Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Оцінювання якості програмних засобів на основі стандартів SQuaRE

(тема)

Виконав:

студент II курсу, групи СПм-22-4

Яценко І.С.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Запорожець О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Яценко Ігорю Станіславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Оцінювання якості програмних засобів на основі стандартів SQuaRE

затверджена наказом по університету від " 01 " квітня 2024 р. № 257Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи 1) Документація стандартів ISO/IEC 25000 (SQuaRE);
2) Відомості про існуючі інструменти та методи оцінювання якості програмних засобів;
3) Метрики та показники, визначені відповідно до стандартів SQuaRE;
4) Експертні оцінки та рекомендації щодо застосування стандартів SQuaRE;
5) Кейс-стаді на тему якості програмних засобів.

4. Перелік питань, що потрібно опрацювати у роботі
1) Основні принципи та історія розвитку SQuaRE;
2) Методологія оцінювання якості програмних засобів;
3) Вимірювання якості програмного продукту;
4) Перспективи та напрямки удосконалення SQuaRE;
5) Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація – 17 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Збір інформації та аналіз літератури по темі	02.04.2024-08.04.2024	
2	Визначення методології оцінювання якості	09.04.2024-16.04.2024	
	програмних засобів на основі стандартів		
	SQuaRE		
3	Розробка критеріїв та метрик оцінювання	17.04.2024-30.04.2024	
	якості програмного забезпечення		
4	Збір та обробка даних для оцінювання якості	01.05.2024-15.05.2024	
	якості програмних засобів		
5	Аналіз отриманих даних	15.05.2024-23.05.2024	
6	Оформлення матеріалів кваліфікаційної роботи	24.05.2024-13.06.2024	
7	Подання кваліфікаційної роботи керівникові	14.06.2024	
8	Подання кваліфікаційної роботи на рецензу-	15.06.2024	
	вання		

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Запорожець О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 61 с., 3 рис., 2 табл., 1 дод., 16 джерел.

ОЦІНЮВАННЯ ЯКОСТІ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СТАНДАРТИ, ОЦІНЮВАННЯ ПРОДУКТИВНОСТІ, МОДЕЛЬ ЯКОСТІ.

Мета роботи: вивчити рекомендації та вимоги стандартів SQuaRE щодо вимірювання якості програмного забезпечення та визначити, наскільки вони корисні.

У роботі були розглянуті моделі та стандарти SQuaRE щодо вимірювання якості. Було також розглянуто, що таке елемент показника якості програмного забезпечення, загальний підхід до вимірювання якості програмного забезпечення, еталонна модель вимірювання якості, вибір і створення показників якості, планування та виконання вимірювань, вимірювання якості програмного забезпечення.

ABSTRACT

Master’s thesis: 61 pages, 2 figures, 2 tables, 1 appendices, 16 sources.

SOFTWARE QUALITY, QUALITY ASSESSMENT, STANDARDS, PERFORMANCE ASSESSMENT, QUALITY MODEL.

The major goal of this thesis is to study the recommendations and requirements of SQuaRE standards for measuring software quality and to determine how useful they are.

The work considered SQuaRE models and standards for quality measurement. It also considered what a software quality indicator element is, a general approach to software quality measurement, a reference quality measurement model, selection and creation of quality indicators, planning and execution of measurements, and software quality measurement.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЕПЯ – елемент показника якості

Інструменти для оцінювання якості – програмні засоби для проведення оцінки якості програмного забезпечення

Кейс-стаді – конкретний випадок або приклад оцінювання якості програмного забезпечення

Метрики якості ПЗ – вимірювані характеристики якості програмного забезпечення

Модель якості ПЗ – концептуальна структура для оцінки якості програмного забезпечення

ПЗ – програмне забезпечення

ПЯ – показник якості

Рекомендації на основі аналізу – рекомендації, що базуються на результатів аналізу даних

Розвиток стандартів SQuaRE – майбутні напрями удосконалення стандартів оцінювання якості програмного забезпечення

CMMI – інтеграція моделі зрілості можливостей (англ., Capability Maturity Model Integration)

IEC – Міжнародна електротехнічна комісія (англ., International Electrotechnical Commission)

ISO – Міжнародна організація зі стандартизації (англ., International Organization for Standardization)

MTBF – середній час між відмовами (англ., Mean Time Between Failures)

MTTR – середній час відновлення (англ., Mean Time to Repair)

SQuaRE – вимоги до якості програмного забезпечення та оцінка (англ., Software Quality Requirements and Evaluation)

ВСТУП

У сучасному інформаційному суспільстві програмне забезпечення відіграє ключову роль у різних аспектах життя, від підтримки бізнес-процесів до особистого використання в повсякденному житті. Якість програмного забезпечення (ПЗ) безпосередньо впливає на його ефективність, надійність та задоволення потреб користувачів. Відповідно, оцінка якості ПЗ стає невід'ємною складовою процесу його розробки та експлуатації.

Один з визнаних міжнародних стандартів для оцінки якості програмного забезпечення є набір стандартів SQuaRE (Software Quality Requirements and Evaluation), розроблений Міжнародною організацією зі стандартизації (ISO). Стандарти SQuaRE визначають методології, моделі та метрики, які дозволяють об'єктивно оцінювати якість ПЗ в різних аспектах його функціонування.

Ця кваліфікаційна робота спрямована на ретельний аналіз стандартів SQuaRE, їх структуру та компоненти, а також на розгляд методології оцінювання якості ПЗ, зокрема процесу вибору метрик, збору та аналізу даних. Магістерська кваліфікаційна робота також включає порівняння різних підходів до оцінювання якості ПЗ з використанням стандартів SQuaRE та аналіз результатів, що дозволяє зробити висновки щодо їх ефективності та застосування у практиці.

На основі отриманих результатів надаються рекомендації для розробників програмного забезпечення щодо покращення процесу його розробки та підтримки, а також висвітлюються перспективи розвитку стандартів SQuaRE у майбутньому.

1 ОСНОВНІ ПРИНЦИПИ ТА ІСТОРІЯ РОЗВИТКУ SQUARE

1.1 Визначення та історія розвитку стандартів SQuaRE

SQuaRE (Software Product Quality Requirements and Evaluation) – це набір міжнародних стандартів, розроблений для визначення, вимірювання та оцінювання якості програмного забезпечення (ПЗ). Ці стандарти надають структуровану методологію для оцінки якості ПЗ за допомогою визначення характеристик і метрик, які відповідають різним аспектам якості, таким як функціональність, надійність, зручність використання, ефективність, підтримуваність і портативність.

Розробка стандартів для оцінки якості програмного забезпечення почалася ще в 1980-х роках, коли розуміння важливості забезпечення високої якості ПЗ стало критичним для успішної розробки та використання інформаційних систем. У той час з'явилися перші стандарти, що фокусувалися на окремих аспектах якості ПЗ. У 1991 році Міжнародна організація зі стандартизації (ISO) та Міжнародна електротехнічна комісія (IEC) випустили стандарт ISO/IEC 9126, який був одним із перших спроб систематизувати підходи до оцінки якості програмного забезпечення. ISO/IEC 9126 визначав шість основних характеристик якості ПЗ: функціональність, надійність, зручність використання, ефективність, підтримуваність і портативність.

На основі досвіду використання ISO/IEC 9126 та розуміння необхідності в більш деталізованих і комплексних підходах до оцінки якості ПЗ, у 2003 році почалася розробка нової серії стандартів, яка отримала назву SQuaRE. Ці стандарти були розроблені з метою об'єднання підходів до оцінювання якості ПЗ, управління вимогами до якості та процесів оцінювання. Перші версії стандартів SQuaRE були випущені в 2005 році і включали в себе серію ISO/IEC 25000, яка замінила ISO/IEC 9126. Нові стандарти включали чіткіші визначення характеристик якості ПЗ та детальні методики їх оцінювання.

На сьогоднішній день стандарти SQuaRE продовжують розвиватися і вдосконалюватися, враховуючи нові тенденції та вимоги до якості програмного забезпечення. Вони активно використовуються в індустрії для забезпечення високої якості ПЗ та його відповідності вимогам користувачів і замовників.

1.1.1 Порівняння SQuaRE з іншими стандартами якості ПЗ

Порівняння стандарту SQuaRE з іншими стандартами якості програмного забезпечення відображає важливі відмінності і спільні аспекти, які визначають їхню ефективність у практичних застосуваннях. Наприклад, стандарт ISO/IEC 9126, що був прийнятий як основа для SQuaRE, зосереджується на визначенні характеристик якості ПЗ, таких як функціональність, надійність, зручність використання та ефективність.

SQuaRE, у свою чергу, розширює цей підхід, вводячи більш деталізовану структуру для вимірювання якості ПЗ. Він включає більшу кількість характеристик і метрик, які охоплюють не лише технічні аспекти, але й аспекти якості, пов'язані з користувацьким досвідом, експлуатаційною безпекою та іншими важливими факторами.

Порівняння з CMMI (Capability Maturity Model Integration) показує, що SQuaRE зосереджується більше на оцінці якості самого продукту, в той час як CMMI більше орієнтований на процеси розробки та управління проектом. Обидва стандарти можуть доповнювати один одного, залежно від потреб організації та специфіки проектів.

1.1.2 Основні положення та принципи SQuaRE

Основні положення та принципи стандарту SQuaRE (Software Quality Requirements and Evaluation) відображають його основні принципи та підходи до оцінки якості програмного забезпечення.

SQuaRE створений з метою забезпечення системного підходу до оцінки якості ПЗ. Він охоплює широкий спектр характеристик і метрик, які враховують як технічні, так і нефункціональні аспекти продукту. Це включає в себе оцінку функціональності, надійності, ефективності, зручності використання, підтримки та інших ключових характеристик.

Один з ключових принципів SQuaRE – це інтегрованість. Стандарт спрямований на інтеграцію різноманітних методів оцінки, що дозволяє здійснювати комплексну оцінку різних аспектів якості ПЗ з урахуванням специфіки проекту та його вимог.

