Issue 15, 2024

https://doi.org/10.23939/sisn2024.15.115

УДК 004.422.4:005.3:004.4

ПРИНЦИПИ СТВОРЕННЯ БАГАТОЦІЛЬОВИХ МОДЕЛЕЙ ЯКОСТІ ПРОГРАМНИХ СИСТЕМ

Антон Шантир

Державний університет інформаційно-комунікаційних технологій,
кафедра штучного інтелекту, Київ, Україна ,
E-mail: anton.shantyr@gmail.com, ORCID: 0000-0002-0466-3659

Запропоновано на математичному рівні оригінальний підхід до створення багатоцільових моделей якості програмних систем (ПС). Основа дослідження – вивчення та узагальнення тенденцій моделювання якості ПС та потреб користувачів з метою визначення оптимальних принципів побудови таких моделей. У статті подано математичні пояснення, які відіграють ключову роль у виявленні та формалізації принципів створення багатоцільових моделей якості програмного забезпечення (ПЗ). Важливим аспектом є розгляд принципів побудови моделей якості на математичному рівні, що уможливлює точніші оцінювання та аналіз різних аспектів якості ПЗ. Результати дослідження вказують на те, що врахування математичних принципів під час створення багатоцільових моделей якості ПС може забезпечити вагомий практичний внесок. Встановлено, що на практичному рівні розроблення багатоцільових моделей якості ПС врахування принципів створення багатоцільових моделей якості ПС може мати доволі численні практичні наслідки. Зокрема, застосування метрик у встановлених принципах дає змогу забезпечити комплексний погляд на якість ПС та визначити сфери, які потребують уваги та покращень. Це допомагає розробникам та інженерам із управління якістю ПЗ приймати обґрунтовані рішення щодо поліпшення та оптимізації систем. Дослідження показало, що створення високоякісних моделей якості потребує уваги до різних аспектів, від потреб користувачів до тестування та постійного вдосконалення, а також використання математичних методів для їх формалізації та аналізу. Розроблені принципи створення багатоцільових моделей якості на рівні математичних моделей дають змогу використовувати ці моделі для оцінювання та аналізування різних аспектів якості ПС, представляючи кожну модель за допомогою відповідної функції, яка визначає залежність між метрикою якості та самою якістю ПС. Очікують, що подальший розвиток та впровадження цих принципів сприятимуть покращенню процесів розроблення програмних продуктів та забезпечить високу якість результуючого ПЗ.

Ключові слова: програмне забезпечення; метрики якості; потреби користувачів; інформаційні технології; методологія Agile; підтримка системи; стандарти якості; математичний апарат.

Вступ

За останні десятиліття істотно зросла важливість ПС у всіх сферах суспільства. Вони стали невід'ємною частиною нашого повсякденного життя, їх використовують у банківській справі, медицині, автомобільній промисловості, електронній комерції та безлічі інших галузей. Це підвищує вимоги до якості ПЗ, оскільки недоліки чи помилки можуть мати серйозні наслідки, зокрема фінансові втрати, порушення безпеки даних та навіть загрози для життя і здоров'я.

Щоб забезпечити високу якість ПС, розробники та інженери використовують різноманітні методи та підходи. Одним із ключових інструментів у цьому процесі є багатоцільові моделі якості [2].

Ці моделі дають змогу оцінити різні аспекти якості програмного забезпечення, такі як функціональність, ефективність, надійність та інші, забезпечуючи комплексний погляд на продукт і допомагаючи визначити його потенційні слабкі місця [3].

Проте створення багатоцільових моделей якості не є простим завданням. Воно потребує уважного аналізу вимог користувачів, ретельного вивчення характеристик системи та врахування різноманітних факторів, що впливають на її якість [4]. Крім того, важливо забезпечити, щоб моделі були гнучкими та адаптованими до змін у вимогах до програмного забезпечення та умов його застосування.

В умовах постійного розвитку інформаційних технологій та зростання складності ПС проблема забезпечення їх якості набуває все більшої актуальності. Відповідно [5] якість ПЗ є одним із ключових факторів успіху в IT-індустрії, оскільки безпосередньо впливає на задоволення користувачів, вартість розроблення та підтримки системи, а також на репутацію компанії-розробника.

Загалом упровадження принципів якості на практиці допомагає розробникам підвищити ефективність та результативність процесу розроблення ПС, а також забезпечити створення високоякісних та відповідальних продуктів.

Постановка проблеми

З метою ефективного управління якістю ПС та вирішення їхніх багатогранних завдань, удосконалення процесів розроблення та підтримки необхідно створити багатоцільові моделі якості. Ці моделі дають змогу враховувати різноманітні аспекти якості, такі як функціональність, надійність, продуктивність, безпека та інші, забезпечуючи комплексний підхід до оцінювання та управління якістю програмних продуктів.

Формулювання проблеми дослідження принципів створення багатоцільових моделей якості ПС для їх ефективного застосування у практиці розроблення ПЗ охоплює кілька ключових аспектів. По-перше, існує потреба в розробленні універсальніших та адаптивніших моделей, здатних враховувати різноманітні вимоги та типи програмного забезпечення. Багато моделей якості можуть бути спрямовані на вирішення конкретних завдань, або областей програмного забезпечення. Проте необхідно дослідження принципів, які дадуть змогу створювати універсальніші моделі, здатні враховувати різноманітні потреби різних типів ПЗ.

Крім того, важливо вирішити суперечності між різними критеріями якості та забезпечити баланс між ними. У багатоцільових моделях якості можуть виникати суперечності між різними аспектами якості, такими як продуктивність, надійність, зручність використання тощо. Дослідження механізмів врегулювання таких конфліктів та забезпечення балансу між різними критеріями є ключовим аспектом.

Не менш важливі розуміння різноманітності вимог різних груп користувачів та адаптація моделей до них. Необхідні також відповідні метрики та методики оцінювання якості, а також інтеграція моделей із процесом розроблення ПЗ для забезпечення постійного покращення продукту.

Очікування користувачів щодо якості програмного забезпечення можуть відрізнятися. Деякі підкреслюють продуктивність, тоді як інші зацікавлені у зручності використання або безпеці. Відповідно [10] дослідження принципів, які дозволять адаптувати моделі якості до різних вимог різних груп користувачів, є важливим завданням.

Ефективне застосування моделей якості вимагає відповідних метрик та методик оцінки. Дослідження оптимальних підходів до вимірювання різних аспектів якості та їхнього відображення у багатоцільових моделях може стати ключовим завданням.

Також важливою проблемою є інтеграція моделей якості з процесом розроблення програмного забезпечення. Дослідження та розроблення методів, що дають змогу враховувати моделі якості

на кожному етапі життєвого циклу розроблення програмного забезпечення, допомагають забезпечити постійне покращення якості продукту.

Вирішення цих проблем може сприяти створенню ефективніших та універсальніших моделей якості програмного забезпечення, що сприятиме покращенню процесів розроблення та забезпечить задоволення різноманітних потреб користувачів.

Аналіз останніх досліджень та публікацій

Під час моделювання принципів створення багатоцільових моделей якості ПС може виникнути декілька ситуацій, які потребують доповнення чи вдосконалення. Аналізуючи ці проблеми, варто відзначити проблему недостатності вхідних даних: якщо вхідні дані для моделювання не є достатньо повними або точними, це може призвести до недостовірних результатів [1]. Рішенням може бути розроблення додаткових механізмів для збирання та аналізування даних або використання додаткових джерел інформації.

