Shantyr, A. S. (2024). Identification of sources of uncertainty in the evaluation of software systems quality. *Actual Issues of Modern Science. European Scientific e-Journal, 31*, __-__. Ostrava: Tuculart Edition, European Institute for Innovation Development. (In Ukrainian)

DOI: 10.47451/inn2024-05-02

The paper is published in Crossref, ICI Copernicus, BASE, Zenodo, OpenAIRE, LORY, Academic Resource Index ResearchBib, J-Gate, ISI International Scientific Indexing, ADL, JournalsPedia, Scilit, EBSCO, Mendeley, and WebArchive databases.

Anton S. Shantyr, Candidate of Technical Sciences, Associate Professor, Department of Artificial Intelligence, State University of Information and Communication Technologies. Kyiv, Ukraine.

Identification of sources of uncertainty in the evaluation of software systems quality

Abstract: Today, software systems play a pivotal role in various aspects of life, with increasing complexity and diversity. However, along with this comes a growing number of challenges related to the quality and reliability of these systems. The need for assessing the quality of software products requires adherence to high accuracy and reliability through various methodological approaches. One of the main problems in this context is the insufficient substantiation of theoretical and methodological approaches in determining sources of uncertainty. This problem requires a comprehensive approach and systematization to address the tasks of evaluating the quality of modern software systems considering uncertainty. Significant contributions to the theoretical and practical aspects of development regarding the generalization and systematization of sources of uncertainty in the development and operation of modern software systems have been made by scholars such as C. Areces, R. Fervari, A. Saravia, F. Velázquez-Quesada, M. Bougeret, A. Pessoa, M. Poss, N. Boukhelifa, C. Johnson, K. Potter, L. Clarté, B. Loureiro, F. Krzakala, L. Zdeborova, D. Tsapetis, M. Shields, D. Giovanis, A. Olivier, et al. The study object is methodological approaches to determining sources of uncertainty. The study's purpose is to generalize the systematization of sources of uncertainty in the development and operation of modern software systems. To achieve the purpose, the following tasks are set and solved in the article: on a comprehensive level, the main sources of uncertainty in software systems are analyzed, such as changes in requirements, design flaws, unforeseen operating conditions, and other factors; a comprehensive model for optimizing the problem of formalizing uncertainty in software requirements is developed based on the application of machine learning methods and data analysis. In the process of this comprehensive research, methods of analysis, synthesis, generalization, and comparison are used. The author concludes that as a result of using methods' combination in the proposed development, it provides a more efficient and objective approach to managing the uncertainty of software requirements, compared to some existing approaches. This allows you to increase the reliability, quality, and efficiency of the developed software product.

Keywords: software system quality, mathematical framework, uncertainty optimization, formalization of sources of uncertainty, identification of sources of uncertainty, critical sources of uncertainty.

Антон Сергійович Шантир, кандидат технічних наук, доцент, кафедра штучного інтелекту, Державний університет інформаційно-комунікаційних технологій. Київ, Україна.

УДК 004.422.4:005.3:004.4

Визначення джерел невизначеності при оцінці якості програмних систем

Анотація: Нині програмні системи відіграють ключову роль у багатьох аспектах життя, зростає їхня складність та різноманітність. Однак, разом із цим збільшується і число викликів, пов'язаних з якістю та надійністю цих систем. Необхідність в оцінці якості програмних продуктів вимагає дотримання високої точності та достовірності за допомогою різноманітних методологічних підходів. Однією з основних проблем у цьому контексті є недостатня обґрунтованість теоретико-методичних підходів у визначенні джерел невизначеності. Ця проблема потребує комплексного підходу та систематизації для вирішення завдань оцінки якості сучасних програмних систем з урахуванням невизначеності. Суттєвий внесок у розробку теоретичних й практичних аспектів, щодо узагальнення систематизації джерел невизначеності при розробці та експлуатації сучасних програмних систем внесли такі вчені, як: C. Areces, R. Fervari, A. Saravia, F. Velázquez-Quesada, M. Bougeret, A. Pessoa, M. Poss, N. Boukhelifa, C. Johnson, K. Potter, L. Clarté, B. Loureiro, F. Krzakala, L. Zdeborova, D. Tsapetis, M. Shields, D. Giovanis, A. Olivier et al. Об'єктом дослідження є методологічні підходи до визначення джерел невизначеності. Метою цієї статті є узагальнення систематизації джерел невизначеності при розробці та експлуатації сучасних програмних систем. Для реалізації мети в статті поставлені і вирішені такі завдання: На комплексному рівні проаналізовано основні джерела невизначеності в програмних системах, такі, як зміни у вимогах, недоліки в проектуванні, непередбачувані умови експлуатації та інші фактори; розроблена комплексна модель оптимізації проблеми формалізації невизначеності у вимогах до програмного забезпечення на базі застосування методів машинного навчання та аналізу даних. В процесі даного комплексного дослідження використано методи аналізу, синтезу, узагальнення, порівняння. Автор приходить до висновку що в результаті використання комбінації методів у запропонованій розробці забезпечує більш ефективний та об'єктивний підхід до управління невизначеністю вимог до програмного забезпечення, порівняно з деякими існуючими підходами. Це дозволяє підвищити надійність, якість та ефективність розроблюваного програмного продукту.

Ключові слова: якість програмних систем, математичний апарат, оптимізація невизначеності, формалізація джерел невизначеності, ідентифікація джерел невизначеності, критичні джерела невизначеності.