Крім того, SQuaRE базується на принципах об'єктивності та стандартизації. Він надає чітко визначені метрики і процедури оцінки, що дозволяють здійснювати порівняння і аналіз якості ПЗ на основі загальноприйнятих критеріїв.

1.2 Структура та компоненти стандартів SQuaRE

SQuaRE складається з декількох основних компонентів, які включають в себе модель якості ПЗ, метрики та характеристики якості ПЗ. Модель якості ПЗ за SQuaRE визначає основні характеристики, що впливають на якість програмного забезпечення, і слугує основою для подальшої оцінки. Метрики і характеристики якості ПЗ надають детальні інструменти для вимірювання і оцінки кожної з цих характеристик.

1.2.1 Модель якості ПЗ за SQuaRE

Модель якості програмного забезпечення за SQuaRE є багаторівневою і складається з декількох основних характеристик та підхарактеристик. Ця модель дозволяє оцінювати якість програмного забезпечення на різних етапах його життєвого циклу, від початкового проектування до остаточного впровадження та підтримки. Основні характеристики моделі якості SQuaRE

включають функціональну придатність, надійність, зручність використання, ефективність, підтримуваність та портативність. Кожна з цих характеристик має свої підхарактеристики, які забезпечують більш детальну оцінку. Наприклад, функціональна придатність може включати такі підхарактеристики, як правильність, повнота та придатність для використання. Надійність може включати зрілість, відмовостійкість та можливість відновлення після збоїв.

Функціональна придатність забезпечує, що програмне забезпечення виконує потрібні функції та відповідає специфікаціям. Надійність гарантує, що ПЗ працює стабільно та без збоїв. Зручність використання оцінює, наскільки легко користувачам взаємодіяти з програмою. Ефективність визначає, наскільки добре програмне забезпечення використовує ресурси системи, такі як час виконання та пам'ять. Підтримуваність включає можливість внесення змін та виправлення помилок, а портативність – здатність ПЗ працювати на різних платформах та середовищах. Модель якості SQuaRE є гнучкою і може бути адаптована для конкретних проектів та потреб. Це дозволяє компаніям враховувати специфічні вимоги та умови, забезпечуючи високий рівень якості для кожного окремого продукту.

1.3 Переваги та обмеження стандартів SQuaRE

Стандарти SQuaRE (Software Product Quality Requirements and Evaluation) мають багато переваг, але водночас і певні обмеження. Однією з основних переваг стандартів SQuaRE є системний підхід до оцінки якості.

Вони пропонують чітко визначені методики та процедури для оцінювання якості програмного забезпечення, що дозволяє уникнути суб'єктивності та отримати об'єктивні результати. Крім того, ці стандарти забезпечують комплексність оцінки, охоплюючи усі аспекти якості ПЗ: функціональність, надійність, зручність використання, ефективність, підтримуваність та портативність. Це дозволяє провести всебічний аналіз

якості продукту.

Ще однією важливою перевагою є використання метрик. SQuaRE пропонує різноманітні метрики для оцінки якості, що дозволяє кількісно оцінити різні характеристики ПЗ і забезпечити точний аналіз. Використання єдиних стандартів і метрик також дозволяє порівнювати якість різних програмних продуктів та їх версій, що корисно для прийняття рішень щодо вибору або поліпшення ПЗ.

Стандарти SQuaRE також підтримують вдосконалення процесів розробки і тестування ПЗ, що сприяє підвищенню загальної якості продуктів і задоволеності користувачів. Важливо зазначити, що ці стандарти є міжнародними, що визнаються в усьому світі, полегшуючи співпрацю між компаніями і підвищуючи довіру до результатів оцінювання.

Однак стандарти SQuaRE мають і свої обмеження. Впровадження цих стандартів може вимагати значних зусиль і ресурсів, особливо для компаній, які раніше не використовували системні методики оцінювання якості. Процес оцінювання якості за стандартами SQuaRE вимагає від фахівців високої кваліфікації та знань у сфері якості ПЗ, що може бути проблематичним для малих компаній або нових команд. Крім того, використання стандартів SQuaRE може бути пов'язане з додатковими витратами на навчання персоналу, адаптацією процесів і впровадженням необхідних інструментів і методик. Хоча ці стандарти є універсальними, вони можуть не завжди повністю відповідати специфічним потребам або умовам окремих проектів чи компаній, що може вимагати додаткової адаптації. Процес оцінювання якості за стандартами SQuaRE також може займати значний час, що може бути критичним для проектів з обмеженими термінами. Отже, стандарти SQuaRE є потужним інструментом для оцінювання якості програмного забезпечення, але їх впровадження і використання пов'язане з певними викликами. Розуміння цих переваг і обмежень допоможе компаніям ефективно використовувати стандарти SQuaRE для підвищення якості своїх продуктів.

1.3.1 Переваги впровадження стандартів

Впровадження стандартів SQuaRE в сфері розробки програмного забезпечення дозволяє організаціям досягати кількох ключових переваг. Вони включають чітке узгодження і стандартизацію процесів оцінювання якості, що сприяє уникненню недорозумінь та знижує ризики. Також, використання чітко визначених метрик і методологій спрощує використання ресурсів і часу, що забезпечує ефективне управління проектами і забезпечує високий рівень якості програмного забезпечення. Покращення довіри споживачів до продукції завдяки можливості демонструвати стабільність і надійність програмного забезпечення також є важливим аспектом.

Впровадження стандартів SQuaRE також сприяє підтримці внутрішніх і зовнішніх аудитів, що дозволяє організаціям відповідати вимогам стандартів і регуляторів. Це стимулює стабільний розвиток компанії та підвищує її конкурентоспроможність, забезпечуючи високу якість програмного забезпечення і задоволення потреб клієнтів.

1.3.2 Обмеження та виклики впровадження

Впровадження стандартів SQuaRE супроводжується кількома обмеженнями і викликами. Серед них слід виділити необхідність значних ресурсів і зусиль на початковому етапі для інтеграції нових процесів і методів оцінювання. Також важливо забезпечити належне супроводження процесу впровадження і навчання персоналу, щоб забезпечити ефективне використання стандартів. Адаптація стандартів до конкретних потреб і характеристик проектів може вимагати додаткових зусиль і ресурсів. Крім того, внутрішня підтримка і впровадження культури якості в організації є важливими аспектами успішної реалізації стандартів SQuaRE.

Впровадження стандартів SQuaRE також стикається з викликами у розумінні і прийнятті нових метрик та критеріїв оцінювання якості, особливо

у складних інформаційних системах і проектах. Крім того, необхідно забезпечити відповідність стандартів зовнішнім вимогам і регуляторним стандартам, що може вимагати додаткових зусиль і уваги з боку організації. Постійне оновлення і адаптація стандартів до змінюваних технологічних і ринкових умов також є важливим аспектом для успішного впровадження SQuaRE.

2 МЕТОДОЛОГІЯ ОЦІНЮВАННЯ ЯКОСТІ ПЗ

2.1 Процес оцінювання якості за SQuaRE

Задача оцінювання якості програмного забезпечення за стандартами SQuaRE включає комплексний процес, що охоплює вибір, застосування та аналіз метрик, які дозволяють об'єктивно визначати рівень його якості. Процес оцінювання розпочинається з визначення цілей оцінки та областей, які потрібно оцінювати. Це може включати функціональність продукту, його продуктивність, надійність, безпеку та інші аспекти, залежно від специфікацій проекту та вимог користувачів.

Далі йде вибір відповідних метрик якості. Це важливий крок, оскільки від обраних метрик залежить об'єктивність та репрезентативність отриманих результатів. Метрики можуть бути вибрані на основі їхньої спроможності адекватно відображати основні аспекти функціонування програмного продукту. Після вибору метрик проводиться адаптація до конкретного контексту проекту. Це включає в себе специфікацію методів вимірювання, встановлення метричних порогів та інтерпретацію отриманих значень. Наприклад, для метрики продуктивності можуть встановлюватися стандарти часу відгуку системи або частоти завантаження сервера. Завершальний етап включає проведення оцінки на практиці, аналіз отриманих даних та надання висновків та рекомендацій з покращення якості ПЗ на основі результатів оцінювання.

Такий циклічний підхід дозволяє не лише об'єктивно оцінити поточний стан програмного забезпечення за стандартами SQuaRE, але і забезпечити підстави для його подальшого вдосконалення з урахуванням виявлених недоліків та потреб користувачів.

Метод вимірювання якості – це логічна послідовність операцій, що використовується для кількісного визначення властивості щодо конкретної шкали.

Результат застосування методу вимірювання називається елементом показника якості (ЕПЯ). ЕПЯ – це показник, визначений у термінах властивості і методу вимірювання, який включає вибірково перетворення за допомогою математичної функції. Функція вимірювання – це алгоритм, який використовується для об'єднання елементів показника якості.

Результат застосування функції вимірювання називається показником якості (ПЯ) програмного забезпечення. ПЯ стають кількісними показниками характеристик і підхарактеристик якості. Для вимірювання характеристики або підхарактеристики якості може використовуватися кілька ПЯ.

Взаємозв’язок властивості кількісного визначення, методу вимірювання, ЕПЯ і ПЯ наведено на рис. 2.1.

