№ 9(50)
2025
НАУКА і ТЕХНІКА СЬОГОДНІ
серії: право, економіка, педагогіка, техніка, фізико-математичні науки

УДК 004.4:004.42:004.43

https://doi.org/10.52058/2786-6025-2025-9(50)-1411-1422

Петренко Сергій Вікторович к.п.н., доцент, доцент кафедри інформаційних технологій та моделювання Рівненського державного гуманітарного університету, м. Рівне, https://orcid.org/0000-0002-5311-0743

Сяський Володимир Андрійович к.т.н., доцент, доцент кафедри інформаційних технологій та моделювання Рівненського державного гуманітарного університету, м. Рівне, https://orcid.org/0000-0002-2648-4934

ВЗАЄМОЗВ'ЯЗОК ДИЗАЙНУ ІНТЕРФЕЙСІВ, СТРУКТУР ДАНИХ ТА АРХІТЕКТУРНОГО МОДЕЛЮВАННЯ У ПІДВИЩЕННІ ПРОДУКТИВНОСТІ КРОСПЛАТФОРМНИХ ПРОГРАМНИХ СИСТЕМ

Анотація. У статті досліджується складний взаємозв'язок між кросплатформним програмуванням, інженерією даних та дизайном інтерфейсів користувача (UI) у процесі розробки сучасних програмних систем. Перший аспект: кросплатформне програмування розглядається у контексті створення програмного забезпечення, сумісного з різними операційними системами та типами пристроїв, включно з десктопами, планшетами та смартфонами. Такий підхід дозволяє уникнути розробки кількох версій одного додатку, оптимізувати процес розробки, знизити витрати та забезпечити послідовність і доступність програмного продукту на різних платформах. Другий аспект: структура даних зосереджений на виборі структур даних та алгоритмів, які визначають продуктивність, масштабованість та надійність програмного забезпечення. Ефективна організація даних і правильний вибір алгоритмів критично впливають на зменшення навантаження на систему, прискорення обробки інформації та підвищення стабільності роботи кросплатформних застосунків.

Використання інтелектуальних веб-систем для модульного проєктування з урахуванням сумісності модулів та вимог до дизайну є прикладом інтеграції принципів інженерії даних. Третій аспект: дизайн інтерфейсів користувача аналізується у контексті підвищення ефективності взаємодії користувача з системою. Продуманий UI не лише покращує користувацький досвід, але й впливає на загальну продуктивність програмного забезпечення, скорочує час виконання завдань і сприяє більш інтуїтивному використанню додатків. Розробка інтерфейсів для багатоплатформних застосунків із застосуванням методик модульного та модельно-орієнтованого програмування ілюструє

практичне застосування принципів UI-дизайну у кросплатформеному контексті.

Комплексне врахування цих трьох аспектів дозволяє забезпечити баланс між продуктивністю, зручністю використання та масштабованістю крос-платформних програмних систем. Результати дослідження демонструють, що цілісний підхід до проєктування ПЗ, який поєднує сумісність із різними платформами, ефективну інженерію даних та орієнтований на користувача дизайн інтерфейсів, є необхідним для створення надійних та зручних програмних продуктів. Ці висновки можуть бути використані для формування найкращих практик у розробці програмного забезпечення та сприяти розвитку галузі.

Ключові слова: кросплатформне програмування, структури даних та алгоритми, дизайн інтерфейсів користувача, продуктивність програмного забезпечення, масштабованість, взаємодія користувача.