Вступ

В сучасному світі програмні системи стають все більш складними та різноманітними, відіграючи ключову роль у багатьох сферах нашого життя. Однак, разом із зростанням їхньої складності, з'являються нові виклики, пов'язані з їх якістю та надійністю. Зокрема враховуючи сучасні особливості та вимоги, які вказуються в різних профільних стандартах якості (ISO 9126, ISO/IEC 25010, IEEE 730 та ін.), котрі в свою чергу охоплюють різносторонні аспекти механізмів оцінюванні якості програмних систем окреме місце заслуговують питання, що пов'язані із вирішенням фундаментальних проблем в спектрі комплексного дослідження методичних підходів визначення джерел невизначеності. Дана тенденція пов'язана із тим, що в сучасних стандартах із регулювання питань, щодо оцінки якості програмних систем вимагається дотримання високої точності та достовірності. Об'єктом дослідження є методологічні підходи до визначення джерел невизначеності. Метою цієї статті є узагальнення систематизації джерел невизначеності при розробці та експлуатації сучасних програмних систем. Для реалізації мети в статті поставлені і вирішені такі завдання:

- на комплексному рівні проаналізовано основні джерела невизначеності в програмних системах, такі, як зміни у вимогах, недоліки в проектуванні, непередбачувані умови експлуатації та інші фактори;
- розроблена комплексна модель оптимізації проблеми формалізації невизначеності у вимогах до програмного забезпечення на базі застосування методів машинного навчання та аналізу даних.

В процесі даного комплексного дослідження використано методи аналізу, синтезу, узагальнення, порівняння.

Опрацьовано наукові роботи C. Areces, R. Fervari, A. Saravia та F. Velázquez-Quesada (*2023*) проаналізовано логіку на основі невизначеності в процесі здійснення дій; D. Behera (*2023*) розглянуто альтернативні методології для розв'язання задач лінійного програмування з епістемічною невизначеністю; M. Bougeret, A. Pessoa та M. Poss (*2023*) розглянуто задачі робустного планування з обмеженою невизначеністю для однієї машини; N. Boukhelifa, C. Johnson та K. Potter (*2023*) досліджено візуалізацію та прийняття рішень при проектуванні в умовах невизначеності; V. Cappelli, S. Cerreia-Vioglio, F. Maccheroni, M. Marinacci та S. Minardi (*2020*) розглянуто джерела невизначеності та особливості формування суб'єктивних цін при розробці програмних систем; L. Clarté, B. Loureiro, F. Krzakala, & L. Zdeborova, (*2023*) розглянута теоретична характеристиа невизначеності в високо-вимірній лінійній класифікації; J. Dorn, S. Apel та N. Siegmund (*Dorn et al., 2023*). розглянуто особливості управління невизначеністю у вимірах продуктивності конфігурованих програмних систем; A. Drozhkin (*2022*) проаналізовано сучасні підходи до оцінки якості програмних продуктів в умовах невизначеності; D. Tsapetis, M. Shields, D. Giovanis, A. Olivier інші (*2023*) розглянуто UQpy v4.1: бібліотеку для квантифікації невизначеності з використанням Python; X. Zhang, Z. Li та інші (*2023*) розглянуто особливості вибору інформативних даних з урахуванням невизначеності для багатомодального виявлення об'єктів; J. Zhao, Y. Wang та інші (*2023*) досліджено вплив візуалізації невизначеності на надійність моделі; M. Liao, D. Shen та P. Lv (*2023*). розглянуто єдину модель невизначеності даних та невизначеності взаємозв'язків даних; та інші.

Незважаючи на суттєві напрацювання вчених залишаються недостатньо обґрунтованими теоретико-методичні підходи визначення джерел невизначеності. Таким чином, комплексний розгляд піднятої тематичної спрямованості є необхідним в вирішенні завдань із оцінки якості сучасних програмних систем із врахуванням вирішення задач в межах тематики визначення джерел невизначеності визначення джерел невизначеності.

Результати дослідження

Математичний апарат та методологічні особливості класифікації джерел невизначеності

В оцінці якості комп'ютерних програмних систем велике значення відводиться правильному визначенню їх кількісних характеристик. Це стає ключовим етапом, як у процесі розробки, так і у підтримці та покращенні програмного забезпечення. Методи

оцінювання цих характеристик відіграють важливу роль у забезпеченні якості продукту та визначенні його відповідності вимогам користувачів. Піднімаючи дану спрямованість зазначимо, що на технічно-практичному рівні – оцінка якості програмних систем є досить складним завданням, оскільки вона часто пов'язана з невизначеністю. Навіть найбільш досконалі методи тестування та аналізу не можуть повністю виключити можливість помилок чи несправностей у програмному коді. Натомість поняття невизначеності у контексті оцінки якості програмних систем охоплює різноманітні аспекти, включаючи непередбачуваність поведінки програми в різних умовах експлуатації, складність аналізу великих обсягів даних, а також неповноту специфікацій та вимог до системи.

