ЗАХИСТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ОБЛАДНАННЯ/ SOFTWARE & HARDWARE ARCHITECTURE SECURITY

DOI: 10.18372/2225-5036.30.19208

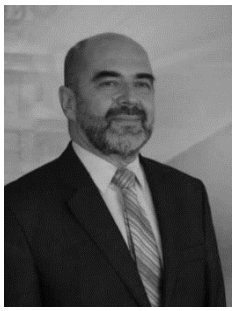
НЕВИЗНАЧЕНІСТЬ ОЦІНЮВАННЯ КІЛЬКІСНИХ ХАРАКТЕРИСТИК ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Антон Шантир, Ольга Зінченко,
Євген Чичкарьов, Олександр Вишнівський

Державний університет інформаційно-комунікаційних технологій

ШАНТИР Антон Сергійович, к.т.н.
Рік та місце народження: 1988 рік, м. Київ, Київська область, Україна.
Освіта: Національний технічний університет України «Київський політехнічний інститут», 2011 р.
Посада: доцент кафедри штучного інтелекту.
Наукові інтереси: стратегічне управління та організаційні трансформації, штучний інтелект та аналіз великих даних, технології прийняття рішень та інтелектуальні системи, кібербезпека та захист інформації.
Публікації: понад 60 наукових публікацій, серед яких наукові статті, навчальні посібники, патенти, тези та матеріали доповідей на конференціях.
E-mail: anton.shantyr@gmail.com.
Orcid ID: 0000-0002-0466-3659.

ЗІНЧЕНКО Ольга Валеріївна, д.т.н., доц.
Рік та місце народження: 1986 рік, м. Київ, Київська область, Україна.
Освіта: Державний університет інформаційно-комунікаційних технологій, 2009 рік.
Посада: завідувач кафедри штучного інтелекту з 2020 року.
Наукові інтереси: методи та засоби штучного інтелекту, проектування систем штучного інтелекту, технології прийняття рішень, аналіз і обробка великих даних.
Публікації: понад 70 наукових публікацій, серед яких наукові статті, навчальні посібники, тези та матеріали доповідей на конференціях.
E-mail: zinchenkoov@gmail.com .
Orcid ID: 0000-0002-3973-7814.



ЧИЧКАРЬОВ Євген Анатолійович, д.т.н., проф.
Рік та місце народження: 1964 рік, м. Маріуполь, Донецька обл., Україна.
Освіта: Московський хіміко-технологічний інститут ім.Д.І.Менделєєва, 1987 рік.
Посада: професор кафедри штучного інтелекту.
Наукові інтереси: проектування систем штучного інтелекту, обробка і розпізнавання зображень, машинне навчання, нейронні мережі глибокого навчання, технології створення веб-додатків та мобільних додатків.
Публікації: більше 200 наукових публікацій, серед яких наукові статті, монографії, навчальні посібники, патенти, тези та матеріали доповідей на конференціях.
E-mail: chychkarovea@gmail.com.
Orcid ID: 0000-0002-4362-5129.

ВИШНІВСЬКИЙ Олександр Вікторович, аспірант
Рік та місце народження: 1989 рік, м. Київ, Україна.
Освіта: Київський національний університет ім. Т.Г. Шевченка, 2014 р.
Посада: аспірант кафедри комп'ютерних наук з 2022 року.
Наукові інтереси: методи та засоби штучного інтелекту, аналіз і обробка великих даних.
Публікації: 3 наукових публікації, 5 тез доповідей на конференціях.
E-mail: o.vyshnivskyi@gmail.com.
Orcid ID: 0009-0008-0209-9549.

Анотація. *Нині оцінка якості програмного забезпечення виступає важливим етапом у процесах його розробки та впровадження. Вона надає розробникам можливість отримати об'єктивну оцінку, щодо розроблених програмних продуктів та визначити їх відповідність до діючих міжнародних стандартів та вимог із оцінки якості програмного забезпечення. Однак даний процес в межах оцінки якості програмного забезпечення часто супроводжується певним рівнем невизначеності в межах реалізації оцінювання кількісних характеристик якості, що може ускладнити прийняття рішень про перспективність використання розробленого продукту та його безпеки. Суттєвий внесок у розробку теоретичних й практичних аспектів, щодо узагальнення проблематики питання невизначеності оцінювання кількісних характеристик якості внесли такі вчені, як: S. Hayashi, M. Kubo, H. Mori, C. Areces, R. Fervari, A. Saravia, F. Velázquez-Quesada, S. Guaman, J. Alamo, J. Caiza, M. Nakamura та ін.. Метою даної статті є вирішення проблеми пов'язаної із невизначеністю оцінювання кількісних характеристик якості програмних систем. Для реалізації мети в статті поставлені і вирішені такі завдання: розглянуті різні аспекти невизначеності оцінювання кількісних характеристик якості комп'ютерних програмних систем; розроблено методологічний підхід до вирішення проблеми пов'язаної із невизначеністю оцінювання кількісних характеристик якості; проведено практичне дослідження розробленого підходу. В процесі вирішення піднятих завдань було використано методи аналізу, синтезу, узагальнення та порівняння.*

Ключові слова: *якість програмних систем, апостеріорний розподіл, стандарти якості, критичні джерела невизначеності, байєсові методи, байєсове оновлення, гібридний підхід.*

Постановка проблеми

В останні десятиліття комп'ютерні програмні системи стали необхідною складовою практично кожної сфери діяльності людей. Зростання їхньої складності та значення викликало необхідність вдосконалення процесів їх розробки, впровадження та підтримки. Одним із ключових аспектів при оцінці програмних систем є їх якість, яка визначається кількісними характеристиками такими, як продуктивність, надійність, швидкодія та інші. Проте, оцінка якості програмних систем часто стикається з проблемою невизначеності. Невизначеність оцінювання кількісних характеристик якості відображає рівень неоднозначності, або варіації у визначенні цих характеристик. Вище вказане може бути наслідком неточності вимірювань, використання різних методик оцінки, впливу людського фактора, або складності самої системи. У зв'язку з цим, розуміння та управління невизначеністю в оцінюванні якості програмних систем стає досить важливим завданням для дослідників та практиків у галузі програмної інженерії.

Аналіз останніх досліджень та публікацій

У сфері оцінки кількісних характеристик якості програмних систем було вирішено та досліджено багато питань. Деякі з ключових аспектів цієї галузі включають розвиток методик і моделей для оцінки якості [17], створення та аналізу стандартів якості [8], а також розробку інструментів і технологій для вимірювання та аналізу характеристик якості [2]. Об'єктом дослідження є методологічні аспекти оцінки кількісних характеристик якості програмних систем.

