leq 4$	недостатньо інформації для Availability (Наявність
$av = 4$	відсутня інформація для Availability
$0 < ft \leq 5$	недостатньо інформації для Fault Tolerance (Відмовостійкість)
$ft = 5$	взагалі відсутня інформація для Fault Tolerance
0 □ *rvb* □ 7	недостатньо інформації для Recoverability (Відновлюваність)
rvb □ 7	відсутня інформація для Recoverability;
$0 < tb \leq 7$	недостатньо інформації для Time Behaviour (Поведінка у часі)
$tb = 7$	відсутня інформація для Time Behaviour
$0 < ru \leq 14$	недостатньо інформації для Resource Utilization (Поведінка ресурсів)
$ru = 14$	відсутня інформація для Resource Utilization
$0 < ccy \leq 5$	недостатньо інформації для Capacity (Ємність)
$ccy = 5$	відсутня інформація для Capacity
$0 < ar \leq 6$	недостатньо інформації для Appropriateness Recognisability (Розпізнавання доцільності)

$ar = 6$	відсутня інформація для Appropriateness Recognisability
$0 < ob \leq 13$	недостатньо інформації для Operability (Керованість)
$ob = 13$	взагалі відсутня інформація для Operability
$0 < uepr \leq 11$	недостатньо інформації для User Error Protection (Захист від помилок користувача)
$uepr = 11$	відсутня інформація для User Error Protection
$0 < uiae \leq 6$	недостатньо інформації для User Interface Aesthethics (Естетичність інтерфейсу користувача)
$uiae = 6$	відсутня інформація для User Interface Aesthethics
$0 < acsb \leq 5$	недостатньо інформації для Accessibility (Доступність)
$acsb = 5$	відсутня інформація для Accessibility
$0 < md \leq 7$	недостатньо інформації для Modularity (Модульність)
$md = 7$	відсутня інформація для Modularity
$0 < rusb \leq 6$	недостатньо інформації для Reusability (Повторне використання)
$rusb = 6$	відсутня інформація для Reusability
$0 < anb \leq 6$	недостатньо інформації для Analysability (Аналізованість)
$anb = 6$	відсутня інформація для Analysability
$0 < mfb \leq 8$	недостатньо інформації для Modifability(Модифікованість)
$mfb = 8$	відсутня інформація для Modifability
$0 < tst \leq 6$	недостатньо інформації для Testability (Тестованість)
$tst = 6$	відсутня інформація для Testability
$0 < cf \leq 10$	недостатньо інформації для Confidentiality (Конфіденційність)
$cf = 10$	відсутня інформація для Confidentiality
$0 < ig \leq 8$	недостатньо інформації для Integrity (Цілісність)

$ig = 8$	відсутня інформація для Integrity
$0 < nrpd \leq 2$	недостатньо інформації для Non Repudiation (Невідхилюваність)
$nrpd = 2$	відсутня інформація для Non Repudiation;
$0 < accb \leq 2$	недостатньо інформації для Accountability (Підзвітність);
$accb = 2$	відсутня інформація для Accountability
$0 < athc \leq 1$	недостатньо інформації для Authenticity (Ідентичність)
$athc = 1$	відсутня інформація для Authenticity
$0 < cex \leq 4$	недостатньо інформації для CoExistence (Співіснування)
$cex = 4$	відсутня інформація для CoExistence;
$0 < io \leq 5$	недостатньо інформації для Interoperability (Взаємодія);
$io = 5$	відсутня інформація для Interoperability
$0 < ab \leq 11$	недостатньо інформації для Adaptability (Адаптованість);
$ab = 11$	відсутня інформація для Adaptability
$0 < ib \leq 4$	недостатньо інформації для Installability (Можливість інсталяції);
$ib = 4$	відсутня інформація для Installability
$0 < rb \leq 3$	недостатньо інформації для Replaceability (Можливість заміни)
$rb = 3$	відсутня інформація для Replaceability

140) якщо *f, ry, ey, uy, my, sy, cy* та *py* дорівнюють нулю, то інформації щодо якості (атрибутів) у КВІ та специфікації вимог до ПЗ достатньо для визначення характеристик якості ПЗ за стандартом ISO 25010:2011, інакше: у КВІ та специфікації вимог до ПЗ недостатньо інформації щодо якості (атрибутів) для визначення певних характеристик якості ПЗ:

$0 < fy \leq 15$	у КВІ та специфікації вимог до ПЗ недостатньо інформації

	для визначення характеристики Functional Suitability (Функційна придатність);
$fy = 15$	у специфікації вимог до ПЗ взагалі відсутня інформація для визначення характеристики Functional Suitability
$0 < ry \leq 30$	недостатньо інформації для визначення характеристики Reliability (Надійність)
$ry = 30$	відсутня інформація для Reliability
$0 < ey \leq 26$	недостатньо інформації для визначення характеристики Performance Efficiency (Ефективність)
$ey = 26$	відсутня інформація для Performance Efficiency
$0 < uy \leq 49$	недостатньо інформації для визначення характеристики Usability (Зручність використання)
$uy = 49$	відсутня інформація для Usability
$0 < my \leq 33$	недостатньо інформації для визначення характеристики Maintainability (Супроводжуваність)
$my = 33$	відсутня інформація для Maintainability
$0 < sy \leq 23$	недостатньо інформації для визначення характеристики Security (Захищеність)
$sy = 23$	відсутня інформація для Security
$0 < cy \leq 9$	недостатньо інформації для визначення характеристики Compatibility (Сумісність)
$cy = 9$	відсутня інформація для Compatibility
$0 < py \leq 18$	недостатньо інформації для визначення характеристики Portability (Можливість переносу)
$py = 18$	відсутня інформація для Portability

Необхідно відсортувати масив mas за спаданням значень елементів (вагових коефіцієнтів відсутніх атрибутів) та вивести індекси тих елементів відсортованого масиву mas, які мають значення, відмінне від 0 – як

рекомендовану пріоритетність доповнення атрибутів у специфікацію вимог до ПЗ.

На основі продукційних правил формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ, розробимо *метод формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до програмного забезпечення* [74]:

- згідно з Методом оцінювання достатності інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до програмного забезпечення на основі онтології, враховуючи порівняльний аналіз онтологій, відбувається формування множини атрибутів $\{qms_1,...,qms_{(138 \square nm)}\}$, відсутніх в онтології для визначення якості конкретного програмного забезпечення;

- за методом здійснення пошуку в ширину в прямому напрямку [266], в підмножині продукційних правил $\{ pr_1,..., pr_{138}\}$ відбувається пошук правила для кожного з елементів множини $\{qms_1,...,qms_{(138 x nm)}\}$, згідно з якими ведеться підрахунок лічильників відсутніх атрибутів для підхарактеристик та характеристик;

- згідно з правилами з підмножини $\{ pr_{139} , pr_{140}\}$, виконується аналіз інформації щодо якості (атрибутів) у специфікації вимог до ПЗ на предмет достатності і, у разі недостатності, формуються висновки, для визначення яких підхарактеристик та характеристик якості ПЗ недостатньо інформації, а також формується сортований (за ваговими коефіцієнтами) список відсутніх атрибутів як рекомендована пріоритетність доповнення атрибутів у КВІ та специфікацію вимог до ПЗ;

- виконується числова оцінка достатності об’єму наявної у специфікації вимог інформації щодо якості (атрибутів) – слід прагнути, щоб ця оцінка була якомога більшою, тобто прямувала до 1:

- за підхарактеристиками:

$$q_{schr} = \frac{fc}{4} + \frac{fcr}{5} + \frac{fa}{6} + \frac{ma}{14} + \frac{av}{4} + \frac{ft}{5} + \frac{rvb}{7} + \frac{tb}{7} + \frac{ru}{14} + \frac{ccy}{5} + \frac{ar}{6} + \frac{lb}{8} +$$
$$+ \frac{ob}{13} + \frac{uepr}{11} + \frac{uiae}{6} + \frac{acsb}{5} + \frac{md}{7} + \frac{rusb}{6} + \frac{anb}{6} + \frac{mfb}{8} + \frac{tst}{6} + \frac{cf}{10} +$$
$$+ \frac{ig}{8} + \frac{nrpd}{2} + \frac{accb}{2} + \frac{athc}{1} + \frac{cex}{4} + \frac{io}{5} + \frac{ab}{11} + \frac{ib}{4} + \frac{rb}{3}.$$

де q_{schr} – кількість підхарактеристик, які неможливо обчислити за наявних у специфікації атрибутів; числа в чисельниках дробів показують кількість відсутніх специфікації атрибутів для певної підхарактеристики якості, числа в знаменниках дробів вказують кількість необхдіних атрибутів для кожної підхарактеристики якості згідно з теоретико-множинною моделлю якості ПЗ на основі стандарту ISO25010:2011:

$$D_{schr} = \frac{31 - q_{schr}}{31} \quad (3.2)$$

де D_{schr} – числова оцінка достатності об’єму наявної у специфікації інформації (атрибутів) для оцінювання якості ПЗ за підхарактеристиками;