Зважаючи на вище відмічене в межах систематизації джерела невизначеності при оцінці якості комп'ютерних програмних систем доцільно відмітити наступні факторіальні першопричині джерела:

- Неоднозначні вимоги: Часто вимоги до програмного забезпечення формулюються неоднозначно, або неповністю. Це може створювати невизначеність в тому, як оцінювати відповідність програми цим вимогам;
- Ризик змін вимог: Вимоги до програмного забезпечення можуть змінюватися протягом розробки, або після випуску продукту. Ці зміни можуть впливати на якість програми та вимагати переоцінки її характеристик;
- Неочікувані зовнішні впливи: Зовнішні фактори такі, як зміни в інфраструктурі, конкурентний тиск, або нові регуляції, можуть впливати на якість програмного забезпечення, створюючи невизначеність в оцінці його характеристик;
- Помилки в розробці та випробуванні: Недоліки у процесі розробки та випробування можуть призводити до появи помилок та невірного функціонування програмного забезпечення, що ускладнює оцінку його якості;
- Варіація у використанні: Реальне використання програмного забезпечення може відрізнятися від очікувань розробників, що призводить до невизначеності в оцінці його ефективності та якості;
- Недоліки у тестуванні: Не ідеальність тестування може призводити до пропуску деяких дефектів, або недостатнього оцінювання якості програмного забезпечення.

Відмітимо, що наведені причини можуть створювати нестабільність і невизначеність у процесі розробки, тестування та використання програмного продукту. Їх важливо враховувати при плануванні, розробці та оцінці якості програмного забезпечення для забезпечення найвищого рівня задоволення користувачів і виконання бізнес-потреб.

Зважаючи на наведені проблеми, розробники програмного забезпечення повинні удосконалювати свої підходи до управління проектами та контролю якості.

Розглядаючи підходи, які можуть допомогти зменшити невизначеність у процесі розробки та оцінки якості програмного забезпечення варто відмітити, що досить вагоме місце має відводитися чіткій постановці вимог і комунікація.

Зважаючи на це в практичній реалізації важливо встановити чіткі та однозначні вимоги до програмного продукту ще на початку проекту. Натомість у відповідності до (*Dorn et al., 2023*) комунікація між розробниками, клієнтами та іншими зацікавленими сторонами повинна бути ефективною та систематичною. Даний підхід потребує

врахування гнучкості та адаптивність: зважаючи на ризик змін вимог, команди розробників повинні працювати у гнучкому середовищі, яке дозволяє швидко реагувати на зміни та адаптувати продукт під нові вимоги.

У комплексі практичної реалізації оцінювання якості програмних систем вимагає постійного формування вимог до тестування і контролю якості: Підвищення якості програмного забезпечення може бути досягнуто шляхом вдосконалення процесів тестування та контролю якості. Відмітимо, що автоматизоване тестування, використання тестових скриптів та інші методи на практиці застосовуються із цілю виявлення різноманітних помилок на ранніх етапах розробки.

Також на всіх етапах розробки програмного забезпечення вагоме місце має відводитися стратегії управління ризиками: Це включає ідентифікацію потенційних загроз для проекту, їх оцінку та планування заходів для їх запобігання, або зменшення наслідків.

Відповідно (*Areces et al., 2023*) для виключення невизначеності необхідно забезпечити механізми інформативного надходження фідбеків від користувачів: Постійний зворотний зв'язок від користувачів допомагає виявляти проблеми та недоліки програмного забезпечення, що дозволяє швидко вносити виправлення та покращення.

Враховуючи дані підходи на комплексному рівні формуються засади, щодо постійного вдосконалення процесів, які пов'язані із визначенням джерел невизначеності при оцінці якості програмних систем.

В межах комплексного розгляду визначенням джерел невизначеності розглянемо більш детально аспектні засади постійного вдосконалення процесів оцінки якості програмних систем, які пов'язані з визначенням джерел невизначеності, а саме:

- аналіз ідентифікації джерел невизначеності;
- оцінку впливу на якість.

Аналіз ідентифікації джерел невизначеності: Першим кроком є аналіз ідентифікації можливих джерел невизначеності в процесі оцінки якості програмних систем. На практиці реалізація даного аспекту може включати аналіз вимог, змін вимог, ризики, недоліки в тестуванні та інші фактори, які можуть впливати на якість програмного продукту.

На узагальненому рівні зв'язок математичного аналізу джерел невизначеності з оцінкою якості програмних систем полягає у тому, що ідентифікація та управління цими джерелами дозволяє краще розуміти та прогнозувати якість програмного забезпечення. Оцінка якості програмних систем включає в себе ряд параметрів таких, як функціональність, надійність, продуктивність, зручність використання та інші. Джерела невизначеності можуть впливати на ці параметри, тому їх аналіз допомагає покращити загальну якість програмного продукту.

Комплексна методологія, яка поєднує математичний аналіз джерел невизначеності з оцінкою якості програмних систем має включати в себе механізм ідентифікації джерел невизначеності. Згідно (*Behera, 2023*) в загальному випадку аналіз можливих джерел невизначеності в процесі розробки та експлуатації програмного продукту, може включати аналіз вимог, ризиків, недоліків у тестуванні та інших факторів. Зважаючи на важливість

математичного підходу, розглянемо модель механізму ідентифікації джерел невизначеності з використанням формул та операторів.

На першому етапі виконується аналіз вимог. Математично зазначений процес можна описати наступним чином: «Нехай R – множина всіх вимог до програмного продукту. Тоді можна визначити джерела невизначеності, як вимоги, які є неоднозначними, або неповними згідно формули (1):

$$U_{req} = \{r_i \in R | r_i\}, \quad (1)$$

де:

r_i – неоднозначна, або неповна вимога.

У відповідності до (*Zhao et al., 2023*) оцінка та аналіз вимог до програмного продукту може допомогти виявити потенційні невизначеності, такі, як неоднозначність вимог, або неповнота опису.