Опрацьовано наукові роботи C. Areces, R. Fervari, A. Saravia, & F. Velázquez-Quesada [1] проаналізовано логіку на основі невизначеності в процесі здійснення дій; S. Guaman, J. Alamo, & J. Caiza, [8] проаналізовано методичні підходи з контролю якості програмного забезпечення, зокрема, з кількісної оцінки приватності в інформаційних системах; S. Hayashi, M. Kubo, H. Mori, , & M. Nakamura, [10] досліджено значення кількісної оцінки та оцінки з використанням автоматизованого програмного забезпечення; D. Behera, [2] розглянуто альтернативні методології для розв'язання задач лінійного програмування з епістемічною невизначеністю; M. Bougeret, A.Pessoa, & M. Poss [3] розглянуто задачі робустного планування з обмеженою невизначеністю для однієї машини; E. Merzlyakova & E. Yanchenko, [17] розглянуто аспекти невизначеності кількісної оцінки якості програмного забезпечення та ін.

Незважаючи на суттєві напрацювання вчених залишаються недостатньо обґрунтованими та дослідженими методологічні підходи та аспекти в межах вирішення питання невизначеності оцінювання кількісних характеристик якості програмних систем.

Мета та постановка завдання

Метою даної статті є вирішення проблеми пов'язаної із невизначеністю оцінювання кількісних характеристик якості програмних систем. Для реалізації мети в статті поставлені і вирішені такі завдання: розглянуті різні аспекти невизначеності оцінювання кількісних характеристик якості комп'ютерних програмних систем; розроблено методологічний підхід до вирішення проблеми пов'язаної із невизначеністю оцінювання кількісних характеристик якості; проведено практичне дослідження розробленого підходу. В процесі вирішення піднятих завдань було використано методи аналізу, синтезу, узагальнення, порівняння.

Виклад основного матеріалу дослідження

Теоретичний розгляд питання невизначеності оцінювання кількісних характеристик якості програмних систем

Невизначеність оцінювання кількісних характеристик якості – це поняття, яке відображає рівень непевності, або варіації в оцінці характеристик якості програмних систем [2].

В загальному випадку невизначеність виникає через низку факторів таких, як неточність вимірювань, різні методики оцінки, людський фактор та складність самих систем. Згідно з [10] до вище зазначених факторів варто віднести:

- неповноту вимог: часто вимоги до програмного продукту формулюються нечітко, або не повністю. Це може призводити до непорозумінь щодо того, що саме потрібно оцінювати та як це робити;

- суб'єктивність оцінок: оцінки якості часто ґрунтуються на суб'єктивних оцінках людей, які можуть

мати різний досвід, розуміння та ставлення до програмного продукту;

- залежність від контексту: якість програмного продукту може залежати від контексту його використання. Наприклад, від технічного середовища, в якому він працює, або від потреб користувачів;

- важкість вимірювання: деякі аспекти якості програмних систем можуть бути складні для об'єктивного вимірювання. Наприклад, складність програми, або її надійність можуть бути складні для точного визначення;

- ефект «бокового впливу»: зміни в одній характеристиці якості можуть мати неочікувані наслідки для інших характеристик. Наприклад, збільшення продуктивності програми може погіршити її надійність;

- час і ресурси: обмеженість часу і ресурсів для проведення оцінки може призвести до того, що деякі аспекти якості будуть оцінені не повністю, або не точно;

- інтерпретація результатів: оцінки якості можуть бути піддані різним інтерпретаціям, особливо коли вони представлені у формі числових значень, або метрик. Різні стейкхолдери можуть мати різний рівень розуміння, або приділяти різну важливість різним аспектам якості;

- динамічність середовища: середовище, в якому працює програмний продукт, може змінюватися з часом. Це може включати зміни в апаратному забезпеченні, операційній системі, або мережевих умовах, що може впливати на його ефективність та інші аспекти якості.

В свою чергу відмітимо, що вище вказані фактори створюють виклики для оцінювання якості програмних систем і підкреслюють важливість уважного планування та методології оцінювання.

У відповідності до праці [1] невизначеність оцінювання якості програмних систем відображається в статистичних параметрах таких, як середнє значення, стандартне відхилення та інші показники розподілу даних. Натомість в праці [17] акцентується увага на методах вимірювання якості, вказуючи на те, що різні методи можуть призводити до різної ступені невизначеності. В праці [21] зазначено, що технічні складнощі програмних систем впливають на невизначеність їх якості, враховуючи такі аспекти, як складність архітектури та інші технічні фактори.

Згідно з [3] сучасні дослідження питань, які пов’язані із невизначеність оцінювання кількісних характеристик якості спрямовуватися на конкретні випадки використання програмних систем. Зокрема досить численна кількість науковців [7] детально досліджують, питання: «Як невизначеність впливає на ефективність програмних систем та якість обслуговування?».

На узагальненому рівні науковцями [8] відмічаються наступні основні аспекти невизначеності кількісного оцінювання якості програмних систем:

- неточність вимірювань: при вимірюванні різних характеристик якості таких, як продуктивність, надійність, або безпека, можуть виникати помилки через недосконаліть інструментів чи методів вимірювання;

- різні методики оцінки: існує багато методик та стандартів оцінки якості програмного забезпечення (наприклад, ISO/IEC 25010: різні методики можуть давати різні результати для тих самих характеристик, що призводить до невизначеності);

- людський фактор: оцінка якості часто залежить від експертних суджень, які можуть варіюватися в залежності від досвіду та знань експертів;

- складність систем: сучасні програмні системи можуть бути дуже складними, що ускладнює їх повне та точне оцінювання.

