https://orcid.org/0009-0005-2094-2876
https://orcid.org/0000-0002-9639-3544

УДК 004.4`2

Д.С. ЛАГУНОВ*, О.Є. КОВАЛЕНКО*

МОДЕЛІ ТА ЗАСОБИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРОЦЕСІВ РОЗРОБКИ ПРОГРАМНИХ КОМПОНЕНТІВ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ

*Національний університет біоресурсів і природокористування України, м. Київ, Україна

Анотація. *Інтернет речей еволюціонує від ізольованих сенсорних рішень до розподілених систем, у яких логіка, дані та інтелектуальні рішення переміщуються між пристроями, периферійними вузлами та хмарою. Сучасні IoT-системи все частіше мають ознаки інтелектуальних кіберфізичних середовищ, у яких оброблення даних, прийняття рішень і керування відбуваються розподілено. Замість монолітних застосунків формуються екосистеми програмних компонентів, що взаємодіють через мережу та виконуються на пристроях із різними ресурсними можливостями, на периферії й у хмарі. Таке середовище природно описується як континуум IoT-Edge-Cloud. Це підвищує рівень вимог до ефективності розробки програмних компонентів, оскільки ускладнюються інтеграція, тестування, розгортання та супровід таких систем у гетерогенних середовищах. У статті систематизовано моделі та засоби підвищення ефективності розробки компонентів інтелектуальних IoT-систем у межах континууму IoT-Edge-Cloud. Розглянуто сучасні архітектурні підходи, зокрема мікросервісні та serverless-рішення, а також континуумні принципи розміщення компонентів. Окрему увагу приділено інженерії вимог, керованим моделями підходам, dataflow-орієнтованим ML-конвеєрам, тестуванню IoT-систем, контейнеризації та інструментам керування даними в континуумі. Сформульовано інтегровану модель підвищення ефективності, яка поєднує вимоги, архітектуру, автоматизацію артефактів, тестові стратегії, практики розгортання та безпекові механізми включно з аудитом прошивок і федеративним навчанням для IDS. Визначено ключові прогалини та напрями подальших досліджень, пов'язані з відтворюваністю тестування, розміщенням компонентів у континуумі, життєвим циклом Edge-AI та операційною складністю федеративних підходів.*
Ключові слова: *інтернет речей, штучний інтелект, керовані моделями процеси розробки, машинне навчання, мікросервіси.*

Abstract. *The Internet of Things is evolving from isolated sensor solutions to distributed systems in which logic, data, and intelligent solutions move between devices, peripheral nodes, and the cloud. Modern IoT systems increasingly have the characteristics of intelligent cyber-physical environments in which data processing, decision-making, and control are distributed. Instead of monolithic applications, ecosystems of software components are formed. They interact via the network and run on devices with different resource capabilities — at the periphery and in the cloud. Such an environment is naturally described as the IoT–Edge–Cloud continuum. This increases the level of requirements for the efficiency of software component development, since integration, testing, deployment, and maintenance of such systems in heterogeneous environments become more difficult. The article systematizes models and methods for increasing the efficiency of developing components of intelligent IoT systems within the IoT-Edge-Cloud continuum. Modern architectural approaches are considered, in particular microservices and serverless solutions, as well as continuum principles of component placement. Particular attention is paid to requirements engineering, model-driven approaches, dataflow-oriented ML pipelines, IoT system testing, containerization, and data management tools in the continuum. An integrated performance improvement model is formulated that combines requirements, architecture, artifact automation, test strategies, deployment practices, and security mechanisms, including firmware auditing and federated learning for*

ISSN 1028-9763. Математичні машини і системи. 2026. № 3

IDS. Key gaps and areas for further research related to test reproducibility, component placement in the continuum, Edge-AI lifecycle, and operational complexity of federated approaches are identified.
Keywords: *Internet of Things, artificial intelligence, model-driven development processes, machine learning, microservices.*

DOI: 10.34121/1028-9763-2026-3-60-71

1. Вступ

Сучасні системи інтернету речей (IoT) усе частіше мають ознаки інтелектуальних кіберконвергентних середовищ, у яких оброблення даних, прийняття рішень і керування відбуваються розподілено. Замість монолітних застосунків формуються екосистеми програмних компонентів, що взаємодіють через мережу і виконуються на пристроях із різними ресурсними можливостями, на периферії та в хмарі. Таке середовище природно описується як IoT-Edge-Cloud континуум (рис. 1), для якого характерні динаміка розміщення, варіативність каналів зв'язку, неоднорідність апаратних платформ і вимоги до низької затримки [1]. У результаті ефективність розробки програмних компонентів визначається не лише швидкістю написання коду, а й якістю інженерії вимог, здатністю архітектури підтримувати незалежну еволюцію модулів, відтворюваністю тестування, швидкістю та безпечністю розгортання, а також керованістю експлуатації.

