Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Інфокомунікацій_____
(повна назва)
Кафедра_____Інформаційно-вимірювальних технологій_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Розробка методики оцінювання

програмного забезпечення

Виконав:
студент _2_ курсу, групи ____$ЯСС_м$-21-1___
_____Столяр І.О._____
(прізвище, ініціали)

Спеціальність __152 Метрологія та_____
___інформаційно-вимірювальна техніка_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма ___Якість, стандартизація
та сертифікація_____
(повна назва освітньої програми)

Керівник ___доц. Козлов Ю.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ ____Захаров І.П.____
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет_____Інфокомунікацій_____

Кафедра_____Інформаційно-вимірювальних технологій_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність____152 Метрологія таінформаційно-вимірювальна техніка___
(код і повна назва)

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

Освітня програма_____Якість, стандартизація та сертифікація_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові___Столяру Іллі Олексійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи ___ Розробка методики оцінювання програмного забезпечення____

_____

_____

затверджена наказом університету від _30_ __листопада___ 2022 р. № _1538Ст__

2. Термін подання студентом роботи до екзаменаційної комісії __ _____ 2022 р.

3. Вихідні дані до роботи _____

Відповідність розроблюваної методики стандартам ISO/IEC 25000: Software Engineering – Software Product Quality Require-ments and Evaluation (SQuaRE)

_____

4. Перелік питань, що потрібно опрацювати в роботі _____

4.1. Якість програмного забезпечення

4.2. Показники якості програмного забезпечення

4.3 Методика оцінювання якості програмного забезпечення

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів,комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____

_____

_____

6. Консультанти розділів роботи (п.6 включається до завдання 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
Основна частина	доц. Козлов Ю.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз сучасного стану проблеми та методів її вирішення	15.11.2022	
2	Підготовка довідкових матеріалів та даних для розробки основної частини	19.11.2022	
3	Розробка основної частини	03.12.2022	
6	Написання пояснювальної записки	06.12.2022	
7	Підготовка презентації	09.12 2022	
8	Представлення закінченої дипломої роботи на кафедру	15.12.2022	

Дата видачі завдання _08_ ___листопада___2022 р.

Студент _____
(підпис)

Керівник роботи _____ ___доц. Козлов Ю.В.___
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи містить 65 сторінок, 6 рисунків, 1 таблицю, перелік посилань з 26 назв.

Галузь інформаційних технологій є однією з найбільш розвинених у сучасному житті. Побут людей в даний час влаштований таким чином, що навіть далека від інформаційних технологій людина щодня використовує їх задля досягнення цілей.

Більше того, комп'ютерні програми на сьогоднішній день використовуються навіть у тих галузях, де ціна помилки дуже висока, тому питання аналізу якості програмного забезпечення є досить актуальним.

Кваліфікаційна робота присвячена питанням аналізу програмного забезпечення, що дозволяє оцінити якість програмних продуктів і мінімальний рівень якості при неповній інформації про програмні системи.

Мета роботи – розробка методики оцінювання якості програмного забезпечення.

У роботі запропоновано оригінальний підхід до оцінювання якості програмного забезпечення на основі експертної оцінки. В основі моделі лежить метод Раша оцінки латентних змінних. Модель дозволяє обчислити підсумковий показник якості кожного програмного продукту за лінійною шкалою і провести аналіз оціночних критеріїв.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЯКІСТЬ, ОЦІНОЧНІ КРИТЕРІЇ, ЕКСПЕРТНЕ ОЦІНЮВАННЯ

ABSTRACT

The explanatory note to the master's qualification work contains 65 pages, figures, 1 table, a list of references from 26 titles.

Subject of information technologies is one of the most widespread in today's life. They will beat people in this hour of power with such a rank that it will take them far from information technology to achieve their goals.

Moreover, computer programs for this day vikoristovuyutsya in quiet closets, the cost of a pardon is even higher, that nutritional analysis of the quality of software security should be actual.

The qualification of the work is related to the analysis of software security, which allows to evaluate the quality of software products and the minimum level of quality in case of incorrect information about the software system.

Purpose of the work - development of a methodology for assessing the quality of software security.

The robot has been propagated with the original id to evaluate the quality of software security on the basis of expert evaluation. The model is based on the Rasch method for assessing latent changes. The model allows you to calculate the bag indicator of the quality of the skin software product on a linear scale and analyze the evaluation criteria.

SOFTWARE, EVALUATION, EVALUATION CRITERIA, EXPERT EVALUATION

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АС - автоматизована система;

ЕОМ - електронно-обчислювальна машина;

ІзОД - інформація з обмеженим доступом;

КЗЗ - комплекс засобів захисту;

КСЗІ - комплексна система захисту інформації;

НД ТЗІ - нормативний документ системи технічного захисту інформації;

НСД - несанкціонований доступ;

ОС - обчислювальна система;

ПЕОМ - персональна електронно-обчислювальна машина;

ПЗ - програмне забезпечення;

СЗІ - служба захисту інформації;

SQM - Software Quality Management

SQA - Software Quality Assurance

V&V - Verification and Validation

ВСТУП

Вимоги до якості програмних засобів постійно підвищуються. Процеси розробки, придбання та впровадження складних систем, до яких належать зокрема програмні комплекси, повинні перебувати під жорстким управлінським контролем. В даний час практично у всіх організаціях забезпечується контроль найважливіших характеристик, пов'язаних із виробництвом та використанням програмних продуктів, таких як час, фінансові засоби, ресурси тощо. Однак у більшості випадків поза межами сфери контролю виявляється найбільш важлива характеристика програмних продуктів, заради якої, власне і здійснюються витрати часу, фінансових засобів та ресурсів - це якість продукту, оскільки «неможливо контролювати те, що не можна виміряти» (“ You cannot control what you cannot measure ”).

Одним із підходів для оцінки програмних засобів є оцінка відповідних атрибутів якості, визначених у серії міжнародних стандартів ДСТУ ISO 9126 «Інформаційна технологія – Оцінка програмної продукції».

Стандарти визначають базову термінологію та загальний підхід до проблеми оцінки якості програмних засобів (характеристики якості, метрики для їх виміру, методологію оцінки), що дозволяє зменшити невизначеність при спільній роботі кількох організацій (замовники розробки, розробники, незалежні оцінювачі). Застосування міжнародних стандартів ІСО МЕК у свою чергу зручно тим, що підходи, що використовуються, можуть бути використані при роботі із зарубіжними партнерами.

Відсутність можливості встановлення повного контролю викликає зростання кількості необґрунтованих рішень, збільшує фінансові та проектні ризики, пов'язані з розробкою та впровадженням систем.

Проте в даний час вже існують організації, в яких накопичено досить великий досвід використання метрик в управлінні якістю програмних продуктів, що розробляються та впроваджуються. Використання апробованих підходів в

управлінні якістю розробки та впровадження великих програмних систем значно підвищує передбачуваність проектів, знижує фінансові та ресурсні витрати.

Серед використовуваних метрик якості програмного забезпечення є універсальні метрики, які можна застосувати практично до всіх видів програмного забезпечення. У той самий час більшість найважливіших метрик в успішних проектах розробляється індивідуально з урахуванням особливостей проекту та характеристик предметної області.

Також особливу увагу слід приділити методикам порівняння програмних продуктів від різних розробників, які мають однаковий або схожий функціонал

1 ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Поняття якості

Що таке якість і чому вона має бути настільки глибоко представлена? Протягом багатьох років окремі автори та цілі організації визначали термін “якість” по-різному. Філ Кросбі (Phil Crosby) в 1979 році дав визначення якості як "відповідність користувачам вимогам" (припускає, що вимоги повинні бути настільки чітко визначені, що вони не можуть бути зрозумілі та інтерпретовані некоректно.). Уотс Хемпфрі (Watts Hamphrey) описує якість як "досягнення відмінного рівня придатності до використання" (бере до уваги вимоги та очікування кінцевих користувачів продукту, які очікують, що продукт або сервіс буде зручним для їх потреб). Компанія IBM, у свою чергу, ввела в обіг фразу "якість, керована ринковими потребами" (market-driven quality). Критерій Белдріджа (Baldrige) для організаційної якості використовує схожу фразу - "якість, що задається споживачем" ("customer-driven quality"), розглядаючи задоволення споживача як головне міркування щодо якості. Найчастіше, поняття якості використовується відповідно до визначення системи менеджменту якості ISO 9001 як "ступінь відповідності властивих характеристик вимогам".

Поняття "якість", насправді, не настільки очевидне і просто, як це може здатися на перший погляд. Для будь-якого інженерного продукту існує безліч інтерпретацій якості, залежно від конкретної системи координат. Безліч цих точок зору необхідно обговорити та визначити на етапі вироблення вимог до програмного продукту. Характеристики якості можуть вимагатися тією чи іншою мірою, можуть бути відсутніми або можуть задавати певні вимоги, все це може бути результатом певного компромісу, що цілком перегукується з розумінням “прийнятної якості”, як менш жорсткої точки зору на забезпечення якості, як досягнення досконалості.

Зараз існує кілька визначень якості, які загалом сумісні один з одним. Наведемо найпоширеніші:

Визначення ISO: Якість - це повнота властивостей та характеристик продукту, процесу або послуги, які забезпечують здатність задовольняти заявлені або потреби.

Визначення IEEE: Якість програмного забезпечення - це ступінь, в якому воно має необхідну комбінацію властивостей.

Якість ПЗ - це відносне поняття, яке має сенс тільки при врахуванні реальних умов його застосування, тому вимоги до якості ставляться відповідно до умов і конкретної сфери їх застосування. Воно характеризується трьома аспектами: якість програмного продукту, якість процесів ЖЦ та якість супроводу чи впровадження (рис 1.1).

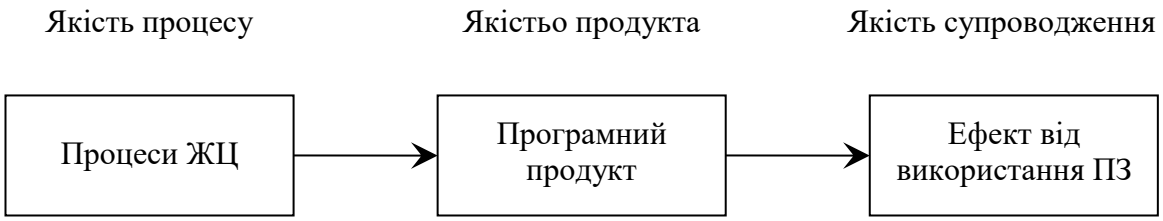

Рисунок 1.1 - Основні аспекти якості ПЗ

Аспект, пов'язаний із процесами ЖЦ, визначає ступінь формалізації, достовірності самих процесів ЖЦ розробки ПЗ, а також верифікацію та валідацію проміжних результатів на цих процесах. Пошук та усунення помилок у готовому ПЗ проводиться методами тестування, які знижують кількість помилок та підвищують якість цього продукту.

Якість продукту досягається процедурами контролю проміжних продуктів процесах ЖЦ, перевіркою їх у досягнення необхідної якості, і навіть методами супроводу продукту. Ефект від застосування ПС значною мірою залежить від знань обслуговуючого персоналу функцій продукту та правил їх виконання.

Прийнятна якість — це бажана ступінь досконалості продукту (послуги), що створюється, здатна задовольнити користувачів і досяжна в рамках заданих проектних обмежень.

Якість у проектній діяльності:

- управління вимогами («атрибути якості» як категорія нефункціональних вимог);
- Тестування (т.зв. напрацювання на відмову, такі метрики як MTTF — Mean Time To Failure , тобто середній час між виявленими збоями системи тощо.)

"Прийнятна якість" можна порівнювати з рівнем обслуговування в рамках заданого SLA - Service Level Agreement . Тобто, прийнятна якість може розглядатися як кількісно виражений компроміс між замовником та виконавцем щодо характеристик продукту, створюваного виконавцем на користь рішення завдань замовника з урахуванням інших обмежень проекту (зокрема, вартістю, що часто називається як « cost of quality » – «вартість якості»).