Таблиця 1

Результати розгляду питання невизначеності оцінювання

Стандарт якості	**Опис**	**Проблеми невизначеності оцінювання**
ISO/IEC 25010 (SQuaRE)	Стандарт визначає модель якості програмного забезпечення та надає набір характеристик якості, які можна оцінити.	Варто відмітити, що не всі характеристики легко вимірювати чи кількісно оцінити. Наприклад, характеристика «ефективність використання ресурсів» може бути виміряна за допомогою кількісних метрик (час відповіді чи швидкодія) але інші аспекти такі, як енергоефективність, можуть бути менш явними для кількісного вимірювання.
CMMI (Capability Maturity Model Integration)	Фокусується на процесах, він також може оцінювати певні кількісні характеристики якості, наприклад, час виконання завдань чи кількість дефектів.	В даному стандарті інтерпретація даних може бути невизначеною, оскільки однією з особливостей CMMI є те, що він фокусується на оцінці процесів, а не на конкретних числових показниках якості.
IEEE 1061	Стандарт надає набір методів та критеріїв для оцінки відповідності програмного забезпечення вимогам.	Хоч даний стандарт може визначати кількісні метрики для вимірювання відповідності вимогам, проте – інтерпретація цих метрик та їх зв'язок із загальною якістю продукту також може бути неоднозначною.
ISO/IEC 9126	Даний стандарт був замінений ISO/IEC 25010, також мав на меті визначення моделі якості програмного забезпечення.	Хоч даний стандарт включає кількісні характеристики, такі як продуктивність та ефективність – інтерпретація результатів може бути невизначеною через широкий спектр можливих метрик і методів їх вимірювання.
ISO/IEC 25023 (SQuaRE)	Даний стандарт зосереджується на визначенні моделі якості для специфікації та оцінки якості продуктів та систем	Хоч даний стандарт містить кількісні характеристики такі, як продуктивність та надійність – інтерпретація результатів може бути складною через різні контексти і потреби користувачів
ISO/IEC 15504 (SPICE)	Даний стандарт визначає модель процесу оцінки процесів розробки та підтримки програмного забезпечення	Незважаючи на те, що даний стандарт фокусується на оцінці процесів, деякі з його атрибутів такі, як продуктивність та ефективність, можуть бути кількісно оцінені. Проте, невизначеність може виникнути при інтерпретації цих результатів в контексті загальної якості продукту.

Наведено результати розгляду питання невизначеності оцінювання кількісних характеристик якості програмних систем у сучасних стандартах (табл. 1), які регулюють питання, щодо дотримання та оцінки якості програмних систем. Відмітимо, що врахування наведених в таблиці 1 проблем та пошук способів їх подолання є важливими завданнями для покращення ефективності оцінювання якості програмних систем.

Зважаючи на вище вказане проблеми які пов'язані з невизначеністю оцінювання кількісних характеристик якості у сучасних стандартах, щодо дотримання та оцінки якості програмних систем включають наступні аспекти:

- неоднозначність метрик: самі метрики можуть бути визначені нечітко, або неоднозначно, що ускладнює їх застосування та порівняння між різними проектами, або організаціями;

- суб'єктивний характер оцінювання: багато метрик якості, особливо ті, що стосуються властивостей користувацького інтерфейсу, або ефективності взаємодії з користувачем, можуть бути суб'єктивно оцінені різними оцінювачами, що призводить до варіативності у вимірюваннях;

- варіативність у вимогах: якості програмних систем можуть вимірюватися у різних контекстах та для різних цільових аудиторій. Це може викликати неоднозначності в тому, які саме метрики слід застосовувати та, як їх інтерпретувати;

- складність вимірювання: деякі аспекти якості програмних систем можуть бути складніше вимірювати, ніж інші. Наприклад, вимірювання продуктивності, або ефективності може потребувати спеціальних інструментів, або складних алгоритмів.

- недостатність даних: у деяких випадках може бути складно зібрати достатню кількість даних для вимірювання певних аспектів якості програмних систем, що робить оцінку менш точною, або навіть неможливою.

- еволюція вимог: вимоги до програмних систем можуть змінюватися з часом, і метрики якості можуть втрачати свою актуальність, або потребувати оновлення для відповідності новим вимогам.

Отже, невизначеність оцінювання кількісних характеристик якості програмних систем може проявлятися у стандартах через різні причини такі, як складність вимірювання деяких характеристик, неоднозначність у визначенні метрик чи інтерпретація результатів.

Розробка та практичне застосування методологічних шляхів вирішення проблеми невизначеності оцінювання кількісних характеристик якості

Нині у сфері оцінювання кількісних характеристик якості програмного забезпечення залишаються досить численна кількість невирішених проблем [20], які потребують додаткового розгляду та дослідження. Однією із таких проблем є усунення невизначеності у вимірюваннях та кількісних оцінках якості програмних систем – дана проблема може впливати на точність оцінок якості. Згідно з [19] більш точні методики вимірювання та аналізу можуть допомогти краще прогнозувати поведінку програмних систем, зменшуючи ризики та підвищуючи якість. Згідно з [6] невизначеність при кількісних оцінках якості програмного забезпечення залишається критичною проблемою, яка може впливати на точність та надійність оцінок якості.

Наявні методи зменшення невизначеності такі, як середнє значення, або фільтри Калмана, мають свої обмеження. Тому необхідно розробити нову методологію, яка ефективніше враховуватиме різноманітні фактори, що впливають на вимірювання. Невизначеність у вимірюваннях можна представити, як випадкову змінну $\boldsymbol{X}$ з певним розподілом ймовірностей. Нехай $\boldsymbol{\mu}$ – справжнє значення вимірюваної характеристики, а $\boldsymbol{\varepsilon}$ похибка вимірювання, тоді виміряне значення $\boldsymbol{X}$ можна записати, як вираз (1):

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{\varepsilon}, \tag{1}$$

де $\boldsymbol{\varepsilon}$ – випадкова змінна з нульовим середнім значенням та дисперсією $\boldsymbol{\sigma}^2$.

Для зменшення невизначеності можна використовувати методи математичної статистики: застосуємо «середнє значення»: використання середнього значення кількох вимірів для зменшення випадкових похибок у відповідності до виразу (2):

$$\hat{\boldsymbol{\mu}} = \frac{1}{\boldsymbol{n}} \sum_{i=1}^{n} \boldsymbol{X}_i, \tag{2}$$

де $\boldsymbol{X}_i$ – випадкова величина, , яка приймає значення на i –тому спостереженні в наборі даних; **n –** це загальна кількість вимірів (досліджень).

При виборі методів фільтрації в межах вирішення даної задачі згідно з [18] можна застосувати фільтри Калмана для динамічних систем, де вимірювання оновлюється в реальному часі (3):

$$\hat{\boldsymbol{x}}_{(\boldsymbol{k}|\boldsymbol{k})} = \hat{\boldsymbol{x}}_{(\boldsymbol{k}|\boldsymbol{k}-1)} + \boldsymbol{K}_{\boldsymbol{k}} \left(\boldsymbol{z}_{\boldsymbol{k}} - \boldsymbol{H}\hat{\boldsymbol{x}}_{(\boldsymbol{k}|\boldsymbol{k}-1)} \right), \tag{3}$$

де $\boldsymbol{K}_{\boldsymbol{k}}$ – коефіцієнт Калмана; $\boldsymbol{z}_{\boldsymbol{k}}$ – вимірювання; $\boldsymbol{H}$ – матриця спостереження.