- за характеристиками (з врахуванням числової оцінки достатності об’єму наявної у специфікації вимог інформації щодо якості атрибутів:

$$q_{chr} = \frac{fy}{15} + \frac{ry}{30} + \frac{ey}{26} + \frac{uy}{49} + \frac{my}{33} + \frac{sy}{23} + \frac{cy}{9} + \frac{py}{18} \quad (3.3)$$

де q_{chr} – кількість характеристик, які неможливо обчислити за наявних у специфікації атрибутів; числа в чисельниках дробів показують кількість відсутніх специфікації атрибутів для певної характеристики якості, числа в знаменниках дробів вказують кількість необхідних атрибутів для кожної характеристики якості згідно з теоретико-множинною моделлю якості ПЗ на основі стандарту ISO25010:2011:

$$D_{chr} = \frac{8 - q_{chr}}{8} \quad (3.4)$$

де D_{chr} – числова оцінка достатності об’єму наявної у специфікації вимог інформації щодо якості (атрибутів) для оцінювання якості ПЗ за характеристиками;

- виконується числова оцінка достатності об’єму наявної (після доповнення) інформації щодо якості (атрибутів) у специфікації вимог до ПЗ (зважаючи на те, що методи оцінювання достатності інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ на основі онтології та на основі зваженої онтології є ітераційними, після формування висновку про недостатність інформації у специфікації відбуватимуться доповнення КВІ та специфікації вимог необхідними атрибутами, які призводитимуть до збільшення достатності об’єму інформації щодо якості у специфікації вимог):

- за підхарактеристиками:

$$q'_{schr} = \frac{fc'}{4} + \frac{fcr'}{5} + \frac{fa'}{6} + \frac{ma'}{14} + \frac{av'}{4} + \frac{ft'}{5} + \frac{rvb'}{7} + \frac{tb'}{7} + \frac{ru'}{14} + \frac{ccy'}{5} + \frac{ar'}{6} + \\ + \frac{lb'}{8} + \frac{ob'}{13} + \frac{uepr'}{11} + \frac{uiae'}{6} + \frac{acsb'}{5} + \frac{md'}{7} + \frac{rusb'}{6} + \frac{anb'}{6} + \frac{mfb'}{8} + \frac{tst'}{6} + \\ + \frac{cf'}{10} + \frac{ig'}{8} + \frac{nrpd'}{2} + \frac{accb'}{2} + \frac{athc'}{1} + \frac{cex'}{4} + \frac{io'}{5} + \frac{ab'}{11} + \frac{ib'}{4} + \frac{rb'}{3}.$$

де q'_{schr} – кількість підхарактеристик, які неможливо обчислити за наявних після доповнення атрибутів, числа в чисельниках дробів показують кількість відсутніх у специфікації атрибутів для певної підхарактеристики якості (після доповнення);

$$D'_{schr} = \frac{31 - q'_{schr}}{31} \qquad (3.6)$$

де D'_{schr} – числова оцінка достатності об'єму наявної після доповнення інформації щодо якості (атрибутів) у специфікації вимог для оцінювання якості ПЗ за підхарактеристиками;

- за характеристиками:

$$q'_{chr} = \frac{fy\prime}{15} + \frac{ry\prime}{30} + \frac{ey\prime}{26} + \frac{uy\prime}{49} + \frac{my\prime}{33} + \frac{sy\prime}{23} + \frac{cy\prime}{9} + \frac{py\prime}{18} \quad (3.7)$$

де q_{chr} – кількість характеристик, які неможливо обчислити за наявних після доповнення (доповнень) атрибутів; числа в чисельниках дробів показують кількість відсутніх у специфікації атрибутів для певної характеристики якості (після доповнення);

$$D'_{chr} = \frac{8-q\prime_{chr}}{8} \quad (3.8)$$

де D_{chr} – числова оцінка достатності об'єму наявної після доповнення інформації щодо якості (атрибутів) у специфікації вимог для оцінювання якості ПЗ за характеристиками;

- обчислюється приріст достатності об'єму інформації щодо якості (атрибутів) у специфікації вимог до ПЗ (після доповнення КВІ та специфікації необхідними атрибутами):

за підхарактеристиками:

$$\Delta q_{schr} = q_{schr} - q'_{schr} = \left(\frac{fc}{4} - \frac{fc'}{4}\right) + \left(\frac{fcr}{5} - \frac{fcr'}{5}\right) + \left(\frac{rb}{4} - \frac{rb\prime}{4}\right) \quad (3.9)$$

де Δq_{schr} – кількість підхарактеристик, які стало можливим обчислити після доповнення (доповнень) специфікації атрибутами;

$$\Delta D_{schr} = D_{schr} - D'_{schr} = \frac{\Delta q_{schr}}{31} \quad (3.10)$$

де ΔD_{schr} – приріст достатності об’єму інформації щодо якості (атрибутів) у специфікації вимог (після доповнення специфікації необхідними атрибутами) для оцінювання якості ПЗ за підхарактеристиками;

- за характеристиками:

$$\Delta q_{chr} = q_{chr} - q'_{chr} = \left(\frac{fy}{15} - \frac{fy'}{15}\right) + \left(\frac{ry}{30} - \frac{ry'}{30}\right) + \left(\frac{py}{18} - \frac{py\prime}{18}\right) \quad (3.11)$$

де Δq_{chr} – кількість характеристик, які стало можливим обчислити після доповнення (доповнень) специфікації атрибутами

$$\Delta D_{chr} = D_{schr} - D'_{schr} = \frac{\Delta q_{schr}}{8} \quad (3.12)$$

де ΔD_{chr} – приріст достатності об’єму інформації щодо якості (атрибутів) у специфікації вимог (після доповнення специфікації необхідними атрибутами) для оцінювання якості ПЗ за характеристиками.

Представимо розроблений метод формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ у вигляді наступної схеми – рис. 3.1.

Методи оцінювання достатності інформації щодо якості (за стандартом ISO 25010) у специфікаціях вимог до ПЗ на основі онтологій

Список відсутніх атрибутів

Пошук правила для кожного відсутнього атрибута у підмножині {pr1,…,pr138}

Лічильники відсутніх атрибутів для визначення підхарактеристик та характеристик якості ПЗ

Аналіз отриманих лічильників за правилами {pr139, pr140}

Інформації достатньо?

\+

-

	Подальша робота над проектом	

Висновок про недостатність інформації. Список відсутніх атрибутів у рекомендованій пріоритетності доповнення

Оцінка достатності об’єму наявної інформації щодо якості у специфікації вимог

Оцінка приросту достатності об’єму інформаці щодо якості у специфікації вимог (після доповнення)

Доповнення КВІ та специфікації
вимог атрибутами

Рисунок 3.1 - Схема методу формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ

3.2 Методи та технологія оцінювання достатності інформації щодо якості у специфікаціях вимог до програмного забезпечення

З метою формування теоретичних засад інформаційної технології оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ проаналізуємо методологію оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ.

Методологію утворюють наступні складові частини:

- моделі предметної галузі «Інженерія ПЗ» (частина «Якість ПЗ»): теоретико-множинні моделі якості ПЗ на основі стандарту ISO; онтологічні моделі предметної галузі «Інженерія ПЗ» (частини «Якість ПЗ»; структурні та онтологічні моделі специфікації вимог до ПЗ (з точки зору наявності інформації щодо якості);

- моделі та методи оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ;

- моделі процесу оцінювання достатності інформації для визначення якості ПЗ на основі стандарту ISO 25010; методи оцінювання достатності інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ на основі онтологій; способи генерування та наповнення шаблону онтології для визначення якості конкретного ПЗ на основі стандарту ISO 25010; методи формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010) у специфікаціях вимог до ПЗ.

Інтеграція моделей та методів у методологію оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ забезпечує наступні результати:

1) висновок про достатність або недостатність інформації щодо якості у специфікації вимог до ПЗ;

2) висновок про необхідність додавання у КВІ інформації щодо вимог, які регламентують характеристики якості, в разі недостатності у специфікації вимог інформації щодо якості;

3) висновок про необхідність доповнення специфікації вимог до ПЗ в разі недостатності у ній інформації щодо якості;

4) висновок про пріоритетність доповнення КВІ та специфікації вимог до ПЗ інформацією щодо якості в разі недостатності інформації щодо якості;

5) числову оцінку достатності об’єму наявної у специфікації вимог інформації щодо якості; 6) числову оцінку прогнозованої якості ПЗ, яке розробляється за специфікацією вимог;

7) висновок щодо очікуваного рівня якості ПЗ, яке розробляється за специфікацією вимог.

На основі моделювання руху інформаційних потоків при формуванні специфікації вимог до ПЗ та дослідження інформаційних потоків в процесі оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ, а також, враховуючи теоретичні засади інформаційної технології оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ, проаналізуємо інформаційну технологію оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ [74] (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 [130]).