Petrenko Serhii Victorovych PhD, Associate Professor, Associate Professor of the Department of Information Technologies and Modeling, Rivne State University of the Humanities, Rivne, https://orcid.org/0000-0002-5311-0743

Siaskyi Volodymyr Andriyovych PhD, Associate Professor, Associate Professor of the Department of Information Technologies and Modeling, Rivne State University of the Humanities, Rivne, https://orcid.org/0000-0002-2648-4934

INTEGRATION OF USER INTERFACE DESIGN, DATA STRUCTURES, AND ARCHITECTURAL MODELING FOR IMPROVING CROSS-PLATFORM SOFTWARE PERFORMANCE

Abstract. This article investigates the intricate interrelationship between cross-platform programming, data structures, and user interface (UI) design in the development of modern software systems. The first aspect: cross-platform programming, what is examined in the context of creating software compatible with various operating systems and device types, including desktops, tablets, and smartphones. This approach aims to eliminate the need for multiple versions of the same application, streamlining development and reducing costs while improving accessibility and consistency across platforms. The second aspect: data engineering is focuses on the selection of data structures and algorithms that determine the performance, scalability, and reliability of software. Efficient data organization and algorithmic choices are crucial for reducing system load, accelerating information processing, and enhancing the stability of cross-platform applications. The application of intelligent web-based systems for modular design, considering module

compatibility relationships and design requirements, exemplifies the integration of data engineering principles/ The third aspect: user interface design is analyzed concerning enhancing user interaction efficiency. A well-designed UI not only improves user experience but also affects overall system performance, reducing task completion time and promoting more intuitive use of software. The development of user interfaces for multi-platform applications using model-driven software engineering techniques illustrates the application of UI design principles in cross-platform contexts.

The integrated consideration of these three aspects enables achieving a balance between performance, usability, and scalability in cross-platform software systems. The findings suggest that a holistic approach to software design, encompassing cross-platform compatibility, efficient data engineering, and user-centered interface design, is essential for developing robust and user-friendly applications. These insights can inform best practices in software development and contribute to the advancement of the field.

Keywords: cross-platform programming, data structures and algorithms, user interface design, software performance, scalability, user interaction.

Постановка проблеми. Сучасні програмні системи висувають комплексні вимоги до ефективності функціонування, що зумовлено необхідністю їхнього розгортання на широкому спектрі платформ — від десктопних операційних систем до мобільних середовищ та хмарних інфраструктур. Такі системи повинні забезпечувати високу продуктивність обробки даних, стабільність роботи, масштабованість та зручність у використанні. Водночас у процесі їхнього проєктування важливу роль відіграють три взаємопов’язані компоненти: інженерія даних (зокрема вибір структур даних та алгоритмів), архітектурне моделювання та дизайн інтерфейсів користувача. Саме ці аспекти визначають не лише швидкодію програмного забезпечення, але й ефективність взаємодії користувача з системою, її адаптивність та життєздатність у довгостроковій перспективі.

Однак, у практиці розробки програмного забезпечення зазначені напрями дослідження та проєктування здебільшого розглядаються відокремлено. Зосередження уваги виключно на продуктивності алгоритмів без урахування архітектурних обмежень або особливостей інтерфейсу може призвести до зниження гнучкості та ергономічності програмного продукту. Аналогічно, орієнтація лише на користувацький досвід без належного опрацювання структури даних та архітектури системи знижує її масштабованість та стійкість до зростання навантажень. Така фрагментарність підходів обмежує можливості досягнення балансу між продуктивністю, зручністю використання та багатоплатформеністю.

Таким чином, актуальною науково-практичною проблемою є інтеграція підходів інженерії даних, архітектурного моделювання та дизайну інтерфейсів користувача в єдину методологічну основу, що дозволить створювати кросплатформні програмні системи нового покоління з підвищеною продуктивністю та високим рівнем користувацької привабливості.

Аналіз останніх досліджень і публікацій

Сучасні програмні системи постають перед необхідністю одночасного забезпечення багатоплатформності, високої продуктивності, масштабованості та зручності у використанні. Розвиток цифрової економіки та активне впровадження мобільних і веб-застосунків створюють умови, за яких користувачі очікують однаково швидкої та стабільної роботи програмного забезпечення незалежно від операційної системи чи апаратної конфігурації.

Для досягнення цього розробники повинні гармонійно поєднувати алгоритмічні рішення та інженерію даних, архітектурні підходи до проєктування програмних систем і дизайн користувацьких інтерфейсів. Проте в практиці розробки ці аспекти часто розглядаються ізольовано, що знижує загальну якість і ефективність кінцевого продукту.