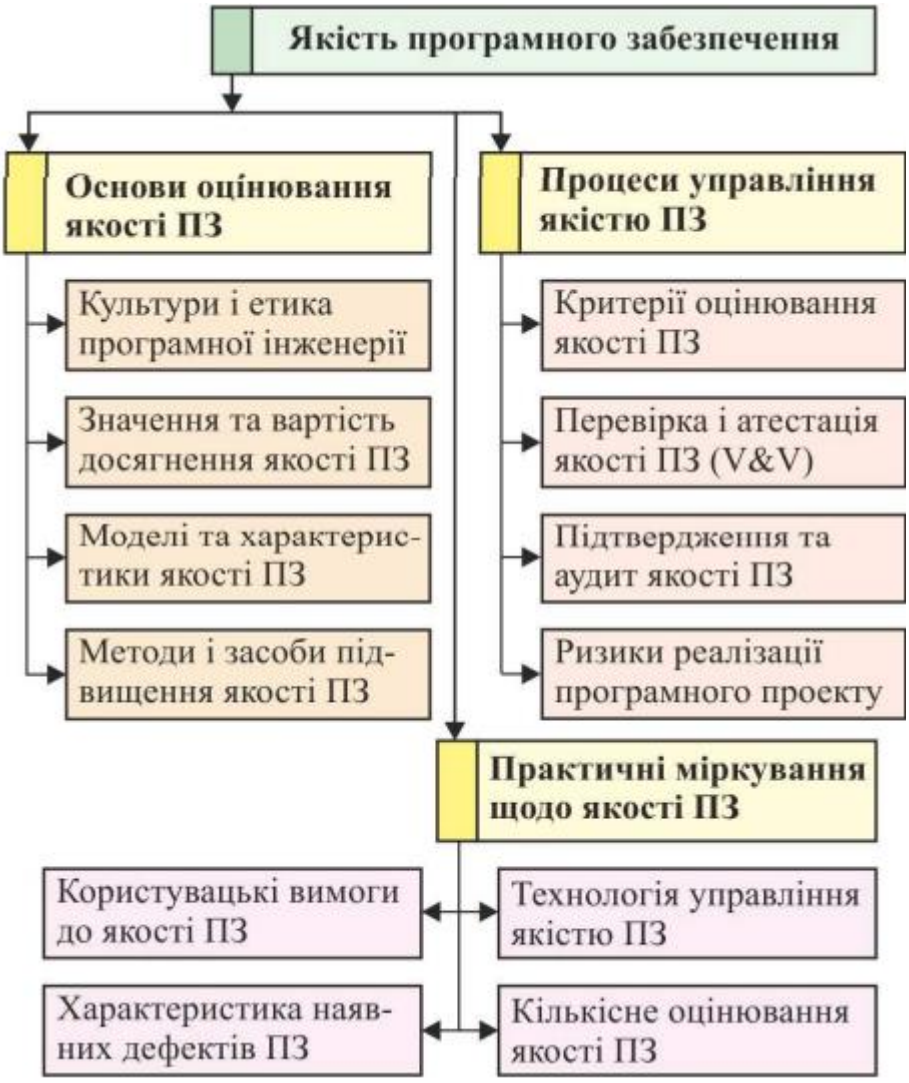

Рисунок 1.2 - Область знань — Якість програмного забезпечення

Інженери повинні розуміти зміст, що вкладається в концепцію якості, характеристики та значення якості щодо програмного забезпечення, що розробляється або супроводжується.

Важливою ідеєю є те, що програмні вимоги визначають необхідні характеристики якості програмного забезпечення, а також впливають на методи кількісної оцінки та сформульовані для оцінки цих характеристик відповідні критерії приймання.

Очікується, що інженери програмного забезпечення сприймають питання якості програмного забезпечення як частину своєї професійної культури. Етичні аспекти можуть відігравати значну роль у забезпеченні якості програмного забезпечення, культурі та відношенні інженерів до своєї роботи. IEEE Computer Society та ACM розробили кодекс етики (“моральний кодекс” – code of ethics) та професійної практики, заснований на восьми принципах, що допомагають інженерам зміцнити їхнє ставлення до якості та незалежність у вирішенні питань забезпечення гідної якості створюваних програмних продуктів у їх повсякденній роботі.

Поняття "якість", насправді, не настільки очевидне і просто, як це може здатися на перший погляд. Для будь-якого інженерного продукту існує безліч інтерпретацій якості, залежно від конкретної системи координат. Безліч цих точок зору необхідно обговорити та визначити на етапі вироблення вимог до програмного продукту. Характеристики якості можуть бути потрібні тією чи іншою мірою, можуть бути відсутніми або можуть задавати певні вимоги, все це може бути результатом певного компромісу.

Вартість якості (cost of quality) може бути диференційована на:

- вартість попередження <дефектів> (prevention cost),
- вартість оцінки (appraisal cost),
- вартість внутрішніх збоїв (internal failure cost),
- вартість зовнішніх збоїв (external failure cost).

Рушійною силою програмних проектів є бажання створити програмне забезпечення, що має певну цінність. Цінність програмного забезпечення може виражатися у формі вартості, а може і ні. Замовник, як правило, має уявлення про максимальні вартісні вкладення, повернення яких очікується у разі досягнення основних цілей створення програмного забезпечення. Замовник може також мати певні очікування щодо якості ПЗ. Іноді замовники не замислюються про питання якості та пов'язану з ними вартість. Чи є характеристики якості чисто декоративними чи, все ж таки, це невід'ємна частина програмного забезпечення? Відповідь, ймовірно, знаходиться десь посередині, як майже завжди буває в таких випадках, і є предметом обговорення ступеня залучення замовника до прийняття рішень та повного розуміння замовником вартості та вигоди, пов'язаної з досягненням того чи іншого рівня якості. В ідеальному випадку більшість таких рішень повинні прийматися в процесі роботи з вимогами, однак ці питання можуть підніматися протягом усього життєвого циклу програмного забезпечення. Не існує якихось «стандартних» правил того, як саме необхідно приймати такі рішення. Однак інженери повинні бути здатні представити різні альтернативи та їх вартість.

1.2 Моделі та характеристики якості (Models and Quality Characteristics)

ISO/IEC 25010 визначає три пов'язані моделі якості програмного забезпечення

- внутрішня якість,
- зовнішня якість та
- якість у процесі експлуатації, а також набір відповідних робіт з оцінки якості програмного забезпечення (ISO14598-98 Software Product Evaluation).

Якість процесів програмного забезпечення (Software engineering process quality)

Управління якістю (software quality management) та якість процесів програмної інженерії (software engineering process quality) мають безпосереднє відношення до якості створюваного програмного продукту.

Існує два найважливіші стандарти в галузі якості програмного забезпечення.

- TickIT стосується розгляду загальної системи менеджменту якості ISO 9001-00 у додатку до програмних проектів.
- Інший важливий стандарт – CMMI , який надає рекомендації щодо вдосконалення процесу. Безпосередньо з управлінням якістю пов'язані процесні галузі (області компетенції) CMMI:
- забезпечення якості процесу та продукту (process and product quality assurance , категорія процесів CMMI " Support "),
- перевірка (verification)і
- атестація (validation).

При цьому, CMMI класифікує огляд (review) і аудит (audit) як методи верифікації, але не як самостійні процеси.

Дані стандарти все ж таки розглядають як взаємодоповнюючі та, що сертифікація за ISO 9001 допомагає у досягненні старших рівнів зрілості за CMMI.

Насамперед інженери повинні визначити цілі створення програмного забезпечення. У цьому контексті особливо важливо пам'ятати, що вимоги замовника — первинні і містять вимоги щодо якості, а не тільки функціональності (функціональні вимоги). Таким чином, інженери відповідальні за отримання вимог до якості, які не завжди представлені явно, а також обговорення їх важливості та ступеня складності їх досягнення. Усі процеси, асоційовані з якістю (наприклад, збирання, перевірка та підвищення якості), повинні проектуватися з урахуванням цих вимог та несуть на собі тягар додаткових витрат (як важливу складову частину вартості програмного забезпечення).

Стандарт ISO 25010 (Software Engineering - Product Quality , Part 1: Quality Model) визначає для двох із трьох описаних у ньому моделей, пов'язані

характеристики та « суб - характеристики» якості, а також метрики, корисні для оцінки якості програмних продуктів.

Розуміння терміна “продукт” розширено включенням всіх артефактів, створюваних на виході всіх процесів, що використовуються створення кінцевого програмного продукту. Прикладами продукту є (але не обмежуються цим):

- повна специфікація системних вимог (system requirements specification),
- специфікація програмних вимог для програмних компонентів системи (software requirements specification , SRS),
- моделі,
- код,
- тестова документація,
- звіти, створювані внаслідок робіт з аналізу якості.

Хоча, найчастіше термін якість використовується щодо кінцевого продукту та поведінки системи в процесі експлуатації, гарною інженерною практикою є вимога до того, щоб відповідність заданим характеристикам якості оцінювалося і для проміжних результатів/продуктів життєвого циклу в рамках усіх процесів програмної інженерії.

1.3 Підвищення якості (Quality Improvement)

Якість програмного забезпечення може підвищуватись за рахунок ітеративного процесу постійного покращення. Це вимагає контролю, координації та зворотного зв'язку в процесі управління багатьма одночасно виконуваними процесами:

1. процесами життєвого циклу,
2. процесом виявлення, усунення та запобігання збоям/дефектам та
3. процесів покращення якості.

До програмної інженерії застосовуються теорії та концепції, що лежать в основі вдосконалення якості. Наприклад, запобігання та рання діагностика

помилок, постійне вдосконалення (continuous improvement) та увага до вимог замовника (customer focus), що становлять принцип “ building in quality ”. Ці концепції ґрунтуються на роботах експертів з якості, що прийшли до думки, що якість продукту безпосередньо пов'язана з якістю процесів, що використовуються для його створення.

Такі підходи, як **TQM** (Total Quality Management – загальне управління якістю) та **PDCA** (Plan , Do , Check , Act – Планування, Дія, Перевірка, Реакція/Коректування) є інструментами досягнення завдань, пов'язаних з якістю. Підтримка менеджменту допомагає у виконанні процесів, оцінці продуктів та отриманні всіх необхідних даних. Крім цього, програма вдосконалення (improvement program , зазвичай є цільовою і охоплює роботу підрозділу чи організації, в цілому) детально ідентифікує всі дії та проекти щодо поліпшення окремих аспектів діяльності в рамках певного періоду часу, за який такі проекти можна здійснити з успішним вирішенням відповідних завдань. При цьому, підтримка менеджменту означає, що всі проекти з поліпшення мають достатні ресурси для досягнення поставленої мети. Підтримка менеджменту тісно пов'язана з реалізацією активної взаємодії в колективі, і повинна запобігати виникненню потенційних проблем (і пасивної чи навіть активної протидії реалізації програми вдосконалення або окремих її проектів). Формування робочих груп, підтримка менеджерів середньої ланки та виділені ресурси лише на рівні проекту – ці питання обговорюються у сфері знань “Процес програмної інженерії”.

Управління якістю програмного забезпечення (SQM, Software Quality Management) застосовується до всіх аспектів процесів, продуктів та ресурсів. SQM визначає процеси, власників процесів, а також вимоги до процесів, вимірювання процесів та їх результатів, плюс – канали зворотного зв'язку.

Процеси управління якістю містять багато процесів. Деякі з них дозволяють безпосередньо знаходити дефекти, тоді як інші допомагають визначити, де саме може бути важливо провести детальніші дослідження, після чого, знову ж таки,

проводяться роботи з безпосереднього виявлення помилок. Багато дій також можуть вестися з метою досягнення тих і інших цілей.

Планування якості програмного забезпечення включає:

1. Визначення необхідного продукту у термінах характеристик якості.
2. Планування процесів отримання необхідного продукту.

Ці процеси відрізняються від процесів SQM як таких, які, у свою чергу, спрямовані на оцінку планованих характеристик якості, а не на реальну реалізацію цих планів. Процеси управління якістю повинні адресуватися питанням, наскільки добре продукт задовольнятиме потребам замовника та вимогам зацікавлених осіб, мати цінність для замовника та зацікавлених осіб та якість, необхідну для відповідності сформульованим вимогам до програмного забезпечення.

SQM може використовуватися для оцінки та кінцевих та проміжних продуктів. Деякі зі спеціалізованих процесів SQM визначено у стандарті 25000:

- Процес забезпечення якості (quality assurance process);
- Процес верифікації (verification process);
- Процес атестації (validation process);
- Процес спільного аналізу (joint review process);
- Процес аудиту (audit process).

Всі ці процеси підтримують прагнення до досягнення якості та, крім того, допомагають у пошуку можливих помилок. Однак вони відрізняються в тому, на чому концентрують увагу.

Процеси SQM складаються із завдань та технік, призначених для оцінки того, як починають реалізовуватись плани щодо створення програмного забезпечення та наскільки добре проміжні та кінцеві продукти відповідають заданим вимогам. Результати виконання цих завдань подаються у вигляді звітів для менеджерів перед тим, як будуть вжиті відповідні коригувальні дії. Управління SQM-процесом ведеться з упевненості, що дані звітів точні.
Як описано в цій галузі знань, процеси SQM тісно пов'язані між собою. Вони можуть перекриватися, інколи ж навіть і поєднуватися. Вони здаються

реактивними за своєю природою, оскільки вони розглядають процеси у тих отриманої практики і вже вироблені продукти. Однак, вони відіграють головну роль на стадії планування, будучи проактивними як процеси та процедури, необхідні для досягнення характеристик та рівня якості, затребуваних зацікавленими особами програмного забезпечення.

Управління ризиками також може відігравати значну роль для випуску якісного програмного забезпечення. Включення “регулярного” аналізу ризиків і відповідних технік управління ризиками до процесів життєвого циклу програмного забезпечення може збільшити потенціал виробництва якісного продукту. Докладнішу інформацію щодо управління ризиками можна знайти в галузі знань “Управління програмною інженерією”.

1.4 Підтвердження якості програмного забезпечення (Software Quality Assurance , SQA)

Процеси SQA забезпечують підтвердження того, що програмні продукти та процеси життєвого циклу проекту відповідають заданим вимогам. Таке підтвердження проводиться на основі планування (planning), постановки (enacting) і виконання (performing) набору дій, спрямованих на те, щоб якість стала невід'ємною частиною програмного забезпечення. Такий погляд має на увазі ясне і точне формулювання проблеми, а також те, що визначені та чітко виражені, повні та однозначно інтерпретовані вимоги до відповідного <програмного> рішення. SQA домагається забезпечення якості в процесі розробки та супроводу за рахунок виконання різних дій на всіх етапах життєвого циклу, що дозволяє ідентифікувати проблеми ще на ранніх стадіях, які практично неминучі в будь-якій складній діяльності.

Управління ризиками (Risk Management) є серйозним додатковим інструментом для забезпечення якості програмного забезпечення.

SQA концентрується на процесах. Роль SQA полягає в тому, щоб забезпечити відповідне планування процесів, подальше виконання процесів на основі заданого плану та проведення необхідних вимірювань процесів із передачею результатів вимірювань заінтересованим сторонам (організаційним структурам та особам).