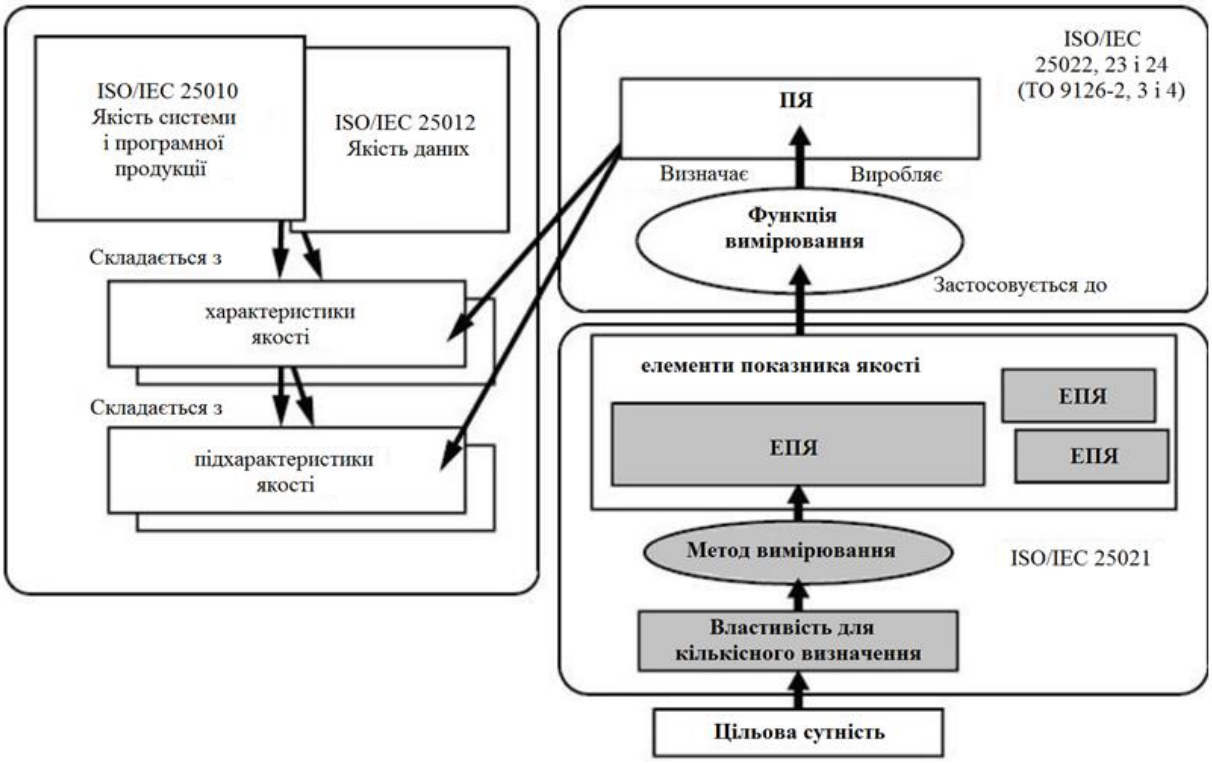

Рисунок 2.1 – Модель вимірювання якості програмних продуктів

Модель якості продукту зводить властивості якості системи/програмного продукту до восьми характеристик, якими є:

функціональна придатність, рівень продуктивності, сумісність, зручність користування, надійність, захищеність, супроводуваність і переносимість (мобільність). Кожна характеристика, в свою чергу, складається з ряду відповідних підхарактеристик (рис. 2.2).

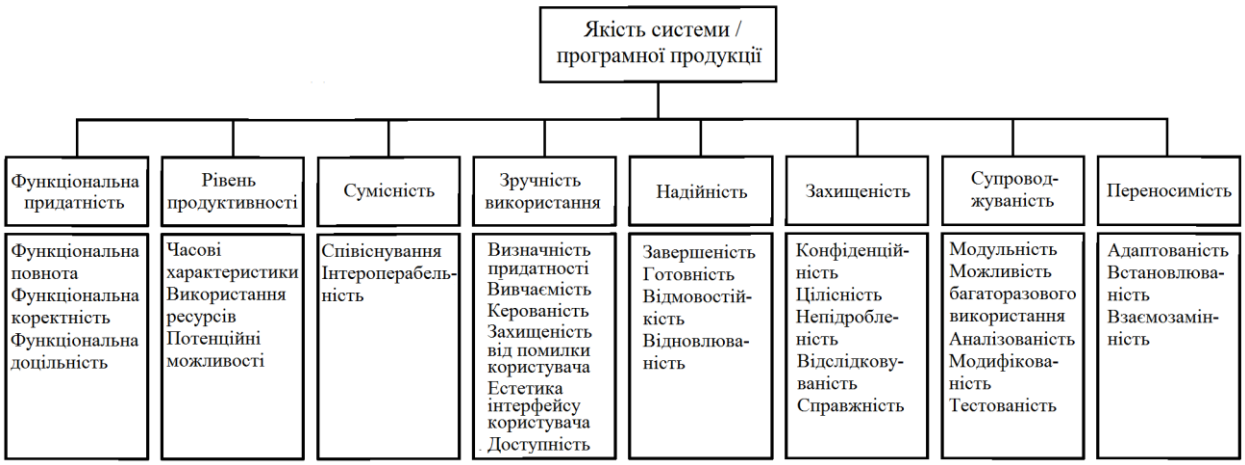

Рисунок 2.2 – Модель якості продукту

Детальний опис характеристик та підхарактеристик моделі якості продукту наведено в таблиці 2.1.

Таблиця 2.1 – Характеристики на підхарактеристики моделі якості продукту

Характеристика	Підхарактеристики
1. Функціональна придатність (functional suitability) – ступінь, в якій продукт або система забезпечують виконання функції відповідно до заявлених і якими мається на увазі потребами при використанні в зазначених умовах.	1.1 Функціональна повнота (functional completeness) - ступінь покриття сукупністю функцій всіх визначених завдань і цілей користувача
	1.2 Функціональна коректність (functional correctness) – ступінь забезпечення продуктом або системою необхідного ступеня точності коректних результатів.
	1.3 Функціональна доцільність (functional appropriateness) – ступінь функціонального спрощення виконання певних завдань і досягнення цілей.
2. Рівень продуктивності (performance efficiency) – продуктивність щодо суми використаних при певних умовах ресурсів	2.1 Часові характеристики (time behaviour) – ступінь відповідності вимогам за часом відгуку, часу обробки і показни-ків пропускної здатності продукту або системи
	2.2 Використання ресурсів (resource utilization) – ступінь задоволення вимог по споживанню обсягів і видів ресурсів продуктом або системою при виконанні ними своїх функцій

Продовження таблиці 2.1

Характеристика	Підхарактеристики
3. Сумісність (compatibility) – здатність продукту, системи або компонента обмінюватися інформацією з іншими продуктами, системами або компонентами, і / або виконувати необхідні функції при спільному використанні одних і тих же апаратних засобів або програмного середовища.	3.1 Співіснування (сумісність) (co-existence) – здатність продукту спільно функціонувати з іншими незалежними продуктами в загальному середовищі з поділом загальних ресурсів і без негативного впливу на будь-який інший продукт.
	3.2 Функціональна сумісність (інтероперабельність) (interoperability) – здатність двох або більше систем, продуктів або компонентів обмінюватися інформацією і використовувати таку інформацію.
4. Зручність використання (usability) – ступінь, в якому продукт або система можуть бути використані певними користувачами для досягнення конкретних цілей з ефективністю, результативністю і задоволеністю в заданому контексті використання.	4.1 Визначність придатності (appropriateness recognizability) – накінець, можливість зрозуміти, чи підходить продукт або система для їх потреб, чи порівнюваний з функціональною доцільністю.
	4.2 Вивчаємість (learnability) – можливість використання продукту або системи певними користувачами для досягнення конкретних цілей навчання для експлуатації продукту або системи з ефективністю, результативністю, свободою від ризику і відповідно до вимог в зазначеному контексті використання.
	4.3 Керованість (operability) – наявність в продукті або системі атрибутів, що забезпечують просте управління і контроль.
	4.4 Захищеність від помилки користувача (user error protection) – рівень системного захисту від помилок користувачів.
	4.5 Естетика інтерфейсу користувача (user interface aesthetics) – ступінь «приємності» і «задоволеності» користувача інтерфейсом взаємодії з користувачем.
	4.6 Доступність (accessibility) – можливість використання продукту або системи для досягнення певної мети в зазначеному контексті використання широким колом людей з різними можливостями.
5. Надійність (reliability) – ступінь виконання системою, продуктом або компонентом певних функцій при зазначених умовах протягом встановленого періоду часу.	5.1 Завершеність (maturity) – ступінь відповідності системи, продукту або компонента при нормальній роботі вимогам надійності.
	5.2 Готовність (availability) – ступінь працездатності та доступності системи, продукту або компонента. Готовність визначається поєднанням завершеності, яка визначає частоту відмов, відмовостійкості та відновлюваності, яка, в свою чергу, визначає тривалість часу бездіяльності після кожного відмови.
	5.3 Відмовостійкість (fault tolerance) – здатність системи, продукту або компонента працювати як призначено, незважаючи на наявність дефектів програмного забезпечення або апаратних засобів.
	5.4 Відновлюваність (recoverability) – здатність продукту або системи відновити дані і необхідний стан системи в разі переривання або збою.

Продовження таблиці 2.1

Характеристика	Підхарактеристики
6. Захист, захищеність (security) – ступінь захищеності інформації та даних, що забезпечується продуктом або системою шляхом обмеження доступу людей, інших продуктів або систем до даних відповідно до типів і рівнів авторизації.	6.1 Конфіденційність (confidentiality) – забезпечення продуктом або системою обмеження доступу до даних тільки для тих, кому доступ дозволений.
	6.2 Цілісність (integrity) – ступінь запобігання системою, продуктом або компонентом несанкціонованого доступу або модифікації комп'ютерних програм або даних.
	6.3 Непідробленість (non-repudiation) – ступінь, з яким може бути доведений факт події або дії таким чином, що цей факт не може бути відкинутий коли-небудь пізніше.
	6.4 Відстежуваність (accountability) – ступінь, до якої дії об'єкта можна простежити однозначно.
	6.5 Справжність (authenticity) – ступінь достовірності тотожності об'єкта або ресурсу необхідному об'єкту або ресурсу.
7. Супроводжуваність, модифікованість (maintainability) – результативність та ефективність, з якими продукт або система можуть бути модифіковані передбачуваними фахівцями з обслуговування.	7.1 Модульність (modularity) – ступінь представлення системи або комп'ютерної програми у вигляді окремих блоків таким чином, щоб зміна одного компонента чинило мінімальний вплив на інші компоненти.
	7.2 Можливість багаторазового використання (reusability) – ступінь, в якій актив може бути використаний в декількох системах або в створенні інших активів. *Примітка: це визначення було адаптовано з IEEE 1517-2004.*
	7.3 Аналізованість (analysability) – ступінь простоти оцінки впливу змін однієї чи більше частин на продукт або систему або простоти діагностики продукту для виявлення недоліків і причин відмов, або простоти ідентифікації частин, що підлягають зміні.
	7.4 Модифікованість (modifiability) – ступінь простоти ефективної і раціональної зміни продукту або системи без додавання дефектів і зниження якості продукту.
	7.5 Тестованість (testability) – ступінь простоти ефективного і раціонального визначення для системи, продукту або компонента критеріїв тестування, а також простоти виконання тестування з метою визначення відповідності цим критеріям.
8. Переносимість, мобільність (portability) – ступінь простоти ефективного і раціонального перенесення системи, продукту або компонента з одного середовища (апаратних засобів, програмного забезпечення, операційних умов або умов використання) в інше.	8.1 Адаптованість (adaptability) – ступінь простоти ефективної і раціональної адаптації для апаратних засобів, програмного забезпечення, інших операційних середовищ або умов використання, що відрізняються або вдосконалених.
	8.2 Встановлюваність (installability) – ступінь простоти ефективної і раціональної, успішної установки і / або видалення продукту або системи в заданому середовищі.
	8.3 Взаємозамінність (replaceability) – здатність продукту замінити інший конкретний програмний продукт для досягнення тих же цілей в тих же умовах.