Також варто звернути увагу на проблему неврахування певних критеріїв якості: Якщо деякі аспекти якості ПС не враховані у початковій моделі, може знадобитися розширення моделі для охоплення цих критеріїв. Це може потребувати аналізу додаткових вимог або експертної оцінки важливості нових аспектів.

В праці [2] висвітлено проблему неефективності методів аналізу та оцінювання: якщо методи, використані для аналізу та оцінки якості, не є достатньо ефективними або точними, може знадобитися розроблення нових методів або використання альтернативних підходів. Це може передбачати вдосконалення алгоритмів, розроблення нових моделей або використання новітніх технологій.

У праці [3] вказано на потребу принципово-стратегічного врахування недоліків у процесах управління якістю: якщо наявні процеси управління якістю не дають достатнього результату, іноді необхідно переглянути та оптимізувати ці процеси. Це може передбачати впровадження нових стратегій, стандартів або інструментів управління якістю.

На загальному рівні вирішити зазначені проблеми можна за допомогою взаємодії між експертами з різних сфер, здійснення аналізу та експериментів, а також упровадження нових методів та практик у розроблення ПС. Додатковою стратегією може бути постійний моніторинг і оцінка якості системи для виявлення та вирішення проблем у реальному часі.

Аналізуючи останні дослідження в галузі моделювання принципів створення багатоцільових моделей якості ПС, потрібно систематизувати їх, а саме:

- на дослідження щодо використання штучного інтелекту: в праці [4] доволі детально розглянуто можливості застосування методів машинного навчання та глибинного навчання для автоматизації процесів аналізу та моделювання якості ПС;
- на дослідження, спрямовані на розгляд та аналіз особливостей інтеграції даних та аналізу великих обсягів інформації: в праці [5] описано дослідження, спрямовані на розроблення алгоритмів та методів для аналізу великих обсягів даних, що дає змогу отримувати точніші та розгорнутіші моделі якості;
- на дослідження, націлені на удосконалення методів тестування та верифікації: наприклад, у праці [6] розглянуто дослідження, спрямовані на розроблення нових методів тестування на основі властивостей та формальної верифікації, що дає змогу виявляти та виправляти помилки в програмному коді ефективніше;
- на дослідження особливостей використання аналізу даних для управління якістю: наприклад, у праці [7] висвітлено дослідження у галузі аналітики даних та бізнес-інтелекту, спрямовані на розроблення інструментів та методів для моніторингу та управління якістю ПС на основі зібраних даних;

– на дослідження специфіки розроблення інтегрованих підходів до управління якістю. Дослідження в цьому напрямі [8] передбачають створення комплексних методологій та інструментів, які об'єднують різні аспекти управління якістю та допомагають ефективно вирішувати завдання забезпечення якості ПС.

Узагальнюючи зазначене вище, варто зауважити, що всі вказані дослідження виконують з метою розроблення нових технологій, методів та інструментів для покращення процесів створення та управління якістю ПС згідно із сучасними вимогами та викликами ринку програмного забезпечення.

Моделюючи принципи створення багатоцільових моделей якості ПС, важливо вирішити низку теоретично-практичних питань. Зокрема, в праці [9] відзначено потребу вирішення питання: "Які критерії якості ПС потрібно враховувати?". Вирішення цього питання може охоплювати розгляд особливостей функціональності ПС [10], продуктивності ПС [11], аналізу механізмів забезпечення безпеки ПС [12], принципово-узагальненого аналізу тенденцій надійності ПС [13], аналізу зручності використання ПС [14] та інші аспекти. В праці [15] на узагальненому рівні порушено питання: "Які метрики будуть використовуватися для кожного критерію якості?". Наприклад, для продуктивності це може бути час відгуку, для безпеки – кількість вразливостей, для надійності – середній час безвідмовної роботи (MTBF) та інші. В праці [16] розглянуто питання: "Які методи та алгоритми будуть використовуватися для оцінки якості?". На практичному рівні в межах вирішення цього питання виникає потреба в чіткому обґрунтуванні щодо вибору застосування методів: тестування, аналізу даних, статистичного або машинного навчання, моделювання та імітації [17].

У праці [18] розглянуто питання: "Які вхідні дані потрібні для моделювання?" Згідно із результатами, які отримали зазначені вище дослідники, врахування різних параметрів системи, середовища розгортання, характеристик користувачів та інших факторів, що впливають на якість [19].

У праці [20] порушено питання: "Які рішення можна приймати на основі результатів моделювання?". З цього приводу дослідники згаданої вище праці вказують, що можна приймати рішення про виправлення помилок, оптимізацію ресурсів, вдосконалення архітектури, вибір стратегій тестування тощо.

У праці [21] порушено питання: "Які методи оптимізації можна застосувати для поліпшення якості системи?". На практичному рівні вирішення цього питання може передбачати оптимізацію процесів розроблення, управління ризиками, вдосконалення архітектури та використання новітніх технологій.

У праці [22] відзначено, що є нагальна потреба у вирішенні питання: "Які стратегії контролю та управління якістю можуть бути використані?". Вирішення цього питання потребує формування на принциповому рівні визначення стандартів якості, а на практичному рівні впровадження процесів забезпечення якості, вдосконалення комунікації та співпраці між командами розробників та тестувальників ПС.

Варто зазначити, що врахування та вирішення усіх зазначених питань допомагає визначити на принциповому рівні підходи та методи, які можна застосувати для створення багатоцільових моделей якості ПС, а також розробити стратегії для управління ними та їх оптимізації. Відповідно до праць [7–9] окреслення принципу використання метрик якості ПС зводиться до визначення конкретних метрик якості, таких як час реакції системи, швидкість виконання, чи рівень безпеки, що дасть змогу визначити та відстежувати рівень якості програмної системи протягом її розроблення та після випуску в експлуатацію. Згідно з [7–11] використання метрик якості є важливою складовою процесу розроблення ПС і може відбуватися так:

– упровадження моніторингу: розробники повинні налаштувати систему моніторингу, яка відстежуватиме ці метрики під час розроблення та після випуску в експлуатацію [10]. Це

може передбачати використання спеціальних інструментів для моніторингу продуктивності, безпеки, якості коду тощо;

- аналіз результатів: отримані дані метрик якості повинна регулярно аналізувати команда розробників та оцінювати, чи відповідає ПС вимогам якості [9]. Цей аналіз може відбуватися на засіданнях спринтів (у випадку Agile) або на інших регулярних зустрічах;
- вживання заходів для вдосконалення ПС: на основі результатів аналізу команда може вжити відповідних заходів для вдосконалення якості програмної системи [8]. Це може охоплювати оптимізацію коду, виправлення потенційних проблем безпеки, вдосконалення процесу розроблення тощо [7, 11]. Варто зауважити, що розглянутий вище цикл моніторингу, аналізу та вдосконалення допомагає команді розробників ПС забезпечити високий рівень якості програмної системи упродовж всього процесу розроблення та після випуску в експлуатацію.

Згідно з [3] окреслення практичного застосування принципу тестування та верифікації зводиться до впровадження систем автоматизованого тестування, що допоможе забезпечити відповідність програмних моделей встановленим стандартам та вимогам користувачів. Це охоплює функціональне тестування, тестування на витривалість, тестування безпеки тощо.

Відповідно до [4] практична реалізація засад принципу орієнтації на користувача потребує дослідження користувачів, опитування яких та аналіз їхніх потреб допоможе розробникам створити моделі, які найкраще задовольнятимуть їхні очікування та потреби. Згідно з [5] принцип "постійного вдосконалення та участі зацікавлених сторін" на практиці передбачає використання Agile методологій, що дає змогу розробникам ПС забезпечити постійне вдосконалення системи за допомогою ітеративного процесу розроблення та залучення замовника та інших зацікавлених сторін до процесу розроблення.

