УДК 004.65:004.052
DOI https://doi.org/10.15589/znp2025.4(502).24

INFORMATION SYSTEMS QUALITY CRITERIA AND ASSESSMENT METHODS AT THE DESIGN STAGE

КРИТЕРІЇ ЯКОСТІ ІНФОРМАЦІЙНИХ СИСТЕМ ТА МЕТОДИКИ ОЦІНЮВАННЯ НА ЕТАПІ ПРОЄКТУВАННЯ

Sergii V. Bataiev[1]
serg.bataiev@gmail.com
ORCID: 0009-0009-9564-9403

Olena M. Husak[2]
gusakolena17@gmail.com
ORCID: 0000-0003-4395-6355

Olga I. Artemenko[3]
olga.hapon@gmail.com
ORCID: 0000-0002-4057-1217

С. В. Батаєв[1],
аспірант

О. М. Гусак[2],
канд. техн. наук, доцент

О. І. Артеменко[3],
канд. техн. наук, доцент

[1] University of Warwick, Great Britain
[2] Yuriy Fedkovych Chernivtsi National University, Chernivtsi
[3] Private Higher Educational Institution "Bukovynian University", Chernivtsi
[1] Ворицький університет, Велика Британія
[2] Чернівецький національний університет імені Юрія Федьковича, м. Чернівці
[3] Приватний вищий навчальний заклад «Буковинський університет», м. Чернівці

Abstract. *Purpose*. To analyze modern quality criteria for information systems and methods for their assessment at the design stage. *Methodology*. The study was conducted by reviewing the literature of recent years (2015–2025), including international standards and scientific publications devoted to assessing the quality of software at the early stages of the life cycle. Methods of analysis, comparison and generalization were applied, the object of the study is the process of ensuring the quality of information systems at the design stage, and the subject is a set of quality criteria and procedures for their early assessment. *Results*. It was established that the quality of information systems is multifaceted and includes functional suitability, reliability, performance, usability, security, maintainability, portability and compatibility. It was determined that at the design stage, partial quantitative and qualitative assessment of these characteristics is possible: analytical models are used to predict performance and reliability, compliance checks, architectural expert methods (reviews, ATAM), prototyping and usability testing to assess convenience, design metrics to assess maintainability, etc. *Scientific novelty*. Modern methods for assessing the quality of IS at the design stage and their correlation with the relevant quality criteria are systematized. It is shown that the complex application of various methods (architecture scenario analysis, simulations, expert review, metrics, etc.) allows identifying potential quality problems before the implementation stage. *Practical significance*. The results can be used by developers and project managers to implement the "shift-left" practice – early quality assurance. The use of a system of quality criteria and relevant assessment methods at the design stage will help reduce the cost of correcting defects and increase the likelihood of successful implementation of information systems.
Key words: quality of information systems; quality criteria; design stage; quality assessment; software attributes; quality assurance.

Анотація. *Мета*. Проаналізувати сучасні критерії якості інформаційних систем та методики їх оцінювання на стадії проєктування. *Методика*. Дослідження проведено шляхом огляду літератури останніх років (2015–2025), включно з міжнародними стандартами та науковими публікаціями, присвяченими оцінюванню якості програмного забезпечення на ранніх етапах життєвого циклу. Застосовано методи аналізу, порівняння та узагальнення, об'єктом дослідження є процес забезпечення якості інформаційних систем на стадії проєктування, а предметом – сукупність критеріїв якості та процедур їх раннього оцінювання. *Результати*. Встановлено, що якість інформаційних систем є багатоаспектною і охоплює функціональну придатність, надійність, продуктивність, зручність користування, безпеку, підтримуваність, переносимість та сумісність. Визначено, що на етапі проєктування можливе часткове кількісне та якісне оцінювання цих характеристик: застосовуються аналітичні

моделі для прогнозування продуктивності та надійності, перевірки на відповідність вимогам, архітектурні експертні методи (огляди, ATAM), прототипування та юзабіліті-тестування для оцінки зручності, метрики проєктування для оцінки підтримуваності тощо. *Наукова новизна*. Систематизовано сучасні методи оцінювання якості ІС на стадії проєктування та їх співвідношення з відповідними критеріями якості. Показано, що комплексне застосування різних методик (сценарного аналізу архітектури, симуляцій, експертного огляду, метрик тощо) дозволяє виявити потенційні проблеми якості до етапу реалізації. *Практична значимість*. Результати можуть бути використані розробниками та менеджерами проєктів для впровадження практики «shift-left» – раннього забезпечення якості. Використання системи критеріїв якості та відповідних методик оцінки на етапі проєктування сприятиме зниженню вартості виправлення дефектів та підвищенню ймовірності успішної реалізації інформаційних систем.

Ключові слова: якість інформаційних систем; критерії якості; етап проєктування; оцінювання якості; атрибути програмного забезпечення; забезпечення якості.

ПОСТАНОВКА ЗАДАЧІ

Якість інформаційних систем (ІС) визначає успішність їх впровадження та експлуатації, впливаючи на задоволеність користувачів, ефективність бізнес-процесів і репутацію організації [1]. При цьому якість не обмежується тільки функціональною правильністю роботи системи – значна частка провалів ІТ-проєктів спричинена недотриманням інших критеріїв, таких як зручність користування чи надійність [1]. Дослідження вказують, що близько 64 % програмних проєктів не повністю відповідають потребам користувачів саме через проблеми з нефункціональними характеристиками (юзабіліті, продуктивністю тощо), навіть якщо основна функціональність реалізована правильно [1]. Крім того, виявлення та виправлення дефектів на пізніх стадіях (після релізу) обходиться на порядок дорожче, ніж на ранніх етапах розробки [2]. Наприклад, усунення помилки, що коштує умовно $100 на етапі модульного тестування, може вимагати до $100 000 у фазі промислової експлуатації [2]. Отже, проблема забезпечення якості на етапі проєктування набуває особливої актуальності: існує потреба в критеріях, що дозволяють оцінити майбутню систему до написання коду, та у відповідних методиках оцінювання, які допоможуть запобігти дефектам і невідповідності вимогам ще “на креслярському столі”. Недостатня увага до цих питань призводить до того, що приблизно 70 % дефектів зароджуються ще на стадіях планування і проєктування [2], а їх несвоєчасне виявлення обумовлює зриви проєктів за термінами, бюджетом або показниками якості.

Таким чином, сучасний стан проблеми характеризується двома ключовими викликами. Перший – багатовимірність поняття якості інформаційних систем, яка ускладнює формулювання універсальних критеріїв та метрик. Другий – дефіцит усталених методик інтеграції оцінювання якості у процес проєктування. Відомо, що чим раніше проводиться перевірка вимог і дизайну на відповідність критеріям якості, тим нижчими будуть сукупні витрати на забезпечення якості і тим вищою – ймовірність успішного впровадження системи [2]. Однак традиційно в практиці багато аспектів якості повноцінно оцінюються лише на етапах тестування готового продукту або навіть дослідної експлуатації, коли можливості впливу на архітектуру системи вже обмежені. Це спричиняє потребу у дослідженні, яке узагальнить відомі критерії якості та методи їх раннього оцінювання, щоб сприяти зменшенню цього розриву між теорією та практикою.