В нашому випадку запропонуємо нове рішення, щодо піднятої проблеми, а саме застосування гібридної методології з використанням байєсового підходу та машинного навчання.

Байєсовий підхід дозволяє інтегрувати попередню інформацію (апріорне знання) з новими даними для покращення оцінки невизначеності. В даному контексті ми будемо використовувати байєсову інтерпретацію для кількісної оцінки якості програмного забезпечення (4):

$$\boldsymbol{P}(\boldsymbol{\theta}|\boldsymbol{D}) = \frac{\boldsymbol{P}(\boldsymbol{D}|\boldsymbol{\theta})\,\boldsymbol{P}(\boldsymbol{\theta})}{\boldsymbol{P}(\boldsymbol{D})}, \tag{4}$$

де $\boldsymbol{P}(\boldsymbol{\theta}|\boldsymbol{D})$ – апостеріорний розподіл параметрів $\boldsymbol{\theta}$ (характеристик якості) з урахуванням даних $\boldsymbol{D}$; $\boldsymbol{P}(\boldsymbol{D}|\boldsymbol{\theta})$ – ймовірність даних при заданих параметрах (правдоподібність); $\boldsymbol{P}(\boldsymbol{\theta})$ – апріорний розподіл параметрів; $\boldsymbol{P}(\boldsymbol{D})$ – маргінальна ймовірність даних. Відмітимо, що вибір апріорного розподілу відбувається на основі попередніх даних, або експертних знань; оновлення на основі нових даних відбувається із врахуванням ймовірності даних при заданих параметрах

(для цього застосовуємо використання нових вимірювань для оновлення розподілу параметрів); обчислення апостеріорного розподілу $\boldsymbol{P}(\boldsymbol{\theta}|\boldsymbol{D})$ передбачає використання байєсового правила для обчислення апостеріорного розподілу.

У відповідності до [15] машинне навчання може допомогти в автоматизації та покращенні процесу оцінки якості. Використання регресійних моделей та нейронних мереж дозволяє врахувати складні залежності між характеристиками програмного забезпечення та їх якістю.

Запропонований гібридний підхід поєднує байєсовий підхід з машинним навчанням для отримання більш точних і надійних оцінок. Це дозволяє враховувати апріорну інформацію та автоматично оновлювати оцінки на основі нових даних і складних залежностей. Етапи даного підходу включають:

- визначення апріорного розподілу параметрів на основі історичних даних, або експертних знань. Нехай параметр $\boldsymbol{\theta}$ (наприклад, надійність програмного забезпечення) має апріорний розподіл $\boldsymbol{P}(\boldsymbol{\theta})$. Тоді виконується умова, яка математично може бути наведена у формі виразу (5):

$$\boldsymbol{P}(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2\right), \quad (5)$$

де $\boldsymbol{\mu}_0$ та $\boldsymbol{\sigma}_0^2$ – апріорне середнє та дисперсія;

- навчання моделей машинного навчання передбачає збір даних та навчання моделей машинного навчання для прогнозування якості.

Відмітимо, що згідно з [14] збір даних про характеристики програмного забезпечення та вимірювання його надійності є ключовими етапами для аналізу якості програмного забезпечення. Дані можуть бути зібрані з різних джерел та за допомогою різних методів. В межах піднятого питання розглянемо основні способи збору даних і вимірювання надійності. Для початку розглянемо збір даних про характеристики програмного забезпечення. Згідно з [1] характеристики програмного забезпечення можуть включати різні метрики, які оцінюють різні аспекти програмного забезпечення. Наведемо основні категорії характеристик та методи їх збору.

1. Метрики коду:

- кількість рядків коду (SLOC) вимірюється за допомогою інструментів аналізу коду, таких, як cloc, або SLOCCount;

- кількість дефектів відстежується за допомогою систем управління дефектами, таких, як Jira, Bugzilla, або інші системи відстеження помилок;

- кількість комітів та змін у коді береться з систем контролю версій, таких, як Git. Інструменти аналізу, як наприклад GitStats, або Gource, можуть допомогти у зборі цих даних.

2. Метрики продуктивності:

- час виконання тестів вимірюється за допомогою автоматизованих систем тестування таких, як Jenkins, або Travis CI;

- споживання ресурсів (ЦП, пам'ять) збираються за допомогою моніторингових інструментів таких, як Prometheus, Nagios, або інші інструменти моніторингу продуктивності.

3. Метрики тестування:

- покриття тестів вимірюється за допомогою інструментів таких, як JaCoCo для Java, Coverage.py для Python, або Istanbul для JavaScript;

- кількість тестів, що пройшли/не пройшли береться з систем автоматизованого тестування.

4. Метрики якості коду:

- цикломатична складність вимірюється за допомогою статичних аналізаторів коду таких, як SonarQube, або Pylint;

- дотримання кодових стандартів: оцінюється за допомогою інструментів лінтингу таких, як ESLint для JavaScript, Pylint для Python, або Checkstyle для Java.

В межах піднятої спрямованості опишемо реалізацію вимірювання надійності програмного забезпечення. Згідно з [5] надійність програмного забезпечення, оцінюється, як здатність системи виконувати свої функції в певних умовах протягом певного періоду часу. Вона може бути виміряна різними способами.

1. Метрики надійності:

- MTTF (Mean Time To Failure): середній час до першого відмови. Вимірюється шляхом відстеження часу між запуском програми та її першою відмовою;

- MTBF (Mean Time Between Failures): середній час між відмовами. Вимірюється шляхом підрахунку середнього часу роботи між відмовами;

- MTTR (Mean Time To Repair): середній час відновлення після відмови. Вимірюється шляхом підрахунку середнього часу, необхідного для виправлення відмови.

2. Тестування надійності:

- стрес-тестування виконується для оцінки поведінки системи під навантаженням. Може бути реалізовано за допомогою інструментів таких, як Apache JMeter, або LoadRunner;

- тестування відмовостійкості виконується для оцінки здатності системи відновлюватися після відмов. Може бути реалізовано за допомогою Chaos Engineering інструментів таких, як Chaos Monkey від Netflix.

3. Аналітика та моніторинг:

- логи та трейсінг збираються та аналізуються за допомогою інструментів логування таких, як ELK Stack (Elasticsearch, Logstash, Kibana), або Splunk;

- моніторинг продуктивності виконується за допомогою систем моніторингу таких, як New Relic, Dynatrace, або Prometheus.