Згідно з [11] практична реалізація принципу "Модульність та повторне використання" зводиться до використання архітектурних шаблонів, які сприяють модульності та повторному використанню компонентів програмної системи, дозволить створити гнучкіші та розширювані системи, які легше підтримувати та розвивати із часом.

Відповідно до [1] в межах окреслення практичного застосування принципу систематичного підходу до розроблення: використання конкретних методологій розробки, таких як Agile, Scrum, або Waterfall, допоможе створити структурований підхід до розроблення системи, що забезпечить відповідність вимогам якості.

Кожна із зазначених вище методологій має певні особливості й підходи до управління процесом розроблення. Зокрема методологія Waterfall – лінійний підхід, де кожна фаза розроблення (вимоги, проєктування, реалізація, тестування, впровадження) виконується послідовно [2–3]. Цей підхід підходить для проєктів з чітко визначеними вимогами до ПС, де потрібна висока передбачуваність і строгий контроль змін [4]. Методологія Agile – ітеративний підхід, що акцентує на гнучкості під час розроблення ПС у межах співпраці з замовником та здатності розробників ПС швидко реагувати на зміни [5]. У цьому разі команди розробників ПС працюють над невеликими ітераціями, відомими як "спринти", і регулярно змінюють пріоритети щодо оцінки якості ПС згідно з вимогами бізнесу. В праці [17] методологію Scrum розглянуто як один з фреймворків Agile, який надає конкретні ролі (Product Owner, Scrum Master, Team), зустрічі (Daily Standups, Sprint Planning, Sprint Review, Sprint Retrospective) та артефакти (Product Backlog, Sprint Backlog, Increment), щоб допомогти команді організувати і виконати роботу щодо розроблення ПС.

Вибираючи між цими методологіями, розробники ПС повинні враховувати особливості проєкту, потреби замовника та умови роботи. Використання будь-якої з цих методологій може сприяти створенню структурованого підходу до розроблення програмної системи, що відповідає вимогам якості.

Однак важливо також зазначити, що успіх проєкту залежить не тільки від вибраної методології, але й від того, як її впроваджує та адаптує команда розробників.

Формулювання цілі статті

Мета статті – дослідити принципи створення багатоцільових моделей якості ПС для подальшого їх ефективного застосування у практиці розроблення програмного забезпечення.

Реалізація поставленої мети передбачає досягнення таких цілей:

1. Проаналізувати взаємозв'язок принципів створення багатоцільових моделей якості ПС у межах окреслення узагальнених тенденцій моделювання та користувацьких потреб.
2. Надати математичні пояснення принципів створення багатоцільових моделей якості ПС.
3. Розробити принципи створення багатоцільових моделей якості на рівні математичних моделей, що дають змогу використовувати математичні моделі для оцінки та аналізу різних аспектів якості ПС.

Виклад основного матеріалу

Принципи створення багатоцільових моделей якості ПС є результатом комплексного аналізу вимог до якості програмного забезпечення та практичного досвіду в розробленні програм. Із урахуванням викладеного вище в табл. 1 наведено результати аналізу взаємозв'язку принципів створення багатоцільових моделей якості ПС в межах окреслення узагальнених тенденцій моделювання та користувацьких потреб.

Таблиця 1

Результати аналізу взаємозв'язку принципів створення багатоцільових моделей якості ПС в межах окреслення узагальнених тенденцій моделювання та користувацьких потреб

Принципи	Опис	Взаємозв'язок	Приклади та деталі
Інкапсуляція та когерентність	Приховування деталей реалізації модулів та надання інтерфейсів для взаємодії Модулі повинні бути логічно зв'язані та виконувати одну або декілька схожих функцій	Інкапсуляція сприяє зменшенню залежності між модулями та полегшує їх використання Когерентність полегшить розуміння та підтримку системи	Використання публічних та приватних методів для доступу до функціоналу модуля. Модуль, відповідальний за опрацювання користувацької аутентифікації, пов'язаний з модулем, відповідальним за управління даними користувачів
Користувацькі потреби та методології розроблення	Визначення функціональних та нефункціональних вимог до системи з орієнтацією на потреби користувачів. Використання певних методологій розроблення (Agile, Scrum, DevOps), що підтримують інтеграцію тестування та забезпечення якості	Вимоги користувачів є основою для визначення критеріїв якості. Забезпечення якості вбудоване у процес розробки	Визначення ефективності, зручності та безпеки для користувачів. Використання тест-драйвен розробки для забезпечення відповідності вимогам

Продовження табл. 1

Принципи	Опис	Взаємозв'язок	Приклади та деталі
Метрики якості та модульність	Визначення метрик для вимірювання якості ПЗ та моніторингу її покращень. Розділення програмного продукту на невеликі, незалежні модулі	Використання метрик для оцінки досягнення поставлених цілей з якості. Модульність взаємодіє з принципом простоти та повторного використання, сприяючи зменшенню складності та повторному використанню	Метрики швидкості відгуку, частоти виникнення помилок тощо. Просте додавання нового функціоналу під час роботи з окремими модулями
Незалежність	Модулі мають бути незалежними один від одного	Незалежність допомагає забезпечити гнучкість та розширюваність системи	Модулі, відповідальні за різні аспекти функціональності, мають мінімальну взаємодію між собою
Постійне вдосконалення та простота використання ПС	Упровадження циклу постійного вдосконалення для поліпшення якості ПЗ в процесі його життєвого циклу. Система повинна бути якомога простішою	Використання знань та досвіду для постійного покращення якості. Простота сприяє розумінню, підтримці та вдосконаленню системи	Застосування PDCA циклу (Plan-Do-Check-Act) для постійного вдосконалення. Відсутність зайвого функціоналу та зайвих залежностей допомагає зробити систему простішою та зрозумілішою
Повторне використання та розширюваність	Модулі повинні бути придатними для використання в інших проєктах Система повинна легко розширюватися для введення нового функціоналу	Повторне використання дає змогу ефективно застосовувати розроблені компоненти Розширюваність забезпечує можливість адаптації системи до змін у вимогах	Бібліотека функцій, яку можна використовувати у різних проєктах, без необхідності переписування коду. Використання розширюваних API або розширюваних механізмів для додавання нового функціоналу без потреби модифікації вже створеного коду
Стандарти якості та тестування та верифікація, а також участь зацікавлених сторін	Урахування відповідних стандартів якості ПЗ (наприклад, ISO 25010) та рекомендаційних документів. Тестування для перевірки відповідності вимогам якості та виявлення дефектів. Залучення зацікавлених сторін до формування вимог та оцінювання якості продукту	Визначення критеріїв якості відповідно до встановлених стандартів. Перевірка та підтвердження відповідності ПЗ вимогам. Забезпечення відповідності продукту очікуванням користувачів та інших зацікавлених сторін	Використання ISO 25010 для визначення атрибутів якості. Автоматизоване тестування, ручне тестування, модульне тестування тощо. Проведення сеансів браундштормінгу та спільного аналізу вимог

З табл. 1 наочно видно, що проведений аналіз дозволив виявити, що успішна реалізація якості ПС вимагає комплексного підходу, яка базується на різноманітних принципах. Ці принципи охоплюють всі етапи життєвого циклу розробки програмного продукту, від визначення вимог та

планування до постійного вдосконалення та взаємодії з зацікавленими сторонами. Зокрема інкапсуляція, модульність та незалежність визначають архітектурні принципи, що сприяють створенню гнучких, легко зрозумілих та розширюваних систем. Крім того, вони взаємодіють з іншими принципами, такими, як простота та повторне використання, що підтримують загальний функціональний розвиток та ефективне використання ресурсів. Підтримка користувацьких потреб є основою для визначення критеріїв якості, використання методологій розробки, вимірювання метрик якості та здійснення тестування та верифікації. Це дозволяє забезпечити високий рівень задоволення користувачів та відповідати їх очікуванням. Нарешті, постійне вдосконалення є ключовим принципом, який забезпечує адаптацію системи до змінних умов та вимог ринку. Цей принцип доповнюється участю зацікавлених сторін, що забезпечує зворотний зв'язок та підтримує постійне покращення системи з урахуванням потреб користувачів та стейкхолдерів.