АНАЛІЗ ОСТАННІХ ДОСЛІДЖЕНЬ І ПУБЛІКАЦІЙ

Проблематика оцінювання якості програмного забезпечення та інформаційних систем досліджується вже кілька десятиліть. Ще у 1970-х роках були запропоновані перші моделі якості. Зокрема, модель Мак-Калла (McCall, 1977) визначила набір факторів якості ПЗ, згрупованих за трьома перспективами: огляд продукту (підтримуваність, гнучкість, тестованість), експлуатація продукту (правильність, надійність, ефективність, цілісність, зручність) та перехід продукту (портативність, можливість повторного використання, сумісність) [3]. Наступна модель Боема (Boehm, 1978) розвинула ці ідеї, виділивши три верхньорівневі характеристики: утилітарність (зручність та ефективність використання), можливість супроводження (модифікованість, тестованість, зрозумілість) та портативність (пристосовність до зміни середовища) [3]. Ці класичні роботи заклали основу для пізніших стандартів. Так, міжнародний стандарт ISO/IEC 9126 (2001) ввів шість базових характеристик якості ПЗ: функціональність, надійність, зручність використання, ефективність, супроводжуваність, переносимість, а також розрізняв внутрішню, зовнішню якість та якість у використанні [3]. Подальший розвиток призвів до створення сімейства стандартів ISO 25000 SQuaRE. Зокрема, чинний стандарт ISO/IEC 25010:2011 визначає вісім характеристик якості продукту (функціональна придатність, продуктивність/ефективність, сумісність, зручність використання, надійність, безпека, підтримуваність, переносимість) та п’ять характеристик якості в користуванні (ефективність у досягненні цілей, продуктивність використання, задоволеність, відсутність ризиків, охоплення контекстів) [1]. Ця модель є нині загальноприйнятою

основою для визначення критеріїв якості програмних систем. Наприклад, за стандартом ISO 25010 програмний продукт вважається якісним не лише якщо він виконує необхідні функції, але й якщо він надійний (стабільно працює без збоїв протягом потрібного часу), захищений (гарантує конфіденційність та цілісність даних), зручний для користувача, ефективно використовує ресурси, легко змінюється та переноситься на інші платформи тощо [1].

Аналіз наукових публікацій останніх років показує, що дослідники приділяють значну увагу як окремим аспектам якості, так і методам їх оцінювання. Формуються специфічні підходи у різних доменах. Наприклад, у галузі медичних інформаційних систем було виявлено щонайменше 13 категорій критеріїв оцінки якості, що охоплюють показники повноти та доступності даних, задоволеності персоналу, тощо [8]. Водночас залишається проблема відсутності єдиного стандартного інструментарію: за результатами систематичного огляду, проведеного Ali Kia та ін. (2022), існує нагальна потреба у розробленні уніфікованої методики оцінювання якості інформаційних систем, оскільки навіть у високорегульованій медичній сфері різні дослідники застосовують різні набори критеріїв [8].

Для етапу проєктування інформаційних систем найбільш актуальними є методики оцінювання архітектурних та дизайнерських рішень на відповідність вимогам до якості. В літературі описано чимало таких підходів. Згідно з оглядами (напр., Fatima & Lago, 2023), сьогодні домінують сценарні методи оцінки архітектури – зокрема, методи аналізу модифікованості та інші техніки, що використовують сценарії використання для перевірки архітектурних рішень щодо певних якостей [5]. Широко застосовується метод ATAM (Architecture Tradeoff Analysis Method), який передбачає виявлення й аналіз сценаріїв якості (наприклад, сценарії надійності, продуктивності, безпеки) спільно з експертами та зацікавленими сторонами, з метою виявлення «вузьких місць» та компромісів архітектури. Сценарно-орієнтовані техніки домінують, оскільки дозволяють в контексті конкретної системи виявити потенційні проблеми, пов'язані з виконанням вимог до певних атрибутів якості [5]. Другу групу складають експертні методи, засновані на досвіді: наприклад, проведення фокус-груп чи оглядів дизайну за участі експертів і користувачів, щоб отримати зворотний зв'язок щодо передбачуваної зручності інтерфейсу, розумності архітектурних рішень тощо [5]. Третя група – прототипування та моделювання. Розробка прототипів (від паперових ескізів інтерфейсу до працюючих модулів) дає змогу провести раннє юзабіліті-тестування, виявити недоліки дизайну, перевірити гіпотези щодо поведінки користувачів. Імітаційне моделювання та симуляції часто застосовуються для оцінки продуктивності систем: шляхом створення моделей навантаження, черг або використання інструментів профілювання ще на рівні архітектури можна передбачити, чи відповідає проєктована система вимогам, наприклад, по часу відгуку або пропускній здатності [5]. Четвертим напрямом є метричний підхід, коли використовуються кількісні метрики дизайну або архітектурних артефактів. Існує багато метричних моделей, зокрема для об'єктно-орієнтованого проєктування широко відомий набір метрик Чідаберра–Кемерера (CK Metrics), що включає показники зв'язності, когезії класів, складності ієрархії тощо [6]. Вони слугують індикаторами таких критеріїв, як підтримуваність та супроводжуваність: дослідження підтверджують значущий зв'язок між значеннями метрик дизайну (кількістю залежностей між модулями, рівнем складності компонентів тощо) та майбутньою складністю підтримки системи [6]. Зокрема, виявлено, що використання набору метрик CK та його розширень дозволяє з достатньою точністю передбачати модулі, схильні до дефектів чи ускладнені в супроводі [6].

Окремо варто відзначити критерії безпеки інформаційних систем. Безпека є критичною якістю для багатьох сучасних ІС. На етапі проєктування її оцінювання здійснюється, як правило, шляхом аналізу загроз (Threat Modeling) – побудови моделей можливих уразливостей та шляхів несанкціонованого доступу, і перевірки, чи запроєктовано достатні механізми захисту. Існують методики STRIDE, DREAD та інші, які допомагають проаналізувати дизайн з точки зору шести категорій загроз (спуфінг, підроблення даних, відмови в обслуговуванні тощо) і закласти у вимоги відповідні контрзаходи. Щодо критеріїв безпеки, українські дослідники Золотухіна О.А. та ін. зазначають, що ключовими складовими якості інформаційних систем у контексті інформаційної безпеки є конфіденційність, цілісність (достовірність) та доступність інформації, тобто система повинна забезпечувати захист даних від несанкціонованого доступу, гарантувати правильність і непротиворічивість даних, а також стійкість до збоїв та атак [7]. На стадії дизайну ці аспекти контролюються через розробку політик безпеки, моделювання зловмисника, закладення резервування і механізмів відновлення.

Попри розмаїття доступних підходів, аналіз публікацій показує, що багато з них лишаються теоретичними. Varajão та ін. (2022), провівши огляд моделей оцінки успішності ІТ-проєктів, дійшли висновку, що більшість знайдених методів за своєю суттю є скоріше концептуальними «вправами», а емпіричних підтверджень їх практичного використання обмаль [4]. Аналогічно, у сфері оцінювання архітектур зазначається, що класичні методики (зокрема сценарні, як ATAM) потребують суттєвих затрат часу та експертних ресурсів, тому в реальних проектах застосовуються не завжди [5]. Ці факти вказують на нера

вирішені частини проблеми: як інтегрувати формальні та напівформальні методи оцінки якості у повсякденний процес проєктування; як автоматизувати або полегшити їх використання; як поєднати різні критерії в єдину систему, що давала б цілісну оцінку якості майбутньої системи. Отже, подальші дослідження спрямовані на розробку методик, які б охоплювали декілька аспектів якості одразу, були підтверджені експериментально та зручні для застосування практиками в умовах реальних проєктних обмежень.