Таким чином для оцінки якості та надійності програмного забезпечення із врахуванням специфіки аспектів невизначеності оцінювання кількісних характеристик якості важливо використовувати різноманітні метрики та методи. Вимірювання характеристик коду, продуктивності, тестування та якості дозволяє отримати об'єктивну оцінку рівня програмного забезпечення. Крім того, використання метрик надійності та відповідних тестувань допомагає забезпечити стабільну та надійну роботу системи під час експлуатації.

Також відмітимо, що в нашому випадку при побудова моделі машинного навчання процес підготовки даних включав комплексну попередню обробку

даних (очистку даних, усунення пропусків та нормалізація характеристик) та розділення даних на навчальну і тестову вибірки (в нашому випадку було задано, 80% на навчання і 20% на тестування).

Комплексна обробка в нашому випадку включає в себе наступні етапи:

- завантаження даних: отримання даних з відповідних джерел, таких як бази даних, або файлові системи;

- очистка даних: видалення дублікатів, корегування помилкових значень, обробка відсутніх, або неправильних даних;

- нормалізація характеристик: приведення значень ознак до одного масштабу для забезпечення стабільності та швидкості збіжності алгоритмів машинного навчання;

- розподіл на навчальну і тестову вибірки: випадкове, або стратифіковане розподілення даних на навчальні та тестові вибірки для оцінювання ефективності моделі;

- крос-валідація: додаткова перевірка моделі шляхом розділення даних на кілька підвибірок для навчання та тестування;

- оновлення на основі нових даних: використання прогнозів моделей, як нових даних для оновлення апостеріорного розподілу. На практиці для цього доцільно застосувати байєсове оновлення, яке забезпечить покращення оцінки параметрів на основі нових даних.

В нашому випадку правдоподібність вимірювань має нормальний розподіл з середнім значенням, передбаченим моделлю лінійної регресії, та невідомою дисперсією. Для цього передбачається використання нових вимірювань для оновлення апостеріорного розподілу, яке задається у відповідності до формули (6):

$$\boldsymbol{P}(\boldsymbol{\theta}|\boldsymbol{D}) \propto \boldsymbol{P}(\boldsymbol{D}|\boldsymbol{\theta})\,\boldsymbol{P}(\boldsymbol{\theta}); \tag{6}$$

- комбінація результатів: комбінація результатів байєсового оновлення та моделей машинного навчання для отримання кінцевої оцінки якості. Даний процес значно підвищує точність прогнозів, використовуючи апріорну інформацію та нові дані для постійного вдосконалення оцінок.

Відмітимо, що у відповідності до [7] на практиці для реалізації попередньої обробки даних та розподілу їх на навчальну і тестову вибірки можна використовувати різні інструменти та бібліотеки, зазвичай в середовищі програмування, такому як Python. Для цього зазвичай застосовують наступні інструменти:

- бібліотеку Pandas: для очистки даних, усунення пропусків та нормалізації характеристик часто використовують бібліотеку Pandas. Вона надає широкі можливості для маніпулювання табличними даними такими, як видалення дублікатів, заповнення пропущених значень, обчислення статистик тощо;

- бібліотеку Scikit-learn: це одна з найпопулярніших бібліотек для машинного навчання в Python. Scikit-learn містить багато утиліт для розподілу даних на навчальну і тестову вибірки таких, як train_test_split;

- бібліотеку NumPy: NumPy може бути використаний для роботи з числовими даними та їх нормалізації;

- бібліотеки Matplotlib, або Seaborn: ці бібліотеки дозволяють візуалізувати дані перед та після обробки, що може допомогти зрозуміти їх розподіл та потенційні аномалії.

В нашому випадку зважаючи на універсальність запропонованого комплексного підходу передбачається застосовувалися всіх вище перерахованих бібліотек.

Математично процес навчання регресійної моделі, або нейронної мережі можна представити у вигляді виразу (7):

$$\boldsymbol{Y} = \boldsymbol{f}\left(\boldsymbol{X};\boldsymbol{\theta}\right). \tag{7}$$

В нашому випадку ми використаємо лінійну регресію для побудови моделі, яка передбачає надійність $\boldsymbol{Y}$ на основі характеристик $\boldsymbol{X_1}$, $\boldsymbol{X_2}$ і $\boldsymbol{X_3}$. Тоді формула лінійної регресії набуває вигляду враза (8):

$$\boldsymbol{Y} = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 \boldsymbol{X}_1 + \boldsymbol{\beta}_2 \boldsymbol{X}_2 + \boldsymbol{\beta}_3 \boldsymbol{X}_3. \tag{8}$$

Після навчання моделі, оцінюється її точність на тестовій вибірці за допомогою метрик таких, як R-квадрат, або середньоквадратична помилка.

В нашому випадку навчання моделі відбувається шляхом мінімізації функції втрат (в нашому випадку для цього була застосована метрика MSE. Згідно з [22] MSE – одна з найпоширеніших метрик для оцінки якості прогнозування моделей машинного навчання, зокрема у задачах регресії. Вона використовується для вимірювання середньої квадратичної помилки між фактичними і передбаченими значеннями). MSE математично можна виразити у відповідності формули (9):

$$\boldsymbol{MSE} = \frac{1}{\boldsymbol{n}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \left(\boldsymbol{Y_i} - \boldsymbol{Y_i}\right)^2, \tag{9}$$

де $\boldsymbol{n}$ – кількість спостережень, $\boldsymbol{Y_i}$ - фактичне значення надійності, $\boldsymbol{Y_i}$ – передбачене значення надійності.

На методологічному рівні підхід дозволяє враховувати невизначеність, адаптуючи моделі до нових умов і забезпечуючи надійніші та обґрунтовані прогнози. В результаті запропонований підхід має сприяти зменшенню ризиків і підвищенню якості програмних систем, забезпечуючи більш комплексний та інформований підхід до оцінки їхньої надійності.

Практичний приклад вирішення проблеми невизначеності у вимірюваннях та оцінках

Постановка задачі: нехай у нас є програмне забезпечення, для якого необхідно оцінити якість. У нас є дані про певні характеристики програмного забезпечення (наприклад, кількість дефектів, час виконання тестів, кількість рядків коду тощо). Необхідно оцінити надійність цього програмного забезпечення з урахуванням невизначеності у вимірюваннях. В межах вирішення практичного завдання передбачається побудова моделі, в якій буде враховуватися

невизначеність у вимірюваннях та реалізовуватися процес оцінювання надійності програмного забезпечення.

Початкові умови: в межах практичного прикладу при аналізі початкових даних в ході реалізації збору даних, щодо характеристики програмного забезпечення в межах формування оцінки його безпеки було отримано цілий ряд дослідних характеристик, які занесені (табл. 2).