SQA-план визначає засоби, які будуть використовуватися для забезпечення відповідності продукту, що розробляється заданим користувачам вимогам з максимальним рівнем якості, можливим при заданих обмеженнях проекту.

Для того, щоб цього досягти, в першу чергу необхідно, щоб цілі якості були чітко визначені та розуміються (а також однозначно інтерпретовані, що є обов'язковою умовою будь-яких цілей та відповідних вимог). Це, в обов'язковому порядку, має бути відображено у відповідних планах управління проектом, розробки та супроводу.

Конкретні роботи та завдання щодо забезпечення якості структуруються з деталізацією вимог щодо їх вартості та асоційованих ресурсів, цілей з погляду управління та відповідним розкладом у контексті цілей, заданих планами управління, розробки та супроводу. План SQA ідентифікує документи, стандарти, практики та угоди, які застосовуються при контролі проекту, а також те, як ці аспекти перевірятимуться та відстежуватимуться для забезпечення достатності та відповідності заданому плану. SQA-план ідентифікує метрики, статистичні техніки, процедури формування повідомлень про проблеми та проведення коригувальних дій, такі засоби як інструменти, техніки та методології, питання безпеки фізичних носіїв, тренінги, а також формування звітності та документації, що стосуються питань SQA.

Крім того, SQA-план стосується і питань робіт із забезпечення якості, що стосуються інших типів діяльності, описаних у різних планах зі створення програмного забезпечення, до яких також належать постачання, встановлення, обслуговування замовних та/або тиражованих/готових програмних рішень (commercial off-the-shelf), необхідні для даного проекту програмного забезпечення.

SQA-план може містити необхідні для забезпечення якості критерії приймання програмного забезпечення та дії щодо формування звітності та управління і контролю над роботами.

1.5 Перевірка (верифікація) та атестація (Verification and Validation)

Перевірка та атестація програмного забезпечення – впорядкований підхід в оцінці програмних продуктів, що застосовується протягом усього життєвого циклу. Зусилля, що докладаються в рамках робіт з перевірки та атестації, спрямовані на забезпечення якості як невід'ємної характеристики програмного забезпечення та задоволення потреб користувача. V&V безпосередньо адресується питанням якості програмного забезпечення та використовує відповідні техніки тестування для виявлення тих чи інших дефектів. V&V може застосовуватися для проміжних продуктів, однак, у тому обсязі, який відповідає проміжним крокам процесів життєвого циклу.

Процес V&V визначає якою мірою продукт (результат) тих чи інших робіт з розробки та супроводу відповідає вимогам, сформульованим у межах цих робіт, а кінцевий продукт задовольняє заданим цілям та користувальницьким вимогам.

Верифікація - спроба забезпечити правильну розробку продукту (продукт побудований правильним чином; зазвичай, для проміжних, іноді, для кінцевого продукту), в тому значенні, що продукт, що отримується в рамках відповідної діяльності, відповідає специфікаціям, заданим у процесі попередньої діяльності.

Атестація (валідація) – спроба забезпечити створення правильного продукту (побудований правильний продукт; зазвичай, у контексті кінцевого продукту), з погляду досягнення поставленої мети.

Обидва процеси – верифікація та атестація – починаються на ранніх стадіях розробки та супроводу. Вони забезпечують дослідженню (експертизу) ключових можливостей продукту як у контексті безпосередньо попередніх результатів (проміжних продуктів), і з погляду задоволення відповідних специфікацій. Метою

планування V&V є забезпечення процесів верифікації та атестації необхідними ресурсами, чітке призначення ролей та обов'язків. Отримуваний план V&V документує і описує різні ресурси, ролі та дії, а також використовувані техніки та інструменти.

План також стосується аспектів управління, комунікацій (взаємодії), політик та процедур щодо дій з верифікації та атестації та їх взаємодії. Крім того, у ньому можуть бути відображені питання формування звітності щодо дефектів та документування вимог.

1.6 Оцінка (огляд) та аудит (Review and Audits)

Існує п'ять типів оцінок та аудитів:

- Управлінські оцінки (management reviews)
- Технічні оцінки (technical reviews)
- Інспекції (inspections)
- Прогонки (walk-throughs)
- Аудити (audtis)
-

1.6.1 Управлінські оцінки (Management Reviews)

Призначення управлінських оцінок полягає у відстеженні розвитку проекту/продукту, визначення статусу планів та розкладів, затвердження вимоги та розподілу ресурсів, або оцінки ефективності управлінських підходів, що використовуються для досягнення поставлених цілей.

Управлінські оцінки підтримують прийняття рішень щодо внесення змін та виконання коригуючих дій, необхідних у процесі виконання програмного проекту.

Управлінські оцінки визначають адекватність планів, розкладів та вимог, водночас, контролюючи їх прогрес чи невідповідність. Ці оцінки можуть виконуватися щодо продукту, будучи фіксованими у формі звітів аудиту, звітів про стан (розвиток), V&V-звітів, а також різних типів планів - управління ризиками

проекту/проектного управління, конфігураційного управління, безпеки програмного забезпечення (safety), оцінки ризиків тощо.

1.6.2 Технічні оцінки (Technical Reviews)

Призначенням технічних оцінок є дослідження програмного продукту визначення його придатності використання у належних цілях. Мета полягає в ідентифікації розбіжностей із затвердженими специфікаціями та стандартами. Для забезпечення технічних оцінок необхідний розподіл наступних ролей: особа, яка приймає рішення (decision-maker); лідер оцінки (review leader); реєстратор (recorder); а також технічний персонал, який підтримує (безпосередньо виконує) дії з оцінки.

Технічна оцінка вимагає, обов'язково, наявності наступних вхідних даних:

- Формулювання цілей
- Конкретного програмного продукту (що оцінюється)
- Заданого плану проекту (плану управління проектом)
- Списку проблем (питань), асоційованих з продуктом
- Процедури технічної оцінки

Команда технічної оцінки слідує заданій процедурі оцінки. Кваліфіковані (з технічної точки зору) особи репрезентують огляд продукту (представляючи команду розробки).

Дослідження продукту проводиться протягом однієї і більше зустрічей (між тими, хто представляє товар і тими, хто оцінює). Технічна оцінка завершується після того, як виконані всі запропоновані події з дослідження продукту.

1.6.3 Інспекції (Inspections)

Призначення інспекцій полягає у виявленні та ідентифікації аномалій у програмному продукті. Існує дві серйозні відмінності інспекцій від оцінок (управлінської та технічної):

1. Особи, які займають управлінські позиції (менеджери) щодо будь-яких членів команди інспектування, не повинні брати участь в інспекціях.

2. Інспекція має вестись під керівництвом неупередженого (незалежного від проекту та його цілей) лідера, навченого технікам інспектування.

Інспектування програмного забезпечення завжди залучає авторів проміжного чи кінцевого продукту, на відміну оцінок, які вимагають цього обов'язково. Інспекції (як тимчасові організаційні одиниці – групи, команди) включають лідера, реєстратора, рецензента та кількох (від 2 до 5) інспекторів. Члени команди інспектування можуть спеціалізуватися у різних галузях експертизи (володіти різними галузями компетенції), наприклад, предметної області, методах проектування, мовою тощо. В даний момент (проміжок) часу інспекції проводяться щодо окремого невеликого фрагмента продукту (у більшості випадків, фокусуючись на окремих функціональних або інших характеристиках; часто, відштовхуючись від окремих бізнес-правил, функціональних вимог чи атрибутів якості, прим. автора). Кожен член команди повинен досліджувати програмний продукт та інші вхідні дані до проведення інспекційної зустрічі, застосовуючи, можливо, ті чи інші аналітичні техніки в невеликих фрагментах продукту або продукту, в цілому, розглядаючи в останньому випадку тільки один його аспект, наприклад, інтерфейси. Будь-яка знайдена аномалія має документуватися, а інформація передаватиметься лідеру інспекції. У процесі інспекції лідер керує сесією інспекції та перевіряє, що всі члени команди підготувалися до інспектування.

Загальним інструментом, що використовується при інспектуванні, є перевірочний лист (checklist), що містить аномалії та питання, пов'язані з аспектами програмного продукту, що викликають інтерес. Результуючий лист часто класифікує аномалії та оцінюється командою з погляду його завершеності та точності.

Рішення про завершення інспекції приймається відповідно до одного (будь-якого) із трьох критеріїв:

1. Прийняття продукту з відсутністю або малою необхідністю переробки
2. Прийняття продукту з перевіркою перероблених фрагментів
3. Необхідність повторної інспекції

Інспекційні зустрічі займають, як правило, кілька годин, на відміну від технічної оцінки та аудиту, що передбачають, як правило, більший обсяг робіт і, відповідно, що тривають довше.

1.6.4 Прогонки (Walk-throughs)

Призначення прогонки полягає у оцінці програмного продукту. Прогін може проводитися з метою ознайомлення (навчання) аудиторії з програмним продуктом.

Головні цілі прогонки полягають у:

- Пошук аномалій
- Поліпшення продукту
- Обговорення альтернативних шляхів реалізації
- Оцінки відповідності стандартам та специфікаціям

Прогонка схожа на інспекцію, проте зазвичай проводиться менш формальним чином. В основному, прогін організується інженерами для інших членів команди з метою отримання відгуку від них на свою роботу, як одного з елементів (технік) забезпечення якості.

1.6.5 Аудити (Audits)

Призначенням аудиту програмного забезпечення є незалежна оцінка програмних продуктів і процесів щодо їх відповідності застосовним регулюючим документам, стандартам, керівним вказівкам, планам і процедурам.

Аудит є формально організованою діяльністю, учасники якої виконують певні ролі, такі як головний аудитор . auditor), другий аудитор (another auditor), реєстратор (recorder) та ініціатор (initiator). В аудиті бере участь представник оцінюваної організації/організаційної одиниці. В результаті аудиту ідентифікуються випадки невідповідності та формується звіт, необхідний команді для розробки коригувальних дій.

При тому, що існують різні формальні назви (і класифікації) оцінок та аудиту, важливо відзначити, що такі дії можуть проводитися майже для будь-якого продукту на будь-якій стадії процесу розробки або супроводу.

1.7 Чинники, впливаючі на якість програмного забезпечення

На планування, управління та вибір SQM-дій та технік впливають різні фактори, серед яких:

- Область застосування системи, в якій працюватиме програмне забезпечення (критичне для безпеки людей), критичне для бізнесу тощо)
- Системні та програмні вимоги
- Які компоненти використовують у системі – комерційні (зовнішні) чи стандартні (внутрішні)
- Які стандарти програмної інженерії застосовні у заданому контексті
- Які методи та програмні інструменти, що застосовуються для розробки та супроводу, а також для забезпечення якості та вдосконалення (продукту та процесів)
- Бюджет, персонал, організація проектної діяльності, плани та розклади для всіх процесів
- Хто цільові користувачі та яке призначення системи
- Рівень цілісності системи

Інформація про ці фактори впливає на те, як саме будуть організовані та документовані процеси SQM, які SQM-роботи будуть відібрані (стандартизовані в рамках проекту, команди, організаційної одиниці, організації), які необхідні ресурси та які обмеження, що накладаються щодо зусиль, що спрямовуються забезпечення якості.

1.8 Дефекти програмного забезпечення

SQM-процеси забезпечують знаходження дефектів.

Опис характеристик дефектів грає основну роль розумінні продукту, полегшує коригування процесу чи продукту, і навіть інформує менеджерів проектів і замовників про статус (стан) процесу чи продукту. Існує безліч таксономій (класифікації та методів структурування) дефектів (збоїв). Характеристика дефектів (аномалій) також використовується в аудиті та оцінки, коли лідер оцінки часто представляє для обговорення на відповідних зустрічах список аномалій, сформований членами оціночної команди.

На тлі еволюції (і появи нових) методів проектування та мов, нарівні з новими програмними технологіями, з'являються нові класи дефектів. Це потребує величезних зусиль щодо інтерпретації (і коригування) раніше певних класів дефектів (збоїв). При відстеженні дефектів інженер цікавиться як їх кількістю, а й типом. Розподіл дефектів за типами особливо важливий для визначення найслабших елементів системи, з точки зору використовуваних технологій та архітектурних рішень, що призводить до необхідності їх поглибленого вивчення, створення спеціалізованих пілотних проектів, додаткової перевірки концепції (proof of concept , POC – підхід, що часто застосовується при використанні нових технологій), залучення сторонніх експертів і т.п. Сама по собі інформація, без класифікації, часто буває просто марною для виявлення причин збоїв, оскільки для визначення шляхів вирішення проблем необхідне їхнє угруповання за відповідними типами. Питання полягає у визначенні такої таксономії дефектів, яка буде значима для інженерів та організації, загалом.

SQM забезпечує збирання інформації на всіх стадіях розробки та супроводу програмного забезпечення. Зазвичай, коли ми говоримо “дефект”, ми маємо на увазі “збій”, відповідно до визначення, наведеного нижче. Проте, різні культури та стандарти можуть припускати різне смислове наповнення цих термінів.

Існують наступні визначення дефектів програмного забезпечення:

- **Помилка (error):** “Відмінність … між коректним результатом та обчисленим результатом отриманим з використанням програмного забезпечення”

- **Недолік (fault):** “Некоректний крок, процес чи визначення даних у комп'ютерній програмі”
- **Збій (failure):** "Некоректний результат, отриманий внаслідок нестачі"
- **Людська/користувацька помилка (mistake):** “Дія людини, що призвела до некоректного результату”

Моделі надійності будуються на підставі даних про збої, зібрані в процесі тестування програмного забезпечення або його використання. Такі моделі можуть бути використані для передбачення майбутніх збоїв та допомагають у прийнятті рішення про припинення тестування.