В табл. 2 наведено основні принципи створення багатоцільових моделей якості ПС разом з їх поясненням на математичному рівні, де кожен принцип описується в контексті його значення для створення високоякісних моделей, а пояснення на математичному рівні намагається формалізувати цей принцип за допомогою математичних термінів або концепцій.

Таблиця 2

Математичні пояснення основних принципів створення багатоцільових моделей якості ПС

Принцип	Пояснення на математичному рівні
Інкапсуляція	Інкапсуляція дозволяє обмежити доступ до деяких компонентів системи і забезпечує їх взаємодію через інтерфейси [13]. Математично це можна представити як об'єкт, який має методи доступу до своїх внутрішніх даних, але не дозволяє безпосередньо змінювати ці дані з зовні: $$Q = \{F, M_d, M_m\},$$ де Q – об'єкт; F – поля (внутрішні дані) об'єкта; M_d – методи доступу; M_m – методи модифікації
Когерентність	Когерентність означає, що компоненти системи повинні працювати разом логічним та послідовним способом [16]. Математично це виражається через взаємовідношення між різними частинами системи та їх взаємодію для досягнення спільної мети [13]: $$K = \sum_{i=1}^{n} \sum_{j=1}^{m} R_{ij} \cdot I_{ij},$$ де K – когерентність; n – кількість компонентів системи; m – кількість можливих взаємодій між компонентами; R_{ij} – рівень взаємовідношення між компонентами i – та j.
Користувацькі потреби	Під час створення багатоцільових моделей якості важливо враховувати потреби користувачів системи. Їх можна виразити за допомогою математичних функцій, які описують очікувані характеристики та вимоги користувачів. Доцільно застосовувати модель мінімізації функції втрат, де функція втрат визначається як різниця між поточним станом програми та станом, який відповідає потребам користувачів [12]: $$F_K = \sum_{i=1}^{n} (P_i - S_i),$$ де F_K – користувацькі потреби; n – кількість ураховуваних характеристик; P_i – очікувана характеристика користувача для i-го аспекту; S_i – поточний стан програми для i-го аспекту
Методології розробки	Методології розроблення надають систематичний підхід до процесу розроблення, що ґрунтується на певних принципах та правилах [17]. Математично це можна подати як алгоритм або набір інструкцій, які описують послідовність дій для досягнення певної мети [16]:

Продовження табл. 2

Принцип	Пояснення на математичному рівні
	$M_{розробки} = \{P, R, I\},$ де $M_{розробки}$ – методологія розроблення; P – множина принципів, які визначають основні концепції та уявлення про те, як повинен бути структурований процес розробки ПС; R – множина правил, які накладають конкретні вимоги та обмеження на виконання кожного етапу розроблення ПС; I – набір інструкцій або алгоритмів, які вказують послідовність дій для виконання кожного етапу розроблення ПС
Метрики якості	Метрики якості використовують для вимірювання рівня якості системи. Це можуть бути числові значення, які виражають рівень відповідності системи певним стандартам або вимогам користувачів [14]. У цьому разі доцільне використання моделі оптимізації, де обмеження визначаються стандартами якості, а цільова функція максимізує відповідність цим стандартам: $\max_{x_1, x_2, \ldots, x_n} F(x_1, x_2, \ldots x_n),$ з обмеженнями: $g_j(x_1, x_2, \ldots x_n) \le 0,$ для $g = 1,2,\ldots, p,$ де $F(x_1, x_2, \ldots x_n)$ – цільова функція, що відображає загальну якість ПС на основі метрик якості; $x_1, x_2, \ldots x_n$ – параметри системи, які можна налаштовувати для покращення якості; $g_j(x_1, x_2, \ldots x_n)$ – обмеження, що відповідають вимогам стандартів якості
Модульність	Модульність означає розділення системи на незалежні модулі, які можна розробляти та тестувати окремо [12]. Це можна виразити математично через розділення функцій системи на окремі компоненти та їх взаємодію. Нехай M це ПС, а $M_1, M_2, \ldots, M_n$ – окремі модулі, які утворюють систему M. Тоді можемо подати модульність як розділення функцій системи на окремі компоненти та їх взаємодію у вигляді: $M = \bigcup_{i=1}^{n} M_i,$ де M – система загалом; M_i – множина окремих модулів, із яких складається система; $\cup$ – оператор об'єднання, що об'єднує всі модулі у системі. Наведена вище формула демонструє, як розділити систему на незалежні модулі, які можна розробити та тестувати окремо, а потім об'єднати в одну цілісну систему
Незалежність	Незалежність означає, що зміна одного компонента системи не повинна впливати на інші компоненти. Математично це можна виразити через відсутність залежності між різними частинами системи. Нехай $X_1, X_2, \ldots, X_n$ – це компоненти системи. Тоді можемо сказати, що ці компоненти є математично незалежними, якщо зміна одного компонента не впливає на значення інших компонентів. Згідно із [13] маємо: $P(X_i \mid X_j) = P(X_i),$ де $P(X_i)$ – ймовірність настання події X_i для всіх i та j – (за умови $i \neq j$). Ця формула виражає відсутність умовної залежності між компонентами системи, що підтверджує їхню незалежність один від одного

З табл. 2 наочно видно, що створення високоякісних моделей полягає у врахуванні різних аспектів, від потреб користувачів до тестування та постійного вдосконалення, а також у використанні математичних методів для формалізації та аналізу цих аспектів.

На практичному рівні розроблення багатоцільових моделей якості ПС врахування принципів, відображених у табл. 2, може мати доволі численну кількість практичних наслідків.

Розробимо принципи створення багатоцільових моделей якості на рівні математичних моделей.

Загальну модель функціональної якості задамо так: нехай Q_f представляє функціональну якість ПС, а M_f – метрику, яка характеризує цю якість. Тоді у загальному вигляді принцип формулюється відповідно до (1):

$$Q_f = f\left(M_f\right), \tag{1}$$

де f – це функція, що визначає зв'язок між метрикою M_f та функціональною якістю Q_f.

У нашому ж випадку ми удосконалюємо загальну модель функціональної якості ПС. Сформулюємо умови, обмеження та область адекватності математично.

Умови: нехай $F(Q)$ – це функція, яка описує функціональну якість Q ПС. Для кожного атрибута якості можна визначити вагу, яка відображає його важливість для кінцевого користувача; Нехай Q_i – значення i-го атрибута якості ПС, де $i = 1,2,\ldots,n$, а n – загальна кількість атрибутів якості; ω_i – вага i-го атрибута якості ПС, де $\omega_i > 0$, для $i = 1,2,\ldots,n$, та $\sum_{i=1}^{n} \omega_i = 1$, тобто сума всіх ваг дорівнює одиниці.