Таблиця 2

Дослідні характеристики

Номер вимірювання	Кількість дефектів (X_1)	Час виконання тестів (X_2, с)	Кількість рядків коду (X_3)	Надійність (Y)
1	15	120	5000	0.85
2	20	150	5200	0.80
3	10	110	4800	0.90
4	25	160	5300	0.75
5	18	130	5100	0.82

Практична реалізація методології на основі даних із табл. 2.

1. Припустимо, що апріорний розподіл параметрів надійності $\boldsymbol{\theta}$ є нормальним з середнім значенням $\boldsymbol{\mu}_0$ **=0.8** та дисперсією $\boldsymbol{\sigma}_0^2 = 0.01$, тоді маємо (10):

$$\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{\mathcal{N}}(0.8, 0.01). \quad (10)$$

2. З використанням початкових даних, навчимо модель лінійної регресії. В, межах дослідження для наведеного прикладу, було отримано наступні коефіцієнти (11):

$$\boldsymbol{Y} = 0.5 + 0.02\,\boldsymbol{X}_1 - 0.001\,\boldsymbol{X}_2 + 0.0001\,\boldsymbol{X}_3. \quad (11)$$

3. Використовуємо модель для прогнозування надійності з врахуванням невизначеності. Нехай нові вимірювання мають наступні значення: Кількість дефектів X1= 17 штук; Час виконання тестів X2= 140 с; Кількість рядків коду X3=5150. Тоді використовуючи модель лінійної регресії, отримаємо прогноз $\boldsymbol{Y} = 1.215$ та $\boldsymbol{\sigma}^2 = 0,05$.

4. Байєсове оновлення: оновлюємо апостеріорний розподіл параметрів на основі нового вимірювання. Нехай правдоподібність нового вимірювання (12):

$$\boldsymbol{P}(\boldsymbol{\theta}|\boldsymbol{D}) = \boldsymbol{\mathcal{N}}(1.215, 0.05). \quad (12)$$

Апостеріорний розподіл розраховується у відповідності до формули (9). Обчислюємо нове середнє значення згідно формули (13) та дисперсію згідно формули (14):

$$\boldsymbol{\mu}_{posterior} = \frac{\frac{\boldsymbol{\mu}_0}{\boldsymbol{\sigma}_0^2} + \frac{\hat{\boldsymbol{Y}}}{\boldsymbol{\sigma}^2}}{\frac{1}{\boldsymbol{\sigma}_0^2} + \frac{1}{\boldsymbol{\sigma}^2}}, \quad (13)$$

$$\boldsymbol{\sigma}^2_{posterior} = \left(\frac{1}{\boldsymbol{\sigma}_0^2} + \frac{1}{\boldsymbol{\sigma}^2}\right)^{-1}. \quad (14)$$

У відповідності до вище наведених формул отримуємо, для нашого практичного прикладу отримуємо: $\boldsymbol{\mu}_{posterior} \approx 0.869$; $\boldsymbol{\sigma}^2_{posterior} \approx 0.0083$.

Тоді кінцева оцінка надійності $\boldsymbol{Y}_{final}$ з врахуванням невизначеності може розрахуватися відповідності до формули (15):

$$\boldsymbol{Y}_{final} = \boldsymbol{\mu}_{posterior} \pm \boldsymbol{\sigma}^2_{posterior}. \quad (15)$$

Відповідно (15) для розглянутого практичного прикладу маємо: $\boldsymbol{Y}_{final} = 0.869 \pm 0.0083$.

Це означає, що для розглянутого прикладу з високою ймовірністю надійність програмного забезпечення в знаходиться в межах від 0.8607 до 0.8773 %.

Обговорення

Запропонований в статті методологічний підхід вирішення проблеми усунення невизначеності у вимірюваннях та кількісних оцінках якості програмних систем - досить перспективним в межах підвищення точності оцінок якості та частково підтверджується в досить численній кількості робіт. Зокрема дана позиція підтверджується тим, що у відповідності до праці [15] більш точні методики вимірювання та аналізу допомагають краще прогнозувати поведінку програмних систем, зменшуючи ризики та підвищуючи якість. При порівнянні запропонованого підходу варто відмітити, що байєсівський підхід у поєднанні з машинним навчанням досить кардинально відрізняється від інших підходів зважаючи, на що він може бути більш ефективним для вирішення цієї проблеми.

У відповідності до [6] врахування апріорної інформації: використання апріорного розподілу параметрів (наприклад, надійності) дозволяє почати з певним рівнем впевненості, який базується на попередніх знаннях.

У відповідності до [12] вагоме місце має відводитися оновленню на основі нових даних: відповідно цій позиції запропоноване рішення передбачає, що застосування байєсівського підходу дозволить оновлювати наші оцінки параметрів на основі нових вимірювань, що постійно покращує точність моделі. Згідно з [13] дана тенденція пов'язана із тим, що важливе місце відводиться саме адаптивності до нових даних: байєсівський підхід постійно оновлює апріорні ймовірності на основі нових даних, що надходять. Це дозволяє моделі адаптуватися до змін та нових інформаційних потоків більш ефективно, ніж традиційні методи, які часто базуються виключно на поточних даних.

Також варто відмітити інтерпретацію результатів: згідно з [6] байєсівські моделі забезпечують більш інтуїтивне розуміння невизначеності та ризиків, оскільки результати представлені у вигляді ймовірнісних розподілів. Це дозволяє отримати більш детальну інформацію про можливі сценарії та їх ймовірності, що корисно для прийняття рішень у умовах невизначеності. В праці [22] зазначено, що постійне оновлення даних допомагає зменшити невизначеність і підвищити точність оцінок якості.

В праці [13] відмічається важливість врахування гнучкості моделювання згідно з поглядами авторів

даної праці байєсівські методи легко інтегруються з різноманітними типами даних та моделями, включаючи складні та багаторівневі структури. Це дозволяє застосовувати байєсівський підхід у широкому спектрі задач машинного навчання, від класифікації до регресії і кластеризації.

В межах оцінки невизначеності в межах запропонованого підходу підкреслюється здатність вирішення проблем із врахуванням невизначеності. Зокрема варто відмітити, що згідно з працею [16] байєсівський підхід оцінює не лише середнє значення параметрів, але й їхню невизначеність, що дозволяє робити більш обґрунтовані прогнози. Перевагою даної методології відповідно стає підтримка кращої інформованість моделі. Згідно з [17] моделі, які враховують невизначеність, надають більш повну інформацію для прийняття рішень, що дозволяє краще оцінювати ризики і планувати дії.