За результатами SQM-робіт, спрямованих виявлення дефектів, виконуються дії з видалення дефектів з досліджуваного продукту. Однак цим справа не обмежується. Є й інші можливі дії, що дозволяють отримати повну віддачу результатів виконання відповідних SQM-работ. Серед них – аналіз та підбиття підсумків (резюмування) за виявленими невідповідностями/дефектами, використання технік кількісної оцінки (отримання метрик) для покращення продукту та процесу, відстеження дефектів та видалення їх із системи (з управлінським та технічним контролем проведення необхідних коригувальних дій). У свою чергу, джерелом інформації для поліпшення процесу, зокрема, є SQM-процес.

Дані про невідповідності та дефекти, знайдені в процесі реалізації відповідних технік SQM, повинні фіксуватися для запобігання їх втрати. Для деяких технік (наприклад, технічної оцінки, аудиту, інспекцій), присутність реєстратора (recorder) – обов'язково, саме для фіксування такої інформації, нарівні з питаннями (у тому числі потребують додаткового розгляду) та прийнятими рішеннями. У тих випадках, коли використовуються відповідні засоби автоматизації, вони можуть забезпечити отримання необхідної вихідної інформації про дефекти (наприклад, зведену статистику за статусами дефектів, відповідальними виконавцями тощо). Дані про дефекти можуть збиратися та записуватися у формі запитів на зміни (SCR, software change request) і можуть,

згодом, заноситись у певні типи баз даних (наприклад, з метою відстеження крос-проектної/історичної статистики для подальшого аналізу та вдосконалення процесів), як вручну, так і в автоматичному режимі з відповідних засобів аналізу (ряд сучасних засобів проектування) та спеціалізованих інструментів дозволяють аналізувати код та моделі із застосуванням відповідних метрик, значущих для забезпечення якості продуктів та процесів). Звіти про дефекти надсилаються управлінській ланці організації/організаційної одиниці або структури (для прийняття відповідних рішень щодо проекту, продукту, процесу, персоналу, бюджету тощо).

1.9 Техніки управління якістю програмного забезпечення (Software Quality Management Techniques)

Техніки SQM можуть бути розподілені за кількома категоріями:

- статичні
- техніки, що потребують інтенсивного використання людських ресурсів, або техніки колективної оцінки
- аналітичні
- динамічні

Статичні техніки припускають детальне дослідження (examination) проектної документації, програмного забезпечення та іншої інформації про програмний продукт без його виконання. Ці техніки можуть включати інші, що розглядаються нижче, дії з "колективної" оцінки або "індивідуального" аналізу, незалежно від ступеня використання засобів автоматизації.

Техніки колективної оцінки (People-intensive techniques). Форма такого роду технік, включаючи оцінку та аудит, може змінюватись від формальних зборів до неформальних зустрічей або обговорення продукту навіть без звернення до його коду. Зазвичай, такого роду техніки припускають очного взаємодії мінімум двох, а здебільшого, і більше фахівців. При цьому такі зустрічі можуть вимагати

попередньої підготовки (практично завжди стосується визначення змісту зустрічей, тобто переліку питань, що виносяться на обговорення). До ресурсів, що використовуються в таких техніках, нарівні з досліджуваними артефактами (продуктом, документацією, моделями тощо) можуть належати різноманітні аркуші перевірки (checklists) та результати аналітичних технік (розглядаються нижче) та робіт з тестування. Дані техніки розглядаються, наприклад, у стандарті 12207 під час обговорення оцінки (review) та аудиту (audit).

Аналітичні техніки (Analytical techniques)

Інженери, які займаються програмним забезпеченням, зазвичай застосовують аналітичні техніки. З точки зору Agile -методик та підходів, individuals and Interactions передбачає безпосереднє спілкування та постійну взаємодію членів команди.

Іноді кілька інженерів використовують одну і ту ж техніку, але щодо різних частин продукту. Деякі техніки базуються на специфіці інструментальних засобів, інші – припускають “ручну” роботу. Багато хто може допомагати знаходити дефекти безпосередньо, але найчастіше вони використовуються для підтримки інших технік. Ряд технік також включає різноманітних експертизу (assessment) як складовий елемент загального аналізу якості. Приклади таких технік - аналіз складності . analysis), аналіз керуючої логіки (або аналіз контролю потоків управління - control flow analysis) та алгоритмічний аналіз (algorithmic analysis).

Кожен тип аналізу має конкретне призначення і не всі типи застосовні до будь-якого проекту. Прикладом техніки підтримки є аналіз складності, який корисний для визначення фрагментів дизайну системи, що мають надто високу складність для коректної реалізації, тестування або супроводу. Результат аналізу складності може застосовуватися для розробки тестових сценаріїв (test cases). Такі техніки пошуку дефектів, як аналіз логіки, що управляє, може також використовуватися і в інших випадках. Для програмного забезпечення з великою алгоритмічною логікою дуже важливо застосовувати алгоритмічні техніки, особливо в тих випадках, коли некоректний алгоритм (не його реалізація, а саме

логіка, прим. автора) може призвести до катастрофічних результатів (наприклад, програмне забезпечення авіоніки, для якої питання безпеки використання – safety грають вирішальну роль).

Інші, формальні типи аналітичних технік відомі як формальні методи. Вони застосовуються для перевірки вимог та дизайну (треба визнати, лише іноді, у реальній сьогоднішній практиці промислової розробки програмного забезпечення). Перевірка коректності застосовується до критичних фрагментів програмного забезпечення (що взагалі кажучи, мало пов'язано з формальними методами – це природний шлях досягнення прийнятної якості при мінімізації витрат). Найчастіше вони використовуються для верифікації особливо важливих частин критично-важливих систем, наприклад, конкректних вимог безпеки та надійності.

Динамічні техніки (Dynamic techniques)

У процесі розробки та супроводження програмного забезпечення доводиться звертатися до різних видів динамічних технік. Здебільшого це техніки тестування. Однак, як динамічні техніки можуть розглядатися техніки симуляції, перевірки моделей і "символічного" виконання (symbolic execution , часто передбачає використання модулів - “ пустушок” з погляду виконуваної логіки, з емулюваним входом і виходом під час розгляду загального сценарію поведінки багатомодульних систем; іноді під цим терміном розуміються й інші техніки, залежно від обраного першоджерела).

Перегляд (читання) коду зазвичай сприймається як статична техніка, але досвідчений інженер може виконувати код безпосередньо “у процесі” його читання (наприклад, використовуючи діалогові засоби покрокової налагодження для ознайомлення чи оцінки чужого коду). Таким чином, дана техніка цілком може обговорюватись і як динамічна. Такі розбіжності у класифікації технік ясно показують, що залежно від ролі людини в організації, він може приймати та застосовувати одні й самі техніки по-різному.

Залежно від організації ведення проекту певні роботи з тестування можуть виконуватися при розробці програмних систем у SQA та V&V процесах. З огляду

на те, що план SQM адресується питанням тестування, ця тема включає деякі коментарі з тестування.

1.10 Тестування (Testing)

Процеси підтвердження якості, описані в SQA та V&V планах, досліджують та оцінюють будь-який вихідний продукт (включаючи проміжний та кінцевий), пов'язаний зі специфікацією вимог до програмного забезпечення, на предмет:

- трасування (traceability),
- узгодженості (consistency),
- повноти/завершеності (completeness),
- коректності (correctness)
- безпосередньо виконання вимог (performance).

Таке підтвердження також охоплює будь-які вихідні артефакти процесів розробки та супроводу, збору, аналізу та кількісної оцінки результатів. SQA-діяльність забезпечує гарантію того, що відповідні (необхідні в заданому контексті проекту) типи тестів сплановані, розроблені та реалізовані, а V&V – розробку планів тестів, стратегій, сценаріїв та процедур тестування .

Два типи тестування слідують завданням, які задаються SQA і V&V, тому що на них лягає відповідальність за якість даних, що використовуються в проекті:

- Оцінка та тестування інструментів, що використовуються в проекті
- Тестування на відповідність (або оцінка тестів на відповідність) компонентів та COTS-продуктів (COTS — commercial of-the-shelf , готовий до використання продукт) для використання у продукті.

Іноді, незалежні V&V-організації можуть вимагати можливості моніторингу процесу тестування і, у певних випадках, завіряти (або, частіше, документувати/фіксувати) реальне виконання тестів на предмет їх проведення відповідно до заданих процедур. З іншого боку, може бути зроблено звернення до V&V може бути спрямоване на оцінку та самого тестування: достатності планів та процедур, відповідності та точності результатів.

Інший тип тестування, яке проводиться під керівництвом V&V-організації - тестування третьою стороною (third-party testing). Така третя сторона сама не є розробником продукту і у жодній формі не пов'язана з розробником продукту. Понад те, третя сторона є незалежним джерелом оцінки, зазвичай акредитованим щодо володіння відповідними повноваженнями (наприклад, організацією-розробником тієї чи іншої стандарту, відповідність якому оцінюється незалежним експертом і чиї дії підтверджені творцем стандарту). Призначення такого роду тестування полягає у перевірці продукту на відповідність певному набору вимог (наприклад, з інформаційної безпеки).

1.11 Кількісна оцінка якості програмного забезпечення (Software Quality Measurement)

Моделі якості програмних продуктів часто включають метрики визначення рівня кожної характеристики якості, властивої продукту.

Якщо параметри якості вибрані правильно, такі вимірювання можуть підтримувати якість (рівень якості) багатьма способами. Метрики можуть допомогти в управлінні процесом ухвалення рішень. Метрики можуть сприяти пошуку проблемних аспектів та вузьких місць у процесах. Метрики є інструментом оцінки якості своєї роботи самими інженерами – як з метою, визначеною SQA, так і з точки зору більш довгострокового процесу вдосконалення якості.

Зі збільшенням внутрішньої складності, витонченості програмного забезпечення, питання якості виходять далеко за межі констатації факту – працює чи працює програмне забезпечення. Питання ставиться – наскільки добре досягаються ціль якості, що кількісно оцінюються.

Існує ще кілька тем, предметом обговорення яких є метрики, які безпосередньо підтримують SQM. Вони включають сприяння у ухваленні рішення про момент припинення тестування. У цьому контексті видаються корисними

моделі надійності та порівняння зі зразками (еталонами, прийнятими як приклади певної якості – benchmarks).

Вартість процесу SQM є одним із проблемних питань, яке завжди спливає у процесі прийняття рішення про те, як буде організовано проект (проектні роботи). Часто використовуються загальні моделі вартості, засновані на визначенні того, коли саме дефект виявлено і як багато зусиль необхідно витратити на його виправлення в порівнянні з ситуацією, якби дефект був знайдений на ранніх етапах життєвого циклу. Проектні дані можуть допомогти отримати більш чітку картину вартості.

Нарешті, сама по собі SQM-звітність має корисну інформацію не тільки про самі процеси, а й про те, як можна покращити всі процеси життєвого циклу.

Хоча, як кількісні оцінки (в даному випадку йдеться про результати оцінок, а не про процес вимірювань) характеристик якості можуть бути корисними власними силами (наприклад, кількість незадоволених вимог і пропорція таких вимог), можуть ефективно застосовуватися математичні та графічні техніки, полегшують інтерпретацію значень метрик. Такі техніки цілком природно класифікуються, наприклад, так:

- Засновані на статистичних методах (наприклад, аналіз Pareto , нормальний розподіл тощо)
- Статистичні тести
- Аналіз тенденцій
- Пророцтво (наприклад, моделі надійності)

Техніки, засновані на статистичних методах і статистичні тести часто надають "знімок" найбільш проблемних областей досліджуваного програмного продукту (і, до речі, те саме часто й щодо процесів). Результуючі графіки та діаграми візуально допомагають особам, які приймають рішення, у визначенні аспектів, на яких необхідно сфокусувати ресурси проекту. Результати аналізу тенденцій можуть демонструвати, що порушується розклад, наприклад, під час тестування; або що збої певних класів стають дедалі частішими до того часу, доки

робляться коригуючі дії у процесі розробки. Техніки передбачення допомагають у плануванні часу тестів та у передбаченні збоїв.

1.12 Якість програмного продукту з різних точок зору

Якість програмного продукту (software quality) - весь обсяг ознак і характеристик програмної продукції, що відноситься до її здатності задовольняти встановленим чи передбачуваним потребам.

Важливість кожної характеристики якості змінюється в залежності від класу програмного забезпечення. Наприклад, надійність найбільш важлива для програмного забезпечення бойових критичних систем, ефективність найбільш важлива для програмного забезпечення критичних систем реального часу, а практичність найбільш важлива для програмного забезпечення діалогу кінцевого користувача.

Важливість кожної характеристики якості також змінюється в залежності від прийнятих точок зору.

Точка зору користувача

Користувачі здебільшого виявляють зацікавленість у застосуванні програмного забезпечення, його продуктивності та результатах використання. Користувачі оцінюють програмне забезпечення без вивчення його внутрішніх аспектів або того, як створювалося програмне забезпечення.

Користувача можуть цікавити такі питання:

- Чи потрібні функції в програмному забезпеченні?
- Наскільки надійне програмне забезпечення?
- Наскільки ефективно програмне забезпечення?
- Чи є програмне забезпечення зручним для використання?
- Наскільки легко переноситься програмне забезпечення та інше середовище?

Точка зору розробника

Процес створення вимагає від користувача та розробника використання тих самих характеристик якості програмного забезпечення, оскільки вони застосовуються для встановлення вимог та приймання. Коли розробляється програмне забезпечення для продажу, вимоги якості повинні відображати передбачувані потреби.