Обмеження: тоді розроблену модель функціональної якості можна виразити способом (2), де ця модель передбачає лінійну залежність між значеннями атрибутів якості та їх вагами:

$$F(Q) = \sum_{i=1}^{n} \omega_i \cdot Q_i, \tag{2}$$

де Q_i – значення i-го атрибута якості; ω_i – вага i-го атрибута якості; n – загальна кількість атрибутів якості.

Сфера застосування моделі обмежена наявністю достовірних даних про атрибути якості та їх відповідні значення. Ця модель дає змогу врахувати важливість кожного атрибута якості, призначивши йому вагу. Наприклад, для вебсайту можуть бути визначені атрибути якості, такі як швидкодія, безпека, доступність тощо. Кожен з цих атрибутів може мати свою вагу залежно від потреб користувача або специфіки проєкту. Цей спосіб моделювання функціональної якості дає змогу систематизувати і враховувати різні аспекти якості програмної системи, беручи до уваги важливість. Такий підхід може бути корисним для розроблення та оцінювання програмних продуктів у контексті вимог та потреб користувачів.

Область адекватності: модель доцільно використовувати у випадках, коли атрибути якості можна виміряти або оцінити числовими значеннями, а ваги цих атрибутів визначити з урахуванням їх важливості. Модель може бути менш ефективною у випадках, коли атрибути якості мають складні взаємозв'язки, або коли важко визначити їх ваги на основі об'єктивних критеріїв. Вказані вище умови, обмеження та область адекватності дають змогу математично визначити та розуміти, в яких умовах та як можна використати модель для оцінювання функціональної якості ПС.

Узагальнену модель ефективності використання ресурсів задаємо так: нехай Q_e представляє ефективність використання ресурсів програмної системи, а M_e – відповідну метрику. Тоді принцип можна виразити як формулу (3):

$$Q_e = g\left(M_e\right), \tag{3}$$

де g_f – функція, що визначає зв'язок між метрикою M_e та ефективністю використання ресурсів Q_e.

В нашому ж випадку аналогічно до минулої моделі удосконалимо загальну модель ефективності використання ресурсів ПС, сформулювавши умови, обмеження та область адекватності математично.

Умови: нехай E_i – ефективність використання i-го ресурсу ПС, де $i = 1,2,\ldots,m$, а m – загальна кількість ресурсів; й ω_i – вага i-го ресурсу ПС, де $\omega_i > 0$, для $i = 1,2,\ldots,m$, та $\sum_{i=1}^{m} \omega_i = 1$, тобто сума всіх ваг дорівнює одиниці.

Обмеження: розроблену модель ефективності використання ресурсів ПС можна подати способом (4), згідно із яким модель передбачає лінійну залежність між ефективністю використання ресурсів ПС та їх вагами:

$$E = \sum_{i=1}^{m} \omega_i \cdot E_i, \tag{4}$$

де ω_i – вага i-го ресурсу ПС; m – загальна кількість ресурсів.

Область застосування моделі обмежена наявністю достовірних даних про ефективність використання ресурсів та їх відповідні значення.

Область адекватності: модель придатна для використання у випадках, коли ефективність використання ресурсів можна виміряти або оцінити числовими значеннями, а ваги цих ресурсів визначити з урахуванням їх важливості. Модель може бути менш ефективною, коли ефективність використання ресурсів має складні взаємозв'язки або коли важко визначити їх ваги на основі об'єктивних критеріїв. Ці умови, обмеження та область адекватності математично визначають умови застосування моделі ефективності використання ресурсів ПС та допомагають зрозуміти, як використати її для оцінювання ефективності використання ресурсів у різних умовах та контекстах.

Задамо узагальнену модель надійності: нехай Q_r представляє ефективність використання ресурсів ПС, а M_r – відповідну метрику. Тоді принцип можна подати як вираз (5):

$$Q_r = h(M_r), \tag{5}$$

де h_f – це функція, що визначає зв'язок між метрикою M_r та ефективністю використання ресурсів Q_r.

Аналогічно до наведених вище моделей удосконалимо узагальнену модель надійності – сформулюємо умови, обмеження та область адекватності математично.

Умови: нехай R_i – рівень надійності i-го компонента ПС, де $i = 1,2,\ldots,k$, а k – загальна кількість компонентів ПС; нехай ω_i – вага i-го компонента ПС, де $\omega_i > 0$, для $i = 1,2,\ldots,k$, та $\sum_{i=1}^{k} \omega_i = 1$, тобто сума всіх ваг дорівнює одиниці.

Обмеження: розроблену модель надійності ПС можна виразити способом (6), згідно із яким модель передбачає лінійну залежність між ефективністю використання ресурсів ПС та їх вагами:

$$R = \sum_{i=1}^{k} \omega_i \cdot R_i, \tag{6}$$

де ω_i – вага i-го компонента ПС; k – загальна кількість компонентів ПС, R_i – рівень надійності i-го компонента ПС.

Область застосування моделі обмежена наявністю достовірних даних про рівень надійності компонентів та їх відповідні значення.

Область адекватності: модель підходить для використання у випадках, коли рівень надійності компонентів можна виміряти або оцінити числовими значеннями, а ваги цих компонентів визначити з урахуванням їх важливості. Модель може бути менш ефективною у випадках складних взаємозв'язків рівня надійності компонентів або коли важко визначити їх ваги на основі об'єктивних критеріїв. Ці умови, обмеження та область адекватності математично визначають умови застосування розробленої моделі надійності програмної системи та допомагають зрозуміти, як використати її для оцінювання надійності системи у різних умовах та контекстах.

Задамо модель безпеки: нехай Q_S – безпека ПС, а M_S – відповідна метрика. Тоді принцип можна подати як вираз (7):

$$Q_S = k(M_s), \tag{7}$$

де k_f – це функція, що визначає зв'язок між метрикою M_s та безпекою ПС Q_S.

Удосконалення наведеної вище моделі безпеки передбачає математичне розроблення умов, обмежень та області адекватності.

Умови: нехай S_i – рівень безпеки i-го аспекту ПС, де $i = 1,2,\ldots,l$, а l – загальна кількість аспектів безпеки ПС; нехай ω_i – вага i-го аспекту безпеки ПС, де $\omega_i > 0$, для $i = 1,2,\ldots,l$, та $\sum_{i=1}^{l} \omega_i = 1$, тобто сума всіх ваг дорівнює одиниці.

Обмеження: тоді розроблену модель безпеки ПС можна виразити способом (8), відповідно до якого модель передбачає лінійну залежність між рівнем безпеки аспектів та їх вагами:

$$S = \sum_{i=1}^{l} \omega_i \cdot S_i, \tag{8}$$

де S_i – рівень безпеки i-го аспекту ПС, де $i = 1,2,\ldots,l$, ω_i – вага i-го аспекту безпеки ПС.

Область застосування моделі обмежена наявністю достовірних даних про рівень безпеки аспектів та їх відповідні значення.

Область адекватності: модель доцільно використовувати у випадках, коли рівень безпеки аспектів можна виміряти або оцінити числовими значеннями, а ваги цих аспектів визначити з урахуванням їх важливості. Модель може бути менш ефективною у випадках, коли на рівні безпеки аспектів є складні взаємозв'язки або коли важко визначити їх ваги на основі об'єктивних критеріїв. Ці умови, обмеження та область адекватності математично визначають умови застосування моделі безпеки програмної системи та допомагають зрозуміти, як використати її для оцінювання рівня безпеки системи у різних умовах та контекстах.

