УДК 004.415.5:621.3.019.3:005.591.6

А. С. Шантир

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ МОДЕЛЕЙ ОЦІНКИ ЯКОСТІ НА ЕТАПАХ РОЗРОБКИ ТА ВИКОРИСТАННЯ ПРОГРАМНИХ СИСТЕМ

Державний університет інформаційно-комунікаційних технологій, Київ

Анотація. В статті аналізуються особливості застосування моделей оцінки якості на різних етапах розробки та використання програмних систем з метою забезпечення їхньої високої якості. Головною метою дослідження є визначення нових комбінованих підходів, щодо оцінки ефективності та впливу використання моделей якості на різних етапах життєвого циклу програмних продуктів. У ході роботи використовується системний підхід до аналізу, що враховує взаємодію моделей якості з різними етапами проектування, розробки, тестування та експлуатації програмного забезпечення. Зокрема, розглядається роль моделей якості у підвищенні ефективності процесів розробки та їх вплив на остаточну якість продуктів. Методологія включає в себе аналіз існуючих моделей якості, їх адаптацію до конкретних умов проекту, а також вивчення практичних прикладів впровадження моделей якості в реальних проектах. Реалізується аналіз існуючих моделей якості, їх адоптацій до конкретних умов програмних системних проектів та аналіз впровадження в реальних проектах. Цей підхід дозволив нам отримати конкретні результати та розкрити ключові аспекти впровадження моделей якості. Розглядається взаємодія моделей якості із загальними стратегіями забезпечення якості та їхній вплив на підвищення продуктивності та надійності програмних систем. В загальнонауковому аспекті дослідження зводиться до оцінки ефективності цього підходу та визначенні його ключових особливостей. Подальший розгляд показав, що використання нових комбінованих моделей якості на етапах проектування, розробки та тестування сприяє покращенню різних аспектів якості програмного забезпечення. Вони не лише визначають критерії якості, але й сприяють забезпеченню відповідності цим критеріям протягом усього життєвого циклу проекту. Отримані результати підтверджують важливість використання комбінованих моделей якості на всіх етапах розробки програмних систем. Це сприяє не лише покращенню якості кінцевого продукту, але і ефективності всього процесу розробки. Наші результати можуть слугувати основою для практичного впровадження моделей якості в проекти програмного забезпечення та покращення загального рівня якості у цій галузі.

Ключові слова: Інтеграційна модель зрілості можливостей, процеси забезпечення якості програмного забезпечення, квадранти гнучкого тестування, Шість Сигм, загальне управління якістю, процес програмного забезпечення, фреймворк архітектури групи.

Abstract. The article examines the peculiarities of applying quality models at different stages of development and utilization of software systems to ensure their high quality. The main aim of the research is to identify new combined approaches for evaluating the effectiveness and impact of quality models usage at various stages of software product life cycles. The work employs a systematic approach to analysis, considering the interaction of quality models with different stages of software design, development, testing, and operation. Specifically, the role of quality models in enhancing the efficiency of development processes and their impact on the final product quality is discussed. The methodology involves analyzing existing quality models, adapting them to specific project conditions, and studying practical examples of quality model implementation in real projects. An analysis of existing quality models, their adaptation to specific conditions of software system projects, and an analysis of implementation in real projects are carried out. This approach has enabled us to obtain specific results and reveal key aspects of quality model implementation. The interaction of quality models with overall quality assurance strategies and their impact on improving the productivity and reliability of software systems is considered. In a general scientific aspect, the research boils down to evaluating the effectiveness of this approach and determining its key features. Further examination showed that the use of new combined quality models in the stages of design, development, and testing contributes to the improvement of various aspects of software quality. They not only define quality criteria but also ensure compliance with these criteria throughout the project life cycle. The obtained results confirm the importance of using combined quality models at all stages of software development. This contributes not only to the improvement of the final product quality but also to the efficiency of the entire development process. Our results can serve as a basis for the practical implementation of quality models in software projects and improving the overall quality level in this field.

Key words: Capability maturity model integration, software quality assurance processes, agile testing quadrants, Six Sigma, total quality managemen, software process, group architecture framework.

DOI: https://doi.org/10.31649/1999-9941-2024-59-1-127-138.

Вступ

В сучасному світі, де програмні системи стають невід'ємною частиною практично будь-якої сфери діяльності, забезпечення їх якості має вирішальне значення. Зростання вимог до надійності, ефективності та безпеки програмних продуктів вимагає від розробників постійного вдосконалення методів тестування та контролю якості.Перед впровадженням будь-якої програмної системи необхідно ретельно проаналізувати її вимоги та визначити критерії якості, які відображають очікувані властивості та параметри. Моделі якості допомагають не лише визначити ці критерії, але й встановити методи їх вимірювання та оцінки. Зважаючи на вищенаведене досить актуальною проблемою є розуміння особливостей та переваг використання моделей якості на різних етапах розробки та використання програмних систем,яка поправу стала ключовою в області забезпечення успішної реалізації проектів та задоволення потреб користувачів. Аналіз цих аспектів допомагає ефективно впроваджувати стратегії контролю якості та забезпечувати високу якість програмних продуктів.

Актуальність

Актуальність проблеми забезпечення якості програмних систем на кожному етапі їх життєвого циклу відображається в потребі постійного пошуку нових ефективних методів оцінки та контролю якості програмних продуктів. Навіть при успішній розробці даних продуктів, важливо забезпечити їх високу якість протягом усього періоду їх використання. Таким чином, розуміння особливостей та вибір оптимальних моделей якості стає ключовим для ефективної розробки та використання програмних систем. За останні роки програмне забезпечення проникає в усі сфери життя, проблема забезпечення його якості стає надзвичайно актуальною. Висока конкуренція, швидкі темпи розвитку технологій та зростання очікувань користувачів створюють виклики для розробників у плані забезпечення надійності, продуктивності та безпеки програмних продуктів. При цьому, складність програмних систем та розмір їх кодової бази ускладнюють процес тестування та виявлення помилок, що може призвести до зниження якості та недоліків у функціональності. Таким чином, необхідність вдосконалення методів оцінки, контролю та забезпечення якості програмного забезпечення є актуальною проблемою, яка потребує пошуку ефективних рішень.

Мета

Метою дослідження є визначення та розробка нових комбінованих підходів, щодо оцінки ефективності та впливу використання моделей якості на різних етапах життєвого циклу програмних продуктів.

Задачі

1. Проаналізувати відомі проблеми які виникають в моделях якості, котрі застосовуються для оцінки та підтримки якості програмних систем на різних етапах їх розробки та використання.
2. Виконати постановку завдання розробки комбінованої системної моделі оцінки якості в межах адаптактивного застосування моделей якості із цілю підвищення ефективності процесів розробки та їх впливу на механізм оцінювання якості програмних продуктів.
3. Розробити комбінований метод оцінки ефективності та впливу використання моделей якості на різних етапах життєвого циклу програмних продуктів.