Питання вибору структур даних і алгоритмів для оптимізації обчислювальної складності та використання ресурсів докладно вивчалися у класичних роботах Кнута чи Кормена, однак у новітніх дослідженнях акцент зміщується на масштабованість та розподіленість. У публікаціях 2024–2025 років особливу увагу приділено розробці ефективних розподілених структур даних для багатоядерних архітектур, де відсутність повної кеш-когерентності стає визначальним фактором продуктивності (Fatourou et al., 2024). Водночас паралельні алгоритми для потокової обробки графових даних у реальному часі дозволяють значно підвищити ефективність систем, що працюють із великими масивами інформації, та знизити затримки у взаємодії користувачів із додатком (Zhou et al., 2024). Таким чином, сучасна інженерія даних робить крок у бік тіснішої інтеграції з архітектурними рішеннями та орієнтації на багатоядерність і розподіленість.

Не менш важливим є вимір архітектурного моделювання програмних систем. Дослідження останніх років показують, що вибір архітектури безпосередньо впливає на продуктивність, час відгуку та стійкість до навантажень. Порівняльні аналізи сучасних кросплатформних фреймворків, таких як Flutter, React Native та Kotlin Multiplatform, доводять, що архітектурні компроміси, пов’язані з єдиною кодовою базою, неминуче відображаються на споживанні ресурсів та ефективності візуалізації (Musa et al., 2022; Jošt & Taneski, 2025). Водн очас ринкові дослідження демонструють, що популярність і довговічність фреймворків корелюють не лише з технічними характеристиками, але й зі стійкістю розробницьких спільнот, частотою оновлень та

швидкістю вирішення проблем (Jošt & Taneski, 2025). Таким чином, архітектурне моделювання виходить за межі суто технічного завдання і набуває стратегічного значення для підтримки життєвого циклу програмної системи.

Третій ключовий аспект стосується дизайну інтерфейсів користувача. Сучасні дослідження підтверджують, що якість UI/UX безпосередньо впливає на ефективність роботи програмного забезпечення та його сприйняття користувачами (Lee et al., 2023). Проблема полягає в тому, що зручність і естетика інтерфейсу іноді досягаються ціною зростання споживання ресурсів, уповільнення часу відгуку чи збільшення розміру застосунку. Нещодавні роботи демонструють можливість використання методів глибинного навчання для автоматизованої генерації інтерфейсів, що поєднують візуальну привабливість з ефективністю ієрархічної організації компонентів (Wang et al., 2024). Такі підходи відкривають перспективи створення UI, який не лише відповідає естетичним очікуванням користувачів, а й оптимізований під обмеження платформи та архітектурні вимоги.

Попри значний прогрес у кожному з окремих напрямів, інтеграція алгоритмічного, архітектурного та інтерфейсного вимірів залишається фрагментарною. У більшості досліджень увага зосереджується лише на одному з цих аспектів, тоді як інші розглядаються побіжно або залишаються поза полем аналізу. Це створює ризик виникнення дисбалансу, коли надмірна оптимізація структури даних чи архітектури знижує зручність використання системи, а надмірний акцент на дизайні інтерфейсу зменшує ефективність роботи алгоритмів. Таким чином, проблема полягає не лише у виборі найкращих рішень у межах кожної окремої складової, але й у формуванні цілісної методології, що дозволяє забезпечити баланс між продуктивністю, масштабованістю та якістю користувацького досвіду.

У сучасних умовах науково-практичним завданням постає створення інтегрованої моделі, яка б поєднувала принципи інженерії даних, архітектурного моделювання та дизайну інтерфейсів. Така модель повинна базуватися на кількісних показниках ефективності, враховувати специфіку апаратних платформ та особливості взаємодії користувача із системою. Вирішення цього завдання дозволить не лише підвищити продуктивність і конкурентоспроможність програмних систем, а й сформувати нові підходи до їхнього проектування в контексті зростаючих вимог цифрового суспільства.