Основу інформаційної технології оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ складають [100]:

- способи генерування та наповнення шаблону онтології для визначення якості конкретного ПЗ на основі стандарту ISO 25010;

- методи оцінювання достатності інформації щодо якості (за стандартом ISO 25010) у специфікаціях вимог до ПЗ на основі онтологій;

- методи формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010) у специфікаціях вимог до ПЗ;

- метод оцінювання результатів проектування та прогнозування характеристик ПЗ на основі ШНМ, а також підсистема оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ на основі порівняльного аналізу онтологій та підсистема оцінювання, які базуються на вищезазначених методах.

Інформаційна технологія оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ забезпечує [100]:

- підтримку процесу оцінювання якості ПЗ на ранніх етапах життєвого циклу на основі оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ;

- опрацювання інформації щодо якості у специфікаціях вимог до ПЗ програмними агентами (ботами), без участі фахівців, що забезпечує можливість автоматизації таких процесів та усунення суб'єктивного впливу фахівців, а також збережуваність цієї інформації у софтверній компанії у випадку звільнення фахівця;

- висновок про достатність інформації щодо якості у КВІ та специфікаціях вимог до ПЗ;

- пріоритетність доповнення КВІ та специфікації вимог необхідною інформацією (в разі недостатності інформації) шляхом формування запиту щодо доповнення бізнес-вимог до ПЗ;

- числову оцінку достатності об'єму наявної у специфікації вимог інформації щодо якості;

- числову оцінку приросту достатності об'єму інформації щодо якості у специфікації вимог після її доповнення;

- висновок щодо очікуваного рівня якості ПЗ на основі опрацювання метричної інформації.

При реалізації зазначених підсистем слід максимально автоматизувати процеси оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ з метою мінімізації впливу людського фактору та з метою спрощення виконання зазначеного оцінювання як розробником, так і замовником.

На вхід підсистеми оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ на основі порівняльного аналізу онтологій подаються множини: 1) $\{qms_1,...qms_{nm}\}$ (nm =38) наявних у специфікації вимог до ПЗ вищеописаних атрибутів якості, необхідних для виконання оцінювання підхарактеристик та характеристик якості ПЗ; 2) $\{sqcxi_1,..., sqcxi_{nqcxi}\}$ (nqcxi = 42) наявних у специфікації вимог до ПЗ вищеописаних показників, необхідних для виконання метричного аналізу.

Результатом роботи такої підсистеми є:

- висновок про достатність інформації щодо якості (за стандартом ISO 25010) у КВІ та специфікації вимог до конкретного ПЗ;
- рекомендації щодо необхідності та пріоритетності доповнення КВІ та специфікації вимог до ПЗ атрибутами якості (в разі недостатності атрибутів);
- оцінка достатності об’єму наявної у специфікації вимог інформації щодо якості (атрибутів);
- висновок про достатність інформації щодо якості (для метричного аналізу) у КВІ та специфікації вимог до конкретного ПЗ;
- рекомендації щодо необхідності та пріоритетності доповнення КВІ та специфікації вимог до ПЗ показниками (в разі недостатності показників);
- оцінка достатності об’єму наявної у специфікації вимог інформації щодо якості (показників).

Підсистема оцінювання оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ на основі порівняльного аналізу онтологій складається з наступних компонентів [100]:

- модуль введення атрибутів специфікації вимог до ПЗ – є складовою частиною інтерфейсу користувача; зчитує інформацію користувача щодо наявних значень атрибутів $\{qms_1,...qms_{nm}\}$ (nm =138) специфікації вимог до ПЗ; користувачу пропонується список всіх 138 атрибутів якості у базовій онтології, з яких він видаляє атрибути, відсутні у аналізованій специфікації, формуючи онтологію для визначення якості конкретного ПЗ;

- модуль введення показників специфікації вимог до ПЗ – є складовою частиною інтерфейсу користувача; зчитує інформацію користувача щодо наявних значень показників $\{sqcxi_1,..., sqcxi_{nqcxi}\}$, (nqcxi = 42) специфікації вимог до ПЗ; користувачу пропонується список всіх 42 показників у базовій онтології, з яких він видаляє показники, відсутні у аналізованій специфікації;

модуль електронної підтримки користувача – є складовою частиною інтерфейсу користувача; надає користувачу інформацію про: структуру специфікації вимог до ПЗ, які є шаблонами специфікації вимог до ПЗ з точки зору наявності атрибутів та показників якості);

- атрибути, необхідні для оцінювання якості ПЗ за стандартом ISO 25010;

- модуль оцінювання достатності інформації у специфікації вимог до ПЗ для визначення якості ПЗ за стандартом ISO 25010 – виконується генерування та наповнення шаблону онтології для визначення якості конкретного ПЗ, враховуючи введені у модулі введення атрибутів наявні атрибути $\{qms_1,...qms_{nm}\}$ (nm =138), згідно зі способом генерування та наповнення шаблону онтології для визначення якості конкретного ПЗ.

Виконується порівняльний аналіз онтології для визначення якості конкретного ПЗ із базовою онтологією предметної галузі «Інженерія програмного забезпечення» (частина «Якість ПЗ»), результатом якого список

відсутніх у специфікації атрибутів. За методом формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010:2011) специфікаціях вимог до ПЗ виконується опрацювання результатів порівняльного аналізу онтології для конкретного ПЗ з базовою онтологією. Якщо при порівняльному аналізі онтологій не виявлено розбіжностей (список відсутніх атрибутів є порожнім), то інформації щодо якості у КВІ та специфікації достатньо для визначення якості ПЗ за стандартом ISO 25010, тоді блок виведення оцінки достатності інформації специфікації вимог до ПЗ для визначення якості ПЗ за стандартом ISO 25010 видає користувачу підсистеми висновок «Інформації КВІ та розглядуваної специфікації достатньо для визначення якості ПЗ за стандартом ISO 25010».

Якщо ж при порівняльному аналізі онтології встановлено розбіжності, то у КВІ та специфікації недостатньо атрибутів для визначення певних підхарактеристик та характеристик якості ПЗ, тоді блок виведення оцінки достатності інформації специфікації вимог до ПЗ для визначення якості ПЗ за стандартом ISO 25010 видає користувачу підсистеми висновок «Інформації КВІ та розглядуваної специфікації недостатньо для визначення якості ПЗ за стандартом ISO 25010», після чого відбувається порівняльний аналіз онтології для визначення якості конкретного ПЗ із зваженою базовою онтологією предметної галузі «Інженерія програмного забезпечення» (частина «Якість ПЗ»).

Далі виконується пошук продукційного правила (у продукційних правилах формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ, які містяться в базі знань) для кожного відсутнього специфікації вимог атрибута, згідно з якими підсистема виводить користувачу висновок щодо підхарактеристик та характеристик якості ПЗ, для визначення яких у специфікації недостатньо інформації, а також сортування всіх відсутніх атрибутів специфікації за спаданням значень вагових коефіцієнтів, тобто

встановлюється рекомендована пріоритетність їх доповнення у КВІ та специфікацію вимог до ПЗ.

За допомогою блоку виведення рекомендацій щодо необхідності та пріоритетності доповнення специфікації вимог до ПЗ атрибутами для визначення якості ПЗ за стандартом ISO 25010, сформовані рекомендації видаються користувачу підсистеми у вигляді висновку «Для підвищення достатності об’єму інформації щодо якості у специфікації вимог рекомендовано доповнити КВІ та специфікацію вимог атрибутами у такій послідовності:», і далі слідує список всіх відсутніх атрибутів згідно з встановленою пріоритетністю доповнення їх у специфікацію.

Крім цього, підсистема на основі результатів виконання певних правил, згідно з етапом 4 методу формування логічного висновку про достатність інформації щодо якості (за стандартом ISO 25010:2011) у специфікаціях вимог до ПЗ, виконує кількісну оцінку достатності об’єму наявної у специфікації вимог інформації (атрибутів) для оцінювання якості ПЗ за підхарактеристиками і характеристиками та виводить користувачу висновок вигляду: «Достатність об’єму наявної у специфікації вимог інформації (атрибутів) для оцінювання якості ПЗ за стандартом ISO 25010 становить: », і далі слідує кількісна оцінка достатності.

Таким чином, описана інформаційна технологія оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ надає замовнику інформацію для вибору специфікації вимог до ПЗ, забезпечує можливість порівняти між собою різні версії специфікацій, тобто є основою для прийняття мотивованого та обґрунтованого рішення щодо вибору специфікації вимог до ПЗ з урахуванням достатності інформації щодо якості у специфікації, а також очікуваного рівня якості ПЗ, що розробляється за специфікацією.

Розглянута інформаційна технологія, за рахунок можливості доповнення специфікації вимог необхідною інформацією, дає можливість підвищити достатність об’єму наявної інформації щодо якості у специфікації

вимог. Такий підхід дозволяє уникнути перевитрат коштів та виконати вимоги замовників щодо якості ПЗ.

3.3 Оцінювання достатності інформації специфікацій вимог для визначення якості програмного забезпечення за стандартом ISO 25010:2011

З метою оцінювання достатності інформації специфікацій вимог для визначення якості програмного забезпечення за стандартом ISO 25010:2011 на прикладі інформаційної системи торгівельного центру було проаналізовано специфікацію вимог до ПЗ останнього.