Розв'язання задач

1. Аналіз проблем. У реаліях швидкозмінюваного інформаційного середовища, розробка програмних систем (ПС) стає відокремлюючою лінією між успіхом та викликами у сфері інформаційних технологій. Постійні та динамічні зміни у вимогах користувачів, ростучі обсяги даних та загострена конкуренція вимагають від розробників не тільки креативності та експертності, але і вдосконалення стратегій управління якістю програмного забезпечення. В останні роки відзначається активний розвиток підходів до розробки, орієнтованих на досягнення високої якості продукту [1]. Застосування моделей якості на різних етапах життєвого циклу програмної системи визнається, як важлива стратегія для покращення процесів розробки та забезпечення високого рівня функціональності та надійності[2]. У цьому контексті, наукове дослідження та аналіз останніх вирішених проблем в галузі застосування моделей якості стають невід'ємною частиною стратегічного планування для компаній і розробників програмного забезпечення. Проте, залишаються відкритими питання невизначеності в розробці нових методичних технологій із оцінки якості ПС [3] та пошуку оптимальних рішень у змінюючихся умовах ринку, що потребує глибокого наукового аналізу ролі та ефективності застосування моделей якості на кожному етапі розробки програмних систем, розглядаючи як вирішені проблеми, так і ті аспекти, які залишаються під великим питанням. Розглядання цих питань має на меті поглиблене розуміння та визначення оптимальних підходів для досягнення високих стандартів якості у розробці програмного забезпечення. Дослідження в області застосування моделей якості в розробці та використанні програмних систем відбувається постійно, і багато проблем вже мають вирішення, але також існують ті, які залишаються відкритими.

Вирішені проблеми:

– Розробка стандартів якості: Великий крок був зроблений у напрямку стандартизації методів та метрик оцінки якості програмних систем [1–15]. Наприклад, ISO 25010 визначає модель якості програмного забезпечення визначає набір характеристик та сценарії їх використання;

– Автоматизація тестування [1, 4]: Автоматизовані засоби тестування дозволяють ефективніше виявляти помилки та проблеми якості на ранніх етапах розробки;

– Моделювання процесів розробки [3–5]: Застосування моделей для опису та аналізу процесів розробки сприяє покращенню якості виробництва програмного забезпечення.

Невирішені, або актуальні проблеми:

– Оцінка користувацького досвіду: Визначення об'єктивних метрик для оцінки користувацького досвіду залишається складною задачею, оскільки він є суб'єктивним поняттям [4];

– Моделі для безпеки та захисту: Забезпечення ефективних моделей для визначення та оцінювання безпеки програмних систем залишається відкритою проблемою [7];

– Врахування вимог до взаємодії та масштабованості: Зростання складності систем та вимог до їх взаємодії може створювати виклики при розробці моделей якості, які повинні враховувати ці аспекти [8];

– Врахування змінливості вимог: В межах технологічного прогресу нині постійно змінюються вимоги до програмних систем, які можуть призводити до потреби в адаптації моделей якості для забезпечення їх актуальності та ефективності [9];

– Врахування впливу різних технологій: Швидкі зміни технологій можуть створювати труднощі у врахуванні їх впливу на якість програмного забезпечення [10];

– Моніторинг у реальному часі: Розробка засобів моніторингу, які надають інформацію в реальному часі та дозволяють швидко реагувати на проблеми, залишається важливою задачею [11].

Зазначені вище фахові тематичні спрямованості відображають сучасні виклики та тенденції в області оцінки якості програмного забезпечення, і вони можуть бути об'єктом активного дослідження та розвитку.

В табл.1. наведено результати порівняльного аналізу щодо застосування популярних моделей якості на різних етапах розробки та використання програмних систем.

З табл.1. видно, що кожна модель якості має свої унікальні особливості, які можуть бути корисними на різних етапах розробки та використання програмних систем. Зокрема деякі моделі, такі як ISO/IEC 25010 та IEEE 730, акцентують на оцінці якості продукту та процесів розробки, зазначаючи ключові критерії, які можна використовувати для цього. Інші, наприклад, CMMI та SPICE, спрямовані на підвищення зрілості процесів розробки програмного забезпечення. Деякі моделі, такі як Agile Testing Quadrants та Lean Software Development, спеціалізуються на використанні в Agile-процесах та мінімізації витрат. Зважаючи на вище наведене вибір моделі якості повинен залежати від конкретних потреб та характеристик проекту. Інтеграція моделей якості з процесами розробки може допомогти забезпечити більш ефективну та якісну розробку програмного забезпечення.

Відповідно аналізу наукових праць [1–15] застосування моделей якості на різних етапах розробки та використання програмних систем може стикатися з рядом проблем. Зокрема в праці [1] піднімається проблема, щодо недооцінки вимог до якості, дана проблема зводиться до того, що не завжди вдається коректно визначити та врахувати всі вимоги до якості програмної системи на ранніх етапах розробки. Аналізуючи біль детально дану проблему можна зробити висновок, що проблема недооцінки вимог до якості програмної системи полягає в тому, що на ранніх етапах розробки не завжди вдається коректно визначити та врахувати всі аспекти якості, які можуть вплинути на подальший процес розробки та експлуатації програмного продукту. Це може призвести до недоліків у функціональності, надійності, ефективності та інших аспектах якості програмного забезпечення. З нашої позиції доцільно відмітити, що для вирішення цієї проблеми важливо застосовувати адекватні методи та моделі управління якістю, які дозволять виявляти, визначати та враховувати вимоги до якості на початкових етапах розробки. Також в даному аспекті досить важливо використовувати практики та методології, що сприяють ранньому виявленню та вирішенню проблем якості, такі як Agile-підходи, Continuous Integration та Continuous Delivery, тестування на ранніх етапах розробки тощо. Загалом, управління якістю на ранніх етапах розробки є критично важливим для успішної поставки програмного забезпечення, і це вимагає уваги до всіх аспектів якості від самого початку процесу розробки.

В праці [2] піднімається проблема, щодо недостатнього охоплення тестування в межах застосування моделей якості на різних етапах розробки та використання програмних систем: не завжди можна забезпечити повне тестування всіх аспектів програми, що може призводити до уникнення або неправильного виявлення деяких дефектів, що загрожує якості та надійності програмного забезпечення. Згідно з [2–4] для вирішення цієї проблеми важливо використовувати комплексний підхід до тестування, який охоплює різні аспекти програмного забезпечення, такі як функціональність, надійність, ефективність, безпека та інші.

Згідно з [5] у вирішенні вище зазначеної проблеми важливо використовувати автоматизоване тестування та інструменти для підвищення ефективності тестування та забезпечення більш широкого охоплення коду.