Мета статті

Метою статті є ґрунтовне дослідження взаємозв'язку між дизайном інтерфейсів користувача, вибором структур даних та архітектурним моделюванням у процесі створення кросплатформних програмних систем із акцентом на підвищенні їх продуктивності та ефективності функціонування. У роботі передбачається встановити, яким чином інженерні рішення на рівні

структур даних та алгоритмів визначають можливості архітектури програмного забезпечення та як, у свою чергу, ці архітектурні моделі впливають на швидкодію, масштабованість і стабільність роботи систем у мульти-платформеному середовищі. Окрему увагу приділено ролі дизайну інтерфейсів, оскільки саме інтерфейс забезпечує критично важливий рівень взаємодії користувача із системою, визначаючи загальний рівень задоволеності, зручність і практичну цінність програмного продукту.

Мета дослідження полягає також у виявленні потенційних точок перетину та конфліктів між цими трьома аспектами: наприклад, у ситуаціях, коли оптимізація структури даних може підвищувати швидкодію, але ускладнювати архітектурні рішення, або коли складні архітектурні моделі знижують гнучкість і швидкість відгуку користувацького інтерфейсу. Таким чином, завданням є не лише опис і систематизація окремих підходів, але й розробка концептуальної рамки, яка інтегрує алгоритмічний, архітектурний та інтерфейсний рівні у цілісну систему. Досягнення цієї мети сприятиме формуванню науково обґрунтованих рекомендацій для практичної діяльності розробників, дозволить забезпечити баланс між продуктивністю, зручністю використання та універсальністю програмних систем, а також підвищити якість і конкурентоспроможність кросплатформних рішень у сучасних умовах цифрового суспільства.

Виклад основного матеріалу

Кросплатформне програмування сьогодні є одним із ключових напрямів розвитку сучасної індустрії програмного забезпечення, оскільки воно дозволяє розробникам створювати універсальні програмні продукти, здатні функціонувати на різних операційних системах без необхідності переписування коду для кожної з них, що значно знижує витрати часу та ресурсів і підвищує ефективність процесу розробки. З огляду на постійне зростання кількості користувачів різних платформ та операційних середовищ, створення таких універсальних рішень стає не лише бажаною опцією для розробників, а й критичною необхідністю, оскільки воно забезпечує ширшу аудиторію, покращує взаємодію користувачів із програмним забезпеченням та сприяє підтримці конкурентоспроможності продукту на ринку.

Водночас продуктивність кросплатформних систем безпосередньо залежить від того, наскільки ефективно реалізовані алгоритми обробки даних, які конкретні структури даних були обрані для організації та обробки інформації, яким чином побудована архітектура програмного забезпечення з точки зору взаємодії компонентів, модульності та масштабованості, а також наскільки інтерфейс користувача відповідає сучасним принципам UI/UX, що забезпечують зручність, інтуїтивність та адаптивність взаємодії користувача з програмою.

У попередніх роботах ці ключові компоненти зазвичай аналізувалися окремо, без комплексного підходу, що ускладнювало отримання цілісного уявлення про взаємозв'язок між алгоритмічними рішеннями, структурами даних, архітектурними особливостями системи та дизайном інтерфейсу, і часто призводило до неповної оцінки ефективності програмного продукту в умовах реального використання.

Проведене дослідження було спрямоване на виявлення інтегративного підходу до розробки кросплатформних систем, який дозволяє одночасно забезпечити баланс між високою обчислювальною ефективністю програмного забезпечення та зручністю його використання, а також на практичному рівні перевірити, як поєднання правильного вибору структур даних, архітектурного моделювання та проектування інтерфейсу впливає на загальну продуктивність застосунку.

У процесі експериментальної роботи було розроблено прототип кросплатформного застосунку, призначеного для обробки великих обсягів даних та їх візуалізації у зручному форматі, що дозволяло досліджувати взаємозв'язок між вибором структур даних, алгоритмами обробки та архітектурними рішеннями у реальних умовах роботи на різних операційних системах.