Оскільки розробники відповідають за створення програмного забезпечення, яке має задовольняти вимогам якості, вони зацікавлені як проміжна продукція так само, як і як кінцева продукція. Для того, щоб оцінити якість проміжної продукції на кожній фазі циклу розробки, розробники повинні використовувати різні метрики для тих самих характеристик, тому що одні й ті самі метрики непридатні для всіх фаз життєвого циклу.

Наприклад, користувач розуміє ефективність у термінах часу реакції, тоді як розробник використовує у проектній специфікації терміни довжини маршруту та часу очікування та доступу. Метрики, що застосовуються для зовнішнього інтерфейсу продукції, заміняються метриками, що застосовуються для її структури.

Точка зору керівника

Керівник може бути більш зацікавлений у спільній якості, ніж у конкретній характеристиці якості, і з цієї причини потребуватиме визначення важливості значень, що відображають комерційні вимоги для індивідуальних характеристик. Керівнику може також знадобитися порівняння підвищення якості з критеріями керованості, такими як планова затримка або перевитрата вартості, тому що він бажає оптимізувати якість у межах обмеженої вартості, трудових ресурсів та встановленого часу.

1.13 Етапи оцінки якості програмного продукту

Наступний рисунок відображає основні етапи, потрібні для оцінювання якості програмного забезпечення.

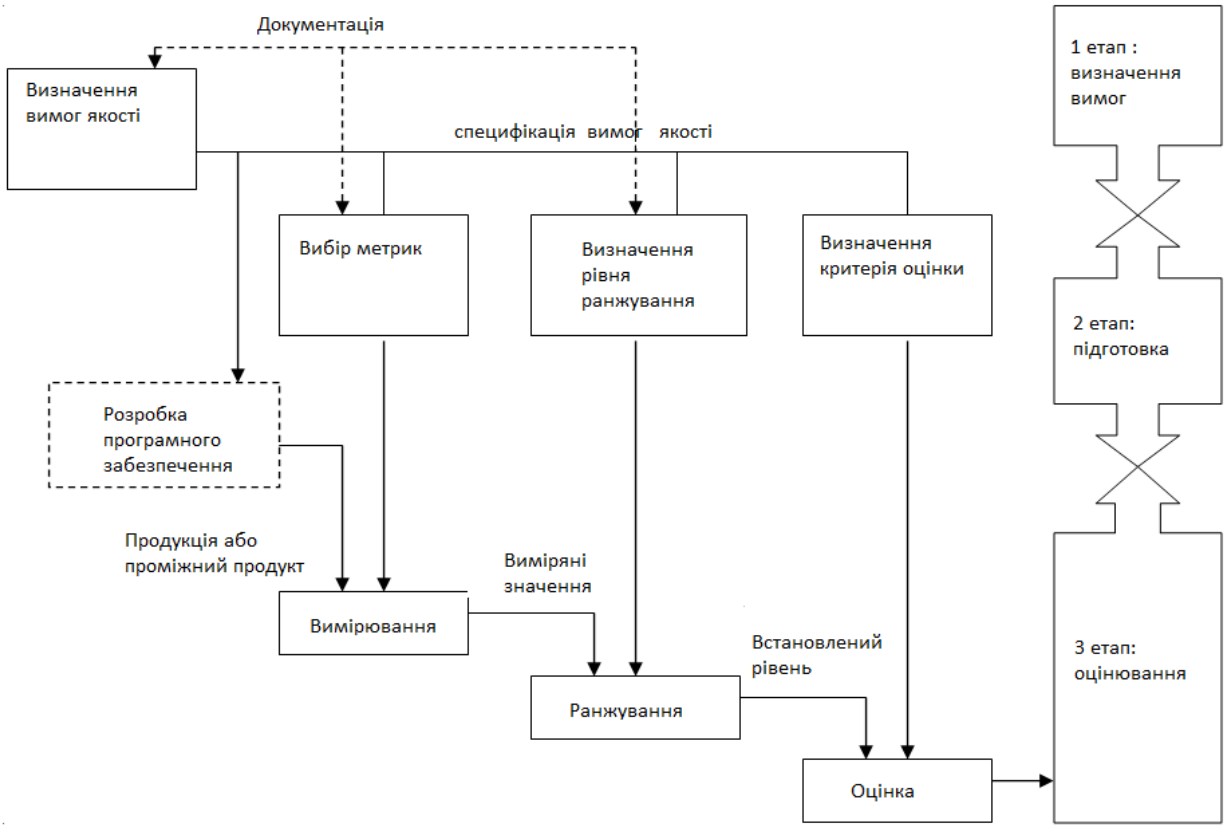

Рисунок 1.3 - Модель процесу оцінювання»

Процес оцінювання складається із трьох стадій: встановлення (визначення) вимог до якості, підготовка до оцінювання та процедура оцінювання. Цей процес може застосовуватися у будь-якій відповідній фазі життєвого циклу для кожного компонента програмної продукції.

Метою початкової стадії є встановлення вимог у термінах показників якості. Вимоги висловлюють потреби зовнішнього оточення для аналізованої програмної продукції і мають бути визначені на початок розробки. Метою другий стадії є підготовка основи для оцінювання. Результатом третьої є висновок якості програмної продукції. Потім узагальнена якість порівнюється з іншими факторами, такими як час і вартість. Остаточне рішення керівництва приймається з урахуванням критерію керованості. Результатом є рішення посібника з приймання або відбраковування, або з випуску або випуску програмної продукції.

Процес розробки має бути побудований таким чином, щоб забезпечити можливість вимірювання якості продукту. Проведені дослідження показують: що

вища якість процесу розробки, то вища якість розробленого у процесі якості програмного забезпечення. Якість кожної стадії проекту зростає, по-перше, як пряме наслідок зрілості процесу, по-друге, внаслідок використання проміжного продукту вищої якості, виробленого попередньої стадії. При цьому підкреслюється, що значення другої причини , що забезпечує наростання якості в процесі життєвого циклу для зрілих процесів, виявляється набагато важливішим. Все це можна подати у вигляді деякої моделі.

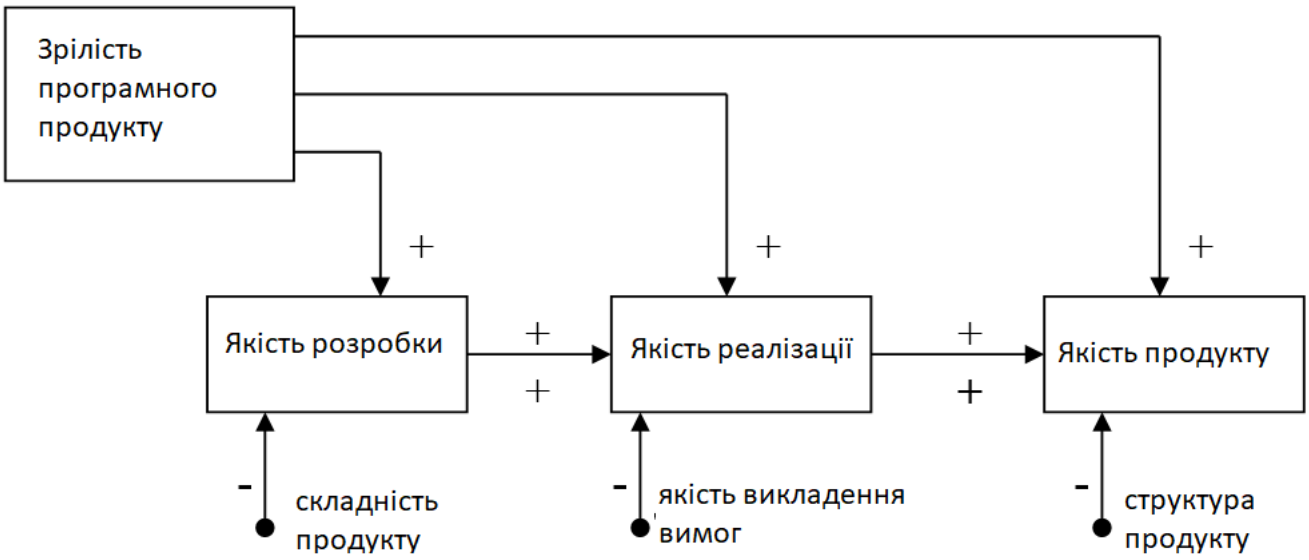

Рисунок 1.4 - Концептуальна модель якості процесу розробки»

Висновки:

Перше: якість накопичується в продукті при складному виробництві кумулятивним чином, причому, внесок у якість, здійснений на ранніх стадіях, має сильніший вплив на кінцевий продукт, ніж на більш пізніх стадіях. Це підтверджується всією практикою програмування, наприклад відомо, що недоліки проектування систем не можуть бути компенсовані високою якістю кодування.

Таким чином, для управління якістю побудови складної системи необхідно проводити вибір виробників на основі вимірювання ступеня зрілості та прозорості процесів розробки, що використовуються. Вимірювання якості процесу розробки підрядників є важливою складовою загального управління якістю, важливішим,

ніж вимір якості результуючого продукту, що виробляється в ході приймально-здавальних випробувань.

Друге: тестування та вимір якості має відбуватися на всіх стадіях життєвого циклу. Вилучення даних про якість, яка була закладена на ранніх стадіях, може бути дуже дорогим, за відсутності повних результатів

2 ПОКАЗНИКИ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Відомо, що показник якості ПЗ представляє собою здатність програмних засобів в певних умовах задовольняти встановленим або передбачуваним вимогам і потребам . Стрімкий розвиток коштів призводить до виникнення нових проблем, на які необхідно звертати увагу. Розробка ПС проводиться для конкретних цілей та завдань. Однак в результаті програмний засіб не завжди відповідає потрібним вимогам і має певні споживчі властивості. Якість будь-якого продукту, що поступає до споживача, є головною складовою для розвитку підприємств. Тому важливо визначити показники, що впливають на якість програмних засобів [1]:

- область, де застосовується програмний засіб;
- призначення ПЗ;
- розв'язувані завдання та їх тип;
- характеристики якості програмного засобу (склад та значення);
- величина допустимої шкоди через недостатню якість ПС;
- обсяг програмного засобу;
- складність програмного засобу;
- зв'язок розв'язуваних завдань із реальною кількістю часу (допустимою тривалістю очікування результатів розв'язання задачі);
- тривалість експлуатації;
- можливість створення нових версій програмних засобів;
- застосування та тиражування програмного засобу.

Відповідно до вимог системи якості ISO 9000, кожен процес потребує моніторингу та вимірювання, а також обробці результатів вимірювань для розробки впливів на якість [2].

В розвинених країнах застосовуються системи, здатні керувати якістю в освітніх установах відповідно до стандарту ISO 9000.

При вирішенні завдань, таких як розробка або модернізація програмних засобів, виникає завдання оцінки якості ПЗ, що використовується. Також відомо,

що при оцінці якості ПС використовується багато моделей і стандартів. Найбільш сучасний і відомий стандарт – ISO/IEC 25000 – являє собою модель якості у вигляді набору характеристик (тобто категорій атрибутів, за допомогою яких можна провести оцінку якості ПС) та відносин між ними. Набір даних характеристик є основою для класифікації вимог до якості та оцінки якості ПЗ [5]. У цьому найбільшого поширення отримали моделі типу «чинники – критерії – метрики» [6]. Щодо оцінки якості ПС у навчальних закладах, необхідно враховувати рівень знань студентів, які використовують дані програмні засоби. Також слід зазначити, що за швидкістю та легкістю освоєння ПС можна судити про якість продукту, про повноту та зручність використання функцій програмного продукту (далі – ПП).

Щоб оцінити якість ПП у ВНЗ, можна провести його тестування серед студентів для виявлення наступних показників [7–10]:

- надійність;
- коректність;
- зручність
- застосування;
- ефективність;
- універсальність;
- супроводжуваність.

Відповідно до вищевказаних показників якості, можна визначити потрібні вимоги до якості ПС. Для того щоб виявити, який програмний продукт найбільше відповідає критеріям, можна використовувати ієрархічну структуру оцінки якості, яка представлена на рис. 1

Значення загальних властивостей відповідають значенням виділених загальних показників якості, наприклад, таких як естетичні властивості, властивості призначення, ергономічні властивості і т. д. Для певних властивостей ці методичні вказівки містять рекомендації щодо їх зміни на більш прості властивості та їх використання.

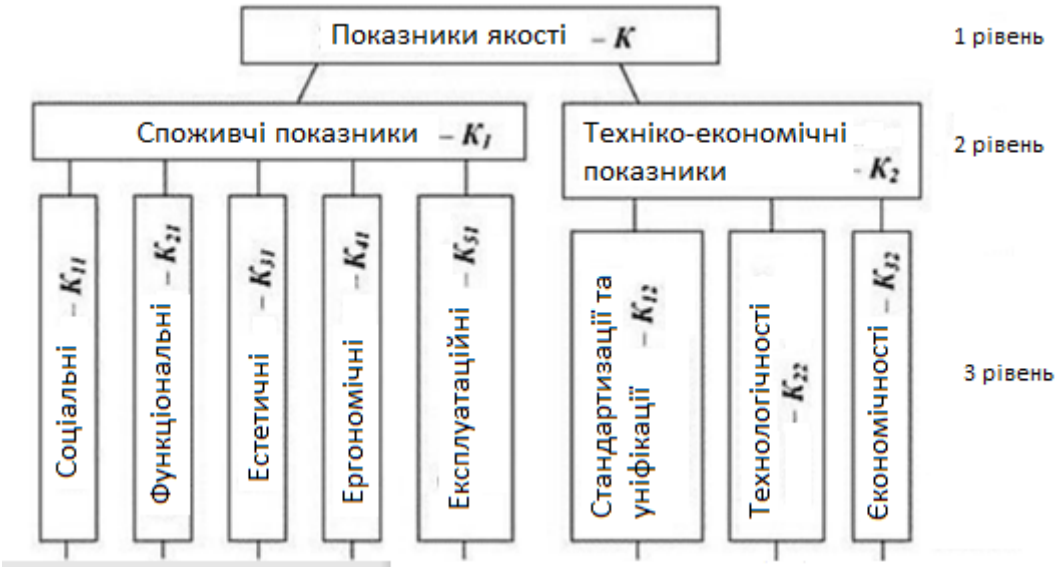

Рисунок 2.1 - Ієрархічна структура оцінки якості

Аналіз інформації, поданої на рис. 1, показує, що до 1-го рівня належать споживчі ($К_1$) і техніко-економічні ($К_2$) показники якості (далі - ПЯ) ПС .