Модель оцінки зручності використання задамо так: нехай Q_u – зручність використання ПС, програмної системи, а M_U – відповідна метрику. Тоді принцип можна подати як вираз (8):

$$Q_u = l(M_u), \tag{9}$$

де l – це функція, що визначає зв'язок між метрикою M_u та зручністю використання ПС Q_u.

Розглянемо математичне визначення умов, обмежень та області адекватності для моделі оцінки рівня зручності використання ПС:

Умови: нехай U_i – рівень зручності i-го аспекту використання ПС, де $i = 1,2,\ldots,z$, а z – загальна кількість аспектів зручності використання ПС; нехай ω_i – вага i-го аспекту зручності використання ПС, де $\omega_i > 0$, для $i = 1,2,\ldots,z$, та $\sum_{i=1}^{z} \omega_i = 1$, тобто сума всіх ваг дорівнює одиниці.

Обмеження: тоді розроблену модель оцінки рівня зручності використання ПС можна виразити способом (9), за яким модель передбачає лінійну залежність між рівнем безпеки аспектів та їх вагами:

$$U = \sum_{i=1}^{z} \omega_i \cdot U_i, \tag{10}$$

де U_i – рівень зручності i-го аспекту використання ПС; ω_i – вага i-го аспекту зручності використання ПС; z – загальна кількість аспектів зручності використання ПС.

Область застосування моделі обмежена наявністю достовірних даних про рівень зручності аспектів та їх відповідні значення.

Область адекватності: модель підходить для використання у випадках, коли рівень зручності аспектів можна виміряти або оцінити числовими значеннями, а ваги цих аспектів визначити з урахуванням їх важливості. Модель може бути менш ефективною у випадках, коли на рівні зручності аспектів є складні взаємозв'язки або коли важко визначити їх ваги на основі об'єктивних критеріїв. Ці умови, обмеження та область адекватності математично визначають умови застосування моделі оцінювання рівня зручності використання ПС та допомагають зрозуміти, як використати її для оцінювання зручності використання системи у різних умовах та контекстах.

Наведені вище принципи дають змогу використовувати математичні моделі для оцінювання та аналізу різних аспектів якості ПС. Кожна модель представлена відповідною функцією, яка визначає залежність між метрикою якості та самою якістю ПС.

У табл. 3 наведено опис метрик якості ПС, застосованих під час розроблення математичного узагальнення принципів створення багатоцільових моделей якості ПС.

Таблиця 3

Опис метрик якості ПС, застосованих для розроблення математичного узагальнення принципів створення багатоцільових моделей якості ПС

Модель	Метрика	Опис	Відповідність до ISO/IEC 25010	Відмінності від ISO/IEC 25010
Модель функціональної якості	Метрика функціональності ПС M_f	Вимірює рівень функціональності системи, такий як кількість функцій, їх складність та відповідність вимогам. Вище значення вказує на більшу функціональність	Повнота функцій, точність, сприйнятливість	Додаткові метрики вимог
Модель ефективності	Метрика ефективності ПС M_e	Визначає ефективність використання ресурсів, таких як час відгуку, використання пам'яті, пропускна спроможність тощо. Вище значення вказує на більшу ефективність	Швидкість відгуку, використання ресурсів	Введення часу реакції

Продовження табл. 3

Модель	Метрика	Опис	Відповідність до ISO/IEC 25010	Відмінності від ISO/IEC 25010
Модель надійності	Метрика надійності ПС M_r	Вимірює стійкість системи до виникнення помилок та відмов. Це може бути середній час між відмовами (MTBF), кількість відмов за певний період та інші параметри. Вище значення вказує на більшу надійність	Частота відмов, MTBF	Введення часу між відмовами
Модель безпеки	Метрика безпеки ПС M_S	Оцінює рівень захищеності системи від несанкціонованого доступу, вразливостей та загроз безпеки. Вище значення вказує на більшу безпеку	Стійкість до атак, безпека даних	Введення метрики захищеності
Модель зручності використання	Метрика зручності використання ПС M_u	Визначає, наскільки легко та зручно користувач може взаємодіяти із системою. Це може бути час навчання, кількість кроків для виконання операцій, рівень задоволення користувача тощо. Вище значення вказує на зручніше використання	Час навчання, задоволення користувача	Введення рівня задоволення

Варто зауважити, що метрики, наведені в табл. 3, дають змогу кількісно оцінити різні аспекти якості ПС та визначити їхній рівень відповідно до встановлених критеріїв. Щоб кількісно оцінити різні аспекти якості програмного забезпечення (ПЗ) та визначити їхній рівень відповідно до встановлених критеріїв, варто використовувати метрики, наведені в табл. 3. Наведемо детальніше пояснення кожного аспекту якості та відповідних метрик і розглянемо їх використання за допомогою прикладів. *Метрика функціональності ПЗ*: вимірює рівень функціональності системи, такий як кількість функцій, їх складність та відповідність вимогам. Вище значення вказує на більшу функціональність. Приклад: якщо у програми є 100 функцій, які вона виконує точно відповідно до специфікації, то її оцінка функціональності може бути близькою до 100 %. *Метрика ефективності ПЗ*: визначає ефективність використання ресурсів, таких як час відгуку, використання пам’яті, пропускна спроможність тощо. Вище значення вказує на більшу ефективність. Приклад: якщо час відгуку програми на запити користувачів низький, наприклад, менше ніж 0,1 секунди, це свідчить про високу ефективність програми. *Метрика надійності ПЗ*: вимірює стійкість системи до виникнення помилок та відмов. Це може бути середній час між відмовами (MTBF), кількість відмов на певний період часу та інші параметри. Вище значення вказує на більшу надійність. Приклад: якщо програма працює безвідмовно впродовж тривалого періоду, наприклад, 1000 годин без жодної відмови, це свідчить про високий рівень її надійності. *Метрика безпеки ПЗ*. Оцінює

рівень захищеності системи від несанкціонованого доступу, вразливостей та загроз безпеки. Вище значення свідчить про більшу безпеку. Приклад: якщо програма виявляє і блокує спроби несанкціонованого доступу до конфіденційної інформації, це свідчить про високий рівень безпеки. *Метрика зручності використання ПЗ*. Визначає, наскільки легко та зручно користувач може взаємодіяти з системою. Вище значення вказує на кращу зручність використання. Приклад: якщо інтерфейс користувача програми інтуїтивно зрозумілий і ним легко користуватися, це може вказувати на високий рівень зручності використання.

Також варто зазначити, що кожна метрика має певні вимоги та критерії оцінки, які можна визначити відповідно до стандартів, таких як ISO/IEC 25010, а також налаштувати відповідно до конкретних потреб та вимог проєкту.

Щоб поєднати ці метрики функціональної якості у математичному апараті під час моделювання, ми можемо використати наведені вище формули та застосувати їх для виконання комплексної оцінки функціональної якості Q_f. Один з можливих підходів – використання зважених середніх, коли метрики оцінюють за їхнім значенням і важливістю. Отже, формулу можна подати так (10):

$$Q_f = \omega_1 \cdot M_1 + \omega_2 \cdot M_2 + \omega_3 \cdot M_3, \tag{11}$$

де M_1 – час відгуку (*Response Time*); M_2 – кількість помилок (*Number of Errors*); M_3 – повнота функціоналу (*Functionality Completeness*); ω_1, ω_2, ω_3 – ваги, що відображають важливість кожної метрики в загальній оцінці. Зазвичай вони додатні та сума їх дорівнює одиниці. Але варто зазначити, що формула (11) дозволяє об'єднати різні аспекти функціональної якості у єдину комплексну оцінку, яка може використовуватися для порівняння та аналізу різних ПС.