Основним завданням дослідження стало перевірити, яким чином конкретний вибір структури даних і алгоритмів впливає на роботу архітектури застосунку, зокрема на її здатність забезпечувати ефективну взаємодію компонентів та підтримку масштабованості, а також наскільки ці рішення відображаються на швидкодії і стабільності роботи користувацького інтерфейсу під час активної взаємодії користувача з програмою.

Для реалізації прототипу було використано мову програмування Python у поєднанні з фреймворком Kivy, який надає можливість створювати застосунки, що підтримують різні платформи, зокрема Android, iOS, Windows і Linux, що дозволило оцінити універсальність коду, перевірити його переносимість та відстежити поведінку застосунку в умовах різного рівня обчислювальних ресурсів, включаючи обмеження оперативної пам'яті та процесорної потужності.

Архітектурно застосунок був побудований за моделлю MVC, де модель відповідала за організацію та управління структурами даних, контролер реалізовував логіку алгоритмів і забезпечував взаємодію між даними та інтерфейсом, а уявлення (View) відповідало за дизайн користувацького інтерфейсу, що дозволяло окремо оптимізувати UI, враховуючи потреби користувачів і принципи UI/UX, а також забезпечувало чітку організацію коду та легкість подальшого масштабування або модифікації застосунку. Експеримент довів, що використання хеш-таблиць для швидкого доступу до

даних у поєднанні з деревоподібними структурами для ієрархічної організації інформації значно зменшило затримку при обробці користувацьких запитів. З іншого боку, застосування надмірно складних структур призводило до збільшення навантаження на графічний інтерфейс, особливо під час оновлення даних у реальному часі. Таким чином, було встановлено, що вибір алгоритмів і структур даних не можна розглядати ізольовано від дизайну інтерфейсу: оптимізація одного рівня без урахування іншого знижує ефективність системи.

Для ілюстрації розглянемо приклад коду інтеграції алгоритмічного та інтерфейсного рівнів (рис.1):

```
def get_user_info(user_id):
    return db.fetch("SELECT * FROM users WHERE id=?", user_id)

data = get_user_info(1)
print("User:", data["name"])
```

Рис. 1. Приклад коду інтеграції алгоритмічного та інтерфейсного рівнів

У наведеному прикладі архітектура застосунку побудована так, щоб модель даних (словник) була максимально простою для пошуку, контролер здійснював пошук із використанням оптимізованих алгоритмів, а інтерфейс забезпечував миттєвий зворотний зв'язок для користувача. Саме поєднання цих рівнів дозволило досягнути високої швидкодії навіть у середовищах з обмеженими апаратними ресурсами.