Аналізуючи погляди авторів праці [6] доцільно погодитися що в піднятій проблемі виникає нагальная необхідність, щодо врахування особливостей конкретного проекту та контексту використання програмного забезпечення при плануванні та виконанні тестування, щоб забезпечити його високу якість та надійність.

Автори праці [9] акцентують увагу на потребі, щодо надання належної уваги тестуванню на ранніх етапах розробки, що в свою чергу дозволить виявляти та виправляти дефекти на ранніх стадіях, коли вони ще не вирішилися в серйозні проблеми.

Таблиця 1 – Результати порівняльного аналізу щодо застосування популярних моделей якості на різних етапах розробки та використання програмних систем

Найменування моделі якості	**Етап Застосування**	**Основна мета**	**Ключові критерії**	**Застосування в Agile**	**Інтеграція з Процесами**
ISO/IEC 25010	Різні етапи від концепції до підтримки	Оцінка якості продукту та процесів	Функціональність, Надійність, Ефективність	Так	Можлива
CMMI	Розробка та підтримка	Зрілість процесів та управління	Зрілість процесів, Продуктивність	Не прямо спрямовано	Є процеси зрілості
IEEE 730	Етапи розробки та тестування	Вимоги до процесів розробки	Вимоги до процесів розробки	Так	Можлива
Agile Testing Quadrants	Особливо актуальна в Agile	Класифікація тестування в Agile	Цілі тестування на кожному квадранті	Так	Можливість інтеграції в Agile процес
Six Sigma	Етап підтримки	Покращення якості та ефективності	Виявлення та виправлення помилок	Неявно	Вимагає особливостей процесів
TQM (Total Quality Management)	Різні етапи	Загальне управління якістю	Клієнтське задоволення, Залученість персоналу, Постійне покращення	Так	Так
SPICE (Software Process Improvement and Capability Determination)	Розробка та експлуатація	Покращення процесів розробки програмного забезпечення	Здатність до процесів, Ефективність	Можливе	Так
TOGAF (The Open Group Architecture Framework)	Різні етапи	Управління та підтримка корпоративної архітектури	Архітектурна якість, Взаємодія з бізнес-потребами	Частково	Так
SPICE (ISO/IEC 15504)	Розробка та експлуатація	Визначення та оцінка процесів розробки ПЗ	Процесна зрілість та продуктивність	Неявно	Так
COBIT (Control Objectives for Information and Related Technologies)	Управління та оптимізація іТ	Керування IT з метою досягнення бізнес-цілей	Забезпечення стійкості, Забезпечення безпеки	Частково	Так
Lean Software Development	Розробка та підтримка	Мінімізація витрат та оптимізація	Велика цінність, Мінімальне марноспоживання	Так	Можлива
ITIL (Information Technology Infrastructure Library)	Управління послугами IT	Забезпечення якісних IT-послуг	Забезпечення належного рівня сервісу, Ефективність процесів	Частково	Так
Capability Maturity Model (CMM)	Розробка та підтримка	Зрілість процесів розробки ПЗ	Зрілість процесів, Продуктивність	Не прямо спрямовано	Є процеси зрілості

В праці [10] порушується проблема, щодо неспроможності передбачити в моделях якості умови експлуатації в яких перебуватимуть ПС: моделі якості можуть не враховувати всі можливі умови реального використання системи, що призводить до недоліків у функціональності або ефективності.

Про дану проблему також згадується в праці [11] в якій вказується, що вона може призвести до недоліків у функціональності, або ефективності ПС в умовах її експлуатації. В межах вирішення цієї проблеми згідно поглядів, які наведені в [10–11] важливо застосовувати методології та практики, які сприя-

ють адаптації програмного забезпечення до різних умов експлуатації: наприклад, Agile-підходи передбачають ітеративний розвиток програмного забезпечення та зміну відповідно до змін у вимогах користувачів або умовах експлуатації. Також важливо використовувати тестування з реальними умовами, або тестування на сценаріях, яке відображає реальні ситуації використання системи. Також доцільно зважати на тещо вирішення зазначеної проблеми потребує налагодження додаткової, комунікації з користувачами та стейкхолдерами може допомогти зрозуміти реальні очікування та умови експлуатації системи, що дозволить врахувати їх у моделях якості та в процесі розробки. Такий підхід сприяє покращенню функціональності та ефективності програмного забезпечення в умовах його реального використання.

В праці [12] відмічаються проблеми з масштабованістю: Деякі моделі якості можуть бути непридатними для масштабування, особливо при рості розміру та складності програмної системи. Для вирішення цієї проблеми необхідно розробляти та використовувати моделі якості, які можуть ефективно працювати на різних рівнях масштабу, від невеликих проектів до великих розподілених систем. Це може включати в себе розробку або адаптацію моделей, які забезпечують гнучкість та можливість розширення, щоб вони могли адаптуватися до зростаючих потреб системи.

Автори праць [12–13] наголошують, що досить важливо враховувати масштабованість в процесі розробки та впровадження моделей якості. Це означає, що моделі повинні бути легко впроваджуватися та масштабуватися разом з ростом системи без великих зусиль зі змінами або модифікаціями. Загалом, для успішного вирішення проблеми масштабованості моделей якості важливо приділяти увагу їх архітектурі та гнучкості, щоб вони могли ефективно функціонувати в умовах зростання розміру та складності програмної системи.

В праці [14] підкреслюється проблематика, щодо «нестабільності вимог якості»: Вимоги до якості можуть змінюватися протягом життєвого циклу проекту, і моделі можуть стати застарілими або непридатними для нових умов. Для вирішення цієї проблеми важливо використовувати гнучкі методології розробки програмного забезпечення, які дозволяють легко вносити зміни у вимоги та моделі якості протягом життєвого циклу проекту. Наприклад, Agile-підходи передбачають ітеративний розвиток та постійне вдосконалення, що дозволяє адаптувати вимоги та моделі якості до змін у вимогах бізнесу або умовах ринку.

Згідно з [15] в межах піднятої вище проблеми виникає необхідність в забезпеченні підтримання постійного моніторингу та оновлення моделей якості, щоб вони відповідали актуальним вимогам та умовам. Це може включати в себе регулярну перевірку та оцінку моделей якості, а також внесення необхідних змін та коригувань. Загалом, для успішного вирішення проблеми «нестабільності вимог якості» важливо мати гнучкість у розробці та управлінні вимогами, а також постійно оновлювати та підтримувати актуальні моделі якості.

В праці [5] підкреслюється проблематика відсутності однозначних метрик якості: Визначення якості може бути суб'єктивним, і відсутність чітких метрик може ускладнити оцінку якості програмної системи:порівняння якості між різними системами, або версіями однієї системи. Автори вищезазначеної праці наголошують на тому, що для вирішення цієї проблеми важливо розробляти та застосовувати стандартизовані метрики якості, які були б об'єктивними та відображали б реальні характеристики програмного забезпечення. Такі метрики можуть включати в себе показники продуктивності, ефективності, надійності, безпеки, інтерфейсу користувача тощо. З даного приводу важливо враховувати контекст використання програмного забезпечення при визначенні метрик якості.