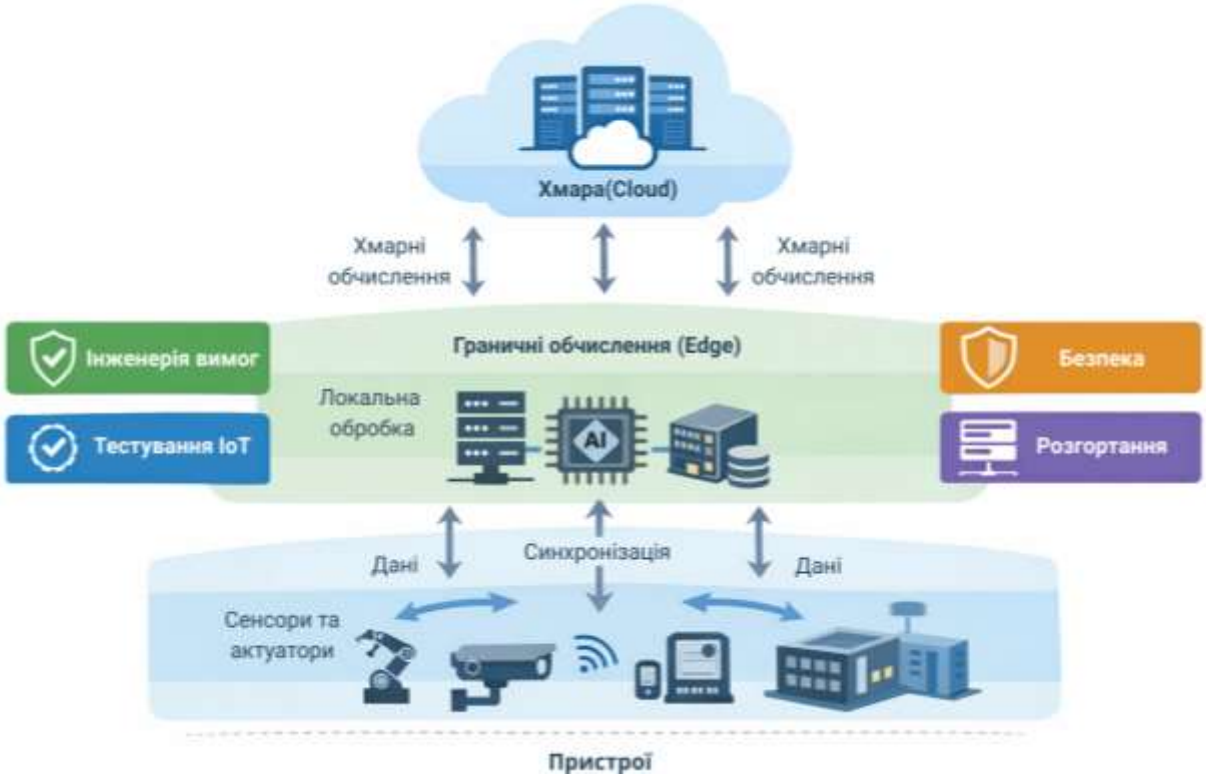

Рисунок 1 — Архітектура IoT-Edge-Cloud континууму

Одним із ключових трендів, що підсилює актуальність проблеми, є інтелектуалізація граничних (edge) обчислень. Перенесення частини аналітики та прийняття рішень до edge-рівня формують окремі вимоги до компонентів: вони мають бути адаптивними, оновлюваними, стійкими до деградації мережі та здатними працювати за обмежених ресурсів [2]. Паралельно систематичні огляди Edge-AI наголошують, що перенесення machine learning/artificial intelligence на периферію зумовлене затримками, приватністю і надійністю, але потребує архітектурних механізмів для гетерогенних платформ і масштабування [3].

Отже, інтелектуальний IoT не лише ускладнює розробку компонентів, а й робить ефективність даної розробки критичною умовою конкурентоспроможності системи.

Ефективність розробки тісно пов'язана з архітектурним стилем. У літературі мікросервісні архітектури розглядаються як один із базових інструментів підвищення модульності та незалежності в IoT-системах [4], а також як перспективний напрям еволюції архітектур IoT-застосунків із відповідними рекомендаціями щодо вибору мікросервісів [5]. Водночас зростає інтерес до поєднання мікросервісів і безсерверних (serverless) архітектур як способу підсилити ізоляцію компонентів, зокрема в безпекових сценаріях [6]. Проте архітектурні переваги супроводжуються новими витратами, серед яких складність інтеграцій, моніторингу та тестування у розподіленому середовищі [7]. Для інтелектуальних IoT-систем ці витрати додатково збільшуються через необхідність підтримки життєвого циклу ML-моделей та даних.

Важливою передумовою ефективності є якісна інженерія вимог. Розвиток IoT супроводжується зростанням нефункціональних вимог та залежності результату від контексту експлуатації, що робить процес інженерії вимог критичним для успішного проєктування IoT-систем [8]. Систематичні огляди підкреслюють, що у практиці RE для IoT залишаються прогалини, пов'язані зі складністю предметної області та різноманітністю потреб користувачів, а також із недостатньою стандартизацією методик і інструментів [9]. Для підвищення ефективності розробки такі прогалини мають бути подолані через переведення вимог до архітектурних рішень, тестових цілей і сценаріїв розгортання.

Окремим аспектом є конвергенція IoT із системами ситуаційного управління, яка змінює постановку задач керування та підсилює значення ситуаційних моделей і контексту [10]. Ситуаційність робить неефективними вузькі підходи, які розглядають компоненти ізольовано від середовища, та підштовхує до інтеграції моделей і засобів, що охоплюють повний життєвий цикл: від вимог до експлуатації та оновлень.

Метою статті є узагальнення моделей і засобів, що підвищують ефективність розробки програмних компонентів інтелектуальних IoT-систем. Завданням є: систематизувати архітектурні та континуумні підходи; охарактеризувати роль інженерії вимог і керованої моделями інженерії (model-driven engineering); розглянути dataflow-орієнтовані ML-конвеєри; узагальнити практики тестування IoT-систем; описати інструменти розгортання, керування даними та контейнеризації; проаналізувати безпекові аспекти включно з аудитом прошивок і федеративним навчанням для систем виявлення вторгнень (IDS); сформувати інтегровану модель ефективності та визначити прогалини для подальших досліджень.

Огляд сформовано як систематизований тематичний аналіз, у якому джерела згруповано за кластерами життєвого циклу та інженерних задач. Підхід до побудови огляду узгоджується з підходами аналізу поточного стану проблеми (state-of-the-art review) для мікросервісних IoT-архітектур, де визначаються категорії архітектурних рішень, типи компонентів і вплив на нефункціональні властивості [4], а також із систематичними оглядами щодо тестування IoT-систем, де класифікуються цілі тестування, підходи, інструменти та виклики [7].