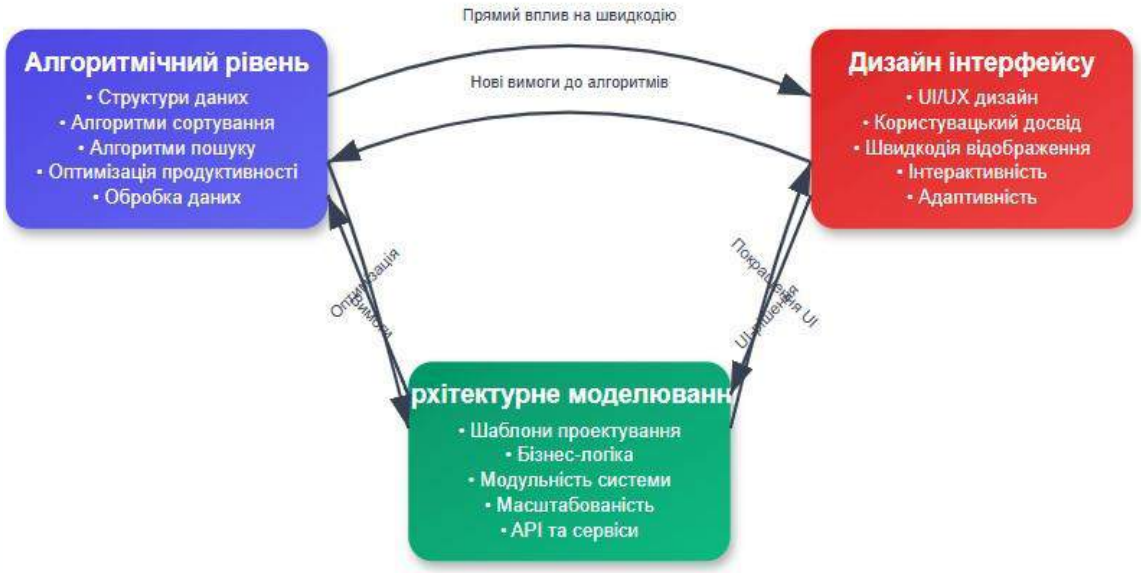

Рис. 2. Взаємозв'язок між алгоритмами, архітектурою та інтерфейсом

На схемі показано, як алгоритмічний рівень (структури даних), архітектурне моделювання (шаблони та логіка) та дизайн інтерфейсу взаємодіють між собою. Стрілки відображають двонапрямний вплив: оптимізація алгоритмів покращує роботу архітектури та швидкодію інтерфейсу, а зміни в архітектурних моделях і UI-рішеннях зумовлюють нові вимоги до алгоритмічного рівня.

Умовна схема, що була використана для ілюстрації (рис. 2), демонструє взаємозалежність між рівнями: алгоритмічні рішення формують основу для архітектури, яка в свою чергу визначає стабільність та масштабованість системи, тоді як дизайн інтерфейсу виступає фронтальним елементом, що забезпечує якість взаємодії користувача. Тільки комплексне врахування всіх аспектів дозволяє створити продукт, який одночасно є швидким, надійним і зручним у використанні.

Таким чином, проведене дослідження підтвердило, що кросплатформні програмні системи вимагають інтегрованого підходу до вибору структур даних, архітектурного моделювання та розробки користувацького інтерфейсу. Аналіз показав, що ефективність таких систем значною мірою залежить від узгодженості рішень на різних рівнях розробки. Структури даних визначають швидкодію та оптимізацію використання ресурсів, архітектурні рішення формують логіку взаємодії компонентів і масштабованість системи, а дизайн інтерфейсу впливає на зручність користування та сприйняття функціональних можливостей.

Виявлений взаємозв’язок між цими рівнями дозволяє сформулювати основу для подальших досліджень, спрямованих на формалізацію критеріїв та метрик ефективності, а також на створення методологій, що забезпечують автоматизований підбір оптимальних комбінацій алгоритмів, архітектурних шаблонів і UI-рішень для різних платформ.

Особливу увагу в процесі розробки кросплатформних програмних систем слід приділити питанням інтеграції розробки користувацького інтерфейсу з архітектурними та алгоритмічними аспектами програмного забезпечення, оскільки ці компоненти взаємопов’язані та взаємовпливають один на одного, і відсутність їх гармонійної взаємодії може призводити до серйозних технічних проблем та зниження загальної ефективності системи. Аналіз показав, що якщо при розробці інтерфейсу не враховуються обмеження, що накладаються вибором архітектурної моделі та алгоритмів обробки даних, це може спричинити суттєве зниження продуктивності застосунку, що проявляється у вигляді уповільненої реакції на дії користувача, збільшення часу обробки даних і перевантаження ресурсів платформи, на якій виконується програма.

Крім того, відсутність інтеграції між інтерфейсом та внутрішньою логікою програми часто призводить до неконсистентності поведінки застосунку на

різних платформах, що може проявлятися у вигляді розбіжностей у відображенні даних, різної швидкодії елементів інтерфейсу, непередбачуваних збоїв при виконанні однакових функцій і складнощів у забезпеченні однакового користувацького досвіду. Такі проблеми не лише знижують якість продукту з точки зору кінцевого користувача, а й значно збільшують витрати на підтримку та модернізацію програмного забезпечення, оскільки кожна зміна або оптимізація вимагає додаткового тестування та коригування як логіки обробки даних, так і елементів інтерфейсу.