Визначено атрибути якості ПЗ, які наявні в конкретній специфікації інформаційної системи торгівельного центру. На основі способу генерування та наповнення шаблону онтології для визначення якості конкретного програмного забезпечення розроблено онтологію для визначення якості конкретного програмного проекту, яку утворюють складові частини для: визначення функціональної придатності, сумісності, ефективності, можливості переносу, зручності використання, надійності, захищеності, супроводжуваності конкретного програмного проекту, а саме інформаційної системи торгівельного центру (див. рис. А.1-А.8 Додатку А).

Порівняння розробленої онтології для програмного проекту з розроблення інформаційної системи торгівельного центру з фрагментами базової онтології для предметної галузі «Інженерія ПЗ», частина «Якість ПЗ» дало можливість виявити, що у розробленій онтології конкретного програмного проекту відсутні шість атрибутів (рис. 3.2): «час роботи», «кількість збоїв», «кількість відмов», «кількість фіксованих відмов», «кількість контрольованих вимог», «взаємозамінність». Тоді множина відсутніх атрибутів може бути представлена наступним чином:

$$\{qms_{1_{IS}},...,qms_{6_{IS}}\} = \{"OperationTime","NumberOfFaults",$$
$$"NumberOfFailures","NumberOfResolvedFailures",$$
$$"NumberOfControllabilityRequirements","DataExchangeability"\}.$$

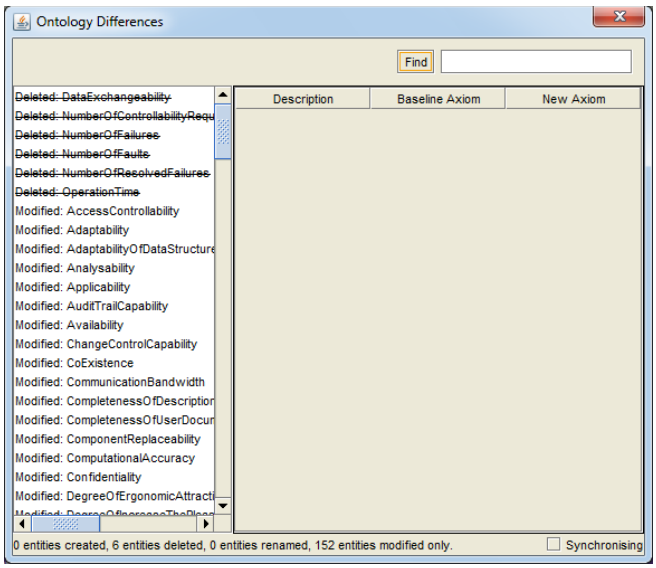

Рисунок 3.2 - Порівняння онтології для визначення якості інформаційної системи торгівельного центру із базовою онтологією предметної галузі «Інженерія програмного забезпечення (частина «Якість ПЗ»)»

За методом формування логічного висновку про достатність інформації специфікації вимог для визначення якості програмного забезпечення виконуємо пошук правила для кожного з елементів множини $\{qms_{1_{IS}},...,qms_{6_{IS}}\}$. Згідно з правил підраховуються відсутні у специфікації вимоги атрибутів для визначення підхарактеристик та характеристик якості ПЗ. Результати вищевказаного пошуку представлені у таблиці 3.1.

Згідно з правилами 139, встановлено, що у специфікації вимог до ПЗ інформаційної системи торгівельного центру недостатньо атрибутів для визначення певних підхарактеристик якості ПЗ, а саме для підхарактеристик: функційна коректність, функційна доцільність, зрілість, наявність, відмовостійкість, відновлюваність, поведінка у часі, поведінка ресурсів, можливість вивчення, керованість, модульність, аналізованість,

модифікованість, тестованість, конфіденційність, цілісність, співіснування, взаємодія, адаптованість, можливість інсталяції.

Таблиця 3.1 - Результати роботи методу формування логічного висновку

щодо достатності інформації специфікації вимог для визначення якості ПЗ

Відсутній атрибут	Номер правила	Результати методу
1. Час роботи	2	$fcr=1$, $fa=1$, $ma=1$, $av=1$, $rvb=1$, $tb=1$, $ru=1$, $lb=1$, $ob=1$, $md=1$, $mfb=1$, $tst=1$, $cf=1$, $ig=1$, $cex=1$, $io=1$, $ab=1$; $fy=2$, $ry=3$, $ey=2$, $uy=2$, $my=3$, $sy=2$, $cy=2$, mas[Operation Time]=17/138
2. Кількість збоїв	10	$ma=2$, $ab=2$, $ib=1$; $ry=4$, $py=3$; mas[Number Of Faults]=3/138
3. Кількість відмов)	11	$ma=3$, $ft=1$, $ru=2$, $md=2$, $anb=1$, $cex=2$; $ry=6$, $ey=3$, $my=5$, $cy=3$; mas[Number Of Failures]=6/138
4. Кількість фіксованих відмов	14	$ma=4$, $md=3$, $mfb=2$, $tst=2$; $ry=7$, $my=8$; mas[Number Of Resolved Failures]=4/138
5. Кількість контрольованих вимог	115	$cf=2$, $ig=2$; $sy=4$; mas[Number Of Controllability Requirements]=2/138
6. Взаємозамінність даних	127	$io=2$; $cy=4$; mas[Data Exchangeability]=1/138

Підхарактеристики, для визначення яких недостатньо наявних у специфікації атрибутів, обведені колом (див. рис. Б.1-Б.8 Додатку Б).

Згідно з першою частиною правила 140 визначено, що у специфікації вимог до ПЗ інформаційної системи торгівельного центру недостатньо атрибутів для всіх восьми характеристик якості ПЗ.

Отже, відсутність у специфікації вимог до ПЗ шести атрибутів призвела до неможливості обчислення 20-ти (з 31) підхарактеристик якості ПЗ, до неможливості обчислення всіх восьми характеристик якості ПЗ з високим рівнем достовірності та згідно з неможливості визначення якості проекту та розроблюваного ПЗ з високим рівнем достовірності. Тоді підсистема СОДІ інформаційної технології оцінювання достатності інформації для визначення

якості програмного забезпечення видасть користувачу висновок: «Інформації розглядуваної специфікації недостатньо для визначення якості ПЗ за стандартом ISO 25010».

Після встановлення факту недостатності інформації специфікації вимог до ПЗ інформаційної системи торгівельного центру для визначення якості програмного забезпечення за стандартом ISO 25010:2011, згідно з другої та третьої частин правила 140, проводиться сортування масиву mas за спаданням вагових коефіцієнтів відсутніх атрибутів та виведення індексів тих елементів відсортованого масиву mas, які мають значення, відмінне від нуля. Список відсутніх у специфікації атрибутів якості ПЗ за спаданням значень вагових коефіцієнтів представлено в табл. 3.2.

Таблиця 3.2 - Список відсутніх у специфікації атрибутів якості ПЗ за спаданням значень вагових коефіцієнтів

Відсутні індекси елементів масиву mas)	Значення елементів масиву mas
1. Час роботи	17/138
2. Кількість відмов	6/138
3. Кількість фіксованих відмов	4/138
4. Кількість збоїв	3/138
5. Кількість контрольованих вимог	2/138
6. Взаємозамінність даних	1/138

Аналізуючи результати, представлені в таблиці 3.2, визначено пріоритетність та порядок розгляду і доповнення атрибутів якості у специфікацію вимог до ПЗ інформаційної системи торгівельного центру.

Таким чином, підсистема СОДІ інформаційної технології оцінювання достатності інформації для визначення якості програмного забезпечення виведе користувачу висновок: «Для підвищення достовірності оцінювання якості ПЗ рекомендовано доповнити специфікацію вимог атрибутами у такій послідовності:

1) час роботи;

2) кількість відмов;

3) кількість фіксованих відмов;

4) кількість збоїв;

5) кількість контрольованих вимог;

6) взаємозамінність даних».

Наступним кроком є оцінювання достовірності оцінювання якості ПЗ на основі наявної у специфікації вимог інформації:

– за підхарактеристиками (за формулами (3.2) та (3.3)):

$$q_{schr_{IS}} = \frac{0}{4}+\frac{1}{5}+\frac{1}{6}+\frac{4}{14}+\frac{1}{4}+\frac{1}{5}+\frac{1}{7}+\frac{1}{7}+\frac{2}{14}+\frac{0}{5}+\frac{0}{6}+\frac{1}{8}+$$
$$+\frac{1}{13}+\frac{0}{11}+\frac{0}{6}+\frac{0}{5}+\frac{3}{7}+\frac{0}{6}+\frac{1}{6}+\frac{2}{8}+\frac{2}{6}+\frac{2}{10}+\frac{2}{8}+\frac{0}{2}+\frac{0}{2}+$$
$$+\frac{0}{1}+\frac{2}{4}+\frac{2}{5}+\frac{2}{11}+\frac{1}{4}+\frac{0}{3}=4,7,$$
$$D_{schr_{IS}} = \frac{31-4,7}{31}=0,85;$$

– за характеристиками (за формулами (3.4) та (3.5)):

$$q_{chr_{IS}} = \frac{2}{15}+\frac{7}{30}+\frac{3}{26}+\frac{2}{49}+\frac{8}{33}+\frac{4}{23}+\frac{4}{9}+\frac{3}{18}=1,54,$$
$$D_{chr_{IS}} = \frac{8-1,54}{8}=0,81.$$

Отже, підсистема СОДІ інформаційної технології оцінювання достатності інформації для визначення якості програмного забезпечення виведе користувачу висновок: «За наявних у специфікації вимог до ПЗ атрибутів, оцінити якість ПЗ за підхарактеристиками можна з точністю 85 %, а за характеристиками – з достовірністю 81 %».