Для забезпечення логічної цілісності використано класифікацію за такими кластерами: 1) етап життєвого циклу (вимоги, проєктування, реалізація, тестування, розгортання, експлуатація, еволюція); 2) тип моделі або методу (архітектурні, модельно-керовані, dataflow-орієнтовані, оптимізаційні); 3) тип засобів (фреймворки, інструменти тестування, платформи розгортання, механізми безпеки); 4) очікуваний ефект на ефективність (скорочення часу розробки, зменшення переробок, підвищення відтворюваності, зниження витрат на експлуатацію); 5) ризики та обмеження (складність розподіленості, гетерогенність, загрози безпеки, firmware-вразливості, комунікаційні витрати федеративного навчання).

2. Архітектурні моделі та континуум IoT-Edge-Cloud як основа ефективності

2.1. Принципи розміщення компонентів у континуумі

IoT-Edge-Cloud континуум описує середовище, в якому функції, дані та керування розміщуються на різних рівнях із різними обмеженнями. Перспектива континууму змінює принципи компонентної інженерії: замість статичного розгортання виникає потреба враховувати міграцію компонентів, зміну топології, доступність ресурсів та мережеві умови. У систематичному огляді континуум-систем підкреслюється, що ключові виклики пов'язані зі складністю керування, інтеграцією технологій підтримки і наявністю відкритих проблем у стандартизації та автоматизації розгортання [1]. Для ефективності розробки це означає, що компонент має проєктуватися із закладеними переносимістю, механізмами конфігурації, спостережуваністю та здатністю працювати при деградації мережі.

Континуумна модель також впливає на вибір інструментів керування даними й інтеграції. У таких системах дані можуть зберігатися й оброблятися в різних вузлах, що потребує узгоджених протоколів, політик і механізмів керування. Практичні фреймворки керування даними в континуумі прагнуть зменшити операційний тягар і, відповідно, підвищити ефективність супроводу, що прямо позначається на загальній ефективності життєвого циклу компонентів [11].

2.2. Мікросервісні архітектури для IoT: ефект і витрати

Мікросервісна архітектура розглядається як одна із провідних для побудови модульних IoT-систем. Її основні переваги у контексті ефективності полягають у можливості незалежного розвитку сервісів, паралельності командної розробки та керованого масштабування. Огляд поточного стану проблеми (state-of-the-art review) мікросервісних архітектур для IoT узагальнює сучасний стан підходу, формує таксономію та демонструє, що мікросервіси часто поєднуються з контейнеризацією, механізмами спостережуваності й автоматизації CI/CD, що прямо впливає на швидкість упровадження змін [4]. Водночас підхід породжує витрати: ускладнюється інтеграція, зростає кількість точок відмови і з'являється потреба у складніших практиках тестування, особливо інтеграційного та end-to-end.

Систематичний огляд [5] підтверджує, що архітектури IoT мають тенденцію до прийняття мікросервісного підходу, але при цьому існують прогалини у практиках проєктування та критеріїв вибору архітектур, що потребує подальшого узагальнення рекомендацій і напрямів розвитку. Для підвищення ефективності це означає необхідність не лише декомпозиції на сервіси, а й формалізації критеріїв меж сервісів, контрактів взаємодії, політик сумісності версій і підходів до тестування.

Отже, перевага мікросервісів полягає в тому, що окремі компоненти можна розробляти, тестувати, оновлювати й масштабувати незалежно. Якщо змінюється логіка обробки телеметрії, не потрібно перебудовувати всю систему. Якщо збільшується навантаження лише на сервіс прийому даних, можна масштабувати саме його, а не весь програмний комплекс. Проте водночас даний підхід потребує додаткової уваги до питань взаємодії сервісів, спостережуваності й тестування.

2.3. Поєднання microservices і serverless підходів

У багатьох IoT-сценаріях навантаження є нерівномірним. Наприклад, більшість часу система може отримувати невелику кількість подій, але в момент аварії, пікового споживання або масового оновлення пристроїв кількість запитів різко зростає. У таких випадках корисним є безсерверний підхід. Він дозволяє запускати окремі функції у відповідь на події без постійного керування серверною інфраструктурою. Це може бути функція перевірки доступу, обробки аномалії, відправлення сповіщення або трансформації даних.

Поєднання microservices і serverless використовується як для прискорення виходу продукту на ринок, так і для розв'язання безпекових задач через ізоляцію компонентів за допомогою чітких інтерфейсів доступу. У роботі, присвяченій мікросервісно-безсерверній архітектурі для безпечної IoT-системи, продемонстровано приклад реалізації, де безпекові вимоги інтегруються в архітектуру через організацію компонентів і механізми захисту [6]. Це підкреслює, що ефективність не може розглядатися окремо від безпеки: швидка доставка функцій повинна супроводжуватися безпековими політиками, контролем доступу та верифікацією взаємодій. Також автори показують, що поєднання serverless і мікросервісної архітектури може зменшити складність платформи, підтримати автоматичне масштабування та знизити операційні витрати, якщо обчислювальні ресурси виділяються залежно від кількості запитів.

Але важливо розуміти, що такі архітектурні підходи не є універсальним вирішенням усіх проблем. Вони створюють і нові вимоги. Потрібно правильно проєктувати API, визначати контракти між сервісами, контролювати версії інтерфейсів, забезпечувати авторизацію, шифрування, логування, відстеження запитів і моніторинг стану компонентів. Якщо цього не зробити, система може стати не простішою, а навпаки складнішою для супроводу. Тому ефективність архітектури залежить не лише від вибору технологій, а й від якості інженерних рішень.