На основі проведеного аналізу можна виділити кілька аспектів, що залишаються недостатньо вирішеними. По-перше, відсутня уніфікована модель оцінювання якості на етапі проєктування, яка б поєднувала різнорідні критерії (функціональні і нефункціональні) та пропонувала інтегрований підхід. Існуючі моделі якості (ISO 25010 та інші) дають таксономію характеристик, але не детальні інструкції, як виміряти кожну з них до реалізації системи. По-друге, обмежена практична апробація багатьох методів: немає достатньо емпіричних даних про ефективність застосування, скажімо, формальних методів перевірки чи розрахункових моделей на ранніх етапах. Це ускладнює вибір методики для конкретного проєкту – проєктувальники часто покладаються на власний досвід та експертні оцінки, які можуть бути суб'єктивними. По-третє, інструментальна підтримка раннього оцінювання якості все ще розвивається: хоча з'являються засоби моделювання (для аналізу продуктивності, надійності), інструменти статичного аналізу дизайну, проте їх інтеграція у загальний процес проєктування не завжди є безшовною. Крім того, деякі критерії (наприклад, юзабіліті чи задоволеність користувачів) важко оцінити без дослідження з реальними користувачами, що на стадії дизайну можливе лише в обмеженій формі (через прототипи, опитування, експертні heuristic evaluation). Це створює ризик, що частина вимог до зручності або прийнятності системи виявиться незадоволеною аж після впровадження, коли виправлення дизайну коштує дуже дорого.

Відповідно, невирішеною частиною загальної проблеми є розробка підходів, які б забезпечили комплексне раннє оцінювання із залученням автоматизованих засобів і користувацького фідбеку, а також узгодження різних метрик та показників між собою. Наприклад, як збалансувати вимоги до продуктивності та безпеки ще на етапі архітектури (коли підвищення безпеки часто знижує продуктивність)? Як кількісно порівняти важливість різних критеріїв для конкретного проєкту і прийняти дизайн-рішення на основі таких порівнянь? Ці питання науковці лише починають вирішувати, тож у даній статті зроблено спробу частково їх адресувати шляхом систематизації критеріїв та методик оцінки якості на етапі проєктування і вказання напрямів, де потрібні подальші дослідження.

МЕТА І ЗАВДАННЯ СТАТТІ

Метою даного дослідження є систематизація критеріїв якості інформаційних систем та методик їх оцінювання на стадії проєктування, а також визначення рекомендацій щодо використання цих методик для підвищення успішності проєктів. Іншими словами, стаття покликана дати відповідь на питання: які основні характеристики (атрибути) якості варто враховувати вже при проєктуванні ІС, і якими способами (інструментами, методами) можна оцінити або перевірити дотримання цих характеристик у проєктних рішеннях до етапу реалізації? Досягнення поставленої мети передбачає розв'язання таких завдань:

- узагальнити та класифікувати критерії якості інформаційних систем, з опорою на сучасні стандарти та наукові підходи;
- проаналізувати існуючі методики оцінювання кожного з виявлених критеріїв на ранніх етапах (наприклад, методи аналізу архітектури для надійності, метрики складності для підтримуваності, прототипування для юзабіліті тощо);
- виявити переваги та обмеження цих методик, а також прогалини, де методів або недостатньо, або вони трудомісткі й потребують покращення;
- сформулювати рекомендації щодо впровадження процесу оцінювання якості під час проєктування на практиці та окреслити напрямки подальших досліджень у цій галузі.

Поставлена мета узгоджується з назвою та очікуваними результатами статті. Досягнення мети має практичну цінність, оскільки дозволить знизити ризики невдач ІТ-проєктів за рахунок врахування всіх важливих аспектів якості на випередження, ще до того, як недоліки проявлять себе у готовій системі.

МЕТОДИ, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Об'єктом даного дослідження є процес проєктування інформаційних систем з точки зору забезпечення якості. Фокус зроблено на процедурі прийняття проєктних рішень (архітектурних, дизайнерських) з врахуванням вимог до якості та на способах перевірки цих рішень. Предметом виступають конкретні критерії якості (характеристики програмної системи) і методики їх оцінювання на етапі проєктування.

Дослідження проведено за допомогою комплексу методів. Основним є метод оглядово-аналітичного дослідження ("literature review"): зібрано та проаналізовано понад 30 наукових джерел, включно зі статтями у журналах, матеріалами конференцій і галузевими стандартами, що стосуються забезпечення якості ПЗ. Було застосовано методику пошуку інформації у наукометричних базах (Scopus, IEEE Xplore, ACM Digital Library) за ключовими словами "software quality attributes", "design stage evaluation", "architecture assessment", "software metrics early

prediction" тощо. Перевага віддавалася роботам останніх 5–10 років, щоб забезпечити актуальність огляду, хоча для історичного екскурсу були враховані й класичні праці (моделі якості 1970–1990-х років). Для відбору джерел використовувалися критерії якості: наявність рецензування (peer-review), індексування у визнаних наукових базах, уникаючи при цьому джерел сумнівного походження. Україномовні та англомовні публікації аналізувалися рівноправно, що дозволило врахувати як глобальний, так і локальний (український) контекст проблеми.

З метою усунення можливих упереджень та помилок, було критично оцінено результати різних авторів. Наприклад, зіставлялися висновки оглядових статей різних дослідників: якщо декілька незалежних оглядів підтверджували важливість певного підходу (скажімо, сценарного аналізу архітектури), ці висновки приймалися як достовірні. Для перевірки повторюваності дослідження описані ключові пошукові запити і критерії відбору джерел, аби інші дослідники за потреби могли відтворити основні етапи.

Обробка даних із джерел здійснювалася шляхом контент-аналізу: з кожної роботи вичленовувалися релевантні факти (визначення критеріїв, опис методів, результати експериментів з оцінювання якості). Далі ці факти групувалися за логічними категоріями (наприклад, "метрики надійності", "методи оцінки архітектури" тощо). Такий підхід забезпечує повторюваність висновків – інші фахівці, застосувавши подібний алгоритм аналізу, мають отримати співставні результати щодо класифікації критеріїв та методик.

Варто наголосити, що оскільки дане дослідження носить оглядово-аналітичний характер, експериментальна перевірка методик не проводилася. Натомість було зроблено акцент на статистичні та емпіричні дані з літератури. Зокрема, розглянуто статистики успішності/невдач проектів, результати емпіричних оцінок кореляції метрик дизайну з дефектністю коду, практичні кейси застосування ATAM тощо. Такий непрямий підхід, хоча і має обмеження (залежність від якості джерел), дозволив охопити ширший спектр методів і критеріїв, ніж було б можливо в межах одного власного експерименту.