Згідно з [7] одні й ті ж метрики можуть мати різне значення для різних типів програмних продуктів або для різних галузей застосування. Наприклад, для веб-додатків можуть бути важливими метрики швидкодії, часу завантаження сторінок, частоти відмов, а для програмного забезпечення в області критичних застосувань – метрики надійності та безпеки. Загалом, розробка і використання об'єктивних та стандартизованих метрик якості є важливою складовою управління якістю програмного забезпечення і дозволяє об'єктивно оцінювати та порівнювати рівень якості різних систем.

В праці [14] підкреслюється проблематика недостатності моніторингу у реальному часі: Відсутність засобів моніторингу та аналізу в реальному часі може призвести до затримок в виявленні та виправленні проблем, що може погіршити якість та надійність програмного продукту. Для вирішення цієї проблеми важливо використовувати засоби моніторингу та аналізу, які дозволяють виявляти проблеми в реальному часі або навіть передбачати їх виникнення на основі аналізу даних. Наприклад, системи моніторингу продуктивності, навантаження та відмов можуть допомогти виявляти проблеми в роботі системи та реагувати на них негайно. Також важливо використовувати автоматизовані засоби тестування та контролю якості, які дозволяють проводити тестування та аналіз результатів в реальному часі. Це дозволяє оперативно виявляти проблеми та вносити необхідні корективи ще на ранніх етапах розробки або в етапі експлуатації. Загалом, моніторинг у реальному часі є ключовою складовою управління якістю програмного забезпечення, оскільки дозволяє швидко реагувати на виявлені проблеми та забезпечує підтримку високої якості та надійності програмного продукту.

В праці [15] особливої уваги заслуговують проблеми з безпекою: моделі якості можуть не враховувати аспекти безпеки, що може призводити до вразливостей та забезпечувати недостатню захист від атак. Для вирішення цієї проблеми важливо включати аспекти безпеки в моделі якості та процеси управління якістю. Це може включати аудит безпеки, тестування на вразливості, перевірку безпеки на ранніх етапах розробки, а також впровадження засобів захисту в програмне забезпечення. Також важливо враховувати конкретні потенційні загрози та вразливості для програмного забезпечення у відповідній галузі або контексті використання. Це дозволить розробникам та управлінцям якості ефективно оцінювати ризики та вживати відповідні заходи забезпечення безпеки. Загалом, безпека програмного забезпечення повинна бути важливою складовою будь-якої моделі якості, і вона повинна бути врахована на всіх етапах розробки та використання програмного продукту.

Автори праці [13] звертають увагу на проблему недостатності врахування користувацького досвіду: В деяких випадках моделі можуть недооцінювати важливість користувацького досвіду, що може впливати на прийняття програмної системи користувачами. Для вирішення цієї проблеми важливо включати аспекти користувацького досвіду в моделі якості та процеси управління якістю. Це може включати оцінку зручності використання, швидкості реакції інтерфейсу, відповідність функціональності потребам користувачів та інші аспекти, що впливають на задоволеність та прийняття програмного продукту. Також важливо залучати користувачів до процесу розробки та тестування програмного забезпечення, щоб отримати зворотний зв'язок щодо їхніх потреб та вимог до користувацького досвіду. Це дозволить розробникам налагодити програмний продукт так, щоб він краще відповідав очікуванням та потребам користувачів. Загалом, врахування користувацького досвіду у моделях якості та процесах управління якістю є ключовим для забезпечення успішного прийняття та використання програмного забезпечення користувачами.

Автори праці [12] окрім вище зазначених проблем також акцентують наукову увагу на проблемі неспроможності враховувати контекст: моделі якості можуть не завжди враховувати контекст використання програмної системи, що призводить до втрати реалізму в оцінці її якості. Це означає, що деякі моделі якості можуть не достатньо, або взагалі не враховувати особливості середовища, в якому програмна система буде використовуватися, що може призводити до втрати реалізму в оцінці її якості. Для вирішення цієї проблеми важливо розробляти моделі якості, які були б призначені для конкретних контекстів використання програмного забезпечення. Це може включати в себе адаптацію існуючих моделей до специфічних умов або розробку нових моделей, які враховують особливості конкретного домену або індустрії. Крім того, важливо збирати та аналізувати дані про реальне використання програмного забезпечення, щоб зрозуміти його характеристики та вимоги в конкретному контексті. Це дозволить розробникам та управлінцям якістю зробити більш об'єктивну оцінку якості програмного продукту і прийняти відповідні заходи для його поліпшення. Загалом, врахування контексту використання є важливим аспектом при розробці та оцінці якості програмного забезпечення, і ця проблема потребує уваги, як у наукових дослідженнях, так і в практичній роботі з програмними системами.

Згідно з результати аналізу праць [1–15] оцінка якості програмних систем може включати в себе різні аспекти, і існує кілька міжнародних стандартів, які можуть бути використані для цього. Зазвичай, ці стандарти визначають загальні принципи оцінки якості та надають рамки для впровадження методів оцінки. Даний метод оцінки якості може бути включений у ряд стандартів, таких як:

– ISO/IEC 9126: Software Engineering –Product Quality: Цей стандарт визначає модель якості програмного забезпечення та надає метрики для вимірювання якості в аспектах, таких як функціональність, ефективність, надійність і т. д.;

– ISO/IEC 25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models: Цей стандарт є оновленою версією ISO/IEC 9126 та надає більш сучасну модель якості програмного забезпечення, включаючи аспекти, такі як сумісність та безпека;

– IEEE Std 730: Software Quality Assurance Processes: Стандарт визначає процеси забезпечення якості програмного забезпечення і може включати стратегії тестування, такі як юніт-тестування.;

– IEEE Std 829: IEEE Standard for Software Test Documentation: Цей стандарт визначає вимоги до документації тестування, що може бути використано для оцінки якості тестування, включаючи юніт-тестування;

– ISO/IEC 33001: Process Assessment – Concepts and Vocabulary: Стандарт надає концепції та терміни, пов'язані з оцінкою процесів розробки програмного забезпечення, які включають у себе тестування та методи оцінки якості;

– ISO/IEC 29119: Software and systems engineering – Software testing: Це набір стандартів, який визначає вимоги до процесів та управління тестуванням програмного забезпечення.

В загальному спектрі наведені вище стандарти визначають фундаментальні принципи оцінки якості та надають рамки для впровадження методів оцінки. Інтеграція цих методів оцінки у різні стандарти дозволяє більш повно і системно оцінювати якість програмного забезпечення на всіх етапах його життєвого

циклу. Комбінована модель дозволяє врахувати специфіку кожного конкретного проекту шляхом адаптації ваг кожної метрики в залежності від його потреб та характеристик. Математично це виражається через можливість зміни ваг для кожної метрики у формулі оцінки. Відповідно [12] кожна метрика якості вказує на певний аспект програмного продукту, що допомагає виявити недоліки та проблеми. Математично це виражається через аналіз показників метрик та їх порівняння з прийнятними стандартами або поріговими значеннями.