2.1.1 Вибір метрик та їх адаптація

Задача вибору метрик якості програмного забезпечення (ПЗ) за стандартами SQuaRE (Software Quality Requirements and Evaluation) є ключовим етапом у процесі оцінювання його ефективності та надійності. Цей процес починається з глибокого аналізу вимог до програмного продукту, що включає у себе ретельне вивчення функціональних, продуктивних та безпекових характеристик, необхідних для задоволення потреб користувачів та вимог бізнес-процесів.

На основі цього аналізу обираються метрики, які найкращим чином відображають ці вимоги та відповідають цілям оцінювання. Наприклад, для вебдодатків можуть бути обрані метрики, що вимірюють час завантаження сторінок, частоту помилкових відповідей сервера або загальну продуктивність системи. Кожна метрика потребує чіткої специфікації її вимірювання та методики інтерпретації отриманих результатів.

Далі проводиться адаптація обраних метрик до конкретного контексту програмного продукту. Цей процес включає в себе визначення оптимальних значень для кожної метрики, врахування унікальних особливостей системи та технічних вимог до її функціонування. Наприклад, метрика «час відгуку системи» може бути адаптована до вимірювання в мілісекундах з урахуванням специфіки обробки запитів користувачів.

Крім того, під час вибору та адаптації метрик необхідно враховувати їхню практичну застосовність та можливість забезпечити об'єктивне вимірювання якості ПЗ. Це є критичним аспектом для забезпечення консистентності оцінки якості програмного забезпечення та розробки обґрунтованих стратегій його вдосконалення.

Таким чином, вибір та адаптація метрик якості програмного забезпечення за стандартами SQuaRE є складним, але необхідним процесом, спрямованим на покращення ефективності та надійності програмних продуктів у відповідності до вимог сучасного ринку та користувацьких очікувань.

2.2 Інструменти для оцінювання якості

Задача оцінювання якості програмного забезпечення за стандартами SQuaRE включає в себе процес вибору та застосування відповідних метрик та методів збору даних, а також їхнього подальшого аналізу. Після вибору відповідних метрик якості, важливим є обрати ефективні методи збору даних. Це може включати автоматизоване тестування, анкетування користувачів, моніторинг реального часу або аналіз журналів системи. Вибір методу залежить від специфічних характеристик програмного продукту та поставлених цілей оцінювання. Отримані дані підлягають подальшому детальному аналізу для виявлення патернів, трендів та аномалій, які можуть вказувати на проблемні аспекти або успіхи програмного продукту.

Аналіз включає статистичні методи, визначення середніх значень, дисперсій, кореляційних зв'язків та інших метрик для об'єктивного порівняння та оцінки результатів. Важливим аспектом є також оцінка достовірності та точності зібраних даних. Це включає перевірку на відповідність методів збору, рівень звільнення помилок та мінімізацію потенційних викривлень. Використання стандартизованих процедур та інструментів для аналізу даних дозволяє підвищити надійність результатів оцінювання.

На основі аналізу зібраних даних формулюються висновки щодо поточного стану якості програмного забезпечення, ідентифікуються ключові проблемні моменти та рекомендації для подальшого вдосконалення. Цей етап є важливим для забезпечення практичної цінності отриманих результатів та їхньої відповідності до поставлених цілей оцінювання.

Для оцінювання якості програмного забезпечення за стандартами SQuaRE важливо мати на увазі наявні інструменти, що допомагають у зборі та аналізі відповідних даних. На сьогоднішній день існує ряд інструментів, які можуть бути використані для реалізації процесу оцінювання якості ПЗ згідно зі стандартами SQuaRE.

Один із найбільш відомих інструментів у цій області є Quality Gate Management. Цей інструмент надає можливість налаштування автоматичних порогів якості для різних аспектів програмного забезпечення, таких як кодування, тестування та інтеграція. Він дозволяє встановлювати метрики якості, які систематично перевіряються під час процесу розробки та деплойменту програмного продукту.

Ще одним корисним інструментом є SonarQube, який забезпечує статичний аналіз коду для виявлення потенційних проблем безпеки, ефективності та підтримки стандартів програмування. SonarQube дозволяє автоматично аналізувати проектний код та надає звіти про виявлені проблеми, що допомагає вчасно виявляти та виправляти дефекти.

Крім того, інструмент Jenkins використовується для автоматизації процесів тестування та інтеграції, що сприяє покращенню якості програмного продукту. Jenkins дозволяє налаштовувати автоматичні тести, забезпечує моніторинг якості під час розробки та може інтегруватися з різними інструментами для збору метрик та аналізу даних.

Кожен із цих інструментів має свої переваги та обмеження, тому вибір конкретного залежить від специфіки проекту, типу ПЗ та поставлених цілей оцінювання якості. Важливо враховувати інтегрованість інструменту в процес розробки, можливість масштабування та підтримку командою розробників.

Для практичного застосування інструментів оцінювання якості програмного забезпечення за стандартами SQuaRE важливо визначити їхню конкретну роль та внедрення в процес розробки програмного продукту.

Один з найефективніших способів застосування цих інструментів полягає в їх інтеграції в процес автоматизації. Наприклад, інтеграція SonarQube в систему Continuous Integration / Continuous Deployment (CI/CD) дозволяє автоматично аналізувати якість коду на кожному етапі його розробки. Це дозволяє вчасно виявляти потенційні проблеми та дефекти, що зменшує витрати часу і ресурсів на подальше виправлення.

Також практичне застосування інструментів включає їхнє використання для моніторингу та звітування про якість програмного продукту. Наприклад, використання Jenkins для автоматизації тестування та інтеграції з іншими інструментами для збору метрик дозволяє забезпечити стабільність і надійність продукту під час його експлуатації.

Практичне застосування також передбачає відповідний підбір метрик якості, які краще відповідають специфіці вашого проекту. Наприклад, для вебзастосунків можуть бути важливими метрики, що стосуються продуктивності та відгуків користувачів, тоді як для вбудованих систем критичними можуть бути метрики надійності та безпеки.

Крім того, практичне застосування інструментів включає їхнє використання для вдосконалення процесів розробки та управління проектами. Наприклад, використання Quality Gate Management для встановлення та контролю якісних стандартів може сприяти покращенню загальної якості продукту та зниженню витрат на його підтримку.

2.3 Кейс-стаді: приклад оцінювання якості конкретного ПЗ

У кейс-стаді аналізується конкретний програмний продукт, в якому обрані відповідні метрики якості, такі як функціональність, надійність та продуктивність. Для збору даних можуть використовуватися методи автоматизованого тестування, збір логів роботи програми та анкетування користувачів.

Отримані дані аналізуються з використанням статистичних методів та інструментів для визначення патернів та трендів у якості ПЗ. Особлива увага приділяється інтерпретації результатів, що включає виявлення основних проблемних аспектів та сильних сторін програмного продукту. На основі аналізу зроблені висновки та рекомендації щодо подальшого удосконалення якості ПЗ. Цей кейс-стаді демонструє практичне застосування стандартів SQuaRE у реальних умовах, а також важливість систематичного підходу до

оцінювання та управління якістю програмного забезпечення.

Для прикладу було оцінено програмне забезпечення, яке є широко відомим і використовується в багатьох компаніях – система управління проектами JIRA від Atlassian.

JIRA є однією з найпопулярніших систем управління проектами, яку використовують для відстеження завдань, управління проектами та контролю за виконанням робіт. Це програмне забезпечення застосовується як малими командами, так і великими компаніями. Основними функціональними вимогами до JIRA є управління завданнями та проектами, призначення завдань користувачам та відстеження їх виконання, інтеграція з іншими інструментами розробки, такими як Bitbucket, Confluence та інші, генерація звітів та аналітики, а також забезпечення безпеки даних і контроль доступу.

Процес оцінювання якості JIRA за допомогою стандартів SQuaRE включав кілька етапів. Спочатку були визначені ключові показники для оцінювання якості JIRA відповідно до стандартів SQuaRE, включаючи метрики функціональності, надійності, зручності використання, ефективності, підтримуваності та портативності. Потім дані для оцінювання якості були зібрані за допомогою різних методів, таких як автоматизовані інструменти для аналізу коду, опитування користувачів, тестування продуктивності та безпеки, а також журнали подій і звіти про помилки. Зібрані дані були проаналізовані для оцінки відповідності кожної метрики встановленим критеріям якості, що дозволило виявити сильні та слабкі сторони JIRA.