Згідно з [13] також варто підкреслити здатність до прогнозування кількісної оцінки якості з довірчими інтервалами: Оцінка невизначеності дозволяє краще розуміти можливі варіації в прогнозах і приймати більш обґрунтовані рішення щодо якості програмних систем.

В межах піднятого дискусійного огляду також відмітимо здатність запропонованого підходу до інтеграції з машинним навчанням. Відповідно [9] моделі машинного навчання можуть обробляти великі обсяги даних і виявляти складні патерни, що підвищує точність прогнозів. Згідно з [8] машинне навчання дозволяє моделі адаптуватися до нових умов і масштабуватися під великі обсяги даних, забезпечуючи ефективність на різних етапах життєвого циклу програмного забезпечення. На методологічному рівні порівнюючи запропонований підхід із традиційним статистичний підходом, який розглянутий в праці [12] варто відмітити, що у традиційних підходів простежується суттєвий недолік: менша гнучкість у врахуванні нових даних і апріорної інформації; точкові оцінки без оцінки невизначеності. Тоді, як зазначалося вище для байєсівського підходу, характерна краща адаптація до нових даних, врахування апріорної інформації та невизначеності.

В праці [10] пропонують застосування фреквентістського підходу, проте в ньому є суттєвий недолік – параметри розглядаються, як фіксовані, а не випадкові змінні, тобто відсутність механізму для врахування апріорної інформації. Тоді, як при застосуванні байєсівського підходу існує можливість враховувати апріорну інформацію і постійно оновлювати оцінки на основі нових даних.

В праці [4] пропонують застосовувати машинне навчання без байєсових методів. Проте даний підхід також має недоліки, а саме: Відсутність механізму для оновлення моделей на основі нових даних; ігнорування невизначеності в прогнозах. Тоді, як у відповідності до [11] для байєсівського підходу характерним є постійне оновлення моделей, врахування невизначеності, що робить прогнози більш обґрунтованими.

В праці [7] вказується, що для піднятої задачі можна застосовувати ансамблеві методи. Проте згідно з дослідженням, яке висвітлено в праці [3] для даного підходу існує суттєвий недолік, який полягає в відсутності механізму для врахування апріорної інформації; менше уваги до оцінки невизначеності. Натомість в запропонованому нашому рішенні, як зазначалося вище при використанні байєсівського підходу передбачається використання апріорної інформації, постійне оновлення моделей і оцінка невизначеності [13].

В межах оцінки невизначеності модель враховує невизначеність, як у початкових даних так і в нових вимірюваннях, що дає можливість робити більш обґрунтовані прогнози. Таким чином, байєсівський підхід у поєднанні з машинним навчанням має значні переваги перед іншими підходами завдяки врахуванню апріорної інформації, адаптивності до нових даних, інтерпретації результатів та гнучкості моделювання.

Висновки. Одним із ключових аспектів при оцінці програмних систем є їх якість, яка визначається різноманітними кількісними характеристиками такими, як продуктивність, надійність, швидкодія та інші. Проте, оцінка якості програмних систем часто стикається з проблемою невизначеності. Невизначеність оцінювання кількісних характеристик якості відображає рівень неоднозначності, або варіації у визначенні цих характеристик.

Запропонована методологія забезпечує інтеграцію байєсового підходу та машинного навчання для ефективного врахування невизначеності у вимірюваннях та оцінках якості програмного забезпечення. Вона дозволяє поєднувати апріорну інформацію з новими даними, автоматично оновлювати оцінки та враховувати складні залежності між характеристиками програмного забезпечення. Байєсівський підхід у поєднанні з машинним навчанням забезпечує більш точні методики вимірювання та аналізу, що допомагає краще прогнозувати поведінку програмних систем, зменшуючи ризики та підвищуючи якість. Цей підхід кардинально відрізняється від інших завдяки врахуванню апріорної інформації, можливості постійного оновлення оцінок на основі нових даних і оцінці невизначеності, що робить його більш ефективним для вирішення проблеми невизначеності у вимірюваннях та кількісних оцінках якості програмних систем. На основі проведеного аналізу встановлено, що запропонований в статті методологічний підхід вирішення проблеми усунення невизначеності у вимірюваннях та кількісних оцінках якості програмних систем є перспективним в межах підвищення точності оцінок якості та частково підтверджується в досить численній кількості робіт.

Список літератури

[1]. Areces, C., Fervari, R., Saravia, A. R., & Velázquez-Quesada, F. R. (2023). Uncertainty-based knowing how logic. Journal of Logic and Computation. https://doi.org/10.1093/logcom/exad056.

[2]. Behera, D. (2023). Alternative methodology for epistemic uncertainty-based linear programming problem. Soft Computing. https://doi.org/10.1007/s00500-023-08725-5.

[3]. Bougeret, M., Pessoa, A., & Poss, M. (2023). Single machine robust scheduling with budgeted uncertainty. Operations Research Letters. https://doi. org/10.1016/j.orl.2023.01.007.

[4]. Boukhelifa, N., Johnson, C. R., & Potter, K. (2023). Visualization and decision-making design under uncertainty. IEEE Computer Graphics and Applications, 43(5), pp. 23–25. https: // doi.org / 10.1109/mcg.2023. 3302172.

[5]. Cappelli, V., Cerreia-Vioglio, S., Maccheroni, F., Marinacci, M., & Minardi, S. (2020). Sources of uncertainty and subjective prices. Journal of the European Economic Association. https: // doi.org /10.1093/jeea/jvaa013.

[6]. Chandra Yadav, D., Singh, Y., Kumar Pandey, A., & Kannagi, A. (2024). Computerized software quality evaluation with novel artificial intelligence approach. Proceedings on Engineering Sciences, 5(4), pp. 363-372. https://doi.org/10.24874/pes.si.24.02.019.

[7]. Clarté, L., Loureiro, B., Krzakala, F., & Zdeborova, L. (2023). Theoretical characterization of uncertainty in high-dimensional linear classification. Machine Learning: Science and Technology. https: //doi.org/10. 1088/2632-2153/acd749.

[8]. Guaman, D. S., Alamo, J. M. D., & Caiza, J. C. (2020). A systematic mapping study on software quality control techniques for assessing privacy in information systems. IEEE Access, 8, pp. 74808-74833. https:// doi. org/10.1109/access.2020.2988408.

[9]. Hahn, S.-J., & Lee, B.-H. (2023). Quality evaluation to small scaled software implementation result. The Journal of Korean Institute of Information Technology, 21(1), pp. 1-10. https: // doi.org / 10.14801 / jkiit. 2023. 21.1.1.