2.4. Конвергенція IoT та систем ситуаційного управління

Конвергенція IoT із системами ситуаційного управління формує додаткову рамку для проєктування інтелектуальних компонентів. У такій постановці ключовим стає не лише збір даних, а й інтерпретація контексту та формування управлінських впливів відповідно до ситуації. Робота про конвергенцію IoT та систем ситуаційного управління підкреслює значення кіберконвергентних аспектів і гетерогенних мереж, що природно поєднується з континуумною перспективою та вимогами до реактивності компонентів [10]. Для ефективності розробки це означає потребу у моделях, які дозволяють описувати ситуації, правила реакції, джерела контексту і трасувати їх до вимог, архітектурних рішень і тестових сценаріїв.

Саме тут стає зрозуміло, чому всі попередні підходи потрібно розглядати разом. Континуум IoT-Edge-Cloud дає змогу гнучко розміщувати обробку даних. Мікросервіси дозволяють поділити систему на функціональні компоненти. Безсерверний підхід забезпечує швидку реакцію на окремі події. А ситуаційне управління перетворює зібрані дані на обґрунтовані рішення. У результаті IoT-система стає не просто технічною інфраструктурою, а інтелектуальним середовищем підтримки рішень.

3. Інженерія вимог як механізм зниження переробок і технічного боргу

3.1. Роль інженерії вимог у життєвому циклі IoT-систем

Інженерія вимог (Requirements Engineering, RE) є ключовою передумовою ефективності, оскільки помилки або неповнота вимог у IoT-середовищах швидко перетворюються на надмірну роботу, інтеграційні дефекти та проблеми експлуатації. У роботі Коваленка О.Є., Смолій В.М. та Лі Л. наголошується, що інженерія вимог є визначальним процесом інженерії систем, а її критичність зростає в умовах інтенсивного розвитку IoT [8]. Для IoT характерні розширені нефункціональні вимоги включно з надійністю, безпекою, затримкою, енергоспоживанням, масштабованістю і приватністю, що зумовлює необхідність системного підходу до формалізації та узгодження вимог.

Ефективність RE у IoT залежить від здатності врахувати контекстність експлуатації. Вимоги часто є залежними від сценаріїв використання, мережевої доступності, доменної

моделі, умов середовища та поведінки користувачів. У цьому контексті підходи до ситуаційного управління можуть розглядатися як методологічна основа формування вимог до реактивності й адаптивності компонентів [10].

3.2. Систематизація практик RE і прогалин

Систематичне мапування інженерії вимог (RE) для IoT підкреслює, що практики RE у цій сфері є складними через широту аспектів IoT, зокрема через різноманітність потреб користувачів, що прямо впливає на якість і керованість вимог [9]. Водночас саме RE дає змогу зменшити витрати на переробки та забезпечити відтворюваність інженерних рішень. Практичний висновок для підвищення ефективності полягає у необхідності поєднання RE з архітектурним аналізом та тестовою стратегією на ранніх етапах, а також у використанні артефактів трасування.

3.3. Трасування «вимоги > архітектура > тести > розгортання»

Ефективність розробки підсилюється, якщо вимоги застосовуються не лише до архітектурних рішень, а й до тестових цілей і сценаріїв розгортання. У розподілених IoT-системах тестування є одним із найдорожчих етапів, і систематичний огляд тестування IoT-систем демонструє різноманітність цілей, підходів та інструментів, а також ключові виклики, пов'язані з гетерогенністю, складністю інтеграцій і відтворюваністю [7]. Отже, зв'язок RE з тестовою моделлю дозволяє зменшити невизначеність і витрати на перевірку якості, а також формалізувати критерії приймання компонентів.

4. Розробка, керована моделями, та автоматизація артефактів розробки й розгортання

4.1. Розробка, керована моделями, як відповідь на складність і неоднорідність

Розробка, керована моделями (Model-Driven Engineering, MDE), розглядається як підхід, що знижує складність розробки шляхом формалізації системи у вигляді моделей, які можуть перевірятися і перетворюватися у реалізаційні артефакти. Для IoT MDE особливо актуальний через гетерогенність пристроїв, протоколів і середовищ розгортання. Систематичне мапінг-дослідження MDE-технік і інструментів для IoT систематизує різноманітні підходи і показує, що модельно-керовані засоби підтримують генерацію артефактів, аналіз архітектури та валідацію залежностей між компонентами [12]. З точки зору підвищення ефективності, це означає зменшення ручних операцій і помилок конфігурації, а також можливість повторного використання шаблонів.

4.2. Кероване моделями розгортання на гетерогенних граничних пристроях IoT

Одним із практично важливих напрямів є модельно-кероване розгортання застосунків на гетерогенних граничних (edge) пристроях. У таких сценаріях MDE використовується для опису топології, ресурсних обмежень, залежностей компонентів і правил розміщення. Робота Guevara та співавторів позиціонується як модельно-керований підхід до розгортання IoT-застосунків у гетерогенних edge-середовищах, що відповідає загальній тенденції зменшення складності розгортання через формалізацію і автоматизацію [13]. З практичної сторони, це дає можливість стандартизувати процес перенесення застосунку між платформами і швидше відтворювати розгортання в тестових середовищах.

4.3. MDE у контексті континууму і оптимізацій розміщення

У континуум-системах модельно-керовані підходи природно поєднуються із задачами оптимального розміщення і прийняття рішень щодо розгортання. Огляд моделей і методів для оптимального прийняття рішень і розгортання в IoT-Edge-Cloud розподілених системах

може розглядатися як рамка для інтеграції оптимізаційних критеріїв у MDE-артефакти [14]. Навіть якщо конкретні алгоритмічні реалізації різняться, ключовою для ефективності залишається ідея формалізованого опису середовища і цілей розміщення, що дозволяє автоматизувати частину інженерних рішень.