Результати оцінювання якості JIRA показали, що система повністю задовольняє більшість функціональних вимог. Вона дозволяє ефективно керувати проектами, призначати завдання, генерувати звіти та інтегруватися з іншими інструментами. Інтеграція з іншими продуктами Atlassian, такими як Confluence і Bitbucket, є бездоганною, що дозволяє створювати єдину екосистему для управління проектами. Система має високу стабільність роботи, середній час між відмовами (MTBF) є дуже високим, що свідчить про

надійність. Середній час відновлення (MTTR) після збоїв також є низьким, що вказує на швидке вирішення проблем.

Опитування користувачів показало високий рівень задоволеності інтерфейсом JIRA. Користувачі відзначають інтуїтивність і зручність використання системи. Кількість помилок, що виникають під час використання, є низькою, що свідчить про добре продуманий дизайн інтерфейсу користувача. Продуктивність системи є на високому рівні, час відгуку і швидкість виконання завдань відповідають вимогам користувачів. Використання процесорного часу і пам'яті є оптимізованим, що забезпечує ефективну роботу системи навіть при великому навантаженні.

Аналіз коду JIRA показав високу підтримуваність. Кількість помилок, виявлених під час ревізії коду, є невеликою, а час, необхідний для внесення змін, є мінімальним. Документація до JIRA є повною і актуальною, що сприяє легкості аналізу і внесення змін. JIRA легко адаптується до різних операційних систем і платформ. Час, необхідний для перенесення системи на нову платформу, є незначним, а кількість проблем, що виникають при перенесенні, є мінімальною.

На основі результатів оцінювання було зроблено кілька рекомендацій для подальшого покращення якості JIRA. По-перше, слід покращити інтеграцію з іншими системами. Хоча JIRA добре інтегрується з продуктами Atlassian, є можливості для поліпшення інтеграції з іншими сторонніми інструментами. По-друге, слід оптимізувати процеси відновлення після збоїв, зменшити час відновлення після збоїв за рахунок впровадження більш ефективних методів моніторингу та автоматизації виправлення проблем. По-третє, слід продовжити роботу над документацією, постійно оновлювати і доповнювати технічну документацію, зокрема для нових функцій та оновлень системи. І, нарешті, слід розширити функціональність, врахувати відгуки користувачів і додати нові функції, які покращать зручність використання та ефективність системи.

Цей кейс-стаді показав, що оцінювання якості програмного

забезпечення за допомогою стандартів SQuaRE є ефективним підходом для всебічного аналізу і покращення якості ПЗ. Впровадження рекомендованих змін допоможе підвищити задоволеність користувачів і забезпечити довготривалу успішну експлуатацію програмного продукту.

3 ВИМІРЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ

3.1 Порівняння результатів з використанням різних стандартів

Оцінка якості ПЗ за допомогою різних стандартів є ключовим етапом у процесі забезпечення його відповідності вимогам та очікуванням користувачів. Порівняння проводиться на основі зібраних даних із використанням метрик та інструментів, визначених у відповідних стандартах.

Наприклад, для порівняння можуть використовуватися такі стандарти як ISO/IEC 25010 (який базується на стандартах SQuaRE), CMMI (Capability Maturity Model Integration), а також інші міжнародні та промислові стандарти, які визначають критерії якості та методи їх оцінювання. Результати порівняння дозволяють ідентифікувати переваги та недоліки кожного стандарту у контексті конкретного проекту. Наприклад, стандарт ISO/IEC 25010 може виявитися більш адаптованим для оцінювання функціональних характеристик ПЗ, тоді як CMMI може забезпечити більш деталізовану оцінку з точки зору процесів розробки та управління проектом.

Крім того, порівняння дозволяє визначити, які стандарти або комбінації стандартів є найбільш ефективними для покращення якості ПЗ в конкретних умовах і призводять до досягнення визначених цілей проекту.

Одним із ключових аспектів порівняльного аналізу стандартів оцінювання якості програмного забезпечення є порівняння стандарту SQuaRE з ISO/IEC 9126. Ці два стандарти відіграють важливу роль у визначенні та оцінці якості програмного забезпечення, проте мають свої особливості та підходи.

ISO/IEC 9126 визначає модель якості ПЗ, яка складається з шести основних характеристик якості: функціональність, надійність, зручність використання, ефективність, супроводжуваність та мобільність. Кожна з цих характеристик поділяється на підхарактеристики, що специфікують критерії

оцінювання. ISO/IEC 9126 часто використовується для оцінювання якості в ранніх стадіях життєвого циклу ПЗ, коли важливо визначити загальну придатність програмного забезпечення до використання.

SQuaRE (Software product Quality Requirements and Evaluation), у свою чергу, розширює підхід до оцінки якості, надаючи більш деталізовану модель, яка включає не лише технічні характеристики якості, а й аспекти, пов'язані з інженерією вимог, оцінкою користувацької задоволеності та процесами управління якістю. SQuaRE складається з набору стандартів, які включають в себе визначення метрик, процесів оцінювання та рекомендацій щодо управління якістю ПЗ.

Порівнюючи SQuaRE з ISO/IEC 9126, можна зазначити, що SQuaRE надає більш глибокий і деталізований підхід до оцінювання якості, охоплюючи більший спектр аспектів від вимог до ПЗ до його експлуатації та підтримки. Водночас, ISO/IEC 9126 пропонує більш структурований підхід до класифікації і оцінки якості, що може бути корисним на ранніх етапах проекту.

Порівняння SQuaRE з іншими підходами до оцінювання якості програмного забезпечення включає аналіз їхніх основних принципів та методів оцінки. SQuaRE, як міжнародний стандарт, надає комплексний підхід до визначення та оцінювання якості програмного забезпечення. Він охоплює не лише технічні аспекти, такі як функціональність та надійність, але й аспекти, пов'язані з інженерією вимог, процесами управління якістю та оцінкою користувацької задоволеності.

Проте, існують інші підходи, такі як ISO/IEC 25010, що акцентують увагу на моделі якості програмного забезпечення, визначаючи основні характеристики якості та їх підхарактеристики. Цей стандарт спрямований на встановлення критеріїв оцінки якості ПЗ та надання структурованого підходу до їх вимірювання.

Крім того, Agile методології, такі як Scrum та Kanban, підходять до оцінювання якості з іншої точки зору, зосереджуючись на швидкому

впровадженні змін та відповідності потребам користувачів. Ці методології вимагають постійного вдосконалення і оцінювання результатів в контексті виконання вимог клієнта та управління ризиками.

Нарешті, моделі Lean і Six Sigma підходять до оцінювання якості через систематичний підхід до виявлення та виправлення дефектів у процесах розробки ПЗ, що спрямоване на забезпечення максимальної ефективності та мінімізацію витрат.

Кожен з цих підходів має свої переваги та недоліки, і вибір конкретного залежить від специфіки проекту, його вимог та етапу життєвого циклу. Порівняння дозволяє об'єктивно оцінити кожен підхід з точки зору відповідності потребам проекту та забезпечення високої якості програмного забезпечення.

Якість програмного продукту може бути оцінена шляхом вимірювання внутрішніх властивостей (як правило, статичних показників проміжних продуктів), або шляхом вимірювання зовнішніх властивостей (зазвичай, шляхом вимірювання поведінки коду при виконанні), або шляхом вимірювання якості використовуваних властивостей (коли продукт знаходиться в реальному або модельованому використанні). Відповідні внутрішні властивості програмного забезпечення є необхідною умовою для того, щоб досягти необхідної зовнішньої поведінки, а відповідна зовнішня поведінка є необхідною умовою для досягнення якості використання (рис. 3.1).

3.2 Метрики функціональності

Метрики функціональності програмного забезпечення (ПЗ) дозволяють оцінити, наскільки добре програмний продукт виконує свої функції відповідно до вимог користувачів і специфікацій. Функціональність є однією з ключових характеристик якості програмного забезпечення і визначає здатність ПЗ виконувати заплановані завдання.

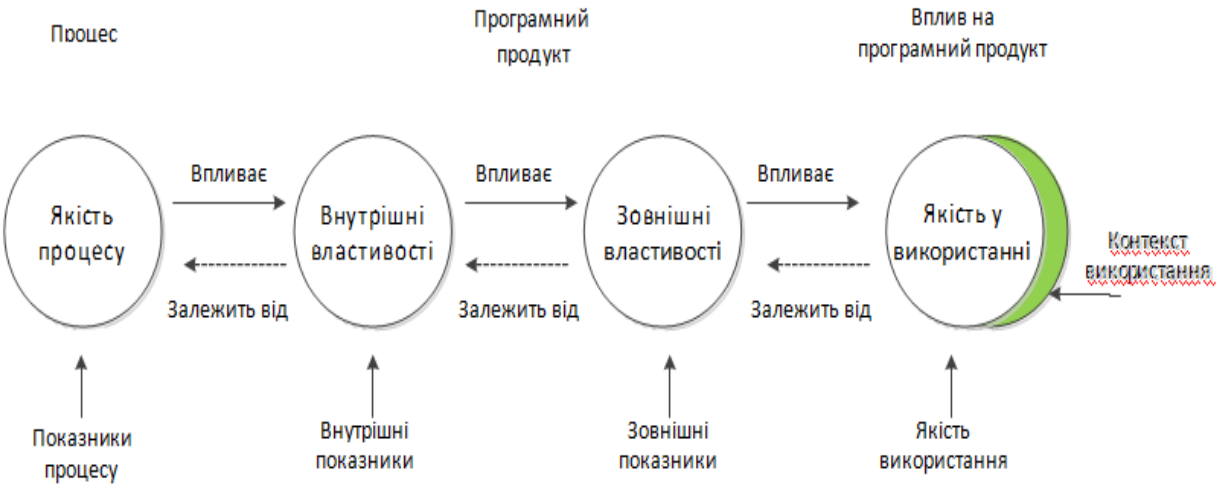

Рисунок 3.1 – Взаємозв'язок між видами показників якості

Одним з основних показників функціональності є відповідність. Вона вимірює, наскільки функції програмного забезпечення відповідають встановленим вимогам і специфікаціям. Для цього можна використовувати такі показники, як кількість виконаних вимог і точність виконання функцій. Наприклад, кількість виконаних вимог показує відсоток вимог, які були успішно реалізовані в програмному продукті, а точність виконання функцій оцінює частоту помилок або відхилень від очікуваних результатів.