На другому етапі виконується аналіз ризиків, який охоплює процес визначення потенційних ризиків, які можуть виникнути під час розробки, або в експлуатації програмного продукту. Даний процес може включати ризики змін вимог, технічні ризики, ризики зв'язані з ресурсами та інші види ризиків. Математично зазначений процес можна описати наступним чином: «Нехай R –. множина всіх можливих ризиків, що можуть вплинути на якість програмного продукту. Тоді можна визначити джерела невизначеності, як ризики, які мають значний вплив на якість (2):

$$U_{req} = \{z_i \in R | z_i\}, \quad (2)$$

де:

z_i – ризиком, який впливає на якість.

На третьому етапі виконується реалізація аналізу історичних даних: в даний аналіз входить розгляд історичних даних з попередніх проектів, або подібних програмних продуктів для виявлення типових джерел невизначеності. Під час аналізу історичних даних згідно (*Boukhelifa et al., 2023*) можна використовувати статистичні показники для виявлення типових джерел невизначеності. Наприклад, U_{hist} – множина джерел невизначеності, що виявлені під час попередніх проектів, то можна обчислити ймовірність появи кожного джерела невизначеності у відповідності до (3):

$$P(\ U_{hist}) = \frac{|U_{hist}|}{n}, \quad (3)$$

де:

n – кількість проектів.

На четвертому етапі виконується експертна оцінка, реалізація котрої передбачає відповідне залучення експертів у галузі розробки програмного забезпечення для виявлення потенційних джерел невизначеності, на які може впливати їх досвід та знання. В узагальненому випадку у відповідності до (*Bougeret et al., 2023*) експертна оцінка може бути виконана за допомогою експертних знань та досвіду. Нехай E – множина всіх експертних оцінок джерел невизначеності. Тоді можна визначити середню експертну оцінку для кожного джерела (4):

$$\overline{U_{exp}} = \frac{\sum_{i=1}^{m} U_i}{m}, \quad (4)$$

де:

m – кількість експертних оцінок.

В межах аналізу даних, які отримують в наслідок застосування формул (1-4) та визначення загальної ймовірності виникнення джерел невизначеності можна використовувати різні оператори такі, як логічні оператори, оператори порівняння, арифметичні оператори тощо. Наприклад, для визначення загальної ймовірності виникнення джерел невизначеності можна використовувати формулу (5):

$$P(U_{total}) = P(U_{req}) + P(U_{risk}) + P(U_{hist}) + P(U_{exp}), \quad (5)$$

де:

$P(U_{total})$ – загальна ймовірність виникнення джерел невизначеності.

На практиці вище наведений математичний аналіз джерел невизначеності з оцінкою якості програмних систем дозволяє здійснювати більш об'єктивну та систематичну оцінку потенційних ризиків, які можуть виникнути під час розробки та експлуатації програмного продукту. Наведемо переваги запропонованого підходу:

- Раннє виявлення ризиків: Аналіз джерел невизначеності може бути проведений на ранніх етапах проекту, навіть перед початком розробки програмного продукту. Це дозволяє розробникам та менеджерам з ризиками виявити потенційні проблеми та прийняти заходи щодо їх управління ще до того, як вони стануть критичними;
- Об’єктивна оцінка ризиків: Використання математичних методів дозволяє об’єктивно оцінювати ймовірність виникнення різних джерел невизначеності та їх вплив на якість програмного продукту. Це допомагає уникнути суб'єктивних оцінок та забезпечити більш точне прийняття рішень;
- Покращення планування проекту: Результати аналізу джерел невизначеності можуть бути використані для покращення планування проекту. Наприклад, виявлені ризики можуть бути враховані при розробці графіку проекту та розподілі ресурсів;
- Мінімізація витрат та збільшення ефективності: Раннє виявлення та управління джерелами невизначеності дозволяє уникнути непередбачених витрат і затримок у розробці. Це допомагає збільшити ефективність проекту та знизити його загальні витрати.
- Забезпечення високої якості продукту: Аналіз джерел невизначеності допомагає виявити та управляти ризиками, які можуть негативно вплинути на якість програмного продукту. Це дозволяє забезпечити високу якість програмного забезпечення та задоволення потреб користувачів.

Отже, математичний аналіз джерел невизначеності на практиці допомагає підвищити ефективність та якість розробки програмних систем, зменшити ризики та забезпечити успішне завершення проекту.

Розглянемо методологічні особливості класифікації джерел невизначеності за впливом на якість. Вище зазначена класифікація зводиться до розподілу ідентифікованих джерел невизначеності на категорії в залежності від їх потенційного впливу на різні аспекти якості програмного забезпечення. В дану класифікацію відповідно входять:

- критичні джерела невизначеності;
- суттєві джерела невизначеності;
- мінорні джерела невизначеності;

- невизначеності, пов'язані з зовнішніми факторами.

Згідно (*Clarté et al., 2023*) до критичних джерел невизначеності відносять джерела невизначеності, які мають найбільший потенційний вплив на ключові аспекти якості програмного продукт такі, як надійність, безпека, або продуктивність. Наприклад, помилки в критичних алгоритмах, або невідповідність основним функціональним вимогам. Математично кількість критичних джерел невизначеності можна розрахувати у відповідності до формули (6):

$$|U_{Critical}| = \sum_{u \in U} \delta(u) \cdot \beta(u) \quad (6)$$

де:

$\delta(u)$ – функція, яка повертає 1, якщо, джерело u є критичним, і 0 в іншому випадку,

$\beta(u)$ – коефіцієнт впливу джерела u на якість програмного забезпечення,

U – множина всіх ідентифікованих джерел невизначеності,

$U_{Critical} \subseteq U$ – множина критичних джерел невизначеності.