5. Інтелектуалізація граничної області та dataflow-орієнтовані ML-компоненти

5.1. Інтелектуалізація граничних обчислень IoT

Перенесення інтелекту на граничний рівень IoT (Edge-AI) формує специфічні вимоги до компонентів: оновлюваність, стійкість до втрати зв'язку, здатність працювати в реальному часі і враховувати обмеження енергоспоживання. У роботі, присвяченій інтелектуалізації граничних обчислень IoT, розглянуто підходи, які підсилюють роль edge-рівня у загальній архітектурі IoT [2]. Це означає, що ефективність розробки має вимірюватися не лише швидкістю створення сервісів у хмарі, а й здатністю швидко проєктувати та оновлювати граничні компоненти та їх взаємодію з хмарною частиною.

5.2. Edge-AI: архітектури, застосування, виклики

Систематичний огляд Edge-AI узагальнює архітектури, застосування і виклики перенесення штучного інтелекту на периферійний рівень. Підкреслюється, що мотиваціями виступають зменшення затримки, збереження приватності і підвищення надійності, але складність зростає через гетерогенність пристроїв, обмежені ресурси та потребу у керуванні життєвим циклом машинного навчання (ML) [3]. Для ефективної розробки з'являються виклики, як-от необхідність інженерних практик для керування версіями моделей, моніторингу якості і безпечного оновлення компонентів.

5.3. Dataflow-орієнтовані ML-конвеєри як спосіб підвищення модульності

У ML-підсистемах ефективність часто знижується через складність конвеєрів (pipelines) і тісну залежність коду від інфраструктури.

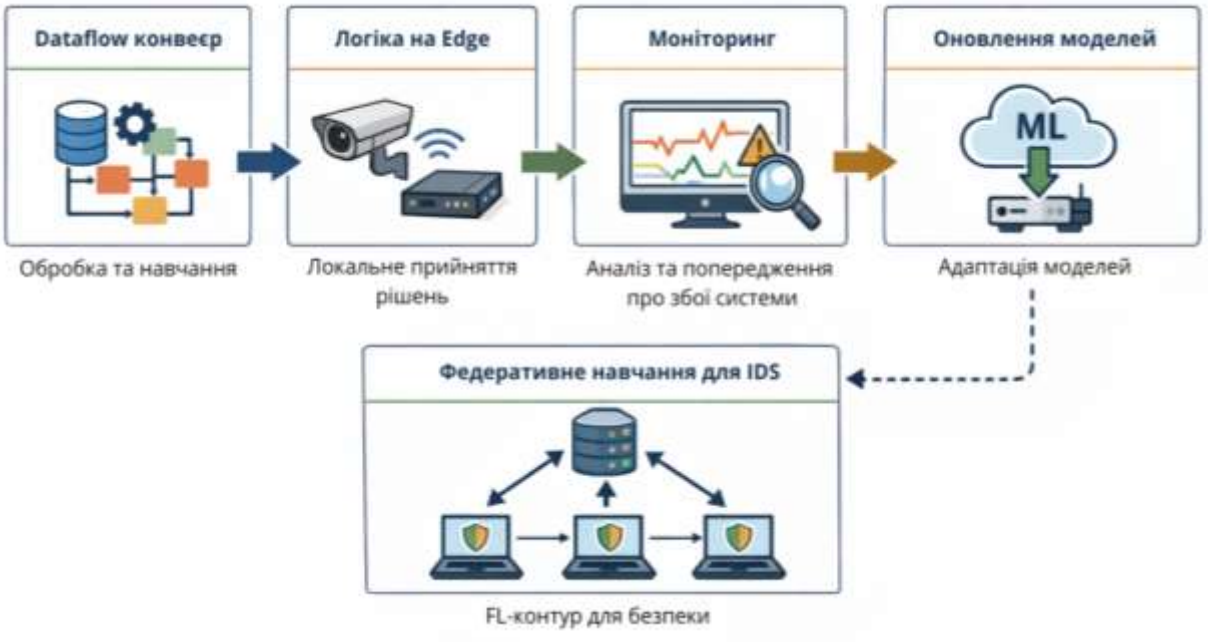

Рисунок 2 — Dataflow-орієнтований підхід

Dataflow-орієнтований підхід дозволяє описувати систему як граф операторів, що полегшує повторне використання, заміну частин конвеєру та розгортання у розподіленому середовищі. У роботі Baldoni та співавторів запропоновано dataflow-орієнтований підхід (рис. 2) для ML-підтримуваних IoT-застосунків, який забезпечує end-to-end конвеєр і спрощує доставку ML-функцій у децентралізованих сценаріях [15]. Практичний ефект полягає у зниженні залежності між бізнес-логікою ML-обробки та конкретними транспортними механізмами, що дозволяє прискорити внесення змін і полегшує експерименти.

6. Тестування IoT-систем як ключовий чинник ефективності

Ефективність розробки в IoT значною мірою визначається здатністю організувати тестування так, щоб воно було відтворюваним, масштабованим і узгодженим з архітектурою. Систематичний огляд тестування IoT-систем узагальнює цілі тестування, підходи, інструменти та виклики й демонструє, що доменна специфіка IoT робить універсальні тестові практики недостатніми [7]. Зокрема, проблеми виникають через наявність фізичних пристроїв, варіативність мережі, неоднорідність протоколів та складність інтеграцій.

У мікросервісних IoT-архітектурах тестування відходить від традиційного системного тесту до поєднання контрактного тестування, інтеграційних перевірок, спостережуваності і контрольованих середовищ виконання. Результати огляду мікросервісних IoT-архітектур підкреслюють чітку взаємозалежність між архітектурним стилем і тестовими практиками [4]. У континуум-системах додаються сценарії перенесення компонентів між edge і хмарою та тестування деградації зв’язку, що потребує спеціальних стендів, симуляторів або цифрових двійників.

Практичний висновок полягає у необхідності поєднання тестування з трасуванням вимог, щоб тестові цілі формувалися на основі нефункціональних вимог і архітектурних рішень, а результати тестів напряму впливали на правила розгортання та політики безпеки.