Аналогічно до попереднього випадку, можна поєднати ці метрики ефективності використання ресурсів у єдину оцінку. Використаємо знову зважену середню для комплексної оцінки ефективності (11):

$$Q_e = \omega_1 \cdot P_1 + \omega_2 \cdot P_2 + \omega_3 \cdot P_3, \tag{12}$$

де P_1 – використання CPU (*CPU Utilization*); P_2 – використання пам'яті (*Memory Utilization*); P_3 – пропускна здатність (*Throughput*); ω_1, ω_2, ω_3 – ваги, що відображають важливість кожної метрики в загальній оцінці.

Варто зауважити, що ваги зазвичай додатні, а їх сума дорівнює одиниці [14]. Відповідно формула (12) дає змогу об'єднати різні аспекти ефективності використання ресурсів у єдину комплексну оцінку, якою можна скористатися для порівняння та аналізу різних ПС.

Аналогічно до попередніх випадків, можемо поєднати ці метрики надійності у єдину оцінку надійності Q_r, використовуючи зважену середню (8):

$$Q_r = \omega_1 \cdot T_1 + \omega_2 \cdot T_2 + \omega_3 \cdot T_3, \tag{13}$$

де T_1 – середній час між відмовами (MTBF); T_2 – інтенсивність відмов (*Failure Intensity*); T_3 – відновний час (*Recovery Time*); ω_1, ω_2, ω_3 – ваги, що відображають важливість кожної метрики в загальній оцінці.

У формулі (12) ваги зазвичай додатні, їх сума дорівнює одиниці, а формула (8) дає змогу об'єднати різні аспекти надійності в єдину комплексну оцінку, яка може використовуватися для порівняння та аналізу різних ПС.

Відповідно за аналогією до попередніх розглянутих вище випадків, можемо поєднати метрики безпеки ПС в єдину оцінку щодо якості безпеки ПС – Q_S, використовуючи зважену середню (13):

$$Q_S = \omega_1 \cdot Z_1 + \omega_2 \cdot Z_2 + \omega_3 \cdot Z_3, \quad (14)$$

де Z_1 – кількість вразливостей (*Number of Vulnerabilities*); Z_2 – рівень аутентифікації (*Authentication Level*); Z_3 – швидкість виявлення та реагування на загрози (*Threat Detection and* Response Time); ω_1, ω_2, ω_3 – ваги, що відображають важливість кожної метрики в загальній оцінці.

Відповідно в (13) ваги зазвичай додатні, їх сума дорівнює одиниці, а саме застосування формули (13) на практиці дає змогу об'єднати різні аспекти безпеки в єдину комплексну оцінку, яку можна використовувати для порівняння та аналізу різних ПС.

Для об'єднання метрик зручності використання ПС у єдину оцінку Q_u доцільно також скористатися зваженою середньою (14):

$$Q_S = \omega_1 \cdot U_1 + \omega_2 \cdot U_2 + \omega_3 \cdot U_3, \quad (15)$$

де U_1 – час навчання (*Learning Time*); U_2 – Кількість кроків для виконання операцій (*Number of Steps*); U_3 – рівень задоволення користувача (*User Satisfaction Level*) ω_1, ω_2, ω_3 – ваги, що відображають важливість кожної метрики в загальній оцінці.

Відповідно час навчання U_1 – час, потрібний користувачеві для ознайомлення з ПС та освоєння основних функцій. Важливо, щоб він був якнайменшим, щоб користувачі могли швидко почати використовувати систему. Кількість кроків для виконання операцій (*Number of Steps*): кількість кроків або дій, які має виконати користувач для здійснення певної операції в ПС. Що менше кроків, то зручніше використовувати систему. Рівень задоволення користувача (*User Satisfaction Level*) – оцінка задоволення користувачів від використання ПС, яку можна виміряти за допомогою опитування користувачів або моніторингу їхніх реакцій під час використання системи.

На практичному рівні застосування формули (14) дає змогу об'єднати різні аспекти зручності використання ПС в єдину комплексну оцінку, яка може використовуватися для порівняння та аналізу різних ПС.

Отже, узагальнені принципи створення багатоцільових моделей якості використовують метрики якості для оцінювання різних аспектів ПС. Кожна модель має свій набір метрик, які відображають специфічні характеристики якості цієї моделі. Під час розроблення програмної системи або оцінювання її якості використовують ці метрики для визначення її ефективності та сфер покращень [3].

Наприклад, для розроблення ПС можна використовувати метрики функціональної якості, щоб визначити, наскільки ефективно система виконує свої функції. Метрики ефективності можна використовувати для оцінки використання ресурсів системи та виявлення можливих вузьких місць. Метрики безпеки допомагають визначити рівень захищеності системи від потенційних загроз.

Застосування метрик в узагальнених принципах дає змогу виробити комплексний погляд на якість програмної системи та визначити області, які потребують уваги та покращень. Це допомагає розробникам та інженерам управління якістю програмного забезпечення приймати обґрунтовані рішення щодо поліпшення та оптимізації систем.

Висновки

1. На практичному рівні розроблення багатоцільових моделей якості ПС врахування принципів створення багатоцільових моделей якості ПС може мати доволі численні практичні

наслідки. Розроблена модель функціональної якості ПС підходить для використання у випадках, коли атрибути якості можна виміряти або оцінити числовими значеннями, а ваги цих атрибутів визначити з урахуванням їх важливості. Модель може бути менш ефективною у випадках, коли атрибути якості мають складні взаємозв'язки або коли важко визначити їх ваги на основі об'єктивних критеріїв. Удосконалена загальна модель ефективності використання ресурсів ПС підходить для використання у випадках, коли ефективність використання ресурсів ПС можна виміряти або оцінити числовими значеннями, а ваги цих ресурсів визначити з урахуванням їх важливості. Модель може бути менш ефективною у випадках, коли ефективність використання ресурсів має складні взаємозв'язки або коли важко визначити їх ваги на основі об'єктивних критеріїв. Розроблена модель оцінки рівня зручності використання ПС підходить для використання у випадках, коли рівень зручності аспектів можна виміряти або оцінити числовими значеннями, а ваги цих аспектів визначити з урахуванням їх важливості. Модель може бути менш ефективною у випадках, коли на рівні зручності аспектів є складні взаємозв'язки або коли важко визначити їх ваги на підставі об'єктивних критеріїв. Ці умови, обмеження та область адекватності математично визначають умови застосування моделі оцінки рівня зручності використання ПС та допомагають зрозуміти, як використати її, щоб оцінити зручність використання системи у різних умовах та контекстах. Розроблена модель безпеки ПС підходить для використання у випадках, коли рівень безпеки аспектів можна виміряти або оцінити числовими значеннями, а ваги цих аспектів визначити з урахуванням їх важливості. Модель може бути менш ефективною у випадках, коли рівень безпеки аспектів має складні взаємозв'язки або коли важко визначити їх ваги на основі об'єктивних критеріїв.

2. Дослідження показали, що створення якісних моделей передбачає урахування різних аспектів, від потреб користувачів до тестування та постійного вдосконалення, а також використання математичних методів для формалізації та аналізу цих аспектів.