Згідно (*Mena, 2020*) до суттєвих джерел невизначеності відносять джерела невизначеності, які можуть впливати на якість програмного продукту, але не настільки критичні, як критичні джерела невизначеності. Наприклад, невідповідність деяких необов'язкових функцій, або можливі зміни в інтерфейсі користувача. Математично кількість суттєвих джерел невизначеності можна розрахувати у відповідності до формули (7):

$$|U_{Significant}| = \sum_{u \in U} s(u) \cdot \gamma(u) \quad (7)$$

де:

$s(u)$ – функція, яка повертає 1, якщо, джерело u є суттєвим, і 0 в іншому випадку,

$\gamma(u)$ – коефіцієнт впливу джерела u на якість програмного забезпечення із врахуванням впливу суттєвих джерел невизначеності.

Згідно (*Tsapetis et al., 2023*) до мінорних джерел невизначеності відносять невизначеності, які мають незначний вплив на якість програмного продукту і можуть бути легко виправлені, або проігноровані. Наприклад, невеликі помилки в тексті документації, або невеликі зміни в зовнішньому вигляді програми. Математично кількість мінорних джерел невизначеності можна розрахувати у відповідності до формули (8):

$$|U_{Minor}| = \sum_{u \in U} m(u) \cdot a(u) \quad (8)$$

де:

$m(u)$ – функція, яка повертає 1, якщо, джерело u є мінорним, і 0 в іншому випадку,

$a(u)$ – коефіцієнт впливу джерела u на якість програмного забезпечення із врахуванням впливу мінорних джерел невизначеності.

Згідно (*Cappelli et al., 2020*) до невизначеностей, які пов'язані з зовнішніми факторами відносять джерела невизначеності, які виникають через зовнішні впливи, такі як зміни в інфраструктурі, конкурентний тиск, або нові регуляції. Ці фактори можуть впливати на якість програмного забезпечення, але їх важко контролювати. Математично кількість джерел невизначеностей, які пов'язані з зовнішніми факторами можна розрахувати у відповідності до формули (9):

$$|U_{External}| = \sum_{u \in U} z(u) \cdot \theta(u) \quad (9)$$

де:

$z(u)$ – функція, яка повертає 1, якщо, джерело u відноситься до джерел невизначеності, які виникають через зовнішні впливи, і 0 в іншому випадку,

$z(u)$ – коефіцієнт впливу джерела u на якість програмного забезпечення із врахуванням впливу джерел невизначеності, які виникають через зовнішні впливи.

В результаті узагальнену формулу в межах врахування класифікації джерел невизначеності за впливом на якість можна представити у вигляд виразу (10):

$$Q = \left(|U_{Critical}|(y_1) + |U_{Significant}|(y_2) + |U_{Minor}|(y_3) + |U_{External}|(y_4)\right) \quad (10)$$

де:

y_1 , y_2 , y_3, y_4 – коефіцієнти важливості для критичних, суттєвих, мінорних джерел невизначеності та зовнішніх факторів.

Вище наведена формула (10) дозволяє врахувати вагомість кожного виду джерел невизначеності, а також враховувати їх кількість, що може бути корисним при оцінці впливу на якість програмного продукту.

Оптимізація проблеми формалізації невизначеності у вимогах до програмного забезпечення

Одним із новітніх математичних рішень, яке можна запропонувати для оптимізації проблеми формалізації невизначеності у вимогах до програмного забезпечення, є застосування методів машинного навчання та аналізу даних. На основі великих обсягів даних про попередні проекти програмного забезпечення можна розробити модель, яка прогнозує ймовірність виконання кожної вимоги на основі її характеристик та контексту. Такий підхід дозволяє автоматизувати процес визначення невизначеності та робити більш точні прогнози. Також можливо застосування методів оптимізації таких, як генетичні алгоритми чи методи оптимізації за допомогою штучних нейронних мереж, для пошуку оптимального розподілу ресурсів та стратегій керування невизначеністю в програмному забезпеченні. Такі підходи дозволяють покращити якість та ефективність процесів розробки програмного забезпечення, зменшуючи вплив невизначеності на кінцевий результат.

Представимо вище вказане у вигляді моделі прогнозування ймовірності виконання кожної вимоги на основі її характеристик та контексту. Нехай

$V = \{\vartheta_1, \vartheta_2, \dots, \vartheta_n,\}$ – множина вимог до програмного забезпечення, де n кількість вимог;

$X = \{x_{i1}, x_{i2}, \dots, x_{im},\}$ – множина характеристик (ознак) вимоги ϑ_i, де m – кількість характеристик;

y_i – цільова змінна, яка відображає ймовірність виконання вимоги ϑ_i.

Тоді модель прогнозування може бути визначена, як функція $f: X \to [0,1]$, яка призначає кожній вимозі ймовірність виконання. Метою є навчання моделі на основі тренувальних даних $D = \{(X_1 y_1), (X_2 y_2), \dots, (X_N y_N),\}$, де N, кількість прикладів у тренувальному наборі.

На початку процесу ми збирали дані про вимоги, включаючи їх характеристики та контекст, які можуть впливати на їх виконання, а саме складність, терміни виконання,

пріоритет, критичність тощо. Також застосували наведені вище класифікацію джерел невизначеності, та формули (1-10).

Задача полягає у мінімізації функції втрат L між прогнозованими ймовірностями і справжніми значеннями (11):

$$L = \sum_{i=1}^{N} L\left(y_i, f(X_i)\right) \quad (11)$$

де:

L може бути різними функціями втрат такими, як середньоквадратична помилка для задач регресії, або крос-ентропія для задач класифікації.