Отримана сукупність даних була структурована для подальшого представлення у розділі "Основний матеріал". У ньому критерії якості і методи їх оцінки згруповано таким чином, щоб читач міг повністю відтворити логіку оцінювання для будь-якого критерію. Фактично, розроблено узагальнену схему (алгоритм) дій: для кожної характеристики якості визначено, які індикатори чи метрики можна використати на етапі дизайну, які інструменти чи процедури застосувати, щоб ці індикатори отримати, і як інтерпретувати результати. Ця схема відображає найкращі практики, описані в літературі, і може слугувати своєрідним шаблоном для повторення оцінювання якості у різних проектах.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Критерії якості інформаційних систем. На основі аналізу стандартів та публікацій можна виділити вісім основних критеріїв якості інформаційних систем (які часто називають характеристиками або атрибутами якості): функціональна придатність, надійність, продуктивність (ефективність), сумісність, зручність використання (юзабіліті), безпека, підтримуваність (супроводжуваність) та портативність. Кожен із цих критеріїв охоплює певний аспект якості програмного продукту і може бути деталізований через підкритерії. Наприклад, надійність включає такі підкритерії, як відмовостійкість, здатність до відновлення та зрілість (стабільність) системи; зручність використання – придатність до навчання, зрозумілість інтерфейсу, ергономічність, захист від помилок користувача тощо; підтримуваність – модульність, аналізованість, модифікованість та тестованість системи. Вказані критерії є нефункціональними вимогами (Non-Functional Requirements), але вони тісно пов'язані з функціональністю: система може повністю виконувати заявлені функції (бути функціонально повною і правильною), але при цьому не задовольняти користувачів через низьку швидкодію чи складність інтерфейсу. Отже, забезпечення якості вимагає балансу всіх критеріїв. При проєктуванні архітектури та дизайну ІС слід явно враховувати кожен з важливих критеріїв та закладати рішення, які забезпечать їх реалізацію.

Методики оцінювання на етапі проєктування. Для кожного з перелічених критеріїв якості існують відповідні методи оцінки, які можна застосувати ще до побудови кінцевої системи – на рівні специфікацій вимог, архітектури, проєктної документації чи прототипів. Розглянемо основні з них.

1. **Функціональна придатність.** Цей критерій відповідає на запитання: чи виконує система потрібні функції повністю і коректно? На стадії проєктування повноту і коректність функціональних вимог оцінюють через аналіз вимог (рев'ю вимог, випадки використання) та створення концептуальних моделей (діаграм прецедентів, BPMN тощо) з подальшою їх перевіркою експертами та стейкхолдерами. Використовується метод інспекції вимог – група експертів переглядає специфікацію на наявність пропущених випадків або зайвих функцій. Для критичних систем може застосовуватися формальна верифікація моделей вимог чи алгоритмів (наприклад, методом скінченних автоматів, перевіркою моделей – model checking). На етапі архітектурного дизайну функціональна придатність опосередковано підтверджується через трасування вимог: кожна вимога повинна бути задоволена певними компонентами архітектури, і проєктувальники перевіряють, що таке трасування повне. Методики типу Requirement Reviews та Use Case Validation дозволяють ще до написання коду

переконатися, що функціональний обсяг системи відповідає очікуванням замовника. Оцінка коректності функцій на рівні дизайну можлива шляхом написання специфікацій (контрактів) для інтерфейсів модулів – це використовується, наприклад, у методах проектування на основі контрактів (Design by Contract). Хоча фактичне підтвердження коректності функцій відбувається під час тестування готового ПЗ, на етапі проєкту розробники можуть, спираючись на ці методи, попередити логічні помилки в постановці задачі.

2. **Продуктивність та ефективність (Performance Efficiency).** Цей критерій стосується швидкодії системи і раціонального використання ресурсів. На рівні архітектури оцінювання продуктивності виконується за допомогою аналітичного моделювання та симуляції. Поширеним підходом є створення моделей черг або мереж Петрі, які представляють потоки запитів через компоненти системи; це дозволяє розрахувати очікуваний час відгуку, пропускну здатність та інші показники під заданим навантаженням. Існують інструменти на кшталт Layered Queuing Network (LQN) моделювання, що беруть на вхід параметри архітектури (кількість серверів, середній час обробки запиту тощо) і дають на виході оцінки продуктивності. Також застосовують нагрузочні профілі: визначається очікувана інтенсивність вхідних транзакцій, і перевіряється, чи обрана архітектура (наприклад, клієнт-серверна з певними характеристиками сервера) здатна її витримати. Для оцінки ефективності використання ресурсів (CPU, пам'ять, мережа) проєктувальники можуть використовувати емпіричні дані або аналогії з існуючими системами. Наприклад, якщо використовується певна СУБД, то на основі її специфікацій і спроектованої схеми бази даних можна приблизно оцінити, який обсяг пам'яті чи диску буде потрібен. Важливим методом є профілювання прототипів: коли створюється прототип ключового алгоритму або компонента та запускається в тестовому середовищі для заміру швидкодії. Це особливо корисно для обчислювально інтенсивних частин системи – ранні вимірювання дозволяють скоригувати дизайн (наприклад, додати кешування, змінити алгоритм) до того, як вся система буде реалізована. Отже, комбінація розрахункових моделей і експериментів з прототипами забезпечує досить надійне передбачення продуктивності майбутньої ІС. Якщо такі оцінки показують невідповідність вимогам (наприклад, очікуваний час відгуку перевищує гранично допустимий), на етапі проектування можна внести зміни: змінити архітектуру (розподілити навантаження, додати балансування), оптимізувати алгоритми або апаратні вимоги.

3. **Надійність (Reliability).** Надійність визначає здатність системи виконувати свої функції безвідмовно протягом потрібного інтервалу часу. Для оцінки надійності на етапі дизайну застосовуються методи аналізу відмов і моделювання відмовостійкості. Одним з підходів є побудова дерев відмов (Fault Tree Analysis) – графічна модель, що показує, як комбінації збоїв компонентів можуть призвести до критичної відмови системи. Проєктувальники розробляють дерево, визначають найбільш вразливі "гілки" і можуть підрахувати ймовірність відмови за допомогою відомих надійностей компонентів (якщо такі дані є). На основі цього приймаються рішення про введення резервування (дублювання компонентів для підвищення надійності) чи інших механізмів підвищення живучості системи. Іншим методом є Markov Models для надійності – система моделюється як сукупність станів (робочий стан, стан з відмовою одного компонента, критичний відмовний стан тощо) і переходів між ними з певними інтенсивностями. Розв'язавши таку модель, отримують метрики надійності: середній час безвідмовної роботи (MTBF), ймовірність відмови за заданий період і т.д. Звичайно, отримання точних чисел на стадії дизайну ускладнене, адже невідомі всі параметри, але порівняльна оцінка варіантів дизайну можлива (наприклад, з резервуванням серверів чи без нього). Крім математичного підходу, використовується евристичний аналіз: експерти з надійності переглядають дизайн на предмет потенційних «єдиних точок відмови» (single points of failure) – компонентів, збій яких «покладе» всю систему. Такі компоненти і вузли намагаються усунути або захистити (наприклад, застосувавши кластеризацію, додавши резервне живлення, тощо). Також проєктують механізми відновлення: плани резервного копіювання даних, відкот транзакцій, автоматичний перезапуск сервісів при збоях. Оцінка наявності та достатності цих механізмів теж входить до перевірки надійності на етапі дизайну. Результатом є певний рівень упевненості, що система зможе працювати без значних простоїв, а в разі збоїв – швидко відновлюватися. Виміряти це повністю можливо лише після впровадження (коли збираються статистичні дані про відмови), але закласти основи надійності потрібно саме при проектуванні.

