УДК 004.413/.415:004.05

С. О. Цимбал
О. О. Коваленко

ВПРОВАДЖЕННЯ ПРАКТИК DEVOPS ДЛЯ АВТОМАТИЗАЦІЇ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вінницький національний технічний університет

Анотація

Стаття присвячена комплексному дослідженню сучасних підходів до автоматизації життєвого циклу розробки програмного забезпечення шляхом глибокої інтеграції методології DevOps. Розглядаються ключові технічні компоненти практик безперервної інтеграції та безперервної доставки, концепція інфраструктури як коду, а також впровадження інструментів автоматизованого тестування, забезпечення безпеки та проактивного моніторингу складних розподілених систем. Особлива увага приділяється трансформації традиційних процесів розробки, подоланню технологічного розриву між командами та забезпеченню надійності й масштабованості розгортання сучасних програмних продуктів в умовах динамічних ринкових вимог.

Ключові слова: DevOps, автоматизація, безперервна інтеграція, безперервна доставка, інфраструктура як код, життєвий цикл ПЗ, контейнеризація, оркестрація, DevSecOps.

Abstract

The article investigates comprehensive modern approaches to automating the software development lifecycle through the deep integration of DevOps methodology. It examines the key technical components of continuous integration and continuous delivery practices, the concept of infrastructure as code, as well as the implementation of automated testing, security assurance, and proactive monitoring of complex distributed systems. Special attention is paid to the transformation of traditional development processes, overcoming the technological gap between teams, and ensuring the reliability and scalability of deploying modern software products under dynamic market requirements.

Keywords: DevOps, automation, continuous integration, continuous delivery, infrastructure as code, software development lifecycle, containerization, orchestration, DevSecOps.

Вступ

Стрімкий розвиток індустрії розробки програмного забезпечення та цифрова трансформація бізнесу вимагають від сучасних інженерних команд максимальної швидкості випуску нових функціональних можливостей при збереженні бездоганної якості, безпеки та стабільності систем. Традиційні каскадні моделі розробки та ізольоване функціонування підрозділів створення коду й системного адміністрування вже не здатні забезпечити необхідну гнучкість, оскільки вони часто стають першопричиною тривалих затримок, довгих циклів зворотного зв'язку, організаційних конфліктів та виникнення критичних помилок під час ручного розгортання продуктів на виробничих серверах. Методологія DevOps виступає фундаментальною відповіддю на ці виклики, пропонуючи філософію глибокого технологічного та культурного об'єднання процесів проєктування, написання коду, тестування, розгортання та безпосередньої експлуатації програмного забезпечення. Автоматизація кожного етапу життєвого циклу в межах цього інтегрованого підходу дозволяє повністю мінімізувати деструктивний вплив людського фактора, суттєво прискорити показник виходу продукту на ринок, оптимізувати використання хмарних ресурсів та забезпечити безперервне надання цінності для кінцевих користувачів.

Результати дослідження

Сучасна парадигма інженерії програмного забезпечення базується на остаточному усуненні технологічних та організаційних бар'єрів між крос-функціональними командами задля створення максимально гнучкого, прозорого та стійкого виробничого середовища [1]. Основою автоматизації життєвого циклу програмного забезпечення в концепції DevOps є проєктування та впровадження наскрізного, повністю автоматизованого конвеєра безперервної інтеграції та безперервної доставки (CI/CD). Практика безперервної інтеграції передбачає регулярне, багаторазове протягом дня злиття вихідного коду всіх залучених розробників у спільну центральну гілку репозиторію за допомогою передових систем контролю версій та спеціалізованих інструментів автоматизації, таких як Jenkins, GitHub Actions або GitLab CI [2]. Кожна така публікація коду автоматично запускає ізольовані процеси збирання проєкту в контейнеризованих середовищах та негайне виконання комплексу базових тестів. Це дозволяє інженерним командам оперативно виявляти конфлікти інтеграції, синтаксичні помилки та логічні дефекти на початкових етапах, що суттєво знижує загальну вартість їх виправлення та запобігає накопиченню технічного боргу.