В свою чергу, споживчий рівень складається з п'яти класів показників якості. Вони визначають суспільну та індивідуальну цінність для людини-споживача: соціальну ($К_{11}$) , функціональну ($К_{21}$) , естетичну ($К_{31}$), ергонометричну ($К_{41}$) , екс-плуатаційну ($К_{51}$) [11]. Однак, якщо ввести наступний рівень ієрархії, що дозволяє розширити список даних характеристик, можна додати такі показники:

- Повнота (всі частини ПС повинні бути представлені, описані, а також повністю реалізовані);

- завершеність (коли програмний засіб має властивість завершеності, в тому числі в ньому повинні бути присутніми всі необхідні компоненти, які розроблені в повному обсязі);

- швидкодія (витрачений час на вирішення завдання користувача);

- ефективність (раціональне ставлення програмного засобу до таких ресурсів, як процесор, пам'ять і т. д., при виконанні своїх завдань);

- тестованість (коли програма виконує перевірку характеристик, підтримує можливість вимірювання продуктивності);

- узгодженість (коли у програмі та в документації використовуються одні й ті самі позначення, угоди, формати);

- розширюваність (можливість програмного засобу збільшувати, якщо це необхідно, обсяг пам'яті для забезпечення зберігання необхідних даних розширювати для окремих модулів обчислювальні функції);

- стислість (відсутність непотрібної чи повторюваної інформації, дублюючі частини коду перетворюються на виклик загальної процедури, те саме стосується й технічної документації);

- рівень вимог технічних засобів (обсяг пам'яті, наявність співпроцесора та інших.);

- супроводжуваність (наявність можливості змінити ПС для задоволення нових вимог споживача).

- простота налаштування та використання технічного середовища (принтер, сканер, модем, монітор та ін);

- портованість (можливість ПС адаптуватися до іншої платформи або операційної системи. Можливість переналаштуватися на нові умови, такі як зміна законодавства і т. д.);

- комунікативність (можливість ПС описувати вхідні дані, а також їх копіювати і виводити при необхідності отриману інформацію);

- інформативність (можливість ПС містити інформацію, зрозумілу і необхідну для користувача);

- можливість працювати у мережі;

- трудомісткість (оцінка тимчасових та матеріальних витрат на освоєння та впровадження ПС);

- оцінюваність (коли ПС використовується для конкретного застосування і забезпечує можливість оцінки якості його функціонування);

- вартість (має задовольняти вимогам замовника та споживача ПС);

- точність (результати ПС повинні мати точну та достатню з погляду основного їх призначення);

- людський фактор (ПС виконує свою роботу, не вимагаючи при цьому додаткових тимчасових витрат від користувача);

- модифікованість (ПС повинен мати структуру, що дозволяє легко вносити необхідні зміни);

- мобільність (коли ПС адаптивно до різних типів платформ і може працювати на ЕОМ іншого типу, що відрізняється від того, для якого призначено);

- доступність (коли ПС допускає вибіркове використання окремих компонентів);

- безпека (коли ПС має та забезпечує якість захисту даних від не-санкціонованого доступу);

- зрозумілість (ПС має бути зрозумілим і супроводжуватися якісною документацією);

- свідомість (технічна документація до ПС не повинна містити надмірної інформації).

Необхідно також відзначити, що кожен із згаданих вище показників може мати різні вагові коефіцієнти і розділятися на більш дрібні складові в залежності від виду ПС.

Техніко-економічні показники якості ПЗ визначають ступінь технічної досконалості та методи проектування. Складаються з трьох класів: стандартизації K_{12} , технологічності K_{22} , економічності K_{32} . Також основними показниками ТЕП є: ефективність; складність розробки; основні витрати; конкурентоспроможність.

Ефективність є співвідношення витрат і результатів функціонування системи. До основних показників ефективності відносяться такі: економічний ефект, термін окупності вкладень, коефіцієнт економічної ефективності вкладень та ін [12].

Економічний ефект – це різниця між результатами діяльності суб'єкта та виробленими їх отримання витратами зміни умов діяльності.

Найчастіше на практиці оцінюється загальний економічний ефект ($E_{заг}$) від виробництва та використання за весь термін служби нового ПП (P), який розраховується за формулою (2.1):

$$Э_{заг} = \left[\left(З_1 \times \frac{B_2}{B_1} \times \frac{P_1 + E_H}{P_2 + E_H} - \frac{(U_1 - U_2) - E_H \times (K_2 - K_1)}{P_2 + E_H} \right) - З_2 \right] \times A_2 , \quad (2.1)$$

де $З_1$, $З_2$ – питомі витрати, витрачені на базовий та новий ПП;

B_1 , B_2 – річний обсяг робіт, який може виконуватись за допомогою базового ($E_н$) та нового ПП;

$У_2 / B_1$ - коефіцієнт зростання продуктивності нового ПП по відношенню до базового;

($P_1 + E_н$) / ($P_2 + E_н$) - Коефіцієнт обліку зміни термінів служби нового ПП в порівнянні з базовим;

U_1 , U_2 - річні середні питомі експлуатаційні витрати користувача при експлуатації одиниці базового та нового ПП;

K_2 , K_1 – питомі середні капітальні вкладення користувача під час використання одиниці базового і нового ПП для розрахунку обсяг робіт, вироблених з допомогою нового ПП;

($U_1 – U_2$) – $E_н \times$ (K_2 - K_1) - середня величина річної економії споживача на наведених витратах ПП;

(($U_1 – U_2$) – $E_н \times$ ($K_2 – K_1$))/($P_2 + E_н$) – середня економія користувача на наведених витратах для всього терміну служби ПП порівняно з базовим $E_н$;

A_2 – обсяг запровадження ПП у аналізований період [13].

Складність розробки ПС переважно залежить від розміру-масштабу ПС. Вона виражається числом рядків мовою програмування чи функціональних точок. Оцінку розміру-масштабу ПС можна наводити в різних одиницях, що, у свою чергу, дозволить змінювати їх чисельні значення для тих самих ПП в кілька разів [14]. Розділимо на дві основні групи одиниці виміру розміру-масштабу ПС:

- одиниці виміру, які розробляються та аналізуються людиною. Так, наприклад, текст програми можна подати у вигляді наступного коду: код, який використовується повторно (придатний для нових програм без внесення змін); новий код (розроблений для нової програми без включення фрагментів раніше написаного коду); спадковий код (розроблений для попередніх додатків і буде використовуватися новими ПП); модифікований код (розроблений для попередніх додатків і може бути використаний у новому ПП після внесення змін);

- одиниці виміру, які розміщуються у використовуваному ПС (характеризують роботу самого комп'ютера, а також обсяг пам'яті, який необхідний для нормального функціонування ПП);

До основних ресурсів, що витрачаються на розробку ПС, можна віднести такі:

- час, що визначає тривалість створення ПЗ;

- допустимі трудовитрати - це витрати, необхідні для розробки ПС з необхідною якістю;

- число фахівців - необхідна і допустима кількість виконавців, що використовується при розробці ПС.

Під час розробки ПС також розраховується загальна трудомісткість (T_0) (2.2):

$$T_0 = K_{сл} \times T_p, \qquad (2.2)$$

де T_p - трудомісткість, у людино/днях чол.-дн. і визначається обсягом ПС, що розробляється;

$K_{сл}$ – додатковий коефіцієнт складності.

Трудомісткість визначається для кожного етапу розробки, після чого загальна трудомісткість ($T_{заг}$) розробки ПС розраховується за формулою (2.3):

$$T_{общ} = \sum_{j=1}^{k} T_j , \qquad (2.3)$$

де $T\,j$ - трудомісткість на стадіях технічного завдання, ескізного проекту, технічного проекту, робочого проекту та впровадження.

Кількість виконавців, необхідне та використовуване при розробці ПС, розраховується за формулою (2.4):

$$Ч = \frac{T_{общ}}{Ф_Д \times D}, \tag{2.4}$$

де $Ч$ – число виконавців;

$Ф_Д$ - фонд часу одного працюючого на місяць;

D – директивний термін розробки ПС.

Конкурентоспроможність ПС – це здатність товарів конкурувати між собою. Це те, наскільки товар відповідає одному або декільком вибраним показникам (комерційним, технічним, економічним тощо) і забезпечує можливість його збуту. Оцінюючи конкурентоспроможності враховується коефіцієнт еквівалентності ($К_{ЕК}$)

ПС, що розраховується за формулою (2.5):

$$К_{ЭК} = \frac{К_{ТН}}{К_{ТБ}}, \tag{2.5}$$

де $К_{ТН}$ та $К_{ТБ}$ – коефіцієнти технічного рівня розвитку нового та базового ПС. Коефіцієнт технічного рівня розвитку ($К_Т$) ПС, що розробляється, у свою чергу, розраховується за такою формулою (2.6):

$$К_Т = \sum_{j=1}^{n} \beta \times \frac{П_i}{П_Э}, \tag{2.6}$$

де – *В ваговий* коефіцієнт i -го технічного параметра ПС;

n – кількість використовуваних параметрів;

$П_i$ - чисельне значення i -го технічного параметра ПС;

$П_Е$ - чисельне значення i -го технічного параметра базового ПС.

В результаті за формулою (7) можна розрахувати інтегральний коефіцієнт конкурентоспроможності $(К_І)$ ПС, що розробляється:

$$К_И = К_{ЭК} \times К_{ФВ} \times К_Н / К_Ц, \tag{2.7}$$

де $К_{ФВ}$ - коефіцієнт функціональних можливостей ПС, що розробляється;

$К_{Н}$ - коефіцієнт відповідності нового ПС нормативним значенням;

$К_{Ц}$ – коефіцієнт ціни споживання ПС.

Оцінка таких характеристик якості, як супровідність та надійність, не може бути виконана, доки не буде отримана перша версія програмного засобу. Ще до кінця розробки програмного засобу метрика складності дозволяє прогнозувати, які компоненти програмного продукту будуть здатні до відмови, можуть складніше тестуватися та негативно реагувати на зміну налаштувань чи конфігурацій.

Метрики складності визначають трудомісткість розробки комплексу програм. Метрики можна розділити на дві групи:

- метрики складності потоку даних (до них відносяться: метрика звернення до глобальних змінних, метрика Чепіна і т. д.). Метрики складності потоку даних визначають конфігурацію, використання, а також розміщення даних у програмі;
- метрики складності потоку управління програмами (до них відносяться: метрика Маккейба, метрика підрахунку точок перетину, метрика Майєрса, метрика граничних значень, метрика Джилба). З допомогою цих метрик оперують як взаємозв'язками цих переходів, і щільністю управляючих переходів усередині програм.

Висновки

Слід зазначити, що оцінку якості програмних засобів може здійснити різними способами. Оцінка якості, що проводиться відповідно до стандарту ISO 25000 дозволяє оцінити якість програмних засобів, у своїй можна використовувати різні системи показників якості. Також є можливість закласти якість засобів ще при розробці технічного завдання, проводити контроль його на всіх етапах життєвого циклу, тобто можна проводити оцінку на мінімальному рівні якості, не маючи повної інформації про ПС.

3 МЕТОДИКА ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Як вже сказано, у сучасному світі роль інформації безперервно зростає у всіх сферах діяльності людини. Неможливо ефективно приймати управлінські рішення на підприємстві, не володіючи певною інформацією. Інформація виступає сьогодні як один з першорядних ресурсів, значення якого можна порівняти зі значенням матеріальних і фінансових ресурсів. У зв'язку з цим вітчизняні підприємства змушені приділяти пильну увагу створенню сучасного інформаційного ресурсу, впровадженню прогресивних інформаційних технологій. Підприємство при визначенні цілей розвитку виявляє можливості їх реалізації виходячи з власного потенціалу, при цьому базою інформатизації є вибір відповідного програмного забезпечення (ПЗ). Для прийняття рішень щодо впровадження використання готового ПЗ або розробки нового виникає проблема оцінки якості програмного продукту.

Насправді важливо оцінювати якість ПЗ у завершеному вигляді, а й у його розробки, модернізації чи перенесення. Що стосується програмного забезпечення, система забезпечення якості - це сукупність методів і засобів організації керуючих і виконавчих підрозділів підприємства, що беруть участь у проектуванні, розробці, модифікації та супроводі комплексів програм з метою надання їм властивостей, що забезпечують задоволення певних потреб замовників і споживачів при мінімальному або допустимому витраченні ресурсів.

Основою для формування необхідних показників ПЗ є аналіз властивостей, характеризуючих якість його функціонування з урахуванням технологічних можливостей розробника. При цьому під якістю функціонування розуміється безліч властивостей, що зумовлюють придатність ПЗ забезпечувати надійне та своєчасне подання необхідної інформації споживачу для її подальшого використання за призначенням. Адекватний набір показників якості програм залежить від функціонального призначення та властивостей кожного ПЗ. Однак

досі немає єдиного системного підходу до оцінки якості ПЗ, не сформована система оцінок якості інформаційних інновацій, не розроблена технологія підвищення якості інформаційних проектів та інновацій.