7. Розгортання, експлуатація, інструментальні засоби та контейнеризація

7.1. Інструменти керування даними в континуумі

Керування даними і потоками телеметрії є однією з головних причин операційної складності IoT-систем. Засоби розробки, націлені на спрощення конфігурації та інтеграції, знижують витрати на підтримку і прискорюють впровадження змін. У роботі [11] описано набір інструментів і набір інструментів керування даними для IoT-Edge-Cloud континууму, що передбачає модульну організацію та можливості візуалізації. Для ефективності розробки це важливо, оскільки зменшує «приховані» витрати на інтеграцію і супровід, а також стандартизує підхід до експлуатації компонентів.

7.2. Контейнеризація в edge intelligence

Контейнеризація широко використовується для забезпечення переносимості й відтворюваності середовищ. Вона полегшує створення однакових середовищ для розробки, тестування і розгортання, що прямо впливає на ефективність. Проте на edge-рівні контейнери мають специфічні обмеження через ресурси і накладні витрати. Огляд контейнеризації в edge intelligence підкреслює контейнери як форму «легкої віртуалізації» для розробки і розгортання, але вказує на вплив затримки та обсягів даних, які стимулюють перенесення обробки ближче до джерел [16]. Це означає, що при проєктуванні компонентів слід закладати профілювання ресурсів, оптимізацію образів і правила оновлення, сумісні з обмеженнями граничних вузлів.

7.3. Практичні архітектури Industrial IoT як шаблон структуризації компонентів

Польові реалізації індустріального інтернету речей (Industrial IoT) корисні як зразки шарування архутектури, інтеграції апаратної та програмної частини і використання інструментів спостережуваності. У роботі [17] продемонстровано впровадження і експериментальне застосування індустріальної IoT-архітектури, де виділено шари sensing, network, middleware та application, а також використано інструменти візуалізації й моніторингу, зокрема Grafana. Для ефективності розробки такі приклади важливі тому, що показують практичний спосіб стандартизації інтеграцій і спостережуваності, а також дозволяють переносити архітектурні шаблони на подібні домени.

8. Безпека як умова ефективності та стійкості життєвого циклу

8.1. Архітектури, загрози та механізми захисту

Безпека в IoT-системах має наскрізний характер і впливає на архітектуру, протоколи, механізми доступу, оновлення та логіку компонентів. Систематичний огляд безпеки IoT узагальнює архітектури, загрози і засоби захисту та демонструє, що рішення мають бути узгоджені з доменом і середовищем застосування [18]. Для ефективності розробки це означає, що «security-by-design» зменшує ризики інцидентів, які здатні нівелювати переваги швидкої розробки, а також зменшує витрати на вимушені зміни, спричинені виявленням критичних вразливостей на пізніх етапах життєвого циклу продукту.

8.2. Вразливості вбудованого програмного забезпечення та аудит як критичний елемент

У IoT важливо враховувати, що програмні компоненти взаємодіють із пристроями, прошивки яких можуть містити типові вразливості. Навіть за якісної хмарної або edge-архітектури, проблеми апаратного забезпечення здатні призвести до збоїв системи та порушення її надійності. Огляд вразливостей прошивок і технік аудиту описує типові підходи до виявлення проблем, а також пов’язує безпеку з процесами доставки й оновлення прошивок [18]. Практичний висновок полягає у необхідності інтегрувати аудит вбудованого програмного забезпечення у життєвий цикл IoT-компонентів включно з політиками оновлень, контролем цілісності та механізмами відкату.

8.3. Федеративне навчання в IoT: можливості і складність

Федеративне навчання (Federative learning, FL) часто подається як спосіб розв’язання проблем приватності й централізації даних. Проте воно створює додатковий рівень операційної складності, зокрема через комунікаційні витрати, гетерогенність клієнтів, а також ризики порушення приватності й атак на моделі. Огляд федеративного навчання для IoT систематизує техніки, виклики і застосування та прямо підкреслює проблеми надлишкової комунікації і гетерогенності як критичні [14]. З точки зору ефективної розробки, FL слід розглядати не лише як алгоритм машинного навчання, а як комплексний інженерний підхід, який змінює вимоги до компонентів моніторингу, збору, безпечного обміну і контролю якості даних.

8.4. FL-підходи до IDS у IoT-мережах

У практичних сценаріях FL активно застосовується для систем виявлення вторгнень (Intrusion Detection Systems, IDS). Це змінює життєвий цикл IDS-компонентів: потрібні модулі локального навчання, збору, перевірки якості, контролю збіжності та захисту обміну. Робота Devine та співавторів демонструє застосування федеративного ML для IDS у IoT-мережах і акцентує важливість підготовки даних і вибору моделей у федеративному сценарії [16–21]. Дослідження Albanbay та співавторів розглядає питання масштабування даних у

FL-IDS і аналізує вплив кількості пристроїв та обсягів даних на точність і збіжність, що має прямі наслідки для інженерії компонентів і ресурсного планування [15].

9. Прикладні класи IoT-систем: моніторинг і керування як контекст для шаблонів ефективності

У системах моніторингу й керування ефективність розробки визначається повторним використанням шаблонів підключення пристроїв, збору телеметрії, нормалізації даних та побудови панелей спостережуваності. Огляд систем моніторингу і керування на базі IoT охоплює домени, зокрема smart agriculture, predictive maintenance, healthcare, довкілля та smart cities, і підкреслює їх вплив на операційну ефективність та підвищення якості сервісів [18].

Практичний сенс включення цього класу джерел в огляд полягає у тому, що він задає типові доменні сценарії, для яких можна оцінювати ефективність компонентної інженерії: швидкість підключення нового сенсора, час інтеграції протоколу, відтворюваність розгортання, керованість оновлень та безпекові політики. Це забезпечує зв'язок між абстрактними моделями і прикладними вимогами.