3. Розроблені принципи створення багатоцільових моделей якості на рівні математичних моделей дають змогу використовувати математичні моделі для оцінювання та аналізування різних аспектів якості ПС. Кожна розроблена модель представлена відповідною функцією, яка визначає залежність між метрикою якості та самою якістю ПС.

Список літератури

1. Abbas, S., Aftab, S., Adnan Khan, M., M. Ghazal, T., Al Hamadi, H., & Yeob Yeun, C. (2023). Data and Ensemble Machine Learning Fusion Based Intelligent Software Defect Prediction System. *Computers, Materials & Continua*, *75*(3), 6083–6100. https://doi.org/10.32604/cmc.2023.037933

2. Aftab, S., Abbas, S., Ghazal, T. M., Ahmad, M., Hamadi, H. A., Yeun, C. Y., & Khan, M. A. (2023). A cloud-based software defect prediction system using data and decision-level machine learning fusion. *Mathematics*, *11*(3), 632. https://doi.org/10.3390/math11030632

3. Arcos-Medina, G., & Mauricio, D. (2020). The influence of the application of agile practices in software quality based on ISO/IEC 25010 standard. *International Journal of Information Technologies and Systems Approach*, *13*(2), 27–53. https://doi.org/10.4018/ijitsa.2020070102

4. Azar, D., Harmanani, H., & Korkmaz, R. (2009). A hybrid heuristic approach to optimize rule-based software quality estimation models. *Information and Software Technology*, *51*(9), 1365–1376. https://doi.org/10.1016/j.infsof.2009.05.003

5. Bharathi, R., & Selvarani, R. (2020). Hidden markov model approach for software reliability estimation with logic error. *International Journal of Automation and Computing*, *17*(2), 305–320. https://doi.org/10.1007/s11633-019-1214-7

6. Chu, Y., & Xu, S. (2007). Exploration of complexity in software reliability. *Tsinghua Science and Technology*, *12*(S1), 266–269. https://doi.org/10.1016/s1007-0214(07)70122-0

7. F. El-Sofany, H., A. Taj-Eddin, I., El-Hoimal, H., Al-Tourki, T., & Al-Sadoon, A. (2013). Enhancing software quality by an SPL testing based software testing. *International Journal of Computer Applications*, *69*(6), 5–13. https://doi.org/10.5120/11844-7574

8. Foidl, H., & Felderer, M. (2016). Integrating software quality models into risk-based testing. *Software Quality Journal*, *26*(2), 809–847. https://doi.org/10.1007/s11219-016-9345-3

9. Helander, M. E., Ming Zhao & Ohlsson, N. (1998). Planning models for software reliability and cost. *IEEE Transactions on Software Engineering*, *24*(6), 420–434. https://doi.org/10.1109/32.689400

10. Janes, A., Scotto, M., Pedrycz, W., Russo, B., Stefanovic, M., & Succi, G. (2006). Identification of defect-prone classes in telecommunication software systems using design metrics. *Information Sciences*, *176*(24), 3711–3734. https://doi.org/10.1016/j.ins.2005.12.002

11. Jones, C. B., & Randell, B. The role of structure: A dependability perspective. У *Structure for dependability: Computer-based systems from an interdisciplinary perspective* (с. 3–15). Springer-Verlag. https://doi.org/10.1007/1-84628-111-3_1

12. Letichevsky, A., Kapitonova, J., Letichevsky, A., Volkov, V., Baranov, S., & Weigert, T. (2005). Basic protocols, message sequence charts, and the verification of requirements specifications. *Computer Networks*, *49*(5), 661–675. https://doi.org/10.1016/j.comnet.2005.05.005

13. Long, Z. (2018). Research on the quality assurance method of spacecraft software based on software testing. *Science Discovery*, *6*(1), 52. https://doi.org/10.11648/j.sd.20180601.19

14. Malik, V., & Singh, S. (2017). Tools, strategies & models for incorporating software quality assurance in risk oriented testing. *Oriental Journal of Computer Science and Technology*, *10*(3), 603–611. https://doi.org/10.13005/ojcst/10.03.08

15. Nasar, M., Johri, P., & Chanda, U. (2014). Software testing resource allocation and release time problem: A review. *International Journal of Modern Education and Computer Science*, *6*(2), 48–55. https://doi.org/10.5815/ijmecs.2014.02.7

16. Pietrantuono, R. (2020). On the testing resource allocation problem: Research trends and perspectives. *Journal of Systems and Software*, *161*, 110462. https://doi.org/10.1016/j.jss.2019.110462

17. Sahu, K., & Srivastava, R. K. (2021). Predicting software bugs of newly and large datasets through a unified neuro-fuzzy approach: Reliability perspective. *Advances in Mathematics: Scientific Journal*, *10*(1), 543–555. https://doi.org/10.37418/amsj.10.1.54

18. Samal, U., & Kumar, A. (2024). A neural network approach for software reliability prediction. *International Journal of Reliability, Quality and Safety Engineering*. https://doi.org/10.1142/s0218539324500098

19. Shrivastava, A. K., Sharma, R., & Pham, H. (2022). Software reliability and cost models with warranty and life cycle. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 1748006X2210762. https://doi.org/10.1177/1748006x221076273

20. Tankala, D. K. (2023). Learning based Software Defect Prediction. *International Journal of Computational Intelligence Research (IJCIR)*, *19*(1), 51–62. https://doi.org/10.37622/ijcir/19.1.2023.51-62

21. Upadhyay, R., & Johri, P. (2013). Review on Software Reliability Growth Models and Software Release Planning. *International Journal of Computer Applications*, *73*(12), 1–7. https://doi.org/10.5120/12790-9769

22. Yu, L., & Mishra, A. (2012). Experience in Predicting Fault-Prone Software Modules Using Complexity Metrics. *Quality Technology & Quantitative Management*, *9*(4), 421–434. https://doi.org/10.1080/16843703.2012.11673302

PRINCIPLES OF CREATING MULTI-OBJECTIVE QUALITY MODELS FOR SOFTWARE SYSTEMS

Anton Shantyr

State University of Information and Communication Technologies,
Department of Artificial Intelligence, Kyiv, Ukraine
E-mail: anton.shantyr@gmail.com, ORCID: 0000-0002-0466-3659

This article proposes an original approach at a mathematical level to the creation of multi-objective quality models for software systems. The research is based on the study and generalization of quality modeling trends of software systems and user needs in order to determine optimal principles for constructing such models. The article provides mathematical explanations that play a key role in identifying and formalizing the principles of creating multi-objective quality models of software. An important aspect is the consideration of quality model construction principles at the mathematical level, allowing for a more precise assessment and analysis of various aspects of software quality. The research results indicate that incorporating mathematical principles into the creation of multi-objective quality models for software systems can have a significant practical impact. It is established that on a practical level of developing multi-objective quality models for software systems, consideration of the principles of creating multi-objective quality models can have numerous practical implications. Specifically, the application of metrics within established principles allows for a comprehensive view of software quality and identifies areas requiring attention and improvement. This helps developers and software quality engineers make informed decisions regarding system improvement and optimization. The research has shown that creating high-quality models of quality requires attention to various aspects, from user needs to testing and continuous improvement, as well as the use of mathematical methods for their formalization and analysis. The developed principles of creating multi-objective quality models at the level of mathematical models allow the use of these models to assess and analyze various aspects of software quality, representing each model using a corresponding function that determines the relationship between quality metrics and the quality of the software itself. It is expected that further development and implementation of these principles will contribute to improving software development processes and ensure high quality of the resulting software.

Key words: software; quality metrics; user needs; information technology; Agile methodology; system support; quality standards; mathematical framework.