В якості пропозиції можна взяти оригінальний підхід до оцінки якості програмного забезпечення, заснований на моделі Раша оцінки латентних змінних. Під латентними (прихованими, неявними) змінними розуміється деякий змінний показник, який не можна безпосередньо виміряти жодним чином, проте можна оцінити за допомогою кінцевого набору змінних змінних, званих індикаторними змінними, які пов'язані з латентними. Обгрунтованість запропонованого підходу пов'язана з тим, що поняття «якість» є типовою латентною змінною, яку можна оцінити за допомогою набору індикаторних змінних, якими є експертні оцінки ПЗ на підставі групи оціночних показників.

Існує кілька підходів до вимірювання латентних змінних, але, на думку багатьох дослідників, найбільш ефективною і зручною для практичного використання є модель Раша [1, 2], що обумовлює її широке застосування в останні роки.

3.1 Математична модель

Розглянемо загальний підхід до вимірювання якості ПЗ, що базується на моделі Раша оцінки латентних змінних.

Нехай є N видів ПЗ, якість яких потрібно оцінити: A_1, A_2, , A_N. Для експертної оцінки використовується L оціночних показників: K_1, K_2, , K_L. Як індикаторних змінних будуть виступати такі змінні: U_{ij} - експертні оцінки i -го ПЗ по j -му оціночному показнику. Ці оцінки можуть мати різну розмірність і різної природи. Для приведення оцінок до єдиної шкали проводять процедуру нормалізації, в результаті якої всі нормалізовані оцінки за критеріями u_{ij} приймуть значення з інтервалу (0; 1). Як алгоритм нормалізації можна вибрати наступний.

У разі максимізації показника (чим більше експертна оцінка, то краще ПЗ):

$$u_{ij} = \frac{U_{ij} - \min\limits_{i}(U_{ij})}{\max\limits_{i}(U_{ij}) - \min\limits_{i}(U_{ij})}, \tag{3.1}$$

у разі мінімізації показника (чим менше оцінка, тим краще ПЗ):

$$u_{ij} = \frac{\max\limits_{i}(U_{ij}) - U_{ij}}{\max\limits_{i}(U_{ij}) - \min\limits_{i}(U_{ij})}. \tag{3.2}$$

Далі використовуємо ймовірнісний підхід. Припустимо, що є можливість вибору двох видів ПО з номерами n і m. Позначимо P_{nj} - ймовірність того, що ПЗ з номером n влаштовує експерта за j - му показнику. Звідси випливає, що ймовірність того, що цей програмний продукт не влаштовує експерта, дорівнює $(1 - P_{nj})$. Приймемо аналогічні позначення і для ПЗ з номером m.

Далі позначимо: N_{11} - число показників, за якими експерта влаштовують обидва програмні продукти; N_{10} - число показників, за якими влаштовує тільки m - е ПЗ; N_{01} - число показників, за якими влаштовує тільки n -е ПЗ і N_{00} - число показників, за якими експерта не влаштовують обидва види ПЗ.

З точки зору порівняння зазначених двох конкретних видів, інформативними можна вважати тільки значення N_{10} і N_{01}. У свою чергу значення N_{11} і N_{00} не дають уявлення про те, у якого виду ПЗ повага вище. При цьому параметр N_{10}, що відображає ступінь переваги ПЗ A_m, відповідно до теореми множення ймовірностей, буде прямо пропорційний добутку ймовірностей $P_{mj}(1 - P_{nj})$. Аналогічно пара-метр N_{01} прямо пропорційний до вірогідності $(1 - P_{mj})P_{nj}$. Таким чином, було отримано вираз, що визначає відношення параметрів N_{10} до N_{01}:

$$\frac{N_{10}}{N_{01}} \sim \frac{P_{mj}(1 - P_{nj})}{P_{nj}(1 - P_{mj})}. \tag{3.4}$$

Якщо припустити, що число показників L нескінченно або дуже велике, воно допоможе визначити різницю в рівні оцінок ПЗ з номерами n і m. Через те, що не було накладено жодних умов на оцінні показники, отримане вираження не залежить від набору самих показників. При розгляді іншого виду з номером k буде отримано аналогічне вираз. При цьому співвідношення оцінок ПЗ залишиться

незмінним. З цього випливає, що для оціночних показників з деякими номерами *k* і j можна записати:

$$\frac{P_{nj}(1-P_{nj})}{P_{nj}(1-P_{mj})}=\frac{P_{mk}(1-P_{nk})}{P_{nk}(1-P_{mk})}. \tag{3.5}$$

У свою чергу, з цього виразу можна записати наступне:

$$\frac{P_{nj}}{1-P_{nj}}=\frac{P_{nk}}{1-P_{nk}}\cdot\frac{1-P_{mk}}{P_{mk}}\cdot\frac{P_{mj}}{1-P_{mj}}. \tag{3.6}$$

Для використання вищеописаного методу на практиці необхідно, щоб результати експертного порівняння привабливості програмних продуктів *n* і *m* були об'єктивні. А для цього потрібно, щоб для будь-яких оціночних показників було справедливо співвідношення будь-якого набору ПЗ *k* і *j* .

З метою забезпечення виконання цієї вимоги в якості вихідних точок для проведення порівняльного аналізу були прийняті ступінь оцінки деякого ПЗ з індексом 0, і деякого оцінного показника з індексом 0. Крім цього, необхідна єдина шкала вимірювання, що об'єднує в собі рівень привабливості ПЗ для експерта і рівень важливостей оціночних показників, при цьому за точку відліку в ній зручно але прийняти ці показники з індексом 0 та вважати їх еквівалентними. Таким чином, значення параметра *P* дорівнюватиме 0,5. При

$$=\frac{P_{n0}}{1-P_{n0}}\cdot\frac{P_{0j}}{1-P_{0j}}. \tag{3.7}$$

Неохідно зазначити, у виразі (3.6)

Необхідно відзначити, у виразі (3.6) величина $\left(\frac{P_{n0}}{1-P_{n0}}\right)=d_n$ узагальнений показник привабливості n -го ПЗ, що є його унікальним власністю, який можна інтерпретувати як якість ПЗ. З іншого боку, величина $\frac{P_{0j}}{1-P_{0j}}=\frac{1}{b_j}$ можна

розглядатися як деяка узагальнена оцінка вразливості, неможливості оціночного показника по всій групі ПЗ. У результаті отримуємо:

$$\frac{P_{nj}}{1-P_{nj}} = \frac{d_n}{b_j}. \tag{3.8}$$

Виходячи з цього, можна обчислити імовірність або ступінь того, що j -й ПЗ влаштовує експерта по n -му оціночному показнику.

Використовуючи записане вище рівняння (7) та прологарифмувавши його частини, отримаємо:

$$\ln\frac{P_{nj}}{(1-P_{nj})} = \theta_n - \beta_j.$$

де $\ln d_n = \ln\frac{P_{n0}}{1-P_{n0}} = \theta,$

та $\ln\frac{P_{0j}}{1-P_{0j}} = \ln b_j = \beta_j$

Виходячи з цього, перейменувавши для зручності індекси, можна обчислити ймовірність

$$P_{ij} = \frac{e^{\theta_i-\beta_j}}{1+e^{\theta_i-\beta_j}}. \tag{3.9}$$

Ці ймовірності можна інтерпретувати як нормалізовані оцінки привабливості ПЗ на основі оціночних показників u_{ij} .

Отриманий вираз аналогічно формулі Г. Раша, отриманий ним при оцінці латентних змінних [1].

Для застосування (3.9) на практиці необхідно знайти оцінки якості ПЗ θ_i та ступеня вразливості оціночних показників β_j на основі відомих оцінок кожного виду програмного продукту за показниками u_{ij} , які отримані емпірично за допомогою експертного оцінювання.

Якщо розглянути модель Раша оцінки латентних змінних [2], то, відповідно до неї, оцінки θ_i та β_j є методом максимальної правдоподібності (МП-метод). Однак у дихотомічній моделі Раша ймовірності P_{ij} можуть приймати лише два

значення - 0 або 1, що не відповідає представленій у роботі моделі, коли ймовірності P_{ij} можуть приймати значення з безперервного спектру від 0 до 1. В силу цього, можна використовувати для цих цілей метод найменших квадратів [3, 4, 5]: параметри θ_i та β_j моделі (8) вибираються так, щоб сума квадратів відхилень емпіричних даних u_{ij} від розрахункових ймовірностей (3.9) була найменшою. Математично це зводиться до мінімізації залишкової суми:

$$S(\theta_i, \beta_j) = \sum_{i=1}^{N}\sum_{j=1}^{L} w_j \cdot (u_{ij} - P_{ij})^2 =$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{L} w_j \cdot \left(u_{ij} - \frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}} \right)^2 \to \min. \tag{3.10}$$

Оцінки θ_i та β_j, отримані за даною моделлю, будуть вимірюватися за лінійними шкалами, і початок відліку в них буде невизначеним. Нульовий відлік шкал можна вибрати так, щоб усі оцінки θ_i та β_j були неотрицательными. Тоді умова (9) доповнюватиметься умовою нормування:

$$\theta_i \geq 0,\ \beta_j \geq 0;\ i = 1, 2, ..., N,\ j = 1, 2, ..., L. \tag{3.11}$$

Можна використовувати інші нормувальні умови: нульове середнє значення оцінок, нормування на одиничну шкалу і т.д.

Розв'язання оптимізаційної задачі (3.8) таможна проводити з використанням надбудови «Пошук рішень» MS Excel.

Представлена модель передбачає, що оціночні показники мають однакову важливість для експертів. Однак у реальних ситуаціях в оцінці якості ПЗ важливість цих показників, зазвичай, різна, і її треба враховувати в оцінці. Врахувати важливість показників можна шляхом введення ваг оціночних показників. Позначимо w_j - вага j -го оцінного показника. Також будемо вважати, що вага вимірюється за шкалою від 0 до 1, і чим більша вага, тим більшу важливість для оцінки якості ПЗ матиме цей показник. Тоді при мінімізації залишкової суми,

кожне доданок (9) буде враховуватися пропорційно до відповідної йому ваги. В результаті, замість (9) буде вирішуватися оптимізаційна задача виду:

$$S(\theta_i, \beta_j) = \sum_{i=1}^{N} \sum_{j=1}^{L} w_j \cdot (u_{ij} - P_{ij})^2 =$$
$$= \sum_{i=1}^{N} \sum_{j=1}^{L} w_j \cdot \left(u_{ij} - \frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}} \right)^2 \to \min. \quad (3.12)$$

3.2 Вибір оціночних показників

Для оцінки якості ПЗ пропонується 17 оціночних показників, які можна групувати на 4 укрупнених показника.

1. *Функціональність* характеризує властивість ПЗ, що визначає його відповідність вимогам до обробки даних та загальносистемним вимогам, встановленим у ТЕЗ.

1.1. *Придатність* характеризує здатність ПЗ забезпечувати наявність і ступінь достатності виконуваних функцій для вирішення завдань відповідно до встановлених вимог і коректність функціонування людино-машинного інтерфейсу.

1.2. *Правильність* характеризує здатність ПЗ забезпечувати можливість отримання тільки правильних результатів обробки даних.

1.3. Здатність до взаємодії характеризує здатність ПЗ взаємодівати із заданою номенклатурою програмнихзасобів чи систем.

1.4. Захищеність характеризує спроможливість ПЗ запобігати випадковому і умисний несанкціонований доступ до своїх функцій і даних, а також виявити результати такого доступу або руйнування самого ПЗ та даних.

1.5. Узгодженість характеризує особливість ПЗ відповідати нормативним документів у частині придатності, правильності, здатності до взаємодії та захищенності.

2. Надійність характеризує властивість ПЗ, що визначає його здатність виконувати необхідні функції в умовах виникнення відхилень у середовищі функціонування.

2.1. Стабільність характеризує спосібність ПЗ безперервно виконувати необхідні функції в умовах виникнення відхиленняній у середовищі функціонування.

2.2. Стійкість до помилок характеризує здатність ПЗ забезпечувати продовження своєї роботи після виникнення відхилень.

2.3. Відновлюваність характеризує здатність ПЗ відновлювати свою працездатність та пошкоджені дані в умовах збоїв і дестабілізуючих дій.

2.4. Своєчасність характеризує особливість ПЗ надавати своєчасний доступ користувачів до своїх функцій в умовах збоїв та інших дестабілізуючих впливів.

2.5. Узгодженість характеризує особливість ПЗ відповідати прийнятим нормативним документам щодо стабільності,

3. Практичність характеризує властивість ПЗ, що визначає його здатність забезпечвати швидке освоєння, застосування та експлуатацію з мінімальними трудовитратами з урахуванням характеру розв'язуваних завдань та кваліфікації обслуговуючого персоналу.

3.1. Зрозумілість характеризує способність ПЗ забезпечувати мінімальні витрати зусиль користувача для розуміння загальної логічної концепції його функціонування.

3.2. Навчаність характеризує способність ПЗ забезпечувати мінімальні витрати зусиль користувача на засвоєння правилйого застосування.

3.3. Простота використання характеризує здатність ПЗ забезпечувати мінімальні витрати зусиль користувача на його експлуатацію.

3.4. Узгодженість характеризує способність ПЗ відповідати нормативним документів у частині зрозумілості, навченості та простоти використання.