10. Інтегрована модель підвищення ефективності розробки IoT-компонентів

На основі узагальнення джерел доцільно сформулювати інтегровану модель, у якій ефективність розглядається як результат узгодженої роботи моделей і засобів на всіх етапах життєвого циклу. Логічною основою є зв'язок між інженерією вимог, архітектурою, автоматизацією артефактів, тестуванням, розгортанням і безпекою. Вимоги задають цілі й критерії якості включно з нефункціональними властивостями, критичними для IoT [8, 9]. Архітектура визначає декомпозицію системи на компоненти і їх взаємодію, зокрема через мікросервісні підходи та можливе поєднання з безсерверними практиками [4–6]. З перспективи континууму випливають правила розміщення і переміщення компонентів між рівнями, що впливає на політики розгортання і спостережуваності [1]. MDE описує формалізацію й автоматизацію артефактів, що зменшує ручну працю і ризики конфігураційних помилок [12, 13]. У ML-підсистемах dataflow-орієнтовані підходи та Edge-AI утворюють принципи модульності і оновлюваності інтелектуальних компонентів [2, 3, 15]. Тестування забезпечує контроль якості і повинно бути узгоджене з архітектурою та вимогами, щоб зменшити витрати на перевірку і переробки [5]. Інструменти керування даними і контейнеризація відповідають за відтворюваність, переносимість і зниження операційного навантаження [11, 16]. Безпека інтегрується як наскрізна властивість, включно з аудитом вбудованого програмного забезпечення і спеціальними сценаріями FL-IDS, які змінюють життєвий цикл компонентів [18–22].

У межах цієї моделі можна виділити три практичні принципи підвищення ефективності. Перший принцип полягає в ранній формалізації вимог та їх трасуванні до архітектури й тестування, що дає змогу запобігти вимушеним змінам на пізніших етапах життєвого циклу. Другий принцип — у використанні моделей і автоматизації (MDE, конфігураційні шаблони, dataflow-опис) для зменшення ручних операцій і підвищення відтворюваності. Третій принцип — у включенні безпеки й оновлюваності (включно з вбудованим програмним забезпеченням та ML/FL) як невід'ємних властивостей компонентів, щоб запобігати втратам ефективності через інциденти та складність супроводу.

11. Прогалини, відкриті проблеми та перспективні напрями досліджень

Незважаючи на активний розвиток архітектурних і інструментальних підходів, існують прогалини, що обмежують ефективність розробки інтелектуальних IoT-систем. По-перше, залишається відкритою проблема узгодження динамічного розміщення компонентів у континуумі з тестовими стратегіями і гарантіями якості. Континуум-огляд підкреслює невирішені

питання керування і стандартизації, а тестовий огляд демонструє складність відтворюваного тестування у гетерогенних умовах [1, 7].

По-друге, Edge-AI потребує більш зрілих інженерних практик для керування життєвим циклом моделей включно з моніторингом дрейфу, політиками оновлення і механізмами безпечної доставки на edge-вузли. Систематичні огляди вказують на виклики гетерогенності й ресурсних обмежень, що ускладнюють уніфікацію процесів [2, 3].

По-третє, безпека прошивок і ланцюжок оновлень залишаються слабкою ланкою, здатною нівелювати ефект від будь-якої архітектури. Огляд вразливостей вбудованого програмного забезпечення демонструє широкий спектр проблем і потребу у системному аудиті [19].

По-четверте, федеративні підходи, хоча й підтримують приватність, створюють складність у експлуатації, зокрема через комунікаційні витрати та гетерогенність клієнтів. Огляди FL для IoT і емпіричні дослідження FL-IDS показують, що ефективність таких систем залежить від масштабування даних, якості збору даних та механізмів захисту федеративного процесу навчання [20–22].

Перспективними напрямами досліджень є поєднання мікросервісних та безсерверних підходів для розробки програмних компонентів інтелектуальних систем інтернету речей. Сюди входять питання правильного розміщення компонентів в IoT-Edge-Cloud континуумі, керування даними в таких розподілених системах, оптимізація тестування та розгортання, конвергенція з системами ситуаційного управління, а також використання методів штучного інтелекту для розробки таких систем.

12. Висновки

У статті систематизовано моделі та засоби підвищення ефективності розробки програмних компонентів інтелектуальних IoT-систем у контексті IoT-Edge-Cloud континууму. Показано, що ключовими факторами ефективності є узгодженість інженерії вимог, архітектурної декомпозиції, автоматизації артефактів, тестування і практик розгортання, а також інтеграція безпеки як наскрізної властивості компонентів. Мікросервісні та мікросервісно-безсерверні архітектури забезпечують модульність і незалежність еволюції, але підсилюють потребу в якісних практиках тестування й спостережуваності. Континуумна перспектива змінює вимоги до переносимості, керованості та розміщення компонентів і підкреслює важливість інструментів керування даними та операційної стандартизації. Model-driven підходи та dataflow-орієнтовані ML-конвеєри зменшують ручні операції і підвищують відтворюваність, що особливо важливо для Edge-AI сценаріїв. Безпека, включно з аудитом прошивок і федеративними IDS-підходами, визначає стійкість життєвого циклу та запобігає втратам ефективності через інциденти.

Сформульована інтегрована модель підвищення ефективності дозволяє розглядати розробку IoT-компонентів як єдиний процес від вимог до експлуатації та еволюції. Подальші дослідження доцільно зосередити на відтворюваному тестуванні у континуумі, стандартизованих життєвих циклах Edge-AI компонентів, інтеграції аудиту вбудованого програмного забезпечення у CI/CD та інженерних патернах федеративних компонентів безпеки.

СПИСОК ДЖЕРЕЛ

1. Gkonis P. et al. A Survey on IoT-Edge-Cloud Continuum Systems: Status, Challenges, Use Cases and Open Issues. *Future Internet*. 2023. Vol. 15, N 12. Art. 383. DOI: https://doi.org/10.3390/fi15120383.