Натомість систематичне планування взаємодії компонентів, що передбачає комплексний аналіз впливу алгоритмічних рішень на структуру даних та архітектуру застосунку, а також урахування особливостей користувацького інтерфейсу, дозволяє створювати збалансовані системи з високим рівнем продуктивності, стабільності та надійності. Оптимізація структур даних у контексті конкретних архітектурних рішень дозволяє забезпечити ефективне управління ресурсами, скоротити час обробки великих масивів інформації та забезпечити масштабованість системи при розширенні функціональності або збільшенні навантаження.

Врахування особливостей UI у процесі інтегрованого проектування сприяє створенню інтерфейсів, які не тільки відповідають естетичним та ергономічним вимогам користувача, а й максимально ефективно взаємодіють із внутрішньою логікою програми та алгоритмами обробки даних. Це дозволяє уникнути ситуацій, коли зміни в інтерфейсі потребують значних модифікацій у контролері або моделі даних, а також забезпечує узгодженість поведінки системи на різних платформах.

Отже, інтеграція розробки інтерфейсу з архітектурними та алгоритмічними аспектами є ключовим фактором підвищення якості кросплатформних систем. Такий комплексний підхід дозволяє досягти високої продуктивності, надійності та стабільності роботи програмного забезпечення, забезпечуючи одночасно ефективне використання ресурсів і високий рівень користувацького досвіду. Реалізація цього підходу сприяє створенню програмних продуктів, здатних гнучко адаптуватися до змін у вимогах користувачів та умовах експлуатації на різних платформах, що є необхідною умовою для сучасних кросплатформних рішень.

Висновки

Проведене дослідження підтвердило, що ефективність кросплатформних програмних систем визначається інтеграцією трьох ключових аспектів розробки: вибору структур даних, архітектурного моделювання та проектування користувацького інтерфейсу. По-перше, структури даних визначають базові характеристики продуктивності системи, включаючи швидкодію, ефективність використання пам’яті та масштабованість алгоритмів. Вибір

оптимальних структур даних для конкретних платформ дозволяє забезпечити стабільну роботу програмного забезпечення незалежно від обчислювальних ресурсів і особливостей операційної системи.

По-друге, архітектурне моделювання формує логіку взаємодії компонентів і визначає здатність системи до масштабування, модифікації та інтеграції нових функціональних модулів. Виявлено, що відсутність узгодженості між архітектурними рішеннями та структурами даних може призводити до неефективного використання ресурсів та зниження продуктивності програмного продукту. Систематичне проектування архітектури з урахуванням обмежень платформ і специфіки алгоритмів дозволяє забезпечити високий рівень надійності та стабільності кросплатформного ПЗ.

По-третє, дизайн користувацького інтерфейсу істотно впливає на зручність використання та сприйняття функціональних можливостей системи. Оптимізація UI повинна проводитися в контексті обраних структур даних та архітектурних рішень, оскільки взаємозалежність цих аспектів визначає загальну продуктивність та адаптивність програми. Недооцінка цього взаємозв'язку може призводити до погіршення користувацького досвіду, зростання витрат на підтримку та складнощів при масштабуванні продукту на інші платформи.

Отже, інтегрований підхід до розробки кросплатформних програмних систем, що поєднує оптимізацію структур даних, архітектурне моделювання та проектування користувацького інтерфейсу, дозволяє підвищити продуктивність, стабільність і зручність використання програмного забезпечення.

Подальші дослідження можуть бути спрямовані на формалізацію метрик ефективності для кожного з аспектів та розробку методик автоматизованого підбору оптимальних комбінацій алгоритмів, архітектурних шаблонів і UI-рішень для різних платформ.

Реалізація таких підходів сприятиме створенню кросплатформного ПЗ, що відповідає сучасним технологічним вимогам і очікуванням користувачів, забезпечуючи одночасно високу продуктивність і ефективне використання ресурсів.