На методичному рівні для навчання вище наведеної моделі у відповідності до (*Gray et al., 2023*) застосували:

- градієнтний спуск;
- методи оптимізації: Adam та RMSprop;
- алгоритми машинного навчання такі, як глибокі нейронні мережі, градієнтні дерева випадкового лісу тощо.

Вище зазначені методи дозволяють навчити модель на тренувальних даних і використовувати її для прогнозування ймовірностей виконання вимог у реальних проектах програмного забезпечення.

Розробляючи систему прогнозування ймовірності виконання вимог відмітимо, що дана система базується на методах машинного навчання і використовується для прогнозування ймовірності виконання кожної вимоги на основі її характеристик. Основна ідея полягає в тому, щоб на основі статистичного аналізу попередніх проектів програмного забезпечення побудувати модель, яка здатна передбачати ймовірність виконання вимоги на основі інформації про неї.

На програмному рівні для розробки цієї системи згідно з (*Cappelli et al., 2020*) доцільно використовувати спеціалізовані бібліотеки машинного навчання такі, як TensorFlow та Scikit-learn, які в повній мірі забезпечують побудову та навчання моделі.

Після навчання модель може бути використана для прогнозування ймовірності виконання вимоги в реальних проектах програмного забезпечення.

У відповідності до (*Drozhkin, 2022*), доцільно передбачити, додаткове застосування модульної системи, яка використовує методи оптимізації для пошуку оптимального розподілу вагових коефіцієнтів між вимогами з метою мінімізації загальної невизначеності. Математично, це може бути сформульовано, як задача оптимізації, де метою є мінімізація загальної невизначеності U, яка обчислюється, як сума вагових коефіцієнтів.

В межах практичної реалізації проектного рішення у відповідності до (*Areces et al., 2023*) доцільно передбачити можливість застосування алгоритмів оптимізації: зокрема генетичних алгоритмів, методів оптимізації за допомогою штучних нейронних мереж, або методів оптимізації на основі градієнтів, для пошуку оптимального розподілу вагових коефіцієнтів. В нашому випадку для розробки цієї системи було вирішено використовувати спеціалізовані бібліотеки оптимізації: SciPy та CVXPY, які забезпечили реалізацію алгоритмів оптимізації та пошуку найкращого розподілу вагових коефіцієнтів (*Рисунок 1*). В результаті подальшої практичної програмно-розробницької реалізації вище

зазначені рішення із розробки комплексної системи були інтегровані разом для забезпечення кращого управління невизначеністю вимог та підвищення якості процесу розробки програмного забезпечення.

В результаті розроблений код програмного рішення (*Рисунок 1*) визначає функцію обчислення загальної невизначеності, яка обчислює суму вагових коефіцієнтів. Початковий розподіл вагових коефіцієнтів встановлюється на рівні 0.5 для кожного коефіцієнта. В межах вирішення задач оптимізації встановлюються обмеження для вагових коефіцієнтів (в нашому випадку поставилась умова, що їх сума повинна бути рівна 1). Після чого було використано метод мінімізації для пошуку оптимального розподілу вагових коефіцієнтів. І в результаті оптимальний розподіл вагових коефіцієнтів виводиться на екран. В межах побудови системи, для створення комплексного підходу до управління невизначеністю вимог у процесі розробки програмного забезпечення були поєднані вище запропоновані системні рішення (поєднано систему прогнозування ймовірності виконання вимог та систему оптимізації розподілу вагових коефіцієнтів). Для даного об'єднання у відповідності до (*Liao, 2023*) була застосована стратегія використання прогнозованої ймовірності, щодо виконання вимог для визначення їх вагових коефіцієнтів у системі оптимізації, що дозволило в кінцевому випадку забезпечити реалізацію комплексного підходу до управління невизначеністю вимог у процесі розробки програмного забезпечення.

Відмітимо, що при розробці забезпечили навчання моделі прогнозування ймовірностей виконання вимог на основі наборів тренувальних даних, а саме:

- Quality Assessment Dataset (QAD);
- Bug Prediction Dataset (BPD) – Набір даних для передбачення виникнення помилок та оцінки якості програм;
- Code Quality Benchmark (CQB) – Стандартний набір даних для порівняння та оцінки якості програмного коду.

Зазначені тренувальні набори було взято із: GitHub та UCI Machine Learning Repository.

Після навчання моделі виконується прогнозування ймовірності виконання вимог для тестового набору даних. Потім застосовується функція uncertainty, яка обчислює загальну невизначеність з урахуванням вагових коефіцієнтів, помножених на прогнозовані ймовірності виконання вимог. У відповідності до (*Boukhelifa et al., 2023*) для того щоб знайти оптимальний розподіл вагових коефіцієнтів з урахуванням цієї загальної невизначеності було застосовано метод мінімізації.

В межах розробки використовувались декілька популярних бібліотек для роботи з машинним навчанням та забезпечення оптимізації, а саме:

- бібліотека Pandas: була застосована для обробки та аналізу даних, яка дозволила зручно завантажувати, обробляти та маніпулювати даними, що забезпечило ефективну роботу з наборами даних;
- бібліотека Scikit-learn: застосувалася для машинного навчання в Python, що містить багато реалізацій алгоритмів машинного навчання та інструментів для оцінки моделей, що робить легким створення, навчання та оцінювання моделей.