[10]. Hayashi, S., Kubo, M., Mori, H., & Nakamura, M. (2023). Significance of quantitative evaluation and assessment using automated volumetric breast density measurements software (Volpara Density). Nihon Nyugan Kenshin Gakkaishi (Journal of Japan Association of Breast Cancer Screening), 32(1), pp. 63-65. https://doi. org/10.3804/jjabcs.32.63.

[11]. Humphrey, W. S., & Singpurwalla, N. D. (1998). A bayesian approach for assessing software quality and productivity. International Journal of Reliability, Quality and Safety Engineering, 05(02), pp. 195-209. https: //doi.org/10.1142/s0218539398000194.

[12]. Indi, M. M., Priyangan, D. M., Herdiani, F. D., Budiman, B., & Mose, Y. (2023). Evaluation of the effectiveness of technology-based project management systems for software development. Global International Journal of Innovative Research, 1(2), pp. 175-181. https:// doi.org/10.59613/global.v1i2.30.

[13]. Kamaletdinov, S., Aripov, N., Khudayberganov, S., Bashirova, A. M., & Akhmedov, M. D. (2023). Evaluation of data quality based on Bayesian networks in railway rolling stock monitoring systems. E3S Web of Conferences, 460, 04014. https: // doi.org / 10.1051/ e3sconf/202346004014.

[14]. Kuetche, F., Noura, A., Ntsama, P. E., Welba, C., & Thierry, S. (2023). Signal quality indices evaluation for robust ECG signal quality assessment systems. Biomedical Physics & Engineering Express. https: //doi.org/ 10.1088/2057-1976/ace9e0.

[15]. Kusuma, M. R., Windu Gata, Sigit Kurniawan, Dedi Dwi Saputra & Supriadi Panggabean. (2023). Software defect prediction for quality evaluation using learning techniques ensemble stacking. Inspiration: Jurnal Teknologi Informasi Dan Komunikasi, 13(2), pp. 1-13. https://doi.org/10.35585/inspir.v13i2.58.

[16]. Liu, C. (2021). Quantitative evaluation of software component behavior discovery approach. IEICE Transactions on Information and Systems, E104.D(1), pp. 117-120. https://doi.org/10.1587/transinf.2020mpl0001.

[17]. Merzlyakova, E. Y., & Yanchenko, E. V. (2023). Review of software quality verification and evaluation methods. The Herald of the Siberian State University of Telecommunications and Informatics, 17(1), pp. 92-106. https://doi.org/10.55648/1998-6920-2023-17-1-92-106.

[18]. Pankov, P. S., & Tagaeva, S. B. (2020). Systems of differential equations and computer phenomena. Herald of Institute Mathematics of the National Academy of Sciences of the Kyrgyz Republic, (2), pp. 86-93. https:// doi.org/10.52448/16948173_2020_2_86.

[19]. Yoon, H. (2023). A quantitative evaluation for usability under software quality models. International Journal on Recent and Innovation Trends in Computing and Communication, 11(3), pp. 24-29. https: //doi.org /10.17762/ijritcc.v11i3.6194.

[20]. Yue, C., Huang, R., Towey, D., Xian, Z., & Wu, G. (2023). An entropy-based group decision-making approach for software quality evaluation. Expert Systems with Applications, 121979. https: // doi.org /10.1016/j. eswa.2023.121979.

[21]. Zhang, X., Li, Z., Zou, Z., Gao, X., Xiong, Y., Jin, D., Li, J., & Liu, H. (2023). Informative data selection with uncertainty for multimodal object detection. IEEE Transactions on Neural Networks and Learning Systems, 1–13. https://doi.org/10.1109/tnnls.2023.3270159.

[22]. Zhao, J., Wang, Y., Mancenido, M. V., Chiou, E. K., & Maciejewski, R. (2023). Evaluating the impact of uncertainty visualization on model reliance. IEEE Transactions on Visualization and Computer Graphics, pp. 1-15. https://doi.org/10.1109/tvcg.2023.3251950.

УДК 004.422.4:005.3:004.4

Shantyr A., Zinchenko O., Chychkarov E., Vyshnivskyi A. Uncertainty in evaluating quantitative quality characteristics of software

Abstract. *Currently, software quality evaluation is a crucial stage in the processes of software development and implementation. It provides developers with the opportunity to obtain an objective assessment of the developed software products and determine their compliance with existing international standards and software quality evaluation requirements. However, this process is often accompanied by a certain level of uncertainty in evaluating the quantitative quality characteristics, which can complicate decision-making regarding the prospects for the use and safety of the developed product. Significant contributions to the theoretical and practical aspects of generalizing the issue of uncertainty in evaluating quantitative quality characteristics have been made by scholars such as S. Hayashi, M. Kubo, H. Mori, C. Areces, R. Fervari, A. Saravia, F. Velázquez-Quesada, S. Guaman, J. Alamo, J. Caiza, M. Nakamura, and others. The purpose of this article is to address the problem associated with the uncertainty in evaluating the quantitative quality characteristics of software systems. To achieve this goal, the article sets and solves the following tasks:*

examining various aspects of uncertainty in evaluating the quantitative quality characteristics of computer software systems; developing a methodological approach to solving the problem of uncertainty in evaluating quantitative quality characteristics; and conducting a practical study of the developed approach. The methods used to solve these tasks include analysis, synthesis, generalization, and comparison.

Keywords: *software systems quality, posterior distribution, quality standards, critical sources of uncertainty, Bayesian methods, Bayesian updating, hybrid approach.*

Шантир Антон Сергійович, кандидат технічних наук, доцент кафедри штучного інтелекту Державного університету інформаційно-комунікаційних технологій.
Anton Shantyr, Ph.D., Associate Professor of the Department of artificial intelligence of the State University of Information and Communication Technologies.

Зінченко Ольга Валеріївна, доктор технічних наук, доцент, завідувач кафедри штучного інтелекту Державного університету інформаційно-комунікаційних технологій.
Olha Zinchenko, Dc.S, Associate Professor, Head of the Department of artificial intelligence of the State University of Information and Communication Technologies.

Чичкарьов Євген Анатолійович, доктор технічних наук, професор, професор кафедри штучного інтелекту Державного університету інформаційно-комунікаційних технологій.
Yevhen Chychkarov, Dc.S, Professor, Professor of the Department of artificial intelligence of the State University of Information and Communication Technologies.

Вишнівський Олександр Вікторович аспірант кафедри комп'ютерних наук Державного університету інформаційно-комунікаційних технологій.
Oleksandr Vyshnivskyi, Dc.S, postgraduate stadent of the Department of Computer Science of the State University of Information and Communication Technologies.

Отримано *19 травня 2024 року,* затверджено редколегією *26 червня 2024 року*