2. Постановка завдання. Завдання полягає в розробці комбінованої системної моделі оцінки якості програмного продукту, яка базується на інтеграції різних моделей якості, таких як CMMI, IEEE 730, Agile Testing Quadrants, Six Sigma, TQM, SPICE та TOGAF, з метою забезпечення комплексного та повного оцінювання якості продукту. Модель передбачає використання ряду метрик, таких, як частота виникнення помилок, час відгуку, використання пам'яті, пропускна здатність, зручність використання та безпека, які оцінюються на різних етапах розробки та експлуатації програмного продукту. Узагальнена ідея постановки даного завдання передбачає створення моделі, яка буде враховувати особливості кожної з використаних моделей та метрик, забезпечуючи об'єктивне оцінювання якості програмного продукту з урахуванням вимог користувача та відповідності стандартам якості. Вище зазначена постановка завдання визначає потребу в розробці комбінованої системної моделі оцінки якості програмного продукту, яка базується на інтеграції різноманітних моделей та метрик якості з різних наукових областей.

Формуваня вирішення даного завдання є досить важливим з ряду наукових та практичних причин, а саме:

– Мультидисциплінарний підхід: узагальнена потреба в узагальнені підходу, який буде базується на інтеграції різних моделей та метрик, враховує різноманітність аспектів якості програмного продукту, таких як функціональність, продуктивність, зручність використання та безпека. Це дозволяє розглядати якість з різних точок зору та забезпечує більш повне оцінювання;

– Використання передових практик: Інтеграція таких відомих моделей як CMMI, Six Sigma, TQM та TOGAF дозволяє використовувати передові методики та практики управління якістю та розробки програмного забезпечення. На пратиці це забезпечує високий рівень якості в розробці програмних продуктів та їх відповідності числених стандартів якості;

– Комплексний підхід до вимірювання якості: потреба взастосуванні метрик, таких, як частота виникнення помилок, час відгуку, використання пам'яті, пропускна здатність тощо, враховують різні аспекти якості, такі як надійність, продуктивність та ефективність використання ресурсів;

– Відповідність науковим стандартам: Розробка комбінованої системної моделі ґрунтується на наукових дослідженнях та теоріях у галузі управління якістю та інформаційних технологій, що забезпечує відповідність вимогам наукової області та підвищує довіру до результатів оцінювання.

Обгрунтування доцільності розроблення комбінованої моделі зводиться до її переваг, а саме:

– Комплексний підхід до якості: Поєднання різних методик та стандартів дозволяє охопити різні аспекти якості програмного продукту, включаючи процеси розробки, тестування, управління якістю та архітектурні вимоги;

– Використання найкращих практик: Включення таких широко визнаних методик, як CMMI, Six Sigma, Agile Testing Quadrants тощо, дозволяє використовувати найкращі практики відповідно до конкретних потреб та характеристик проекту;

– Гнучкість та адаптивність: Модель може адаптуватися до різних типів проектів та специфічних потреб команди розробників. Ваги кожного з методик можуть змінюватися в залежності від контексту;

– Постійне поліпшення: Використання методик Total Quality Management (TQM), Six Sigma та SPICE дозволяє постійно вдосконалювати процеси розробки та управління якістю, що сприяє підвищенню якості програмного продукту;

– Об'єктивність оцінки: Включення різних метрик та стандартів дозволяє забезпечити більш об'єктивну оцінку якості програмного продукту та процесів його розробки.

Отже постановка завдання визначає потребу, щодо створення високоефективної та науково обгрунтованої моделі оцінки якості програмного продукту, яка враховує різноманітність вимог та стандартів якості.

3. Розробка методу. В межах розробки комбінаційної моделі була задіяна комбінація наступних моделей: CMMI (Capability Maturity Model Integration), IEEE 730 (Software Quality Assurance Processes), Agile Testing Quadrants, Agile Testing Quadrants, Six Sigma, TQM (Total Quality Management), SPICE (Software Process Improvement and Capability Determination) та TOGAF (The Open Group Architecture Framework).

В запропонованій комбінованій моделі передбачається застосування наступних метрик:

– Частота виникнення помилок (Error Rate): Цю метрику можна застосувати на будь-якому етапі розробки та використання програмного продукту, але особливо важливо виміряти її під час тестування та експлуатації для виявлення недоліків та вдосконалення якості [5];

– Час відгуку (Response Time): Цю метрику можна виміряти на етапах тестування та використання для оцінки продуктивності та швидкодії програмного продукту [7];

– Використання пам'яті (Memory Usage): Вимірювання використання пам'яті зазвичай проводиться під час тестування та використання програмного продукту для визначення ефективності використання ресурсів [4];

– Пропускна здатність (Throughput): Ця метрика зазвичай оцінюється під час тестування програмного продукту для визначення його продуктивності та швидкодії під навантаженням [9];

– Зручність використання (Usability): Оцінка зручності використання зазвичай проводиться на етапі дизайну та тестування для забезпечення зручного та інтуїтивно зрозумілого інтерфейсу користувача [10];

– Безпека (Security): Оцінка безпеки може бути важливою на будь-якому етапі розробки, але особливо на етапах аналізу вимог, розробки та тестування для забезпечення захисту від зловмисних атак та вразливостей [13].

Представимо повну комбіновану модель оцінки якості програмного продукту з використанням вказаних методик та стандартів. Для кращого розуміння представимо модель у вигляді модулів для кожного етапу розробки та експлуатації програмного продукту:

❖ Етапи розробки:

➢ Вимоги та архітектура:

▪ CMMI: Використання процесів забезпечення якості (QA) для визначення вимог та архітектури.

▪ TOGAF: Використання архітектурного підходу для розробки архітектури ПЗ.

➢ Проектування та розробка:

▪ Six Sigma: Використання методів для мінімізації дефектів та вдосконалення процесів розробки.

▪ SPICE: Використання процесів забезпечення якості для розробки ПЗ.

– Тестування та QA:

▪ Agile Testing Quadrants: Використання різних видів тестування для забезпечення високої якості ПЗ.

▪ IEEE 730: Визначення процесів тестування та QA.

❖ Етапи експлуатації:

➢ Впровадження та підтримка:

▪ TQM: Впровадження стратегій управління якістю для підтримки та поліпшення програмного продукту під час експлуатації.

▪ SPICE: Використання методів оцінки якості під час експлуатації для постійного поліпшення програмного продукту.

• Цикли оцінювання якості ПЗ:

➢ Під час розробки:

▪ Використання Agile Testing Quadrants для тестування на кожному етапі розробки.