Ще однією важливою метрикою є повнота, яка визначає, наскільки всі необхідні функції були реалізовані в програмному забезпеченні. Для цього використовуються показники перекриття функціональних вимог і частота незадоволених вимог. Перекриття функціональних вимог показує відсоток функціональних вимог, які були повністю реалізовані, а частота незадоволених вимог оцінює кількість функціональних вимог, які не були реалізовані або були реалізовані не повністю.

Придатність для використання оцінює, наскільки зручно та ефективно користувачі можуть використовувати функції програмного забезпечення для досягнення своїх цілей. Ця метрика включає такі показники, як час на виконання завдань і кількість помилок користувачів. Час на виконання

завдань вимірює середній час, необхідний користувачам для виконання певних завдань за допомогою ПЗ, а кількість помилок користувачів показує кількість помилок, які роблять користувачі під час взаємодії з програмою.

Інтероперабельність вимірює здатність програмного забезпечення взаємодіяти з іншими системами або компонентами. Це важлива характеристика для забезпечення сумісності та інтеграції ПЗ. Метрики інтероперабельності можуть включати кількість успішних інтеграцій і частоту виникнення проблем при інтеграції. Кількість успішних інтеграцій показує відсоток успішних інтеграцій ПЗ з іншими системами, а частота виникнення проблем при інтеграції оцінює кількість проблем, що виникають під час інтеграції ПЗ з іншими системами.

Безпека оцінює здатність програмного забезпечення захищати дані та забезпечувати конфіденційність і цілісність інформації. Метрики безпеки можуть включати кількість виявлених вразливостей і час реагування на інциденти безпеки. Кількість виявлених вразливостей показує кількість вразливостей, виявлених під час тестування або експлуатації ПЗ, а час реагування на інциденти безпеки оцінює середній час, необхідний для реагування на інциденти безпеки.

Відповідність функціональним вимогам вимірює, наскільки добре програмне забезпечення відповідає встановленим функціональним вимогам. Метрики можуть включати рівень задоволеності користувачів і кількість функціональних дефектів. Рівень задоволеності користувачів оцінюється за допомогою анкетування або опитувань, а кількість функціональних дефектів показує кількість дефектів, що стосуються реалізації функціональних вимог.

3.3 Метрики надійності

Метрики надійності програмного забезпечення (ПЗ) дозволяють оцінити, наскільки стабільно і безпечно працює програмний продукт протягом часу. Надійність є критично важливою характеристикою якості ПЗ,

оскільки від неї залежить безперервність бізнес-процесів і задоволеність користувачів.

Однією з ключових метрик надійності є середній час між відмовами (MTBF). Ця метрика вимірює середній час, протягом якого ПЗ працює без збоїв. Високий MTBF свідчить про високу надійність продукту, оскільки це означає, що збої трапляються рідко. MTBF можна розрахувати шляхом ділення загального часу роботи ПЗ на кількість відмов. Чим вищий цей показник, тим краще.

Ще однією важливою метрикою є середній час відновлення (MTTR). MTTR вимірює середній час, необхідний для відновлення ПЗ після збою. Менший MTTR свідчить про те, що проблеми вирішуються швидко, і система повертається до нормальної роботи у найкоротший термін. Ця метрика є особливо важливою для критичних систем, де навіть короткий час простою може мати серйозні наслідки.

Кількість відмов за певний період є ще одним показником надійності. Він вимірює, скільки разів ПЗ зазнало відмови протягом визначеного періоду часу. Менша кількість відмов свідчить про більш високу надійність. Ця метрика може бути корисною для виявлення тенденцій і проблемних областей у ПЗ.

Здатність до відновлення є важливою характеристикою, що оцінює, наскільки легко і швидко ПЗ може відновитися після збою. Для оцінки цієї метрики використовуються показники, такі як час, необхідний для відновлення, і кількість успішних відновлень після збоїв. Висока здатність до відновлення означає, що ПЗ здатне швидко відновити свою роботу після збоїв, мінімізуючи негативні наслідки для користувачів.

Метрики відмовостійкості оцінюють здатність ПЗ продовжувати роботу в умовах помилок або несправностей. Відмовостійкість може бути оцінена через кількість помилок, які система здатна витримати без повного збою, і кількість критичних помилок, що призводять до повної зупинки роботи ПЗ. Висока відмовостійкість свідчить про те, що система може

продовжувати функціонувати навіть у несприятливих умовах.

Резервування і дублювання компонентів системи є ще одним підходом до підвищення надійності ПЗ. Метрики, що оцінюють ці аспекти, включають частоту використання резервних компонентів і ефективність системи дублювання. Висока ефективність резервування забезпечує додатковий рівень захисту від збоїв, гарантуючи, що система зможе продовжувати працювати навіть у разі відмови основних компонентів.

Загалом, метрики надійності забезпечують всебічний аналіз стабільності та безпеки програмного забезпечення, дозволяючи розробникам і менеджерам приймати обґрунтовані рішення щодо покращення якості продукту. Впровадження надійного ПЗ сприяє підвищенню довіри користувачів, зменшенню ризиків і забезпеченню безперервності бізнес-процесів.

3.4 Метрики зручності використання

Метрики зручності використання програмного забезпечення дозволяють оцінити, наскільки легко та ефективно користувачі можуть взаємодіяти з програмним продуктом для досягнення своїх цілей. Зручність використання є важливою характеристикою якості ПЗ, оскільки вона безпосередньо впливає на задоволеність користувачів і ефективність роботи з програмним продуктом.

Однією з ключових метрик зручності використання є час на виконання завдань. Ця метрика вимірює середній час, необхідний користувачам для виконання певних завдань за допомогою ПЗ. Менший час виконання завдань свідчить про високу зручність використання, оскільки користувачі можуть швидше досягати своїх цілей. Цей показник можна вимірювати шляхом проведення тестувань з реальними користувачами, які виконують типові завдання.

Кількість помилок користувачів є ще однією важливою метрикою

зручності використання. Вона оцінює кількість помилок, які роблять користувачі під час взаємодії з ПЗ. Менша кількість помилок свідчить про те, що інтерфейс ПЗ є інтуїтивно зрозумілим і легким у використанні. Для вимірювання цієї метрики можна використовувати спостереження за користувачами під час виконання завдань або аналіз журналів подій, де фіксуються дії користувачів.

Задоволеність користувачів є важливою характеристикою, що вимірює, наскільки користувачі задоволені взаємодією з ПЗ. Ця метрика може включати проведення анкетувань або опитувань, у яких користувачі оцінюють свої враження від роботи з програмним продуктом. Високий рівень задоволеності користувачів свідчить про те, що ПЗ відповідає їхнім очікуванням і потребам.

Метрики ефективності використання оцінюють, наскільки добре ПЗ допомагає користувачам досягати своїх цілей з мінімальними зусиллями. До таких метрик належать показники успішного виконання завдань і швидкість виконання завдань. Успішне виконання завдань вимірює відсоток завдань, які користувачі змогли виконати без помилок і затримок, тоді як швидкість виконання завдань оцінює, наскільки швидко користувачі можуть досягти своїх цілей.

Легкість навчання є ще однією важливою метрикою зручності використання, яка оцінює, наскільки швидко нові користувачі можуть навчитися працювати з ПЗ. Ця метрика може включати час, необхідний для досягнення певного рівня майстерності, і кількість тренувальних сесій, потрібних для освоєння основних функцій ПЗ. Менший час навчання і менша кількість тренувальних сесій свідчать про високу легкість навчання.

Метрики задоволеності користувачів можуть також включати оцінку емоційного стану користувачів під час роботи з ПЗ. Це може бути досягнуто за допомогою спостереження за виразами обличчя, тілесними реакціями або аналізу відгуків користувачів. Високий рівень позитивних емоцій і відсутність стресу свідчать про високу зручність використання ПЗ.

3.5 Метрики ефективності

Метрики ефективності програмного забезпечення дозволяють оцінити, наскільки добре програмний продукт використовує ресурси системи для виконання своїх завдань. Ефективність є важливою характеристикою якості ПЗ, оскільки вона безпосередньо впливає на продуктивність, швидкість роботи та загальне задоволення користувачів.

Однією з ключових метрик ефективності є час відгуку системи. Ця метрика вимірює час, необхідний програмному забезпеченню для обробки запиту користувача і повернення результату. Менший час відгуку свідчить про високу ефективність, оскільки користувачі отримують результати швидше і можуть працювати більш продуктивно. Час відгуку можна вимірювати шляхом проведення тестів з використанням типових сценаріїв роботи користувачів.

Продуктивність системи є ще однією важливою метрикою ефективності. Вона оцінює кількість завдань або операцій, які ПЗ може виконати за певний період часу. Висока продуктивність свідчить про здатність програмного забезпечення обробляти великі обсяги даних і виконувати складні операції швидко та ефективно. Для вимірювання продуктивності можна використовувати тести навантаження, що моделюють реальні умови роботи системи.

Використання процесорного часу є метрикою, що вимірює кількість часу, протягом якого процесор зайнятий виконанням завдань ПЗ. Менше використання процесорного часу при високій продуктивності свідчить про ефективну оптимізацію коду та алгоритмів. Цю метрику можна вимірювати за допомогою моніторингу системних ресурсів під час виконання програмного забезпечення.

Використання пам'яті є ще однією критично важливою метрикою ефективності. Вона оцінює кількість оперативної пам'яті, яку використовує ПЗ під час своєї роботи. Менше використання пам'яті свідчить про ефективне

управління ресурсами та оптимізацію роботи програмного забезпечення. Моніторинг використання пам'яті дозволяє виявити потенційні проблеми з витоками пам'яті або надмірним споживанням ресурсів.

Пропускна здатність системи є метрикою, що вимірює обсяг даних, який може бути переданий або оброблений програмним забезпеченням за одиницю часу. Висока пропускна здатність свідчить про ефективність роботи системи в умовах високих навантажень. Цю метрику можна вимірювати за допомогою тестів, що імітують умови великої кількості запитів або обробки великих обсягів даних.