4. **Сумісність (Compatibility).** Сумісність означає здатність системи коректно працювати у визначеному оточенні та взаємодіяти з іншими системами. Цей критерій охоплює інтероперабельність (можливість обміну даними з зовнішніми системами за узгодженими форматами та протоколами) та коекзистенцію (відсутність конфліктів при спільній роботі з іншими програмами). Оцінювання сумісності під час дизайну полягає в перевірці дотримання стандартів та специфікацій інтеграції. Проєктувальники аналізують, чи відповідають вибрані інтерфейси загальноприйнятим стандартам (наприклад, якщо система повинна інтегруватися через API – чи використовується REST/JSON за стандартними схемами, чи забезпечує система імпорт/експорт даних у форматах,

прийнятних партнерами, тощо). Проводиться аналіз сценаріїв інтеграції: моделюються випадки використання, коли ІС отримує/передає дані іншій системі, та перевіряється, чи покриває дизайн ці випадки (наприклад, передбачено модуль-конвертор даних, або використовується сумісна з іншою системою версія протоколу). Для коекзистенції розглядаються системні вимоги: які версії ОС, бібліотек, оточення потрібні, та чи не виникне конфліктів, якщо система встановлюється поряд з іншими (наприклад, перевірка портів, що використовуються, сумісність з версіями СУБД або сервера застосувань наявними у замовника). Такі перевірки зазвичай реалізуються у вигляді чек-листів сумісності. Відповіді «так/ні» на питання чек-листа (чи використовуються стандартизовані формати? чи передбачено ручне налаштування параметрів інтеграції? тощо) дають первинну оцінку. Якщо виявляється невідповідність (скажімо, система генерує дані у власному форматі, не підтримуваному іншими), вже на дизайні можна запланувати розвиток функціоналу для усунення цього – до впровадження.

5. **Зручність використання (Usability).** Це критерій, що відповідає за ергономічність та легкість освоєння системи для кінцевих користувачів. Оцінювання юзабіліті на ранніх етапах проводиться методами, запозиченими з галузі Human-Computer Interaction. Перш за все, створюються прототипи інтерфейсу користувача (UI prototypes) – статичні макети або інтерактивні прототипи з обмеженою функціональністю. Далі застосовуються юзабілітітести з участю потенційних користувачів або експертів. Один з поширених методів – експертна оцінка за принципами Нільсена (heuristic evaluation): кілька досвідчених експертів з UX переглядають прототипи інтерфейсу і перевіряють їх на відповідність сформульованим евристикам (зрозумілість навігації, консистентність дизайну, запобігання помилкам тощо). Результати такої оцінки дозволяють виявити більшість явних проблем інтерфейсу ще до розробки. Також проводяться тестування з користувачами на прототипах: користувачам пропонуються типові завдання, які вони мають виконати, і спостерігач фіксує, з якими труднощами вони зіштовхуються. Паралельно можуть вимірюватися базові метрики юзабіліті: час виконання завдання, кількість помилок, рівень задоволеності (шляхом опитування після тесту). Хоча прототип – не кінцевий продукт, такі тести дозволяють зробити обґрунтовані висновки про зручність: якщо користувачі плутаються в структурі меню або не розуміють значення іконок, дизайн потрібно переробити на цьому етапі. Крім того, застосовується когнітивний аналіз завдань: дизайнер «програє» уявний сценарій роботи користувача, крок за кроком, щоб перевірити, чи не є робочий процес занадто складним. Інструменти прототипування (Figma, Adobe XD та ін.) дозволяють швидко вносити правки і перевіряти нові ідеї, що значно пришвидшує цикл покращення юзабіліті на стадії проєкту. Важливо зазначити, що деякі аспекти зручності (наприклад, естетичне сприйняття, емоційна задоволеність) можуть потребувати більш розвинутого прототипу або навіть бета-версії, але основи – простоту, інтуїтивність інтерфейсу – цілком реально оцінити завчасно. Отже, методики юзабілітіоцінювання є критично важливими, адже вони дозволяють виправити дизайн «під користувача», уникнувши дорогих доопрацювань після релізу, коли виявиться, що система не приймається аудиторією.

6. **Безпека (Security).** Безпека інформаційних систем включає здатність протистояти несанкціонованому доступу, забезпечувати цілісність, конфіденційність та доступність даних. На етапі проєктування безпеку аналізують через моделювання загроз: команда розробників визначає, які активи системи (дані, функції) можуть бути ціллю атак, які потенційні зловмисники можуть діяти (інсайдер, зовнішній хакер тощо) і які вектори атак можливі. Один з популярних підходів – метод STRIDE, де розглядаються шість типів загроз (Spoofing – підроблення особи, Tampering – модифікація даних, Repudiation – відмова від дій, Information Disclosure – розголошення інформації, Denial of Service – відмова в обслуговуванні, Elevation of Privilege – підвищення привілеїв). Для кожного компонента дизайну (модуль, база даних, інтерфейс) аналізується, які з цих загроз йому притаманні, і чи передбачені засоби протидії. Результат оформлюється як таблиця або діаграма загроз, де навпроти кожної загрози зазначені контрзаходи (наприклад, шифрування для захисту даних, аутентифікація користувачів для запобігання спуфінгу, журналювання дій для відмови від відповідальності і т.д.). Якщо для якоїсь суттєвої загрози не знайдено контрзаходів у поточному дизайні – це сигнал до доробки архітектури (наприклад, додати модуль авторизації, запровадити двофакторну автентифікацію, розділити привілеї). Також проводиться аналіз архітектурних рішень з точки зору безпеки: чи не закладені у структуру можливості обходу безпекових механізмів (наприклад, пряме звернення до бази даних, оминаючи бізнес-логіку), чи розділені належним чином мережеві сегменти (DMZ для зовнішніх інтерфейсів, захищена внутрішня мережа для критичних сервісів). Корисним є залучення безпекового аудитора на етапі проєкту, який проведе design review з акцентом на безпеці. Крім превентивних заходів, дизайн оцінюють на готовність до інцидентів: наприклад, чи є резервне копіювання, чи дублюються критичні компоненти (щоб протидіяти DoS), чи ведуться логи, які допоможуть розслідувати можливий інцидент. Усі ці моменти можна і потрібно врахувати в дизайні. Критерій доступності (Availability), що часто розглядається як частина безпеки, перекривається з критерієм надійності – тут важливі механізми резервування та

відновлення, які вже згадані вище. В підсумку, оцінка безпеки на стадії проєкту – це здебільшого перевірка на відповідність відомим практикам безпеки (стандарти OWASP, рекомендації щодо розмежування прав тощо) та внесення необхідних змін, щоб убезпечити систему ще до написання коду. Це суттєво знижує вразливість системи, адже дешевше спроєктувати безпечну архітектуру одразу, ніж «латати дірки» після того, як тестери або, що гірше, зловмисники знайдуть експлойти.