▪ Проведення аудитів з використанням CMMI для визначення рівня зрілості процесів.

➢ Під час експлуатації:

▪ Використання TQM для постійного вдосконалення програмного продукту під час його експлуатації.

▪ Впровадження моніторингу та аналізу даних з використанням Six Sigma для виявлення та виправлення проблем.

Математично застосування вказаних метрик в комбінованій моделі може бути обґрунтоване через їхню спроможність забезпечити об'єктивність, комплексність оцінки, врахування специфіки проекту та виявлення недоліків та проблем програмного продукту. Матемаатично підходи до оцінки якості навдено в формулах (1– 11.):

$$Q_{CMMI} = \sum_{j=1}^{M_{CMMI}} \left(w_{CMMI_j} \times Q_{CMMI_j} \right), \quad (1)$$

де: Q_{CMMI} – це оцінка якості ПС за моделлю CMMI; w_{CMMI_j} – вага підкритерію j для моделі CMMI; Q_{CMMI_j} – оцінка підкритерію j для моделі CMMI; M_{CMMI} – підкритерії моделі CMMI.

$$Q_{IEEE730} = \sum_{j=1}^{M_{IEEE730}} \left(w_{IEEE730_j} \times Q_{IEEE730_j} \right), \quad (2)$$

де: $Q_{IEEE730}$ – це оцінка якості ПС за стандартом оцінки якості IEEE730; $w_{IEEE730_j}$ – вага підкритерію j для моделі за стандартом оцінки якості IEEE730; $Q_{IEEE730_j}$ – оцінка підкритерію j для моделі за стандартом оцінки якості IEEE730; $M_{IEEE730}$ – підкритерії моделі за стандартом оцінки якості IEEE730.

$$Q_{AgileTesting} = \sum_{j=1}^{M_{AgileTesting}} \left(w_{AgileTesting_j} \times Q_{AgileTesting_j} \right), \quad (3)$$

де: $Q_{AgileTesting}$ – це оцінка якості ПС за Agile Testing Quadrants; $w_{AgileTesting_j}$ – вага підкритерію j для моделі згідно з Agile Testing Quadrants; $Q_{AgileTesting_j}$ – оцінка підкритерію j для моделі оцінки якості ПС згідно з Agile Testing Quadrants; $M_{AgileTesting}$ – підкритерії моделі згідно з Agile Testing Quadrants.

$$Q_{SixSigma} = \sum_{j=1}^{M_{SixSigma}} \left(w_{SixSigma_j} \times Q_{SixSigma_j} \right), \quad (4)$$

де: $Q_{SixSigma}$ – це оцінка якості ПС за методологією Six Sigma; $w_{SixSigma_j}$ – вага підкритерію j для моделі оцінки якості ПС згідно з засадами Six Sigma; $Q_{SixSigma_j}$ – оцінка підкритерію j для моделі оцінки якості ПС згідно з методологією Six Sigma; $M_{SixSigma}$ – підкритерії моделі згідно з Six Sigma.

$$Q_{\text{TQM}} = \sum_{j=1}^{M_{\text{TQM}}} \left(w_{\text{TQM}_j} \times Q_{\text{TQM}_j} \right), \quad (5)$$

де: Q_{TQM} – це оцінка якості ПС за Total Quality Management; w_{TQM_j} – вага підкритерію j для моделі оцінки якості ПС згідно з засадами Total Quality Management; Q_{TQM_j} – оцінка підкритерію j для моделі оцінки якості ПС згідно з Total Quality Management; M_{TQM} – підкритерії моделі згідно з Total Quality Management.

$$Q_{\text{SPICE}} = \sum_{j=1}^{M_{\text{SPICE}}} \left(w_{\text{SPICE}_j} \times Q_{\text{SPICE}_j} \right), \quad (6)$$

де: Q_{SPICE} – це оцінка якості ПС згідно засад SPICE (Software Process Improvement and Capability Determination); w_{SPICE_j} – вага підкритерію j для моделі оцінки якості ПС згідно з засадами SPICE; Q_{SPICE_j} – оцінка підкритерію j для моделі оцінки якості ПС згідно з SPICE; M_{SPICE} – підкритерії моделі згідно з SPICE.

$$Q_{\text{TOGAF}} = \sum_{j=1}^{M_{\text{TOGAF}}} \left(w_{\text{TOGAF}_j} \times Q_{\text{TOGAF}_j} \right) \quad (7)$$

де: Q_{TOGAF} – це оцінка якості ПС згідно засад TOGAF (The Open Group Architecture Framework); w_{TOGAF_j} – вага підкритерію j для моделі оцінки якості ПС згідно з засадами TOGAF; Q_{TOGAF_j} – оцінка підкритерію j для моделі оцінки якості ПС згідно з TOGAF; M_{TOGAF} – підкритерії моделі згідно з TOGAF.

Загальна оцінка якості розробленої комбінованої моделі оцінки якості ПС буде сумою оцінок кожної моделі, помноженої на її вагу та розраховуватиметься згідно (8):

$$\begin{aligned} Q = {} & w_{CMMI} \times Q_{CMMI} + w_{IEEE730} \times Q_{IEEE730} + w_{AgileTesting} \times Q_{AgileTesting} + w_{SixSigma} \times Q_{SixSigma} + \\ & + w_{TQM} \times Q_{TQM} + w_{SPICE} \times Q_{SPICE} + w_{TOGAF} \times Q_{TOGAF} \end{aligned}, \quad (8)$$

Комбінована оцінка якості $EQ_{комб.і.}$ розраховується шляхом врахування оцінки ефекстивності та впливу для кожного етапу розробки та використання ПС у відповідності до (9):

$$EQ_{комб.i} = w_e \times E_i + w_i \times I_i, \quad (9)$$

де: $EQ_{комб.i}$ – це комбінована оцінка якості із врахуванням оцінки ефекстивності та впливу для кожного етапу розробки та використання ПС; w_e та w_i – ваги ефективності та впливу відповідно, які визначаються експертно або на основі аналізу даних які надходять із спеціальних програмних застосунків із оцінювання якості ПС.; M_{TOGAF} – підкритерії моделі згідно з TOGAF.