- бібліотека SciPy: застосувалася для забезпечення наукових обчислень так, як дана бібліотека у відповідності до (*Tsapetis et al., 2023*) містить багато інструментів для оптимізації, лінійної алгебри, інтегрування та інших математичних обчислень. В межах даної бібліотеки застосовувався метод «minimize з SciPy», що дозволяло знаходити оптимізаційний мінімум функції.

Відмітимо, що в межах розробки вибір стратегії управління невизначеністю був здійснений за допомогою теорії прийняття рішень (дана математична теорія досліджує процеси прийняття рішень в умовах невизначеності та ризику).

Дискусія

Запропонований в статті механізм ідентифікації джерел невизначеності в комплексній методології для оцінки якості програмних систем відрізняється від підходів, які описані в праці (*Haiderzai & Khattab, 2019*) тим, що завдяки його застосуванню вдається ретельно систематизувати всі можливі джерела невизначеності, що можуть виникнути в процесі розробки та експлуатації програмного забезпечення.

Описані в статті рішення можна порівняти з різними підходами до управління невизначеністю вимог у програмному забезпеченні. Зокрема порівнюючи із працею (*Areces et al., 2023*) де пропонуються застосовувати традиційні методи без використання машинного навчання, або оптимізації. В таких підходах невизначеність вимог зазвичай керується за допомогою ручного аналізу та експертних оцінок. Порівняно з цим, запропоновані в нашій статі рішення пропонують реалізовувати автоматизований та більш об'єктивний підхід до управління невизначеністю за допомогою методів машинного навчання та оптимізації. Даний підхід за своїм спрямуванням частково схожий із підходами, які висвітленні в праці (*Behera, 2023*) проте рішення, які наведені в тій праці можуть використовувати лише моделі машинного навчання для прогнозування ймовірності виконання вимог без подальшої оптимізації. У порівнянні з такими рішеннями, описана в нашій роботі система додає елемент оптимізації, що дозволяє автоматично керувати ваговими коефіцієнтами для кращого управління невизначеністю.

Порівнюючи підходи, що базуються виключно на оптимізації варто зазначити, що рішення, які описані в праці (*Galli et al., 2023*) використовують лише методи оптимізації для розподілу ресурсів та керування невизначеністю. Тоді, як описана система використовує комбінацію методів машинного навчання та оптимізації, що може призвести до більш точного та ефективного управління невизначеністю.

Використання запропонованих в роботі методів машинного навчання для прогнозування ймовірності виконання вимог та методів оптимізації для знаходження оптимального розподілу вагових коефіцієнтів в розробленій системі відрізняється від деяких існуючих рішень у наступних аспектах:

- Комбінація методів: У запропонованій розробці використовується комбінація методів машинного навчання та оптимізації для керування невизначеністю вимог. Це означає, що система використовує, як прогнози, котрі отримані з моделей машинного навчання так і оптимізовані вагові коефіцієнти для кращого управління невизначеністю;

- Реалізація конкретних алгоритмів: У даному випадку, для прогнозування ймовірності виконання вимог використовується алгоритм RandomForestClassifier. Цей алгоритм добре працює з різними типами даних та мають високу точність прогнозування. Також для оптимізації використовується бібліотека SciPy, яка містить різноманітні алгоритми оптимізації такі, як метод мінімізації, що дозволяє знаходити оптимальні розподіли вагових коефіцієнтів;
- Більш гнучкий підхід: Використання методів машинного навчання дозволяє моделі адаптуватися до нових даних та змінюватися відповідно до зміни умов, що робить систему більш гнучкою та ефективною у різних сценаріях;
- Об'єктивність: Використання алгоритмів машинного навчання та оптимізації дозволяє системі працювати на основі об’єктивних даних та алгоритмів уникнувши суб'єктивності та помилок, які пов’язані з ручним аналізом, або прийняттям рішень експертом.

Отже, використання комбінації методів машинного навчання та оптимізації із застосуванням формалізації невизначеності у вимогах до програмного забезпечення на базі застосування методів машинного навчання та аналізу даних у запропонованій розробці дозволяє створити більш ефективну та об'єктивну систему управління невизначеністю вимог.

Висновки

1. Механізм ідентифікації джерел невизначеності допомагає зрозуміти, з якими факторами може стикнутися команда розробки під час реалізації програмного продукту. Це дає можливість підготуватися до них, розробити стратегії управління ризиками та невизначеністю і забезпечити більш високу якість програмного забезпечення.
2. Механізм ідентифікації джерел невизначеності в комплексній методології взаємодіє з іншими етапами оцінки якості програмного забезпечення такими як аналіз ризиків, тестування та моніторинг якості. Це дозволяє забезпечити комплексний та збалансований підхід до управління невизначеністю.
3. Використання комбінації методів машинного навчання та оптимізації з формалізацією невизначеності у вимогах до програмного забезпечення на базі методів машинного навчання та аналізу даних у запропонованій розробці дозволяє створити більш ефективну та об’єктивну систему управління невизначеністю вимог у порівнянні з деякими існуючими підходами. Перш за все, використання методів машинного навчання дозволяє побудувати модель прогнозування ймовірності виконання вимог на основі їх характеристик, що допомагає визначити ймовірність успішності виконання кожної вимоги. Це дозволяє отримувати об'єктивні прогнози та підвищує надійність системи.