2. Коваленко О. Є. Інтелектуалізація граничних обчислень інтернету речей. *Математичні машини і системи*. 2024. № 3–4. С. 50–68. URL: https://www.immsp.kiev.ua/publications/articles/2024/2024_3_4/03_04_24_Kovalenko.pdf.

3. Gauttam H., Nain G., Pattanaik K.K. Edge-AI: A systematic review on architectures, applications, and challenges. *Journal of Network and Computer Applications*. 2025. Vol. 245. Art. 104375. DOI: https://doi.org/10.1016/j.jnca.2025.104375.
4. Siddiqui H. et al. Microservices based architectures for IoT systems — State-of-the-art review. *Internet of Things*. 2023. Art. 100854. DOI: https://doi.org/10.1016/j.iot.2023.100854.
5. Razzaq A. A Systematic Review on Software Architectures for IoT Systems and Future Direction to the Adoption of Microservices Architecture. *SN Computer Science*. 2020. Vol. 1, N 6. Art. 350. DOI: https://doi.org/10.1007/s42979-020-00359-w.
6. Ouyang R. et al. A Microservice and Serverless Architecture for Secure IoT System. *Sensors*. 2023. Vol. 23, N 10. Art. 4868. DOI: https://doi.org/10.3390/s23104868.
7. Minani J.B., Sabir A., Moha N., Guéhéneuc Y.-G. A Systematic Review of IoT Systems Testing: Objectives, Approaches, Tools, and Challenges. *IEEE Transactions on Software Engineering*. 2024. DOI: https://doi.org/10.1109/TSE.2024.3363611.
8. Коваленко О.Є., Смолій В.М., Лі Л. Інженерія вимог систем інтернету речей. *Математичні машини і системи*. 2025. № 1. С. 64–75. DOI: https://doi.org/10.34121/1028-9763-2025-1-64-75.
9. Aguilar-Calderón J.-A., Tripp-Barba C., Zaldívar-Colado A., Aguilar-Calderón P.-A. Requirements Engineering for Internet of Things (IoT) Software Systems Development: A Systematic Mapping Study. *Applied Sciences*. 2022. Vol. 12, N 15. Art. 7582. DOI: https://doi.org/10.3390/app12157582.
10. Коваленко О.Є. Конвергенція інтернету речей та систем ситуаційного управління. *Математичні машини і системи*. 2023. № 3. С. 89–103. DOI: https://doi.org/10.34121/1028-9763-2023-3-89-103.
11. Judvaitis J. et al. A Set of Tools and Data Management Framework for the IoT-Edge-Cloud Continuum. *Applied System Innovation*. 2024. Vol. 7. Art. 130. DOI: https://doi.org/10.3390/asi7060130.
12. Mardani Korani Z., Moin A., Rodrigues da Silva A., Ferreira J.C. Model-Driven Engineering Techniques and Tools for Machine Learning-Enabled IoT Applications: A Scoping Review. *Sensors*. 2023. Vol. 23, N 3. Art. 1458. DOI: https://doi.org/10.3390/s23031458.
13. Lim K S., Ooi S.Y., Sayeed M.S., Chew Y.J., Ahmad N.M. Securing the Internet of Things: Systematic Insights into Architectures, Threats, and Defenses. *Electronics*. 2025. Vol. 14, N 20. Art. 3972. DOI: https://doi.org/10.3390/electronics14203972.
14. Guevara I., Ryan S., Singh A., Brandon C., Margaria T. Edge IoT Prototyping Using Model-Driven Representations: A Use Case for Smart Agriculture. *Sensors*. 2024. Vol. 24, No. 2. Art. 495. DOI: https://doi.org/10.3390/s24020495.
15. Baldoni G., Teixeira R., Guimarães C., Antunes N. A Dataflow-Oriented Approach for Machine-Learning-Powered Internet of Things Applications. *Electronics*. 2023. Vol. 12. Art. 3940. DOI: https://doi.org/10.3390/electronics12183940.
16. Urblik L., Kajati E., Papcun P., Zolotová I. Containerization in Edge Intelligence: A Review. *Electronics*. 2024. Vol. 13, N 7. Art. 1335. DOI: https://doi.org/10.3390/electronics13071335.
17. Calderón D., Folgado F.J., González I., Calderón A.J. Implementation and Experimental Application of Industrial IoT Architecture Using Automation and IoT Hardware/Software. *Sensors*. 2024. Vol. 24. Art. 8074. DOI: https://doi.org/10.3390/s24248074.
18. Bakhshi T., Ghita B., Kuzminykh I. A Review of IoT Firmware Vulnerabilities and Auditing Techniques. *Sensors*. 2024. Vol. 24, N 2. Art. 708. DOI: https://doi.org/10.3390/s24020708.
19. Witczak D., Szymoniak S. Review of Monitoring and Control Systems Based on Internet of Things. *Applied Sciences*. 2024. Vol. 14, N 19. Art. 8943. DOI: https://doi.org/10.3390/app14198943.
20. Dritsas E., Trigka M. Federated Learning for IoT: A Survey of Techniques, Challenges, and Applications. *Journal of Sensor and Actuator Networks*. 2025. Vol. 14, N 1. Art. 9. DOI: https://doi.org/10.3390/jsan14010009.
21. Devine M., Ardakani S.P., Al-Khafajiy M., James Y. Federated Machine Learning to Enable Intrusion Detection Systems in IoT Networks. *Electronics*. 2025. Vol. 14, N 6. Art. 1176. DOI: https://doi.org/10.3390/electronics14061176.
22. Albanbay N. et al. Federated Learning-Based Intrusion Detection in IoT Networks: Performance Evaluation and Data Scaling Study. *Journal of Sensor and Actuator Networks*. 2025. Vol. 14. Art. 78. DOI: https://doi.org/10.3390/jsan14040078.

Стаття надійшла до редакції 12.06.2026 / прийнята до друку 13.08.2026