Практика безперервної доставки розширює можливості інтеграційного конвеєра, повністю автоматизуючи складні процеси перенесення перевіреного та схваленого коду в різноманітні тестові та продуктивні середовища. Для цього активно застосовується декларативний синтаксис конфігураційних файлів у форматі YAML, який дозволяє розробникам чітко й однозначно визначати всі етапи збирання, тестування, пакування та розгортання додатків безпосередньо в кодовій базі проєкту [3]. Завдяки використанню інструментів автоматизації розгортання, таких як ArgoCD або Flux, що працюють за принципами GitOps, розробники отримують можливість підтримувати актуальний стан системи у повній відповідності до конфігурацій, описаних у Git-репозиторії. Це нівелює ризики несанкціонованих змін у налаштуваннях серверів та забезпечує високу повторюваність процесів деплою, незалежно від масштабу цільової інфраструктури.

Наступним критично важливим етапом у забезпеченні стабільності та передбачуваності життєвого циклу програмного забезпечення є повний перехід від застарілого ручного або напівавтоматичного налаштування серверних потужностей до передової концепції інфраструктури як коду (IaC), що дає змогу ефективно керувати віртуальними машинами, мережами та сховищами даних за допомогою структурованих конфігураційних файлів [4]. Детальний опис параметрів мережевої топології, конфігурацій серверів, систем управління базами даних та супутніх конфігураційних файлів за допомогою декларативних мов програмування, яскравим прикладом яких є інструмент HashiCorp Terraform або OpenTofu, дозволяє версіонувати, тестувати та рецензувати інфраструктуру аналогічно до звичайного вихідного коду прикладних програм [5]. Такий підхід гарантує абсолютну ідентичність оточення розробки, тестування та промислової експлуатації, повністю усуваючи поширену проблему невідповідності конфігурацій середовищ, яка раніше призводила до раптових збоїв додатків після релізу.

У поєднанні з методологією IaC стандартом для сучасної індустрії розробки стало обов'язкове використання технологій контейнеризації на базі платформи Docker спільно з потужними системами оркестрації, такими як Kubernetes. Контейнеризація дозволяє ізолювати додаток разом із усіма його залежностями, системними бібліотеками та налаштуваннями в єдиний легковагий пакунок, що забезпечує стабільну роботу програми на будь-якому фізичному чи хмарному сервері. Оркестрація за допомогою Kubernetes, у свою чергу, бере на себе завдання автоматичного управління життєвим циклом цих контейнерів: вона забезпечує балансування мережевого навантаження, автоматичне масштабування обчислювальних ресурсів залежно від поточного трафіку, самовідновлення систем шляхом перезапуску пошкоджених контейнерів та безперервне оновлення додатків без зупинки сервісу для користувачів [6].

Невід'ємною та фундаментальною частиною автоматизованого життєвого циклу є проєктування та розгортання комплексних систем безперервного моніторингу, централізованого логування та проактивного аналізу стану інфраструктури й прикладного рівня в реальному часі.

Автоматизація процесів збору, агрегації та візуалізації метрик продуктивності високо інтенсивних додатків за допомогою зв'язки систем Prometheus та Grafana дає змогу експлуатаційним командам оперативно помічати найменші деградації в роботі сервісів та реагувати на потенційні збої ще до того, як вони почнуть негативно впливати на кінцевих користувачів [7]. Сучасні платформи моніторингу та управління логами, такі як ELK Stack (Elasticsearch, Logstash, Kibana) або Grafana Loki, здатні обробляти гігабайти текстових логів щомиті, надаючи інженерам вичерпну інформацію для швидкого пошуку першопричин виникнення помилок (Root Cause Analysis). При цьому сучасні системи моніторингу дедалі частіше інтегрують алгоритми штучного інтелекту та машинного навчання (AIOps) для проведення повністю автоматичного аналізу аномалій у поведінці інфраструктури, що дозволяє прогнозувати вичерпання дискового простору, перевантаження процесорів або витоки оперативної пам'яті [8].