7. **Підтримуваність (Maintainability).** Підтримуваність – це здатність системи до модифікації, розширення, виправлення зусиллями супроводу. Іншими словами, наскільки легко інженерам супроводжувати код і дизайн системи. На етапі проектування цей критерій оцінюється через аналіз структурної якості проєкту. По-перше, використовується метричний аналіз дизайну: розраховуються такі метрики, як зв'язність (coupling) між модулями, когезія (cohesion) всередині модулів, розміри модулів (кількість функцій, передбачених класів тощо), глибина ієрархії успадкування (для ООП-дизайну). Значення цих метрик порівнюються з емпіричними порогами або значеннями з аналогічних успішних проектів. Наприклад, високий рівень зв'язності та низька когезія – провісники складнощів у супроводі, адже зміна в одному місці коду може тягнути непередбачувані ефекти в інших місцях. Метрики CK (Chidamber & Kemerer), такі як CBO (Coupling Between Objects), LCOM (Lack of Cohesion of Methods) та інші, тут є основним інструментом: про них згадувалося раніше, і вони добре підходять саме для оцінки підтримуваності. По-друге, проводиться рев'ю архітектури та проєктних рішень досвідченими розробниками (peer review): на цьому рев'ю звертають увагу на відповідність принципам чистої архітектури (Clean Architecture), принципам SOLID, на наявність чи відсутність «code smells» у проектних схемах (наприклад, надто великі класи, «божественний» об'єкт, що робить усе, циклічні залежності між модулями і т.д.). Цей аналіз виявляє потенційні проблеми, які ускладнять підтримку, ще до того, як вони матеріалізуються у програмному коді. Наприклад, якщо дизайн передбачає сильне переплетення модулів без чітких шарів абстракції, рецензенти можуть порекомендувати переробити архітектуру, розбити систему на підсистеми з чітко визначеними інтерфейсами – навіть якщо поточний дизайн функціонально працездатний, але надалі його розвиток буде ризикованим. Також оцінюється документованість дизайну: підтримуваність покращується, якщо до архітектурних рішень додаються описи, коментарі, діаграми. Хоча це не чисто технічний аспект, але на рев'ю можна помітити, що деякі рішення «неочевидні» без пояснень – це сигнал, що потрібна краща документація чи спрощення рішення. Нарешті, для критично важливих систем інколи застосовується формальний аналіз модифікованості: створюються моделі залежностей між вимогами та модулями, які дозволяють прикинути, як потенційна зміна вимоги вплине на систему (так званий impact analysis). Якщо модель показує, що одна зміна тягне за собою правки у багатьох модулях – це поганий знак, що підтримуваність низька, і дизайн потребує рефакторингу (введення більш чітких інтерфейсів, меншої зв'язності). Таким чином, оцінка підтримуваності – це певною мірою діагностика архітектурної якості: вона дозволяє підвищити «якість коду» ще до написання коду, шляхом закладання правильної модульної структури.

8. **Портативність (Portability).** Портативність – здатність системи бути перенесеною з одного оточення в інше (змінити платформу, ОС, апаратне забезпечення) при мінімальних зусиллях. Для сучасних інформаційних систем це часто проявляється як вимога підтримувати декілька операційних систем, типів пристроїв або інфраструктур (on-premise та cloud). На стадії дизайну портативність забезпечується і оцінюється шляхом дотримання принципів незалежності від платформи. Проєктне рішення перевіряється: чи не прив'язаний код (або задум архітектури) до конкретних технологій? Наприклад, якщо система розробляється для Windows, оцінюється, що потрібно, аби перенести її на Linux: використання кросплатформних фреймворків, уникання системних викликів, специфічних для однієї ОС, застосування портативних мов програмування. Тут багато залежить від вибору технологічного стеку: проєктувальники обирають ті інструменти, які мають версії під різні платформи, стандартні бібліотеки тощо. Також портативність може стосуватися переносимості даних – наприклад, використання СУБД, яка підтримується на різних системах, або інкапсуляція доступу до БД через шар абстракції, щоб було легше змінити тип бази даних у майбутньому. Методика оцінки тут в основному експертна: рев'ю системних архітекторів, які мають досвід крос-платформної розробки, і які можуть вказати: «Ось це рішення прив'язане до конкретного середовища, треба зробити по-іншому». Допоміжними можуть бути і чек-листи: наприклад, перелік типових непортативних практик (hardcode шляхів у файловій системі, залежність від апаратних особливостей, використання застарілих API). Чим менше таких практик – тим вища портативність дизайну. Крім того, оцінюється структура розгортання: якщо планується переносити систему, варто мати можливість зібрати її під нову платформу з мінімальними змінами – тобто дизайн має передбачати чітку конфігурацію залежностей, можливість переведення у віртуалізоване або контейнерне оточення (наприклад, Docker), що дуже підвищує портативність. На етапі проектування це перевіряється шляхом спроби підготовити "інсталяційний сценарій" для іншого оточення або хоча б концептуально оцінити,

які модулі доведеться переписувати при переносі. Ідеально, якщо виявиться, що система побудована з використанням мови програмування і бібліотек, які вже є мультиплатформними – тоді портативність максимальна, і оцінка покаже, що ризиків мінімум. Якщо ж залежність від платформи висока, рішення може бути – обмежити перелік підтримуваних середовищ або адаптувати дизайн (наприклад, винести платформо-залежні частини у відокремлений модуль-"шар" для спрощення майбутнього портування).

Узагальнення результатів оцінювання. У процесі застосування зазначених методик проєктна команда отримує різного роду результати: кількісні оцінки (метрики, розрахункові показники продуктивності, імовірність відмов), а також якісні висновки експертів (список знайдених проблем юзабіліті, перелік потенційних загроз безпеці, рекомендації щодо архітектури тощо). Важливо ці результати правильно інтерпретувати і покласти в основу прийняття подальших рішень. На етапі проєктування результати оцінювання якості мають стати частиною зворотного зв'язку у процесі розробки: якщо якійсь критерій не задовольняє вимоги (наприклад, моделювання показало, що при прогнозованому навантаженні сервер не впорається), проєкт потрібно скоригувати (обрати потужніший сервер, оптимізувати алгоритм, змінити архітектурний шаблон). Таким чином, оцінювання якості – це ітеративний процес: дизайн → оцінка → висновки → вдосконалений дизайн. В ідеалі, ці ітерації продовжуються, поки всі ключові критерії якості (особливо ті, що визначені в вимогах як критичні) не досягнуть прийнятного рівня. Результати оцінювання також документуються: складаються звіти, діаграми, матриці відповідності вимогам. Надалі, на стадії реалізації і тестування, ці результати слугуватимуть орієнтиром (наприклад, якщо під час ранньої оцінки очікувалось 1000 користувачів одночасно, то при тестуванні навантаження слід перевірити систему саме на цю цифру, щоб підтвердити правильність прогнозів).

Описані методи дозволяють вже на стадії проєктування сформувати обґрунтовані очікування щодо якості майбутньої системи і вжити превентивних заходів для усунення недоліків. Варто підкреслити, що результати оцінювання на цій стадії мають імовірнісний характер – вони не гарантують стовідсоткової якості, але значно знижують ризики. Наприклад, якщо аналіз загроз виконано ретельно, ймовірність того, що критична вразливість залишиться непоміченою, суттєво менша. Якщо ж проігнорувати ці методики, система може формально відповідати технічному завданню, але згодом виникатимуть проблеми (повільна робота, складність додавання нового функціоналу, незадоволені користувачі) – і їх вирішення на етапі супроводу обійдеться значно дорожче.

ОБГОВОРЕННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Здійснений огляд і систематизація критеріїв якості та методик їх оцінювання на етапі проєктування підтвердили загальну гіпотезу про те, що раннє врахування і перевірка якості здатні суттєво підвищити успішність проектів і знизити витрати у подальшому. Отримані результати узгоджуються з висновками попередніх досліджень. Зокрема, виявлене домінування сценарно-орієнтованих методів оцінки архітектури відповідає тому, про що повідомляють Fatima і Lago (2023): у їхньому огляді підтверджено, що саме методи на основі сценаріїв та сценарних аналізів є найпоширенішими при оцінюванні архітектур різних систем [5]. Це логічно, адже сценарії (використання, відмов, атак) дозволяють змоделювати різні аспекти поведінки системи і побачити, чи задовольняє дизайн вимоги до якості. Наші результати показали, що доповнення до сценарних методів дають експертні огляди, прототипування і метрики, і їх комбінація є бажаною. Це перегукується із сучасною тенденцією «shift-left testing»: перевірки, раніше віднесені до етапу тестування (наприклад, юзабіліті-тестування чи аналіз продуктивності), зараз переносяться якнайраніше, ближче до етапу дизайну, що підтверджується і практиками в індустрії [2].

Проведений аналіз літератури виявив також певний розрив між теорією і практикою, який згадувався у попередньому розділі. Наші результати це підтверджують: багато методик, хоч і описані докладно в наукових джерелах, у реальних проектах застосовуються фрагментарно. Наприклад, формальні моделі надійності чи продуктивності – вони можуть дати точні прогнози, але вимагають значних зусиль і експертизи, тому на практиці частіше покладаються на спрощені оцінки. Результати нашого дослідження пропонують набір практично орієнтованих методів (чек-листи, експертні рев'ю), які, можливо, менш строгі, зате простіші у впровадженні. Це відповідає реальним потребам індустрії: як зазначають Varajão та ін. (2022), існуючі моделі оцінки часто залишаються на папері, і потрібні більш практичні, емпірично перевірені інструменти [4]. Отже, наш огляд робить акцент на методиках, що мають підтверджену корисність (наприклад, юзабіліті-прототипування давно зарекомендувало себе у UX-практиці, метрики CK – стандарт де-факто в аналізі коду), аби рекомендувати їх до використання вже на стадії проектування.

Співставляючи наші висновки з іншими роботами, варто відзначити, що у деяких сферах (скажімо, медичні ІС або авіоніка) оцінювання якості на ранніх етапах обов'язкове за стандартами (наприклад, серії DO-178 для авіаційного ПО або IEC 62304 для медичного). Наш систематизований підхід узгоджується з вимогами цих стандартів, що вимагають аналізу ризиків і якості ще на етапі дизайну. Наприклад, для

безпеки життєво важливих систем регуляторні вимоги фактично змушують проводити аналіз відмов та моделювання надійності – без цього проєкт не сертифікують. Таким чином, наші рекомендації мають під собою не лише теоретичне, а й нормативне підґрунтя: вони можуть допомогти організаціям відповідати стандартам якості.

Виявлено також, що деякі критерії якості, зокрема «якість в користуванні» (яка охоплює задоволеність, ефективність користувача), оцінювати на стадії проєктування важче, ніж технічні атрибути на кшталт продуктивності або безпеки. Наш підхід до юзабілітіпрототипування та залучення користувачів на ранніх етапах є спробою подолати цю складність. Це відповідає сучасним agile-практикам, де інкрементальні прототипи демонструються користувачам мало не з перших спринтів. Таким чином, ми підтримуємо тренд на людино-центричне проектування, коли зручність і потреби користувача формують архітектуру з самого початку, а не «прикручуються» потім.

Наприкінці обговорення повернемося до гіпотез, висунутих у розділі «Постановка проблеми». Ми припускали, що раннє оцінювання за визначеними критеріями дозволить зменшити частку дефектів, які зазвичай проявляються на пізніх етапах. Отримані результати підтверджують правомірність цієї гіпотези: проективні методи оцінки якості охоплюють більшість потенційних проблем. Якщо їх ретельно застосувати, то відомі закономірності типу «дефект, знайдений на етапі експлуатації, дорожчий у 100 разів» [2] починають працювати навпаки – дефект просто не доходить до експлуатації, бо був усунений у дизайні. Звичайно, не всі проблеми можна передбачити: завжди можливі непередбачувані фактори, помилки реалізації. Але системна робота з якістю на стадії проектування істотно знижує ризики. Це підтверджується і нашими знахідками (усі наведені методики спрямовані на профілактику проблем), і даними інших авторів. Наприклад, у дослідженні Noel et al. (2022), що стосувалося оцінювання якості медичних ІС, було виявлено, що організації потребують стандартизованого інструменту оцінки для підвищення надійності результатів [8]. Наш узагальнений підхід може розглядатися як крок до такого інструменту: фактично він пропонує каркас, який можна покласти в основу практичного керівництва з забезпечення якості при проєктуванні.

Ще одна гіпотеза стосувалася підтримки попередніх теорій: чи не спростовують наші результати якихось усталених уявлень про фактори успіху проектів. Власне, ми підтвердили, що фактор "якість системи" (System Quality), який присутній у відомій моделі успіху ІС DeLone & McLean, є визначальним: якщо система якісна (надійна, зручна, продуктивна), вона значно більше шансів має привести до задоволеності користувачів і успішності впровадження. Наші результати конкретизують, як саме досягти високої якості системи, доповнюючи модель DeLone & McLean практичними механізмами. Таким чином, можна сказати, стаття підтримує попередні теорії, а не спростовує їх, і розвиває їх у частині методів досягнення задекларованих факторів успіху.

Загалом, обговорення показало, що запропонований у статті підхід відповідає сучасному рівню розвитку методології розробки ІС та запитам практики. Результати дослідження можна розглядати як своєрідний місток між теоретичними напрацюваннями (моделями якості, формальними методами) та їх практичною реалізацією у повсякденній проектній роботі. Це особливо важливо з огляду на прискорення темпів розробки: у світі Agile і DevOps знайти час і ресурси на якість непросто, але наш огляд показує, що існують досить легковагі методи (як-от чек-листи чи евристики), що дозволяють інтегрувати забезпечення якості без надмірних витрат.

Наприкінці обговорення варто торкнутися питання подальших досліджень. Очевидно, що нашу роботу можна розвинути в напрямку більш глибокої емпіричної перевірки: наприклад, провести експеримент за участі двох команд розробників, де одна застосовує описані методики при проектуванні, а інша – ні, і порівняти метрики успішності проекту. Також перспективним є створення програмних інструментів, що автоматизують окремі аспекти раннього оцінювання (автоматичний розрахунок метрик з UML-моделей, інтеграція з CASE-засобами для підказок дизайнеру тощо). Ще одним напрямком є розширення критеріїв якості – наприклад, все більш важливою стає стійкість (sustainability) програмних систем, її теж слід враховувати при дизайні (енергоефективність, екологічність ІС). Це відносно новий критерій, який поки не увійшов до стандартних моделей, але вже обговорюється в науковій спільноті [5].

У цілому, обговорення підтверджує валідність отриманих результатів та їх відповідність існуючим знанням, а також висвітлює, як ці результати можна використати для подальшого поступу в галузі методів оцінювання якості інформаційних систем.