Енергоспоживання є важливою метрикою ефективності для портативних і вбудованих систем, де обмежені ресурси живлення. Вимірювання енергоспоживання дозволяє оцінити, наскільки економно ПЗ використовує енергію. Низьке енергоспоживання при високій продуктивності свідчить про ефективну оптимізацію роботи програмного забезпечення. Цю метрику можна вимірювати за допомогою спеціалізованих інструментів, що моніторять споживання енергії під час виконання ПЗ.

Показники ефективності можуть також включати оцінку масштабованості програмного забезпечення, що вимірює здатність системи підтримувати збільшення навантаження без значного погіршення продуктивності. Для оцінки масштабованості використовуються тести навантаження, що поступово збільшують кількість запитів або обсяг даних, щоб визначити, як система реагує на зростання навантаження.

3.6 Показники підтримуваності

Показники підтримуваності програмного забезпечення дозволяють оцінити, наскільки легко та ефективно програмний продукт може бути змінений, виправлений або вдосконалений протягом його життєвого циклу. Підтримуваність є важливою характеристикою якості ПЗ, оскільки вона безпосередньо впливає на вартість і тривалість його експлуатації.

Однією з ключових характеристик підтримуваності є легкість аналізу. Цей показник вимірює, наскільки легко розробники можуть зрозуміти структуру коду, виявити проблеми та визначити необхідні зміни. Для оцінки легкості аналізу можна використовувати такі критерії, як середній час, необхідний для аналізу коду, та кількість виявлених помилок під час ревізії коду. Чим менше часу потрібно на аналіз коду і чим менше помилок виявляється, тим вищою є легкість аналізу.

Легкість внесення змін є ще однією важливою характеристикою підтримуваності. Вона оцінює, наскільки легко та швидко можна внести зміни в програмне забезпечення. Для вимірювання цієї характеристики можна використовувати середній час, необхідний для внесення змін, та кількість помилок, що виникають після внесення змін. Менший час на внесення змін і менша кількість помилок свідчать про високу легкість змін.

Тестованість програмного забезпечення вимірює, наскільки легко можна тестувати ПЗ для виявлення помилок і перевірки його коректної роботи. Показники тестованості можуть включати відсоток тестового покриття коду та кількість помилок, виявлених під час тестування. Високий відсоток тестового покриття і менша кількість невиявлених помилок свідчать про високу тестованість ПЗ. Тестованість також включає оцінку ефективності тестових сценаріїв і автоматизації тестування.

Час, необхідний для виправлення помилок, є важливим критерієм підтримуваності, який вимірює середній час, потрібний для виявлення і виправлення помилок у ПЗ. Менший час на виправлення помилок свідчить про високу підтримуваність, оскільки розробники можуть швидко реагувати на проблеми та усувати їх. Цю характеристику можна вимірювати шляхом аналізу журналів помилок і часу, витраченого на їх виправлення.

Частота оновлень є метрикою, що вимірює, як часто виходять нові версії або оновлення ПЗ. Часті оновлення можуть свідчити про активну підтримку продукту і швидке реагування на змінювані вимоги або виявлені проблеми. Однак надто часті оновлення можуть також свідчити про

нестабільність початкових версій, тому важливо знаходити баланс.

Доступність документації є критично важливою характеристикою підтримуваності, оскільки якісна документація полегшує розробникам розуміння і модифікацію ПЗ. Показники, що оцінюють доступність документації, можуть включати повноту та актуальність технічної документації, а також зручність її використання. Висока якість документації сприяє кращій підтримуваності ПЗ.

Здатність до адаптації є ще однією важливою характеристикою підтримуваності, яка оцінює, наскільки легко програмне забезпечення може бути адаптоване до нових вимог або умов експлуатації. Показники адаптації можуть включати час, необхідний для внесення адаптивних змін, і кількість необхідних змін для забезпечення сумісності з новими середовищами або платформами.

3.7 Показники портативності

Показники портативності програмного забезпечення дозволяють оцінити, наскільки легко програмний продукт може бути перенесений на інші платформи, середовища або конфігурації без втрати функціональності та продуктивності. Портативність є важливою характеристикою якості ПЗ, оскільки вона впливає на здатність програмного забезпечення адаптуватися до різних операційних систем, апаратних платформ та умов експлуатації.

Одним із ключових показників портативності є адаптованість. Цей показник вимірює, наскільки легко програмне забезпечення може бути змінене для роботи в іншому середовищі або на іншій платформі. Для оцінки адаптованості використовуються такі критерії, як час, необхідний для адаптації ПЗ, та кількість змін, необхідних для забезпечення сумісності з новим середовищем. Менший час на адаптацію і менша кількість необхідних змін свідчать про високу адаптованість.

Здатність до встановлення є ще одним важливим показником

портативності. Вона вимірює, наскільки легко програмне забезпечення може бути встановлене в новому середовищі. Для вимірювання цієї характеристики можна використовувати показники, такі як час, необхідний для встановлення ПЗ, та кількість проблем, що виникають під час встановлення. Менший час на встановлення і менша кількість проблем свідчать про високу здатність до встановлення.

Здатність до заміни оцінює, наскільки легко можна замінити одне програмне забезпечення іншим у певному середовищі без значних зусиль. Цей показник включає такі критерії, як сумісність з існуючими даними і процесами, а також час, необхідний для заміни. Висока здатність до заміни свідчить про те, що ПЗ може бути легко замінене іншим продуктом без значних перебоїв у роботі.

Сумісність з різними середовищами експлуатації є важливою характеристикою портативності, яка вимірює, наскільки добре ПЗ працює в різних операційних системах, апаратних платформах і конфігураціях. Для оцінки сумісності використовуються показники, такі як кількість підтримуваних платформ і середовищ, а також кількість проблем, що виникають при зміні середовища експлуатації. Висока сумісність свідчить про те, що ПЗ може бути використане в широкому спектрі умов без значних змін.

Інтероперабельність з іншими системами та компонентами також є важливим аспектом портативності. Цей показник вимірює здатність ПЗ працювати з іншими системами або компонентами в різних середовищах. Для оцінки інтероперабельності використовуються такі критерії, як кількість успішних інтеграцій з іншими системами і кількість проблем, що виникають під час інтеграції. Висока інтероперабельність свідчить про те, що ПЗ може без проблем взаємодіяти з іншими системами у різних умовах.

Ефективність перенесення є ще одним важливим показником портативності, який вимірює, наскільки ефективно ПЗ працює після перенесення на нову платформу або в нове середовище. Для оцінки

ефективності перенесення використовуються такі показники, як продуктивність і час відгуку ПЗ після перенесення. Висока ефективність перенесення свідчить про те, що ПЗ зберігає свою продуктивність і функціональність після перенесення.

Таким чином, показники портативності дозволяють здійснювати всебічний аналіз здатності програмного забезпечення до адаптації, встановлення, заміни і взаємодії з іншими системами в різних умовах експлуатації. Використання цих показників допомагає розробникам і менеджерам забезпечити високу якість ПЗ, що може бути легко перенесене і використане в різних середовищах, знижуючи витрати на підтримку і підвищуючи задоволеність користувачів.

3.8 Вимірювання якості продукту по SQuaRE

Алгоритм вимірювання якості програмного забезпечення пропонується на основі аналізу розділів моделі якості та вимірювання якості SQuaRE:

- крок 1 – розробити моделі якості по ISO/IEC 25010 для визначення відповідних характеристик якості програмного забезпечення;
- крок 2 – застосувати ISO/IEC 25023 для визначення показників якості для кожної характеристики;
- крок 3 – виміряти елементи показника якості за допомогою методів вимірювання відповідно до ISO/IEC 25021;
- крок 4 – визначити вибрані показники якості за допомогою функції вимірювання.

Приклади застосування стандартів SQuaRE для вимірювання якості програмного забезпечення представлені в таблиці 3.1.

Таблиця 3.1 – Приклади застосування стандартів SQuaRE для вимірювання якості програмного забезпечення

Характеристики/ підхарактеристики якості	Показник якості. Опис	Функція вимірювання	Елемент показника якості	Метод вимірювання
Функціональна повнота	Функціональне покриття. Яка частина зазначених функцій реалізована?	$X = 1 - A / B,$ A – кількість відсутніх функцій; B – кількість передбачених опцій	Кількість доступних функцій	Перегляд і аналіз окремих функцій системи/програмн огозабезпечення, доступних користувачеві з обмеженими можливостями для виклику і виконання, і підрахунок кількості функцій, які не вдалося успішно використовувати
Часові характеристики	Середній час відповіді. Скільки в середньому часу потрібно системі, щоб відповісти на призначене для користувача завдання або системну задачу?	$X = \sum_{i=1}^{n} A_i \Big/ n,$ A_i – час, витрачений системою на відповідь на конкретне завдання користувача або системну задачу при і-му вимірюванні, n – кількість виміряних відповідей	Тривалість	Тривалість залежить від загальної кількості часу і прив'язана до Міжнародної системи одиниць (VIM)

4 ПЕРСПЕКТИВИ ТА НАПРЯМКИ УДОСКОНАЛЕННЯ SQUARE

4.1 Рекомендації для розробників ПЗ

Під час розробки програмного забезпечення (ПЗ), особливо з огляду на вимоги до якості, розробники мають дотримуватися низки важливих рекомендацій для забезпечення успішного впровадження та ефективного функціонування продукту:

- стандартизація процесів розробки: використання відомих методологій розробки, таких як Agile або Scrum, для забезпечення систематичного підходу до розробки, тестування та впровадження ПЗ;

- використання автоматизації тестування: імплементація автоматизованих тестів для постійної перевірки функціональності, надійності і продуктивності ПЗ під час розробки;

- забезпечення кодування відповідно до стандартів: використання сучасних практик програмування, таких як чистий код і правила оформлення коду, для зменшення кількості помилок та підвищення читабельності програмного коду;

- моніторинг якості через всі етапи життєвого циклу ПЗ: постійне відстеження і аналіз даних про якість програмного продукту під час його розробки, тестування, випробувань і експлуатації;

- регулярні аудити і перевірки безпеки: впровадження регулярних аудитів коду та тестів безпеки для виявлення потенційних вразливостей і забезпечення захисту даних користувачів;

- постійне вдосконалення і оптимізація: включення механізмів збору зворотнього зв'язку від користувачів і регулярне вдосконалення функціональних можливостей та інтерфейсу ПЗ з урахуванням отриманих даних.