Таким чином, здійснилося доповнення специфікації вимог до ПЗ. Після доповнення специфікації вимог знову розробляється онтологія (версію 2) для визначення якості конкретного програмного забезпечення. Порівняльний аналіз розробленої онтології (версія 2) для програмного проекту з розробки інформаційної системи торгівельного центру з фрагментами базової онтології предметної галузі «Інженерія ПЗ» (частина «Якість ПЗ») виявив, що специфікацію вимог до ПЗ було доповнено атрибутами «Час роботи» (перший

у сортованому списку), «Кількість збоїв (четвертий у сортованому списку), «Взаємозамінність даних» (шостий у сортованому списку), тоді множину відсутніх атрибутів запишемо у вигляді:

$$\{qms'_{1_{IS}},...,qms'_{3_{IS}}\} = \{"NumberOfFailures","NumberOf\,\mathrm{Re}solved\,Failures","NumberOfControllability\,\mathrm{Re}quirements"\}.$$

За методом формування логічного висновку про достатність інформації специфікації вимог для визначення якості програмного забезпечення знову виконується пошук правила для кожного з елементів множини $\{qms'_{1_{IS}},...,qms'_{3_{IS}}\}$ – результати пошуку представлені у таблиці 3.3.

Таблиця 3.3 - Список відсутніх у специфікації атрибутів якості ПЗ та результати роботи методу формування логічного висновку про достатність інформації специфікації вимог для визначення якості ПЗ

Відсутній атрибут	Номер правила	Результати методу (правила)
1. Кількість відмов	11	$ma'=1$, $ft'=1$, $ru'=1$, $md'=1$, $anb'=1$, $cex'=1$; $ry'=2$, $ey'=1$, $my'=2$, $cy'=1$; mas[Number Of Failures]=6/138
2. Кількість фіксованих відмов	14	$ma'=2$, $md'=2$, $mfb'=1$, $tst'=1$; $ry'=3$, $my'=5$; mas[Number Of Resolved Failures]=4/138
3. Кількість контрольованих вимог	115	$cf'=1$, $ig'=1$; $sy'=2$; mas[Number Of Controllability Requirements]=2/138

Згідно з правилом 139, встановлено, що у специфікації вимог до ПЗ інформаційної системи торгівельного центру все ще недостатньо атрибутів для визначення десяти підхарактеристик якості ПЗ (із зазначенням цих підхарактеристик), але порівняльний аналіз списку підхарактеристик, які все ще неможливо визначити після доповнення специфікації, із списком підхарактеристик, які не можна було визначити на основі початкової версії

специфікації вимог, показав, що з'явилась можливість визначення функціональної коректності, функціональної доцільності, наявності, відновлюваності, поведінки у часі, можливості вивчення, керованості, взаємодії, адаптованості, можливості інсталяції конкретного програмного проекту.

За першою частиною правила 140 було встановлено, що у специфікації вимог до ПЗ інформаційної системи торгівельного центру все ще недостатньо атрибутів для визначення п'яти з восьми характеристик якості ПЗ. Тому підистема СОДІ інформаційної технології оцінювання достатності інформації для визначення якості програмного забезпечення знову виведе користувачу висновок: «Інформації розглядуваної специфікації недостатньо для визначення якості ПЗ за стандартом ISO 25010».

Після встановлення факту недостатності інформації специфікації вимог до ПЗ інформаційної системи торгівельного центру для визначення якості програмного забезпечення за стандартом ISO 25010:2011 після її доповнення, згідно з другої та третьої частин правила 140, проводиться сортування масиву mas за спаданням значень елементів та виведення індексів тих елементів відсортованого масиву mas, які мають значення, відмінне від нуля (табл. 3.4).

Таблиця 3.4 - Список відсутніх у специфікації атрибутів якості ПЗ за спаданням значень вагових коефіцієнтів

Індекси елементів масиву mas	Значення елементів масиву mas
1. Кількість відмов	6/138
2. Кількість фіксованих відмов)	4/138
3. Кількість контрольованих вимог	2/138

Отже, підсистема СОДІ інформаційної технології оцінювання достатності інформації для визначення якості програмного забезпечення виведе користувачу висновок: «Для підвищення достовірності оцінювання

якості ПЗ рекомендовано доповнити специфікацію вимог атрибутами у такій послідовності:

1) кількість відмов;

2) кількість фіксованих відмов;

3) кількість контрольованих вимог».

Далі оцінюється ймовірність оцінювання якості ПЗ на основі наявної у специфікації вимог інформації:

– за підхарактеристиками (за формулами (3.6) та (3.7)):

$$q'_{schr_{IS}} = \frac{0}{4}+\frac{0}{5}+\frac{0}{6}+\frac{2}{14}+\frac{0}{4}+\frac{1}{5}+\frac{0}{7}+\frac{0}{7}+\frac{1}{14}+\frac{0}{5}+\frac{0}{6}+\frac{0}{8}+\frac{0}{13}+\frac{0}{11}+\frac{0}{6}+$$
$$+\frac{0}{5}+\frac{2}{7}+\frac{0}{6}+\frac{1}{6}+\frac{1}{8}+\frac{1}{6}+\frac{1}{10}+\frac{1}{8}+\frac{0}{2}+\frac{0}{2}+\frac{0}{1}+\frac{1}{4}+\frac{0}{5}+\frac{0}{11}+\frac{0}{4}+\frac{0}{3}=1,63,$$

$$D'_{schr_{IS}} = \frac{31-1,63}{31}=0,95;$$

– за характеристиками (за формулами (3.8) та (3.9)):

$$q'_{chr_{IS}} = \frac{0}{15}+\frac{3}{30}+\frac{1}{26}+\frac{0}{49}+\frac{5}{33}+\frac{2}{23}+\frac{1}{9}+\frac{0}{18}=0,49,$$

$$D'_{chr_{IS}} = \frac{8-0,49}{8}=0,94.$$

Тоді підсистема СОДІ інформаційної технології оцінювання достатності інформації для визначення якості програмного забезпечення виведе користувачу висновок: «За наявних у специфікації вимог до ПЗ атрибутів, оцінити якість ПЗ за підхарактеристиками можна з достовірністю 95 %, а за характеристиками – з достовірністю 94 %».

Приріст достовірності оцінювання якості ПЗ після доповнення специфікації вимог трьома необхідними атрибутами складає:

– за підхарактеристиками (за формулами (3.10) та (3.11)):

$$\Delta q_{schr_{IS}} = q_{schr_{IS}} - q'_{schr_{IS}} = 4{,}7 - 1{,}63 = 3{,}07,$$

$$\Delta D_{schr_{IS}} = D'_{schr_{IS}} - D_{schr_{IS}} = 0{,}95 - 0{,}85 = 0{,}1,$$

$$\Delta D_{schr_{IS}} = \frac{\Delta q_{schr_{IS}}}{31} = \frac{3{,}07}{31} = 0{,}1;$$

– за характеристиками (за формулами (3.12) та (3.13)):

$$\Delta q_{chr_{IS}} = q_{chr_{IS}} - q'_{chr_{IS}} = 1{,}54 - 0{,}49 = 1{,}05;$$

$$\Delta D_{chr_{IS}} = D'_{chr_{IS}} - D_{chr_{IS}} = 0{,}94 - 0{,}81 = 0{,}13;$$

$$\Delta D_{chr_{IS}} = \frac{\Delta q_{chr_{IS}}}{8} = \frac{1{,}05}{8} = 0{,}13.$$

Процес доповнення специфікації вимог є ітераційним і продовжується до тих пір, поки не буде визначено всі підхарактеристики та характеристики якості або доки не буде сформовано висновок, щодо недостатності даних для визначення якості ПЗ.

Замовник розробленої інформаційної системи торгівельного центру прийняв рішення, що подальше доповнення специфікації економічно недоцільне, тому було сформовано висновок про недостатність даних для визначення якості ПЗ, а за наявних (після доповнення) у специфікації атрибутів оцінити якість ПЗ за підхарактеристиками можна з ймовірністю 95 %, а за характеристиками – з достовірністю 94 %. Оскільки зазначена ймовірність оцінок якості ПЗ влаштувала замовника, то процес доопрацювання специфікації вимог було припинено.