В даному разі до спеціальних програмних застосунків із оцінювання якості ПС відносимо:

– Project Management Software (PM): Програми для управління проектами, такі як Microsoft Project, Jira, Asana, Trello тощо, можуть збирати дані про виконання завдань та календарний графік. Ці дані можуть бути використані для аналізу впливу різних методик на продуктивність та результативність проекту;

– Software Testing Tools: Інструменти для тестування програмного забезпечення, такі як Selenium, JUnit, TestRail тощо, зазвичай надають детальні дані про результати тестування, включаючи кількість та тип дефектів. Ці дані можуть бути використані для оцінки впливу методик тестування на якість продукту;

– Version Control Systems (VCS): Системи контролю версій, такі як Git, SVN, Mercurial тощо, зберігають історію змін коду. Ці дані можуть бути використані для аналізу ефективності розробки та виявлення зв'язку між використанням різних методик та якістю коду;

– Quality Management Software (QMS): Програми для управління якістю, такі як Qualityze, MasterControl, Sparta Systems тощо, можуть надавати інструменти для збору та аналізу даних про якість та впровадження різних методик та стандартів управління якістю;

– Business Intelligence Tools (BI): Інструменти бізнес-аналітики, такі як Tableau, Power BI, Google Data Studio тощо, можуть бути використані для аналізу даних з різних джерел та створення звітів та візуалізації результатів аналізу.

Для врахування інтеграції з процесом розробки, візуалізації результатів та стратегій управління якістю у нашій комбінованій моделі оцінки ефективності та впливу використання моделей якості застосуємо наступні підходи:

1. Модифікуємо нашу комбіновану оцінку $EQ_{комб.і.}$ додаючи член, що відображає наявність API яка застосовуватиметься для автоматизованої оцінки якості на різних етапах розробки ПС (10):

$$EQ_{комбі.} = \frac{1}{N}\sum_{i=1}^{N}\left(w_{адаптовані} \times M_i\right) \times p_j \times API_i, \quad (10)$$

де: API_i – параметр, який відображає наявність API для інтеграції моделі якості API_i i з процесом розробки; M_i – оцінка якості, отримана за допомогою моделі i; p_j – рівень застосовності моделі до проекту j; $w_{адаптовані}$ – адаптовані ваги кожної моделі (моделі, які були включені в комбіновану модель), які можуть змінюватися в залежності від контексту проекту; N – кількість проектів.

Відповідно в межах інтеграції з процесом розробки формула (10) дозволяє урахувати гнучкість, різноманітність методів оцінки та застосовність до різних проектів у комбінованій моделі оцінки якості програмного продукту.

2. Модифікуємо нашу комбіновану оцінку $EQ_{комб.і.}$ додаючи параметр, який відображає рівень візуалізації результатів оцінки якості для моделі (10):

$$EQ_{комбі.} = \frac{1}{N}\sum_{i=1}^{N}\left(w_{адаптовані} \times M_i\right) \times p_j \times T_{Віз.ij}, \quad (11)$$

де: $T_{Віз.ij}$ – параметр, який відображає рівень візуалізації результатів оцінки якості для моделі.

3. Включемо в комбіновану оцінку $EQ_{комб.і.}$ врахування застосування стратегії управління якістю на основі засад включення коефіцієнту стратегії управління якістю ($T_{Стр.ij}$) для проекту та коефіцієнта візуалізації результатів (віз) для і-ї метрики на проекті можуть бути сформовані з урахуванням різних факторів (12):

$$EQ_{комбі.} = \frac{1}{N}\sum_{i=1}^{N}\left(w_{адаптовані} \times M_i \times Q_{ij}\right) \times p_j \times API_i \times T_{Віз.ij} \times T_{Стр.ij}, \quad (12)$$

де: $T_{Стр.ij}$ – коефіцієнт стратегії управління якістю.

Коефіцієнт стратегії управління якістю може бути сформований на основі особливостей та вимог конкретного проекту [11]. Наприклад, якщо проект вимагає великої уваги до якості та неперервного моніторингу, коефіцієнт може бути вищим. Це може бути числове значення, що відображає рівень важливості управління якістю для проекту. Згідно [14] при оцінки ризиків коефіцієнт візуалізації результатів для конкретної метрики може бути сформований на основі оцінки ризиків. Згідно [4,6] якщо певна метрика відіграє критичну роль у проекті та має великий вплив на його успішність, коефіцієнт візуалізації може бути вищим. Згідно [11] залучення експертів у галузі керування якістю та візуалізації результатів може допомогти сформувати адекватні коефіцієнти. Експерти можуть враховувати різні аспекти проекту та метрик якості для визначення оптимальних значень цих коефіцієнтів. Отже, коефіцієнт стратегії управління якістю та коефіцієнт візуалізації результатів формуються на основі аналізу із застосуванням метрик якості. Застосування вказаних метрик якості в комбінованій моделі може бути обгрунтоване математично з точки зору досягнення об'єктивної та комплексної оцінки якості програмного продукту. Комбінування різних метрик якості дозволяє забезпечити об'єктивність оцінки, оскільки кожна метрика вимірює конкретний аспект якості. Математично це виражається через розрахунок середнього значення або вагованого середнього значення метрик для отримання загального показника якості. Використання різних метрик дозволяє оцінити якість програмного продукту з різних точок зору, що забезпечує комплексність оцінки. Математично це виражається через включення різних метрик у формулу для комбінованої оцінки якості. Загалом, подібна комбінована модель допомагає забезпечити високу якість програмного продукту, оптимізувати процеси розробки та управління якістю, а також забезпечити відповідність стандартам та вимогам клієнтів.

Висновки

1. Кожна модель якості має свої унікальні особливості, які можуть бути корисними на різних етапах розробки та використання програмних систем. Відповідно вибір моделі якості повинен залежати від конкретних потреб та характеристик проекту. Інтеграція моделей якості з процесами розробки може допомогти забезпечити більш ефективну та якісну розробку програмного забезпечення. Хоч дослідження в області застосування моделей якості в розробці та використанні програмних систем відбувається постійно, і багато проблем вже мають вирішення, проте також існують ті, які залишаються відкритими.

2. Математично застосування метрик якості в комбінованій моделі може бути обґрунтоване через їхню спроможність забезпечити об'єктивність, комплексність оцінки, врахування специфіки проекту та виявлення недоліків та проблем програмного продукту.

3. Запропонована комбінована модель дозволяє використовувати найкращі практики та методики з кожної з вказаних областей для забезпечення високої якості програмного продукту на кожному етапі його життєвого циклу.

Список літератури

[1] H. Foidl and M. Felderer, "Integrating software quality models into risk-based testing," *Software Quality Journal*, vol. 26, pp. 809–847, 2018.

[2] K. Sahu and R. K. Srivastava, "Predicting software bugs of newly and large datasets through a unified neuro-fuzzy approach: Reliability perspective," *Advances in Mathematics: Scientific Journal*, vol. 10, no. 1, pp. 543–555, 2021.

[3] K. Sahu, F. A. Alzahrani, R. K. Srivastava, and R. Kumar, "Evaluating the impact of prediction techniques: Software reliability perspective," *Computers, Materials & Continua*, vol. 67, no. 2, pp. 1471–1488, 2021.