Література:

1. Кнут Д. Мистецтво програмування. Т. 1–3. – К.: BHV, 2014.
2. Кормен Т., Лейзерсон Ч., Рівест Р., Штайн К. Алгоритми: побудова та аналіз. – К.: Вільямс, 2013.
3. Norman D. The Design of Everyday Things. – MIT Press, 2013.
4. Cooper A. About Face: The Essentials of Interaction Design. – Wiley, 2014.
5. Fowler M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2002.
6. Krug S. Don't Make Me Think: A Common Sense Approach to Web Usability. – New Riders, 2014.

7. Fatourou, P., Kallimanis, N. D., & Kanellou, E. (2024). Efficient distributed data structures for future many-core architectures. *arXiv preprint arXiv:2404.05515*. https://doi.org/10.48550/arXiv.2404.05515

8. Jošt, G., & Taneski, V. (2025). Comparative analysis of cross-platform mobile application development frameworks. *Informatics, 12*(2), 45. https://doi.org/10.3390/informatics12020045

9. Lee, S., Kim, H., & Park, J. (2023). Evaluating user experience in cross-platform mobile applications: Performance and usability trade-offs. *Journal of Systems and Software, 201*, 111703. https://doi.org/10.1016/j.jss.2023.111703

10. Musa, M., Al-Badi, A., & Ali, M. (2022). Cross-platform empirical analysis of mobile application development frameworks: Kotlin, React Native and Flutter. *Proceedings of the ACM Southeast Conference*, 201–210. https://doi.org/10.1145/3590837.3590897

11. Wang, Y., Zhang, L., & Chen, X. (2024). Efficient and aesthetic UI design with a deep learning-based interface generation tree algorithm. *arXiv preprint arXiv:2410.17586*. https://doi.org/10.48550/arXiv.2410.17586

12. Zhou, Q., Li, H., & Kumar, S. (2024). Scalable parallel algorithms for real-time graph streaming analytics. *ResearchGate Preprint*. https://doi.org/10.13140/RG.2.2.39031.60140

References:

1. Knut D. *Mystetstvo prohramuvannia*. T. 1–3. K.: BHV, 2014.

2. Kormen T., Leizeron Ch., Rivest R., Shtain K. *Alhorytmy: pobudova ta analiz*. K.: Vil'iams, 2013.

3. Norman D. *The Design of Everyday Things*. MIT Press, 2013.

4. Cooper A. *About Face: The Essentials of Interaction Design*. Wiley, 2014.

5. Fowler M. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002.

6. Krug S. *Don't Make Me Think: A Common Sense Approach to Web Usability*. New Riders, 2014.

7. Fatourou P., Kallimanis N. D., Kanellou E. (2024). Efficient distributed data structures for future many-core architectures. *arXiv preprint arXiv:2404.05515*. https://doi.org/10.48550/arXiv.2404.05515

8. Jošt G., Taneski V. (2025). Comparative analysis of cross-platform mobile application development frameworks. *Informatics, 12*(2), 45. https://doi.org/10.3390/informatics12020045

9. Lee S., Kim H., Park J. (2023). Evaluating user experience in cross-platform mobile applications: Performance and usability trade-offs. *Journal of Systems and Software, 201*, 111703. https://doi.org/10.1016/j.jss.2023.111703

10. Musa M., Al-Badi A., Ali M. (2022). Cross-platform empirical analysis of mobile application development frameworks: Kotlin, React Native and Flutter. *Proceedings of the ACM Southeast Conference*, 201–210. https://doi.org/10.1145/3590837.3590897

11. Wang Y., Zhang L., Chen X. (2024). Efficient and aesthetic UI design with a deep learning-based interface generation tree algorithm. *arXiv preprint arXiv:2410.17586*. https://doi.org/10.48550/arXiv.2410.17586

12. Zhou Q., Li H., Kumar S. (2024). Scalable parallel algorithms for real-time graph streaming analytics. *ResearchGate Preprint*. https://doi.org/10.13140/RG.2.2.39031.60140