4. Продуктивність характеризує властивість ПЗ, що визначає його здатність використовувати, відповідно до встановленими вимогами, обсяг виділених часових, технічних, матеріальних і людських ресурсів.

4.1. Часоємність характеризує способність ПЗ використовувати відповідно до встановлених вимог обсяг виділяються для виконання необхідних функцій тимчасових ресурсів.

4.2. Ресурсоємність характеризує способність ПЗ використовувати відповідно зі встановленими вимогами обсяг розподілених для виконання необхідних функцій технічних, матеріальних та людських ресурсів.

4.3. Узгодженість характеризує способність ПЗ відповідати прийнятим нормативним документам у частині часу ємності та ресурсомісткості.

Слід зазначити, що в моделі можна виділити також стійкость до помилок, відновлюваність та доступность.

3.3 Практична реалізація методики оцінки якості програмних засобів

Ставиться завдання експертної оцінки значення показника «Практичність» продуктів ринку ПЗ підприємницької діяльності в будівельній компанії. Для експертної оцінки було складено анкету, що складається із тверджень, кожному з яких дана оцінка. Правильність кожного затвердження оцінюється за п'ятибальною шкалою якісних рівнів. При цьому у разі повної згоди з запропонованим твердженням було поставлено оцінку «4». У разі повної незгоди з запропонованим твердженням було поставлено оцінку «0». Інші значення визначають відповідні проміжні якісні рівні використовуваного ПЗ: «1» - «швидше ні, ніж так», «2» - «все одно», «3» - «швидше так, ніж ні».

Досліджувалися 7 програмних продуктів (системі управління контентом для реалізацій на їх основі інтернет магазина):

A_1 – OpenCart

A_2 – Joomla!

A_3 – MODX.

A_4 – WooCommerce

A_5 – Magento

A_6 – 1C-Bitrix

A_7 - UMI.CMS

Для оцінювання експерту пропонувалися 12 тверджень:

K_1 – Завжди відомо, що робити далі із програмою

K_2 - Працюючи з програмою, завжди знаєш, яку команду (дію) необхідно виконати наступною

K_3 – Інтерфейс програми інтуїтивно зрозумілий

K_4 – не змушує порушувати звичні послідовності операцій, за допомогою яких зазвичай здійснюється робота у цій предметній галузі

K 5 – Під час роботи не доводиться періодично заглядати у супровідну документацію

K 6 – Програма несподівано не зависає

K 7 – Організація меню та інформаційних списків виглядає досить логічною

K 8 – Вбудована довідка дозволяє справитися з проблемами, що виникають у ході експлуатації

K 9 – Програму можна освоювати поетапно

K 10 – Програма дозволяє швидко вводити-отримувати дані

K 11 – Використовуючи програму, завдання можуть виконуватися без надмірних дій

K 12 – Функції програми досить легко узгоджуються з вимогами користувача.

Результати експертної оцінки наведено у табл. 1.

Таблиця 3.1 Результати експертизи ПЗ за показником «Практичність»

Експертні оцінки												
Вид ПЗ	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	$До_{10}$	$До_{11}$	$До_{12}$
A_1	3	4	4	4	3	3	4	3	3	4	3	3
A_2	2	3	3	3	2	3	4	2	3	4	3	2
A_3	1	2	3	2	1	3	3	1	3	4	2	3
A_4	2	3	3	3	2	3	3	2	2	4	2	2
A_5	1	2	3	2	2	3	3	1	2	4	3	3
A_6	2	3	3	3	3	3	4	2	2	4	2	3
A_7	2	2	3	3	2	3	3	2	3	4	2	3
Нормалізовані оцінки												
Вид ПЗ	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	$До_{10}$	$До_{11}$	$До_{12}$
A_1	0,75	1	1	1	0,75	0,75	1	0,75	0,75	1	0,75	0,75
A_2	0,5	0,75	0,75	0,75	0,5	0,75	1	0,5	0,75	1	0,75	0,5
A_3	0,25	0,5	0,75	0,5	0,25	0,75	0,75	0,25	0,75	1	0,5	0,75
A_4	0,5	0,75	0,75	0,75	0,5	0,75	0,75	0,5	0,5	1	0,5	0,5
A_5	0,25	0,5	0,75	0,5	0,5	0,75	0,75	0,25	0,5	1	0,75	0,75
A_6	0,5	0,75	0,75	0,75	0,75	0,75	1	0,5	0,5	1	0,5	0,75
A_7	0,5	0,5	0,75	0,75	0,5	0,75	0,75	0,5	0,75	1	0,5	0,75

Там же наводяться нормалізовані оцінки, які потрібні для застосування представленої вище моделі. З огляду на те, що оцінна шкала була від 0 до 4, для нормалізації всі експертні оцінки ділилися на 4. Ваги оціночних критеріїв будемо вважати рівними.

Результати розрахунку показано на рис. 1. У осередках А2-А8 представлені оцінки якості ПЗ за показником «Практичність». Видно, що найкращу оцінку отримав програмний продукт A_1. У діапазоні В1-М1 представлені оцінки нездійсненності (нереалізованості) оцінних критеріїв по всій множині аналізованих програмних продуктів. Найбільш здійсненним (добре ралізованим) виявився критерій K_{10}, який отримав найвищі експертні оцінки з усіх видів ПЗ, а найбільш

нездійсненним з мінімальними оцінками став критерій K_8. Слід зазначити, що оцінки отримані за лінійною шкалою.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Оценки	6,52928	5,65524	5,04576	5,48687	6,22715	5,14159	4,63154	6,52935	5,72871	0	5,90163	5,53162
2	7,45058	0,75	1	1	1	0,75	0,75	1	0,75	0,75	1	0,75	0,75
3	6,48535	0,5	0,75	0,75	0,75	0,5	0,75	1	0,5	0,75	1	0,75	0,5
4	5,87578	0,25	0,5	0,75	0,5	0,25	0,75	0,75	0,25	0,75	1	0,5	0,75
5	6,18648	0,5	0,75	0,75	0,75	0,5	0,75	0,75	0,5	0,5	1	0,5	0,5
6	5,97798	0,25	0,5	0,75	0,5	0,5	0,75	0,75	0,25	0,5	1	0,75	0,75
7	6,48487	0,5	0,75	0,75	0,75	0,75	0,75	1	0,5	0,5	1	0,5	0,75
8	6,2961	0,5	0,5	0,75	0,75	0,5	0,75	0,75	0,5	0,75	1	0,5	0,75
9	Расчет отклонений суммы (7)												
10		0,0012	0,02028	0,00686	0,01515	0,00051	0,02548	0,00317	0,0012	0,00968	3,4E-07	0,00559	0,01489
11		0,00012	0,00288	0,00341	0,00037	0,00412	0,00186	0,01834	0,00012	0,00481	2,3E-06	0,01168	0,04922
12		0,0085	0,00302	0,00288	0,00922	0,02659	0,00552	0,00069	0,0085	0,0455	7,8E-06	4,2E-05	0,02716
13		0,0072	0,01445	6,1E-05	0,00671	0,0001	0,0001	0,00572	0,00721	0,01265	4,2E-06	0,005	0,025
14		0,01335	0,0064	0,00105	0,01449	0,00384	0,00273	0,0019	0,01335	0,00384	6,4E-06	0,05332	0,01966
15		0,00012	0,00289	0,0034	0,00037	0,03457	0,00185	0,01836	0,00012	0,03259	2,3E-06	0,02011	0,0008
16		0,00337	0,02401	0,00075	0,00337	0,0003	0,00011	0,00825	0,00337	0,01251	3,4E-06	0,00948	0,00458
17	Целевая	0,742											

Рисунок 3.1 - Результати розрахунку оцінок ПЗ за показником «Практичність»

Аналогічно можна отримати оцінки ПЗ за іншими оціночними показниками, а потім знову використовувати модель, взявши в якості вихідних даних отримані оцінки за показниками, але попередньо нормувавши їх на одиничну шкалу. У результаті отримаємо оцінки якості програмного забезпечення по всій групі показників.

Висновки

Запропоновано методику оцінювання якості ПЗ, засновану на методі Раша оцінки латентних змінних. Оцінки, отримані за описаною методикою, мають такі переваги в порівнянні класичними адитивними методами оцінювання:

1. Крім оцінок якості ПЗ, метод дозволяє отримати оцінки здійсненності оціночних показників, що характеризують, наскільки програмні продукти в своїй сукупності задовольняють кожному показнику.
2. Оцінки вимірюються за лінійною шкалою.

3. Оцінки якості ПЗ є їх унікальними властивостями і не залежать від оцінних критеріїв.

4. Отримані оцінки більш гнучкі, тому що враховують здійсненність показників.

Запропоновану методику можна застосовувати і для оцінки ПЗ за кожним показником окремо на основі деяких оціночних критеріїв.

ВИСНОВКИ

В результаті аналізу сучасних нормативних документів з якості програмних засобів зроблений висновок, що якість накопичується в продукті при складному виробництві кумулятивним чином, причому, внесок у якість, здійснений на ранніх стадіях, має сильніший вплив на кінцевий продукт, ніж на більш пізніх стадіях. Це підтверджується всією практикою програмування, наприклад відомо, що недоліки проектування систем не можуть бути компенсовані високою якістю кодування.

Таким чином, для управління якістю побудови складної системи необхідно проводити вибір виробників на основі вимірювання ступеня зрілості та прозорості процесів розробки, що використовуються. Вимірювання якості процесу розробки підрядників є важливою складовою загального управління якістю, важливішим, ніж вимір якості результуючого продукту, що виробляється в ході приймально-здавальних випробувань.

В роботі запропоновано методику оцінювання якості ПЗ, засновану на методі Раша оцінки латентних змінних. Оцінки, отримані за описаною методикою, мають такі переваги в порівнянні класичними адитивними методами оцінювання:

Запропоновану методику можна застосовувати і для оцінки ПЗ за кожним показником окремо на основі деяких оціночних критеріїв.

СПИСОК ВИКОРАСТАНИХ ДЖЕРЕЛ

1. Андон Ф.І., Суслов В.Ю., Коваль Г.І., Коротун Т.М. Основи якості програмних систем.–Київ, Академперіодика.– 2002.–502с.
2. Бабенко Л.П., Лаврищева О.М Основи програмної інженерії. Підручник Київ: Знання, 2001. - 269 с
3. Боем Б.У. Інженерне проектування програмного забезпечення. Пров. з англ. / За ред. А.А. Красилів. - М.: Вид-во Радіо і зв'язок, 1985. - 512 с.
4. Боем Б.У., Браун Дж., Каспар Х. та ін. Характеристики якості програмного забезпечення. М. Світ, 1981.
5. Воробйов В. І., Копилцов А. В., Пальчун Б. П., Юсупов Р. Методи та моделі оцінювання якості програмного забезпечення. М. С-Пб.: СПІІРАН.1992.-33с.
6. Колдовський В. Розробка ПЗ: оцінка результату. Комп'ютерний огляд №34 (553) 2006
7. Кулаков А.Ю. Оцінка якості програм ЕОМ .-Київ: Техніка.-1984.-167с.
8. Липаєв В. Якість програмного забезпечення. - М.: Фінанси та статистика, 1983.
9. Липаєв В.В. Методи забезпечення якості великомасштабних програмних систем. - М.: СИНТЕГ. - 2003. - 510 с.
10. Липаєв В.В. Забезпечення якості програмних засобів. Методи та стандарти. - М.: Синтег, 2001. - 380 с.
11. Орлов З. Технології розробки програмного забезпечення: Підручник/ - СПб.: Пітер, 2002. - 464 з.: ил.
12. Соммервіл І. Інженерія програмного забезпечення. 6-видання.-Москва-Санкт-Петербург-Київ, 2002.-623 с.
13. Фокс Дж. Програмне забезпечення та її розробка М.: " Світ " , 1982.
14. Холстед М.Х. Початок науки про програми. - М.: Фінанси та Статистика, 1981.
15. Boehm BW The COCOMO 2.0 Software Cost Estimation Model . - American Programmer . - 2000. - 586 p.
16. ISO/IEC 9126 Software engineering - Product quality - Part 1: Quality model , 2001

17. ISO/IEC, ISO/IEC 25000: Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. Geneva: International Organization for Standardization, 2005. URL: http://www. iso.org/iso/iso_catalogue/catalogue_tc/catalogue_ detail.htm?csnumber=35683 (дата звернення: 08.01.2018).

18. Качество программных средств. Основные понятия и определения. URL:

19. ISO/IEC, ISO/IEC 25000: Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. Geneva: International Organization for Standardization, 2005. URL: http://www. iso.org/iso/iso_catalogue/catalogue_tc/catalogue_ detail.htm?csnumber=35683 (дата обращения: 08.01.2018).

20. Бураков В. В. Модель качества программных средств // Информационно-управляющие системы. 2009. № 2. С. 75–78.2017. № 4. С. 166–168.

21. Заде Л.А. Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.: Мир, 1976.

22. А. Кофман. Введение в теорию нечетких множеств. – М.: Радио и связь, 1982.

23. С.А. Орлов. Технологии разработки программного обеспечения: Учебник. – СПб: Питер, 2002. – 464 с.

24. Липаев В.В. Управление разработкой программных средств: Методы, стандарты, технология. – М.: Финансы и статистика, 1993.

25. Черников Б. В., Поклонов Б. Е. Оценка качества программного обеспечения. М. : ФОРУМ : ИНФРА-М, 2012. 400 с.

26. Методы оценки технико-экономических показателей проектов программных средств. URL: http://docplayer.ru/33018659-Metody-ocenki-tehniko-ekonomicheskih-pokazateleyproektov-programmnyh-sredstv.html (дата обращения: 08.01.2018).