Висновок до розділу 3

Описана інформаційна технологія оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ надає замовнику інформацію для вибору специфікації вимог до ПЗ, забезпечує можливість порівняти між собою різні версії специфікацій, тобто є основою для прийняття мотивованого та обґрунтованого рішення щодо вибору специфікації вимог до ПЗ з урахуванням достатності інформації щодо якості у специфікації.

Визначено, що використання інформаційної технології оцінювання достатності інформації специфікації вимог для визначення якості ПЗ навіть після одного доопрацювання специфікації вимог дало можливість підвищити ймовірність оцінювання якості ПЗ на 13 % за характеристиками для інформаційної системи торгівельного центру, але при цьому було зменшено розмір розриву у знаннях при оцінюванні якості програмного забезпечення.

ВИСНОВКИ

У першому розділі проаналізовано метод оцінки якості (згідно ISO 25010: 2011) у специфікаціях програмного забезпечення онтології, який відрізняється від відомих тим, що дає можливість оцінити адекватність інформації (атрибутів) у специфікаціях програмних вимог для порівняльний онтологічний аналіз. визначення певних субхарактеристик та характеристик якості програмного забезпечення та формулювання висновку про необхідність доповнення CVI та специфікацій атрибутами.

Розроблено метод оцінювання адекватності інформації про якість (згідно ISO 25010: 2011) у специфікаціях програмних вимог на основі зваженої онтології, який відрізняється від відомих тим, що шляхом позначення атрибутичних вагових коефіцієнтів у базовій онтології можна розібратися в усіх технічних характеристиках, відсутніх у специфікаціях. Програмне забезпечення приписує якість шляхом зменшення значень ваг, тобто встановлення пріоритету їх доповнення в КВІ та конкретизації вимог.

Розглянуті методи оцінки адекватності інформації про якість (згідно ISO 25010: 2011) у специфікаціях програмного забезпечення, заснованих на онтології, завдяки їх здатності доповнювати CVI та технічні характеристики необхідною інформацією дозволяють зменшити розрив у знаннях та сектор з невідомою інформацією про програмну систему та підвищити адекватність інформації про якість в специфікації вимог до програмного забезпечення.

Проаналізовано метод формування логічного висновку про достатність інформації про якість (згідно ISO 25010: 2011) у специфікаціях програмних вимог, який на основі запропонованих правил виробництва формує логічний висновок про якість якість інформації (згідно ISO 25010: 2011).

Специфікація вимог до програмного забезпечення дає можливість: сформувати висновок про достатність або недостатність інформації про якість (атрибути) в КВІ та конкретизацію вимог до програмного забезпечення; у разі недостатньої інформації робити висновки,

Щоб визначити, які підхарактеристики та характеристики якості не є достатньою інформацією, сформуйте відсортований (за ваговими коефіцієнтами) список відсутніх атрибутів як рекомендованого пріоритету атрибутів у КВІ та конкретизацію вимог до програмного забезпечення та оцініть достатність наявної інформації про якість (атрибути) в специфікації до програмного забезпечення (до та після оновлення) та визначити приріст кількості інформації (атрибутів) в специфікації вимог до програмного забезпечення (після оновлення).

Для оцінки відповідності інформації специфікаціям вимог до визначення якості програмного забезпечення відповідно до стандарту ISO 25010: 2011 специфікація програмних вимог останнього була проаналізована на прикладі інформаційного центру торгового центру.

Визначаються атрибути якості програмного забезпечення, які доступні в конкретних специфікаціях інформаційного центру торгового центру. На основі методу генерування та заповнення шаблону онтології для визначення якості конкретного програмного забезпечення було розроблено онтологію для визначення якості конкретного програмного проекту, що складається з компонентів для: визначення функціональної придатності, сумісності, ефективності, трансферність, зручність використання, надійність, безпека, наявність конкретного програмного проекту, а саме інформаційної системи торгового центру.

Замовник розробленого інформаційного центру торгового центру вирішив, що подальше оновлення специфікації не є економічно доцільним, тому було зроблено висновок щодо недостатності даних для визначення якості програмного забезпечення.

За характеристиками - з достовірністю 94%. Оскільки ця ймовірність оцінки якості програмного забезпечення задовольнила замовника, процес уточнення специфікації вимог припинився.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO 10002:2014. Quality management. Customer satisfaction. Guidelines for complaints handling in organizations. [Introduced 15.07.2014]. Geneva (Switzerland), 2014. 26 p. (International standard).
2. ISO 19011:2011. Guidelines for auditing management systems. [Introduced 15.11.2011]. Geneva (Switzerland), 2011. 44 p. (International standard).
3. ISO/IEC 15504-2:2003. Information technology. Process assessment. Part Performing an assessment. [Introduced 15.10.2003; revised by ISO/IEC 33002:2015]. Geneva (Switzerland), 2003. 16 p. (International standard).
4. ISO/IEC 25000 series of standards. Web-site. URL: http://iso25000.com/index.php/en/iso-25000-standards (Last accessed: December 7, 2017).
5. ISO/IEC 33002:2015. Information technology. Process assessment. Requirements for performing process assessment. [Introduced 01.03.2015]. Geneva (Switzerland), 2015. 16 p. (International standard).
6. ISO/IEC 9126-1:2001. Software engineering. Product quality. Part 1: Quality model. [Introduced 15.06.2001; revised by ISO/IEC 25010:2011]. Geneva (Switzerland), 2001. 32 p. (International standard).
7. ISO/IEC/IEEE CD 12207:2008. Systems and software engineering. Software life cycle processes. [Introduced 01.02.2008; revised by ISO/IEC/IEEE CD 12207:2016]. Geneva (Switzerland), 2008. 123 p. (International standard).
8. Андон Ф. И. Semantic Web как новая модель информационного пространства Интернет. Проблеми програмування. 2008. №2-3. Спеціальний випуск. С. 417-430.
9. Бабенко Л. П. Основи програмної інженерії: навч.посіб. для студ. вищ. навч. закл. Київ: Знання, 2001. 269 с.

10. Бачинський А. В. Оцінювання ефективності метрик складності програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2012. № 2. С. 171-179.

11. Бездиханюк В. В. Експертна система формування характеристик проекту програмного забезпечення для вибору метрик якості. Інтелектуальні технології в системному програмуванні: матеріали Всеукр.

12. Вендров А. М. Проектирование программного обеспечения экономических информационных систем: учебник. Москва: Финансы и статистика, 2006. 544 с.

13. Винничук Р. О. Особливості розвитку ІТ-ринку в Україні: стан та тенденції. Вісник Національного університету «Львівська політехніка». Серія «Логістика». 2015. № 833. С. 3-8.

14. Вовк І. В. Аналіз інцидентів, спричинених помилками програмного забезпечення. Інтелектуальні технології в системному програмуванні: матеріали Всеукр. наук.-практ. конф. молодих вчених та студентів (Хмельницький, 18-19 квітня 2013 р.). Хмельницький, 2013. С. 187-

15. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. Санкт-Петербург: Питер, 2001. 384 с.

16. Глухих М., Ицыксон В. М. Программная инженерия. Обеспечение качества программных средств методами статического анализа: учебное пособие. Санкт-Петербург: Изд. политехн. ун-та, 2011. 150 с.

17. Глушков В. Введение в кибернетику. Киев: Издательство АН УССР, 1964. 324 с.

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

19. Говорущенко Т. О. Дослідження результатів виявлення помилок програмного забезпечення на різних етапах життєвого циклу. Системний аналіз та інформаційні технології : тези доповідей міжнар. наук.-техн. конф. (Київ, 27- травня 2013 р.). Київ, 2013. С. 411-412.

20. Говорущенко Т. О. Емерджентні властивості програмного забезпечення. Теоретичні та прикладні аспекти побудови програмних систем: матеріали міжнар. наук.-практ. конф. (Київ, 23-26 листопада 2015 р.). Київ, 2015. С. 48-52.

21. Говорущенко Т. О. Інформаційна технологія оцінювання достатності інформації для визначення якості програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2017. № 3. 186-195.

22. Говорущенко Т. О. Характеристики та показники якості програмного забезпечення: аналіз, тенденції, проблеми. Фізико-технологічні проблеми радіотехнічних пристроїв, засобів телекомунікацій, нано- та мікроелектроніки: тези доповідей міжнар. наук.-практ. конф. (Чернівці, 24-26 жовтня 2013 р.). Чернівці, 2013. С. 134-136.

23. Даревич Р.О. Підвищення точності пошуку текстових документів на основі адаптивної онтології. Комп'ютинг. 2007. Т. 6. Вип. 1. С. 51-58.

24. Досин Д.В. Інтелектуальні системи, базовані на онтологіях: монографія. Львів : Цивілізація, 2009. 414 с.

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

26. Дюваль П.Э. Непрерывная интеграция. Улучшение качества программного обеспечения и снижение риска. Москва: Вильямс, 2008. 240 с.