Другою ключовою складовою є використання оптимізаційних методів для знаходження оптимального розподілу вагових коефіцієнтів, що дозволяє керувати невизначеністю вимог та максимізувати якість рішень. Цей підхід забезпечує ефективне розподілення ресурсів та забезпечує оптимальний результат в умовах невизначеності. Третім аспектом є використання аналізу даних та методів машинного навчання для

ідентифікації джерел невизначеності вимог до програмного забезпечення, що дозволяє заздалегідь визначити потенційні ризики та прийняти відповідні заходи для їх управління. В результаті використання комбінації цих методів у запропонованій розробці забезпечує більш ефективний та об'єктивний підхід до управління невизначеністю вимог до програмного забезпечення, порівняно з деякими існуючими підходами. Це дозволяє підвищити надійність, якість та ефективність розроблюваного програмного продукту.

References:

Areces, C. et al. (2023). Uncertainty-based knowing how logic. *Journal of Logic and Computation*, exad056. https://doi.org/10.1093/logcom/exad056

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

Bougeret, M., Pessoa, A., & Poss, M. (2023). Single machine robust scheduling with budgeted uncertainty. *Operations Research Letters*, *51*(2), 137-141. https://doi.org/10.1016/j.orl.2023.01.007

Boukhelifa, N., Johnson, C. R., & Potter, K. (2023). Visualization and decision-making design under uncertainty. *IEEE Computer Graphics and Applications*, *43*(5), 23-25. https://doi.org/10.1109/mcg.2023.3302172

Cappelli, V. Et al. (2020). Sources of uncertainty and subjective prices. *Journal of the European Economic Association*, *19*(2), 872-912. https://doi.org/10.1093/jeea/jvaa013

Clarté, L. et al. (2023). Theoretical characterization of uncertainty in high-dimensional linear classification. *Machine Learning: Science and Technology*, *4*(2). https://doi.org/10.1088/2632-2153/acd749

Dorn, J., Apel, S., & Siegmund, N. (2023). Mastering uncertainty in performance estimations of configurable software systems. *Empirical Software Engineering*, *28*(2). https://doi.org/10.1007/s10664-022-10250-2

Drozhkin, A. A. (2022). Analyzing current approaches to assessing the quality of software products. *Vestnik of the Plekhanov Russian University of Economics*, *6*, 207-218. https://doi.org/10.21686/2413-2829-2022-6-207-218

Galli, T., Chiclana, F., & Siewe, F. (2023). Practical consequences of quality views in assessing software quality. *Axioms*, *12*(6), 529. https://doi.org/10.3390/axioms12060529

Gray, N., de Angelis, M., & Ferson, S. (2023). Towards an automatic uncertainty compiler. *International Journal of Approximate Reasoning*, 108951. https://doi.org/10.1016/j.ijar.2023.108951

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*, 74808-74833. https://doi.org/10.1109/access.2020.2988408

Haiderzai, M. D., & Khattab, M. I. (2019). How software testing impacts the quality of software systems? *International Journal of Engineering in Computer Science*, *1*, 47-51. https://doi.org/10.33545/26633582.2019.v1.i1a.14

Liao, M., Shen, D., & Lv, P. (2023). A unified model of data uncertainty and data relation uncertainty. *Knowledge-Based Systems*, 110811. https://doi.org/10.1016/j.knosys.2023.110811

Mena, R. J. (2020). *Modelling uncertainty in black-box classification systems* [Doctoral thesis, Universitat de Barcelona]. TDX (Tesis Doctorals en Xarxa). http://hdl.handle.net/10803/670763

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), 86-93. https://doi.org/10.52448/16948173_2020_2_86

Tikhanychev, O. V. (2020). On improving indicators for assessing the decision support systems' software quality. *IOP Conference Series: Materials Science and Engineering*, *919*, 052009. https://doi.org/10.1088/1757-899x/919/5/052009

Tsapetis, D. et al. (2023). UQpy v4.1: Uncertainty quantification with Python. *SoftwareX*, *24*, 101561. https://doi.org/10.1016/j.softx.2023.101561

Vinogradova, M. et al. (2021). Assessing the sources of uncertainty in supply chain management. *Strategic Change*, *30*(5), 453-460. https://doi.org/10.1002/jsc.2465

Walkinshaw, N., & Hierons, R. M. (2023). Modelling second-order uncertainty in state machines. *IEEE Transactions on Software Engineering*, 1-16. https://doi.org/10.1109/tse.2023.3250835

Zhang, X. et al. (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

Zhao, J. et al. (2023). Evaluating the impact of uncertainty visualization on model reliance. *IEEE Transactions on Visualization and Computer Graphics*, 1-15. https://doi.org/10.1109/tvcg.2023.3251950

Appendix

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from scipy.optimize import minimize

# Loading the training data
data = pd.read_csv('training_data.csv')

# Splitting into training and testing sets for prediction system
X_train, X_test, y_train, y_test = train_test_split(data.drop('target', axis=1), d

# Training the prediction model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Predicting probabilities of fulfilling requirements for optimization system
predicted_probabilities = model.predict_proba(X_test)[:, 1]

# Function to compute total uncertainty considering weighted coefficients
def uncertainty(weights):
    total_uncertainty = sum(weights * predicted_probabilities)
    return total_uncertainty

# Initial distribution of weighted coefficients
initial_weights = [0.5] * len(predicted_probabilities)

# Constraints for weighted coefficients (e.g., sum = 1)
constraints = ({'type': 'eq', 'fun': lambda weights: sum(weights) - 1})

# Minimizing total uncertainty
result = minimize(uncertainty, initial_weights, constraints=constraints)

# Optimal distribution of weighted coefficients
optimal_weights = result.x
```

Рисунок 1. Фрагмент коду системи, що оптимізує задачу формалізації невизначеності у вимогах до програмного забезпечення