[4] S. Sackey, D. E. Lee, and B. S. Kim, "Duration Estimate at Completion: Improving Earned Value Management Forecasting Accuracy," *KSCE Journal of Civil Engineering*, vol. 24, no. 3, pp. 693–702, 2020. DOI: 10.1007/s12205-020-0407-5

[5] P. Sharma and A. L. Sangal, "Building and Testing a Fuzzy Linguistic Assessment Framework for Defect Prediction in ASD Environment Using Process-Based Software Metrics," *Arabian Journal of Science and Engineering*, vol. 45, no. 12, pp. 10327–10351, 2020.

[6] Y. Hassouneh et al., "Boosted Whale Optimization Algorithm With Natural Selection Operators for Software Fault Prediction," *IEEE Access*, vol. 9, pp. 14239–14258, 2021. DOI: 10.1109/ACCESS.2021.3052149.

[7] R. Al-Qutaish, "Quality Models in Software Engineering Literature: An Analytical and Comparative Study," *Journal of American Science*, vol. 6, pp. 10, 2010.

[8] J. Estdale and E. Georgiadou, "Applying the ISO/IEC 25010 Quality Models to Software Product," in *Systems, Software and Services Process Improvement. EuroSPI 2018*, X. Larrucea, I. Santamaria, R. O'Connor, and R. Messnarz, Eds., vol. 896, 2018, pp. 12.

[9] O. Fonseca-Herrera, A. E. Rojas, and H. Florez, "A Model of an Information Security Management System Based on NTC-ISO/IEC 27001 Standard," *IAENG International Journal of Computer Science*, vol. 48, pp. 213, 2021.

[10] M. N. Aziz, I. M. Sapta, and S. Rochimah, "Security Characteristic Evaluation Based on ISO/IEC 25023 Quality Model, Case Study: Laboratory Management Information System," in *2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), IEEE*, pp. 332–336.

[11] J.-X. Chen, "Overall performance evaluation: new bounded DEA models against unreachability of efficiency," *The Journal of the Operational Research Society*, vol. 65, no. 7, pp. 1120–1132, 2014.

[12] M. Filz, C. Herrmann, and S. Thiede, "Simulation-based Assessment of Quality Inspection Strategies on Manufacturing Systems," *Procedia CIRP*, vol. 93, pp. 777–782, 2020.

[13] A. Golabchi, S. Han, and S. AbouRizk, "A simulation and visualization-based framework of labor efficiency and safety analysis for prevention through design and planning," *Automation in Construction*, vol. 96, pp. 310–323, 2018.

[14] P. Han, L. Wang, and P. Song, "Doubly robust and locally efficient estimation with missing outcomes," *Statistica Sinica*, vol. 26, no. 2, pp. 691–719, 2016.

[15] L. Hund, B. Schroeder, K. Rumsey, and G. Huerta, "Distinguishing between model- and data-driven inferences for high reliability statistical predictions," *Reliability Engineering and System Safety*, vol. 180, pp. 201–210, 2018.

Стаття надійшла: 21.03.2024

References

[1] H. Foidl and M. Felderer, "Integrating software quality models into risk-based testing," *Software Quality Journal*, vol. 26, pp. 809–847, 2018.

[2] K. Sahu and R. K. Srivastava, "Predicting software bugs of newly and large datasets through a unified neuro-fuzzy approach: Reliability perspective," *Advances in Mathematics: Scientific Journal*, vol. 10, no. 1, pp. 543–555, 2021.

[3] K. Sahu, F. A. Alzahrani, R. K. Srivastava, and R. Kumar, "Evaluating the impact of prediction techniques: Software reliability perspective," *Computers, Materials & Continua*, vol. 67, no. 2, pp. 1471–1488, 2021.

[4] S. Sackey, D. E. Lee, and B. S. Kim, "Duration Estimate at Completion: Improving Earned Value Management Forecasting Accuracy," *KSCE Journal of Civil Engineering*, vol. 24, no. 3, pp. 693–702, 2020. DOI: 10.1007/s12205-020-0407-5

[5] P. Sharma and A. L. Sangal, "Building and Testing a Fuzzy Linguistic Assessment Framework for Defect Prediction in ASD Environment Using Process-Based Software Metrics," *Arabian Journal of Science and Engineering*, vol. 45, no. 12, pp. 10327–10351, 2020.

[6] Y. Hassouneh et al., "Boosted Whale Optimization Algorithm With Natural Selection Operators for Software Fault Prediction," *IEEE Access*, vol. 9, pp. 14239–14258, 2021. DOI: 10.1109/ACCESS.2021.3052149.

[7] R. Al-Qutaish, "Quality Models in Software Engineering Literature: An Analytical and Comparative Study," *Journal of American Science*, vol. 6, pp. 10, 2010.

[8] J. Estdale and E. Georgiadou, "Applying the ISO/IEC 25010 Quality Models to Software Product," in *Systems, Software and Services Process Improvement. EuroSPI 2018*, X. Larrucea, I. Santamaria, R. O'Connor, and R. Messnarz, Eds., vol. 896, 2018, pp. 12.

[9] O. Fonseca-Herrera, A. E. Rojas, and H. Florez, "A Model of an Information Security Management System Based on NTC-ISO/IEC 27001 Standard," *IAENG International Journal of Computer Science*, vol. 48, pp. 213, 2021.

[10] M. N. Aziz, I. M. Sapta, and S. Rochimah, "Security Characteristic Evaluation Based on ISO/IEC 25023 Quality Model, Case Study: Laboratory Management Information System," in *2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), IEEE*, pp. 332–336.

[11] J.-X. Chen, "Overall performance evaluation: new bounded DEA models against unreachability of efficiency," *The Journal of the Operational Research Society*, vol. 65, no. 7, pp. 1120–1132, 2014.

[12] M. Filz, C. Herrmann, and S. Thiede, "Simulation-based Assessment of Quality Inspection Strategies on Manufacturing Systems," *Procedia CIRP*, vol. 93, pp. 777–782, 2020.

[13] A. Golabchi, S. Han, and S. AbouRizk, "A simulation and visualization-based framework of labor efficiency and safety analysis for prevention through design and planning," *Automation in Construction*, vol. 96, pp. 310–323, 2018.

[14] P. Han, L. Wang, and P. Song, "Doubly robust and locally efficient estimation with missing outcomes," *Statistica Sinica*, vol. 26, no. 2, pp. 691–719, 2016.

[15] L. Hund, B. Schroeder, K. Rumsey, and G. Huerta, "Distinguishing between model- and data-driven inferences for high reliability statistical predictions," *Reliability Engineering and System Safety*, vol. 180, pp. 201–210, 2018.

Відомості про авторів

Шантир Антон Сергійович – кандидат технічних наук, доцент кафедри Штучного інтелекту ДУІКТ

Shantyr Anton – candidate of technical sciences, associate professor of the Department of artificial intelligence SUICT

A. S. Shantyr

SPECIFICS OF QUALITY ASSESSMENT MODELS APPLICATION AT DEVELOPMENT AND USE STAGES OF SOFTWARE SYSTEMS

State University of Information and Communication Technologies, Kyiv