27. Загоруйко Л. І. Аналіз впливу типу проекту на вибір метрик складності та якості програмного забезпечення на етапі проектування. Інтелектуальні технології в системному програмуванні: матеріали конференції наук.-практ. конф. молодих вчених та студентів (Хмельницький, 26-квітня 2012 р.). Хмельницький, 2012. С. 176-183.

28. Зыль С. Проектирование, разработка и анализ программного обеспечения систем реального времени. Санкт-Петербург: БХВ-Петербург, 2010. 200 с.

29. Инвариантно-ориентированная оценка качества программного обеспечения космических систем: монография / Конорев Б. М. и др. ; под ред. М. Конорева, В. С. Харченка. Харьков: НАУ «ХАИ», 2009. 224 с.

30. Інтелектуальні системи підтримки прийняття рішень на основі адаптивних онтологій / В. Литвин та інші. Штучний інтелект. 2011. № 2. С. 35–44.

31. Клейн Т. Дневник охотника за ошибками. Путешествие через джунгли проблем безопасности программного обеспечения. Москва: ДМК Пресс, 2013. 240 с.

32. Козак М. В. Реалізація та дослідження нейромережної складової методу оцінювання та прогнозування якості програмного забезпечення. Науковий вісник Чернівецького національного університету. Серія «Комп'ютерні системи та компоненти». 2011. Т. 2. Вип. 1. - 19-26.

33. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория практика. Москва: Горячая линия-Телеком, 2001. 382 с.

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

35. Лавріщева К. М. Програмна інженерія: підручник. Київ: Академперіодика, 2008. 320 с.

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

37. Литвин В. Моделювання інтелектуальних систем підтримки прийняття рішень з використанням онтологічного підходу. Радіоелектроніка, інформатика, управління. 2011. № 2. С. 93-101.

38. Луценко Е. В. Существование, несуществование и изменение как эмерджентные свойства систем. Квантовая магия. 2008. Т. 5 Вып. 1. С. 1215-1239.

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

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

41. Медведев В. С., Потемкин В. Г. Нейронные сети Matlab 6. Москва: Диалог-Мифи, 2002. 496 с.

42. Мельник А.М. Метод класифікації текстових документів із використанням онтологічного підходу. Вісник Тернопільського національного технічного університету. 2011. Т. 17. № 2. С. 208-215.

43. Месарович М., Такахара Я. Общая теория систем: математические основы. Москва: Мир, 1978. 344 с.

44. Минухин С.В.Формирование информационного обеспечения системы управления бизнес-процессами предприятия. Актуальні проблеми економіки. 2006. № 10. С. 170–178.

45. Мищенко В. О. CASE-оценка критических программных систем: в 3 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.

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

47. Онищук О. С. Оцінювання результатів проектування та прогнозування характеристик якості програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2011. № 2. С. 168-178.

48. Оценка и обеспечение качества программных средств космических систем: монография / Харченко В. С. и др. ; под ред. В. С.

Харченко, М. Конорева. Харьков: Нац. косм. агентство Украины, Гос. центр регулирования качества, НАУ «ХАИ», 2007. 244 с.

49. Павлова О. О. Метод оцінювання достатності інформації для визначення якості програмного забезпечення на основі зваженої онтології. Вісник Хмельницького національного університету. Серія «Технічні науки». 2016. № 5. С. 146-156.

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

51. Питлик Є. В. Визначення ефективності метрик якості на етапі проектування програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2012. № 2. С. 149-155.

52. Поморова О. В. Дослідження засобів оцінки якості на різних етапах життєвого циклу програмного забезпечення. Вісник Національного університету «Львівська політехніка». Серія «Комп'ютерні системи та мережі». 2011. № 717. С. 141-146.

53. Поморова О. В. Моделювання процесу оцінювання достатності інформації щодо якості у специфікаціях вимог до програмного забезпечення. Вісник Хмельницького національного університету. Серія «Технічні науки». 2017. № 6. С. 70-80.

54. Поморова О. В. Проблеми галузі забезпечення якості програмних продуктів. Інтелектуальні технології в системному програмуванні: матеріали Всеукр. наук.-практ. конф. молодих вчених та студентів (Хмельницький, 23-25 квітня 2014 р.). Хмельницький, 2014. С. 390-391.

55. Поморова О. В., Говорущенко Т. О. Дослідження характеристик навчальної вибірки для штучної нейронної мережі оцінювання якості програмного забезпечення. Поступ в науку. Збірник наукових праць Бучацького інституту менеджменту і аудиту. 2012. № 8. С. 147-152.

56. Поморова О. В., Говорущенко Т. О. Інтелектуальний метод оцінювання результатів проектування та прогнозування характеристик якості

програмного забезпечення. Радіоелектронні і комп'ютерні системи. 2010. № 6. С. 211-218.

57. Поморова О. В.Інтелектуальна підтримка процесу оцінювання і прогнозування складності та якості програмного забезпечення. Вісник Національного університету «Львівська політехніка». Серія «Комп'ютерні системи та мережі». 2011. № 717. С. 134-141.

58. Поморова О.О. Сучасні проблеми оцінювання якості програмного забезпечення. Радіоелектронні і комп'ютерні системи. 2013. №5. С. 319-327.

59. Попов В. П., Крайнюченко И. В. Глобальный эволюционизм и синергетика ноосферы. – Ростов-на-Дону: Издательство АПСН СКНЦ ВШ, 2003. 333 с.

60. Саттон М., Грин А., Амини П. Fuzzing: исследование уязвимостей методом грубой силы. Санкт-Петербург: Символ-Плюс, 2009. 560 с.

61. Сеньківський В. М. Синтез моделей пріоритетного впливу факторів на якість процесу створення програмного забезпечення мобільних пристроїв. Поліграфія і видавнича справа. 2016. №1. С. 67-78.

62. Стасевич А. Приклад написання функціональних вимог до Enterprise-системи. Веб-сайт. URL: http://it-ua.info/news/2014/12/11/priklad-napisannya-funkconalnih-vimog-do-enterprise-sistem.html (дата звернення 07.12.2017).

63. Тарасек С. Я. Аналіз та опрацювання метрик якості програмного забезпечення на етапі проектування. Вісник Хмельницького національного університету. Серія «Технічні науки». 2010. №1. С. 54-63.

64. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. Москва: Мир, 1992. 240 с.

65. Управление качеством. Дистанционный консалтинг. Веб-сайт. URL: http://www.dist-cons.ru/modules/qualmanage/index.html .

66. 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.

67. Ховард М., Лебланк Д., Вьега Дж. 24 смертных греха компьютерной безопасности. Санкт-Петербург: Питер, 2010. 400 с.

68. Ховард М., Лебланк Д., Вьега Дж. Уязвимости в программном коде и борьба с ними. Москва: ДМК-Пресс, 2011. 288 с.

69. Черна Т. Використання адаптивних онтологій в задачі квазіреферування текстових документів. Автоматизированные системы управленя и приборы автоматики. 2014. № 166. С. 27-32.

70. Черников Б. В., Поклонов Б. Е. Оценка качества программного обеспечения: практикум, учебное пособие / под ред. Б. В. Черникова. Москва: ИД ФОРУМ: НИЦ Инфра-М, 2013. 400 с.

71. 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.

72. Яковина В.С. Якість програмного забезпечення. Інженерія програмного забезпечення. 2010. № 2. С. 24-29.

ДОДАТКИ

Додаток А

Порівняння розробленої онтології для програмного проекту з розроблення інформаційної системи торгівельного центру з фрагментами базової онтології для предметної галузі «Інженерія ПЗ», частина «Якість ПЗ»