Рекомендації щодо удосконалення процесу оцінювання якості програмного забезпечення включають в себе кілька ключових аспектів. Перш

за все, важливо активно впроваджувати нові метрики та характеристики якості, які краще відповідають специфіці проектів та потребам клієнтів.

Другим важливим кроком є розвиток інтегрованих платформ і інструментів для автоматизації процесів оцінювання, що дозволяють знижувати витрати часу і зусиль на тестування і аналіз.

Також рекомендується регулярно аналізувати та оновлювати методології оцінювання, зокрема, враховуючи сучасні тенденції в розробці ПЗ, такі як Agile та DevOps.

Для забезпечення високої якості програмного забезпечення важливо акцентувати увагу на навчанні персоналу і підтримці стандартів SQuaRE у всіх етапах життєвого циклу розробки.

Ці рекомендації спрямовані на підвищення ефективності, точності та надійності процесів оцінювання якості програмного забезпечення, що відіграє ключову роль у досягненні успіху в сучасній індустрії програмної інженерії.

4.2 Перспективи розвитку стандартів SQuaRE

Стандарти оцінки якості програмного забезпечення SQuaRE (Software Quality Requirements and Evaluation) знаходяться на передньому краї індустрії, пропонуючи важливі стратегії для поліпшення якості і надійності програмних продуктів. Основними напрямками їх розвитку є: заведення нових метрик і характеристик якості, що відповідають сучасним вимогам ринку програмного забезпечення; розширення сфери застосування стандартів на нові технології та платформи, такі як хмарні рішення та мобільні додатки; покращення методологій оцінювання якості з метою забезпечення більш точних і об'єктивних результатів; стимулювання інтеграції стандартів SQuaRE з іншими міжнародними стандартами якості програмного забезпечення; розвиток інструментальних засобів для автоматизації процесів оцінювання та моніторингу якості програмних продуктів.

Ці напрями розвитку спрямовані на підтримку актуальності стандартів SQuaRE в умовах швидкозмінного ринку програмного забезпечення та на забезпечення ефективного впровадження для підтримки високої якості та конкурентоспроможності продуктів.

4.2.1 Потенційні напрямки вдосконалення

З напрямків вдосконалення можна виділити деякі з них:

- інтеграція нових технологій: адаптація стандартів SQuaRE до новітніх технологій, таких як штучний інтелект, блокчейн або інтернет речей, для вдосконалення методів оцінки якості програмного забезпечення;
- розширення масштабів застосування: впровадження стандартів SQuaRE для оцінки не лише окремих програмних продуктів, але й комплексних інформаційних систем і сервісів;
- забезпечення відкритості та гнучкості: розробка модульних підходів до використання стандартів SQuaRE, що дозволяють адаптувати їх до різних умов і потреб користувачів;
- зростання важливості безпеки: покращення аспектів безпеки в оцінці якості ПЗ з урахуванням зростаючих загроз кібербезпеці;
- вдосконалення інструментальних засобів: розробка нових інструментів і програмних платформ для автоматизації процесів тестування та аналізу якості.

Ці напрямки вдосконалення спрямовані на забезпечення актуальності та ефективності стандартів SQuaRE у сучасному інформаційному середовищі.

4.2.2 Можливі зміни у стандартах

Можливі зміни у стандартах SQuaRE можуть включати розширення області застосування, включення нових категорій ПЗ, таких як штучний

інтелект, інтернет речей або блокчейн. Такі зміни дозволяють стандартам відповідати сучасним вимогам індустрії.

Також можливе удосконалення метрик і характеристик якості, щоб врахувати нові технології та ринкові потреби. Це включає оновлення і розширення набору метрик, що використовуються для оцінки програмного забезпечення.

Зміни в процесах оцінювання також можуть бути важливими. Впровадження нових методологій, які враховують Agile та DevOps підходи до розробки ПЗ, може значно покращити ефективність оцінки якості продуктів.

До інших можливих змін входить покращення вимог до безпеки і приватності. З урахуванням зростаючих загроз кібербезпеці і вимог законодавства, стандарти SQuaRE можуть зміцнювати вимоги до захисту персональних даних і безпеки програмного забезпечення.

Інтеграція з іншими міжнародними стандартами також є важливою. Це дозволяє забезпечити сумісність і обмін кращими практиками в галузі оцінювання якості програмного забезпечення.

Усі ці зміни спрямовані на покращення якості, ефективності і відповідності сучасним вимогам стандартів SQuaRE, що допомагає зберігати конкурентоспроможність індустрії програмного забезпечення.

ВИСНОВКИ

У кваліфікаційній роботі було досліджено та проаналізовано оцінювання якості програмних засобів на основі стандартів SQuaRE (Software Product Quality Requirements and Evaluation). Під час дослідження було досягнуто наступних висновків.

Стандарти SQuaRE (ISO/IEC 25000) забезпечують комплексний підхід до оцінювання якості програмного забезпечення, охоплюючи всі аспекти від визначення вимог до якості до методів їх оцінювання. Це робить їх універсальним інструментом для розробників, менеджерів проєктів та тестувальників ПЗ.

Модель якості, визначена стандартом ISO/IEC 25010, охоплює такі основні характеристики, як функціональна придатність, надійність, зручність використання, ефективність, підтримуваність та портативність. Це дозволяє враховувати всі ключові аспекти якості при оцінюванні програмного продукту.

Впровадження стандартів SQuaRE має численні переваги, включаючи покращення якості ПЗ, підвищення задоволеності користувачів та зниження витрат на виправлення помилок. Однак, існують і певні обмеження та виклики, такі як необхідність значних ресурсів для впровадження та необхідність навчання персоналу.

Процес оцінювання якості за SQuaRE включає визначення критеріїв якості, вибір та адаптацію метрик, вимірювання якості та аналіз результатів. Це забезпечує систематичний та послідовний підхід до оцінювання, що підвищує точність та надійність отриманих результатів.

Кейс-стаді, проведене в роботі, показало, що застосування стандартів SQuaRE для оцінювання якості конкретного програмного продукту дозволяє виявити слабкі місця та надати конкретні рекомендації щодо їх усунення. Це підтверджує ефективність та практичну цінність SQuaRE у реальних

проєктах.

Аналіз показав, що стандарти SQuaRE мають значні переваги у порівнянні з іншими стандартами якості, такими як ISO/IEC 9126, завдяки більш детальному підходу до оцінювання якості та більш широкому охопленню різних аспектів якості ПЗ.

Стандарти SQuaRE продовжують розвиватися, враховуючи нові технологічні тенденції та вимоги ринку. Удосконалення стандартів включає інтеграцію з іншими сучасними методологіями розробки ПЗ, такими як Agile та DevOps, що дозволить підвищити їхню актуальність та ефективність.

На основі проведеного дослідження можна рекомендувати розробникам ПЗ активне впровадження стандартів SQuaRE у процеси розробки та оцінювання якості програмних продуктів. Це сприятиме покращенню якості програмного забезпечення, підвищенню задоволеності користувачів та конкурентоспроможності на ринку. Подальші дослідження можуть бути спрямовані на адаптацію стандартів SQuaRE до специфічних вимог різних галузей та розробку нових метрик для оцінювання якості ПЗ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Вимірювання якості програмного забезпечення на основі стандартів SQuaRE / О. В. Запорожець, Н. В. Штефан, І. С. Яценко // Системи управління, навігації та зв'язку. – 2024. – Вип. 2 . – С. 39-43.

2. ISO/IEC 25000:2014 Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Guide to SQuaRE.

3. Kazuhiro Esaki. Introduction of Quality Requirement and Evaluation Based on ISO/IEC SQuaRE Series of Standard. // Global Perspectives on Engineering Management. May 2013, Vol. 2 Iss. 2, pp. 52-59.

4. ISO/IEC/IEEE 15939:2017 Systems and software engineering. Measurement process.

5. JCGM 200:2012. International vocabulary of metrology – Basic and general concepts and associated terms (VIM).

6. ISO/IEC 25010:2023 Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Product quality models.

7. ISO/IEC 25020:2019 Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Quality measurement framework.

8. ISO/IEC 25022:2016 Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of quality in use.

9. ISO/IEC 25023:2016 Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality.

10. ISO/IEC 25024:2016 Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of data quality.

11. ISO/IEC 25021:2016 Systems and software engineering. Systems and

software Quality Requirements and Evaluation (SQuaRE). Quality measure elements.

12. Вимірювання якості програмного забезпечення на основі міжнародних стандартів / Н. В. Штефан, О. В. Запорожець // Тезисы докладов VIII Международной научно-технической конференции «Метрология, информационно-измерительные технологии и системы» (МИИТС-2021), г. Харьков, 20-21 мая 2021 г. – С. 86–87.

13. Измерение качества программного обеспечения на основе международных стандартов / О. В. Запорожец, Н. В. Штефан // Радиотехника : Всеукр. межвед. науч.-техн. сб. – 2021. – Вып. 206. – С. 152–157.

14. Software Quality Model on the Base of SQuaRE Standards / N. Shtefan, O. Zaporozhets // Radiotekhnika : All-Ukr. Sci. Interdep. Mag. – 2021. – № 207. – P. 159 – 165.

15. Інженерія якості програмного забезпечення : навчальний посібник / Г. В. Табунщик, Р. К. Кудерметов, Т. І. Брагіна. – Запоріжжя: ЗНТУ, 2013. – 180 с.

16. Вавілов Є. В. Серія стандартів SQuaRE як основа забезпечення вимог до якості та оцінки програмних засобів / Збірник наукових праць ОДАТРЯ. – № 1(6). – 2015. – С. 129-139.