ВИСНОВКИ

У статті узагальнено критерії якості інформаційних систем та показано, якими методиками їх доцільно оцінювати ще на етапі проєктування, коли ключові архітектурні та дизайнерські рішення лише формуються і можуть бути змінені з мінімальними витратами. Проведений огляд підтверджує, що якість інформаційної системи є багатовимірною характеристикою і не зводиться до коректної реалізації функцій: для прогнозованого успіху впровадження необхідно одночасно враховувати функціональну придатність, продуктивність і ефективність використання ресурсів, надійність, сумісність, зручність використання, безпеку, підтримуваність і портативність. Саме

ці критерії, узгоджені з сучасними уявленнями про якість програмного забезпечення, повинні бути явно представлені у вимогах і цілеспрямовано перевірятися на проєктних артефактах, а не відкладатися «до тестування», коли архітектура вже фіксована.

Систематизація методик оцінювання показала, що для більшості критеріїв існують практично застосовні підходи, які дозволяють отримати обґрунтовані результати без реалізації повного продукту: для функціональної придатності ключовими є верифікація та валідація вимог і трасування вимог до архітектури; для продуктивності – аналітичне моделювання, симуляція та ранні вимірювання на прототипах критичних компонентів; для надійності – аналіз сценаріїв відмов, виявлення «єдиних точок відмови», моделювання та закладення механізмів відновлення; для сумісності – перевірка протоколів і форматів даних та аналіз інтеграційних сценаріїв; для юзабіліті – прототипування, експертні евристичні огляди та тестування сценаріїв з користувачами; для безпеки – моделювання загроз, перевірка проєктних рішень на принципи мінімальних привілеїв, сегментації та захисту даних; для підтримуваності – рев'ю архітектури, аналіз модульності й залежностей та застосування метрик дизайну як індикаторів складності супроводу; для портативності – оцінювання рівня платформної прив'язки й сценаріїв перенесення та інкапсуляції залежностей. Сукупно ці методики формують логіку «критерій → проєктний артефакт → процедура оцінки → результат», яка дозволяє організувати якість як системний процес, а не як разову перевірку.

Отримані узагальнення підтримують підхід раннього забезпечення якості, оскільки дають змогу виявляти і коригувати критичні ризики до початку або на ранніх стадіях реалізації: недоопрацьовані вимоги, невідповідність нефункціональних характеристик, уразливості безпеки, потенційну низьку прийнятність інтерфейсу, надмірну складність архітектури, що унеможливить швидкий розвиток системи. У підсумку це знижує імовірність ситуацій, коли система формально «працює», але не відповідає очікуванням користувачів або організації через неприйнятні витрати на супровід, повільну роботу, нестабільність чи ризики інформаційної безпеки. Практична цінність роботи полягає в тому, що запропонована структура критеріїв і відповідних методик може бути використана як основа для плану забезпечення якості під час проєктування, де оцінювання виконується ітеративно: результати перевірок повертаються в проєктні рішення, уточнюють вимоги та роблять компроміси між атрибутами якості прозорими й керованими. Перспективним напрямом подальших досліджень є розроблення інтегрованих інструментів підтримки раннього оцінювання (автоматизація аналізу архітектурних моделей і метрик, поєднання з керуванням вимогами), а також емпірична перевірка ефективності різних комбінацій методик у реальних проєктах з урахуванням домену, масштабів і обмежень команд розробки.

REFEREN4CES

[1] Obed, C. (2025). Understanding ISO/IEC 25010: A comprehensive framework for software quality evaluation. Medium. Retrieved from: https://medium.com/@oczz/understanding-iso-iec-25010-a-comprehensive-framework-for-software-quality-evaluation-ae3cc5250057

[2] Dechand, S. Rule of Ten: How To Cut Your Software Development Costs. *Code Intelligence Blog*. Retrieved from: https://www.code-intelligence.com/blog/rule-of-ten

[3] Miguel, J. P., Mauricio, D., & Rodríguez, G. (2014). A review of software quality models for the evaluation of software products (arXiv:1412.2977). arXiv. Retrieved from: https://arxiv.org/pdf/1412.2977

[4] Varajão, J., Lourenço, J. C., & Gomes, J. (2022). Models and methods for information systems project success evaluation – a review and directions for research. *Heliyon, 8*(12), e11977. https://doi.org/10.1016/j.heliyon.2022.e11977

[5] Fatima, I., & Lago, P. (2023). A Review of Software Architecture Evaluation Methods for Sustainability Assessment. In *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)* (pp. 191–194). IEEE. https://doi.org/10.1109/ICSA-C57050.2023.00050

[6] Gradišnik, M., Beranič, T., & Karakatič, S. (2020). Impact of historical software metric changes in predicting future maintainability trends in open-source software development. *Applied Sciences, no. 10*(13), pp. 4624. https://doi.org/10.3390/app10134624

[7] Zolotukhina, O. A., Negodenko, O. V., Reznyk, S. Yu., & Razina, S. Ya. (2020). *Yakist ta testuvannia informatsiinykh system* [Quality and testing of information systems]. Kyiv : DUT. Retrieved from: http://duikt.edu.ua/uploads/l_2177_57414302.pdf [in Ukrainian].

[8] Ali Kia, A., Mohseni, M., Safdari, R., & Ghazisaeedi, M. (2022). Health information systems evaluation criteria: overview of systematic reviews. *Frontiers in Health Informatics, no. 11*(1), pp. 120. https://doi.org/10.30699/FHI.v11i1.376

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Obed, C. (2025). Understanding ISO/IEC 25010: A comprehensive framework for software quality evaluation. Medium. URL: https://medium.com/@oczz/understanding-iso-iec-25010-a-comprehensive-framework-for-software-quality-evaluation-ae3cc5250057

[2] Dechand, S. Rule of Ten: How To Cut Your Software Development Costs. Code Intelligence Blog. *Code Intelligence Blog*. URL: https://www.code-intelligence.com/blog/rule-of-ten

[3] Miguel, J. P., Mauricio, D., & Rodríguez, G. (2014). A review of software quality models for the evaluation of software products (arXiv:1412.2977). arXiv. URL: https://arxiv.org/pdf/1412.2977

[4] Varajão, J., Lourenço, J. C., & Gomes, J. (2022). Models and methods for information systems project success evaluation – a review and directions for research. *Heliyon, 8*(12), e11977. https://doi.org/10.1016/j.heliyon.2022.e11977

[5] Fatima, I., & Lago, P. (2023). A Review of Software Architecture Evaluation Methods for Sustainability Assessment. In *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)* (pp. 191–194). IEEE. https://doi.org/10.1109/ICSA-C57050.2023.00050

[6] Gradišnik, M., Beranič, T., & Karakatič, S. (2020). Impact of historical software metric changes in predicting future maintainability trends in open-source software development. *Applied Sciences, № 10*(13), С. 4624. https://doi.org/10.3390/app10134624

[7] Золотухіна, О. А., Негоденко, О. В., Резник, С. Ю., Разіна, С. Я. (2020). *Якість та тестування інформаційних систем*. Навчальний посібник. Київ : ДУТ. ISBN: 978-617-7020-72-1 URL: http://duikt.edu.ua/uploads/l_2177_57414302.pdf

[8] Ali Kia, A., Mohseni, M., Safdari, R., & Ghazisaeedi, M. (2022). Health information systems evaluation criteria: overview of systematic reviews. *Frontiers in Health Informatics, № 11*(1), С. 120. https://doi.org/10.30699/FHI.v11i1.376

Дата надходження статті до редакції: 23.11.2025
Дата затвердження статті до друку: 14.12.2025
Опубліковано: 30.12.2025