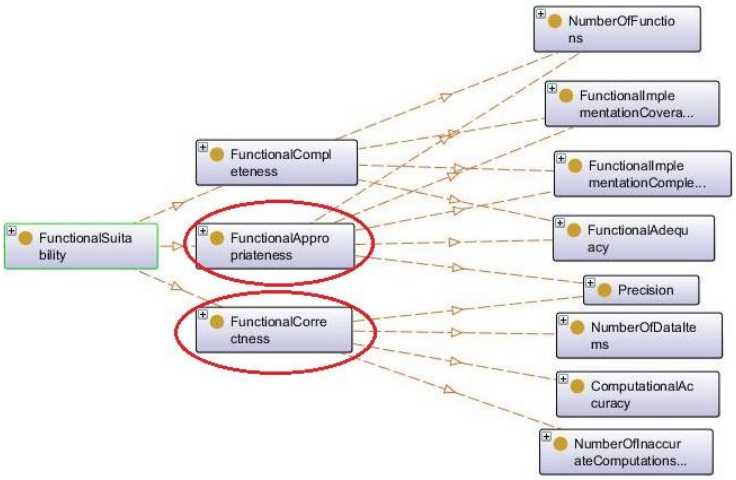

Рис. А.1. Функційна придатність інформаційної системи торгівельного центру

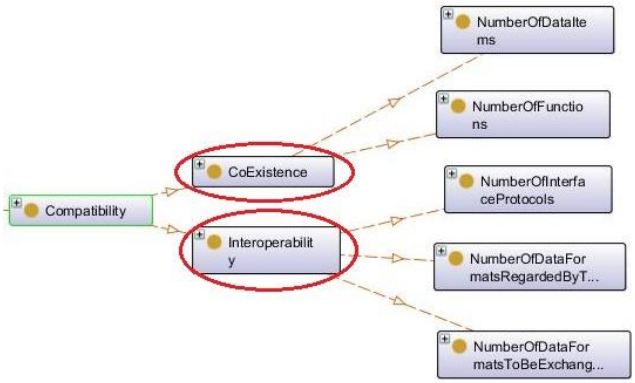

Рис. А.2. Сумісність інформаційної системи торгівельного центру

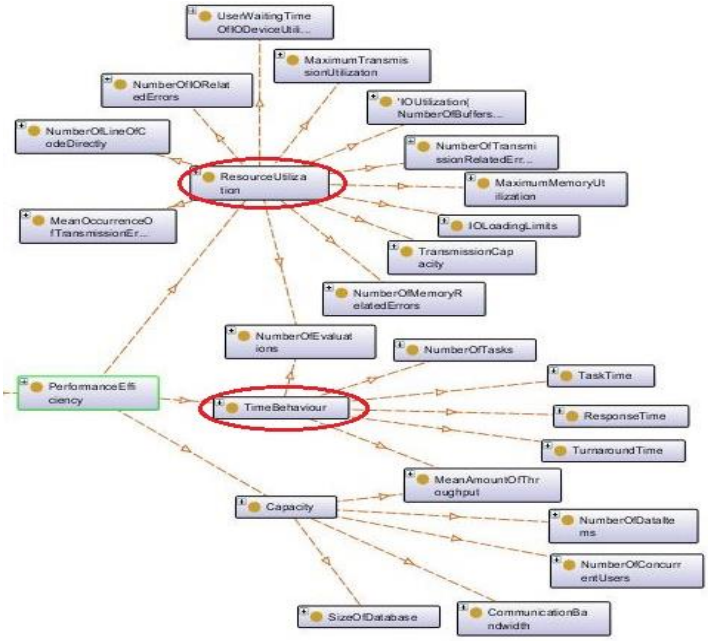

Рис. А.3. Ефективність інформаційної системи торгівельного центру

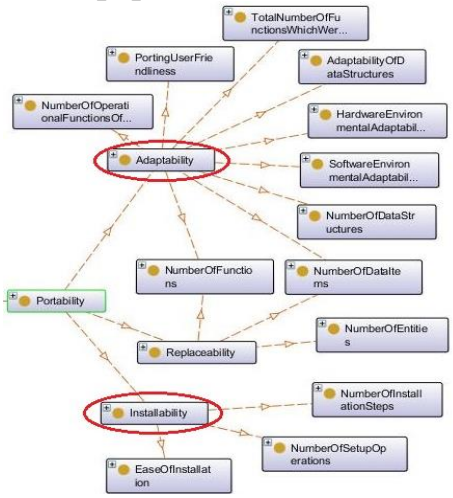

Рис. А.4. Складова онтології для Можливості переносу
інформаційної системи магазину
та складу запчастин для вантажних автомобілів

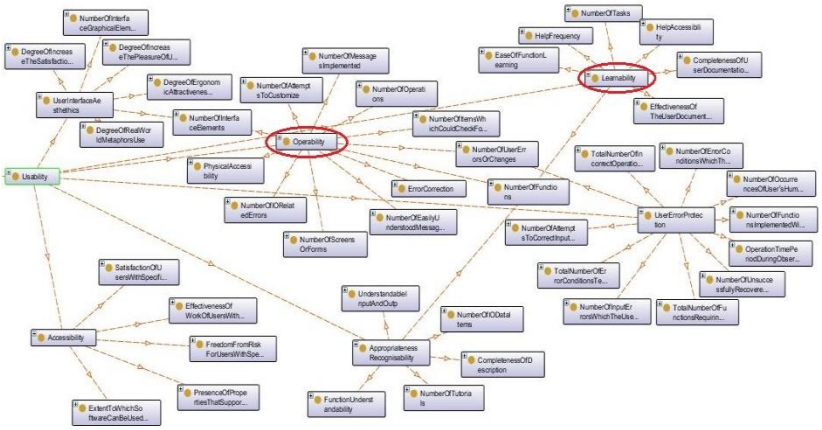

Рис. А.5. Зручність використання інформаційної торгівельного центру

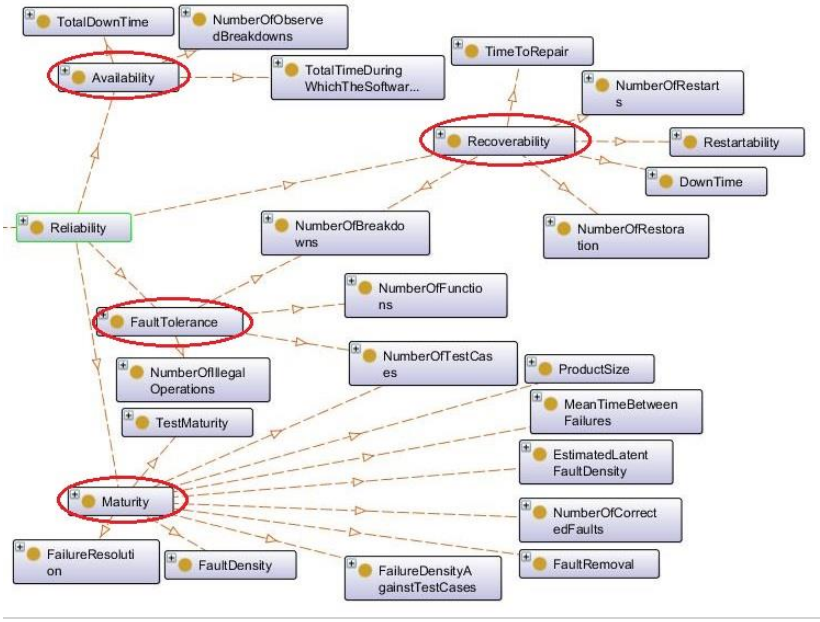

Рис. А.6. Надійність інформаційної системи торгівельного центру

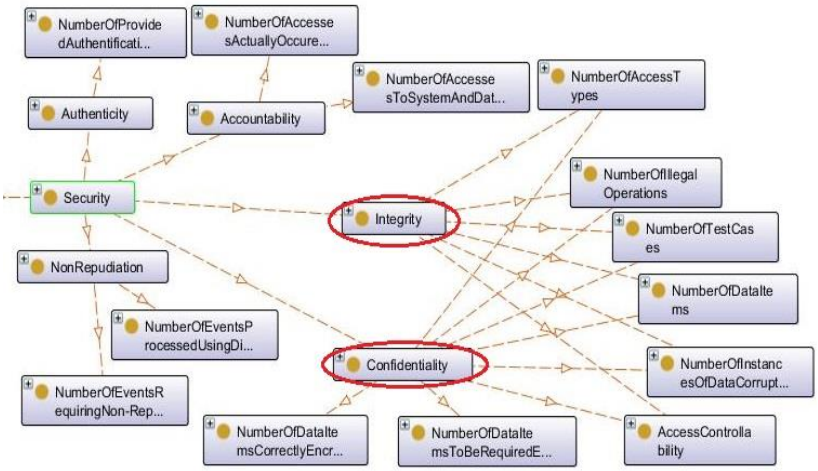

Рис А7. Захист інформаційної системи торгівельного центру

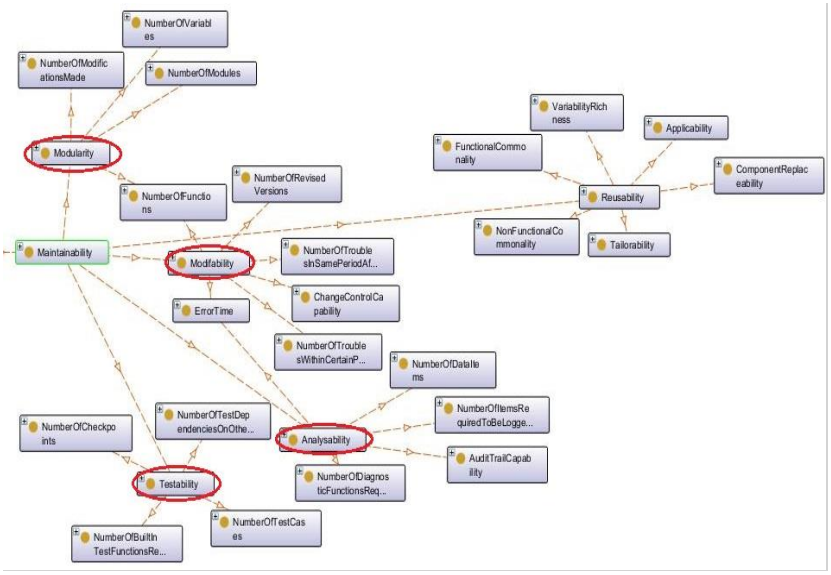

Рис.А.8. Супроводжуваність інформаційної системи торгівельного центру