Паралельно з розвитком класичного моніторингу в сучасні процеси автоматизації життєвого циклу програмного забезпечення активно впроваджуються прогресивні концепції DevSecOps, що передбачають безперервну інтеграцію жорстких засобів та стандартів комп'ютерної безпеки на кожному етапі створення продукту, а не лише перед безпосереднім релізом. Це досягається завдяки інтеграції інструментів автоматизованого статичного (SAST) та динамічного (DAST) сканування коду, а також систем аналізу сторонніх залежностей (SCA), таких як Snyk, SonarQube чи Trivy, безпосередньо в конвеєри розгортання [9]. Автоматична перевірка вихідного коду на наявність вразливостей, залишених паролів чи застарілих бібліотек із відомими вразливостями дозволяє виявляти архітектурні та програмні помилки безпеки на етапі написання коду. Це суттєво зменшує загальний відсоток відмов при фінальних релізах, мінімізує потенційні вектори кібератак на цифрову інфраструктуру організації та гарантує відповідність розроблюваного програмного продукту міжнародним стандартам захисту інформації [10]. Завдяки наявності таких глибоко автоматизованих систем сповіщення та інтегрованих конвеєрів, інженери можуть миттєво реалізувати стратегії автоматичного захисту, ізоляції вразливих вузлів, швидкого горизонтального масштабування або оперативного відкоту змін до попередньої завідомо стабільної версії системи у разі фіксації критичних аномалій чи ознак цілеспрямованої хакерської атаки.

Висновки

Впровадження практик DevOps є безальтернативним стратегічним кроком для глибокої технологічної модернізації та підвищення конкурентоспроможності процесів розробки сучасного програмного забезпечення. Автоматизація життєвого циклу через створення надійних конвеєрів безперервної інтеграції та доставки, впровадження концепції управління інфраструктурою як кодом, контейнеризації та інтеграції систем наскрізного інтелектуального моніторингу й безпеки дозволяє кардинально трансформувати інженерну культуру всередині IT-компаній. Практичні результати реалізації такого комплексного підходу полягають у кратному скороченні часу випуску нових релізів, мінімізації часу простою систем, значному підвищенні загальної відмовостійкості програмних комплексів та нівелюванні операційних і безпекових ризиків. Подальший розвиток цього науково-практичного напряму тісно пов'язаний із розширенням використання інтелектуальних агентів для автоматичного написання та виконання тестів, самостійного усунення інфраструктурних інцидентів та оптимізації архітектури хмарних обчислень.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. IEEE Xplore. Enhancing Operational Efficiency through CI/CD and DevOps. URL: https://ieeexplore.ieee.org/document/10596596/ (дата звернення: 12.04.2026).
2. ScienceDirect. Mapping DevOps capabilities to the software life cycle. URL: https://www.sciencedirect.com/science/article/pii/S0950584924001885 (дата звернення: 15.04.2026).
3. arXiv. DevOps Automation Pipeline Deployment with IaC. URL: https://arxiv.org/abs/2503.16038 (дата звернення: 12.04.2026).
4. SSRN. Exploring IaC Using Terraform in Multi-Cloud. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5075647 (дата звернення: 15.04.2026).
5. American Journal of Technology. IaC Terraform vs CloudFormation. URL: https://gprjournals.org/journals/index.php/ajt/article/view/351 (дата звернення: 18.04.2026).
6. Wiley. Kubernetes in Action. URL: https://onlinelibrary.wiley.com/doi/10.1002/spe.70000 (дата звернення: 18.04.2026).

7. WJAETS. Proactive monitoring with Prometheus, Grafana. URL: https://doi.org/10.30574/wjaets.2024.13.2.0543 (дата звернення: 25.04.2026).
8. ScienceDirect. AIOps for log anomaly detection in era of LLMs. URL: https://www.sciencedirect.com/science/article/pii/S2667305325001346 (дата звернення: 20.04.2026).
9. ResearchGate. Integrating Security into CI/CD Pipelines: A DevSecOps Approach with SAST, DAST, and SCA Tools. URL:https://www.researchgate.net/publication/390459514_Integrating_Security_into_CICD_Pipelines_A_DevSecOps_Approach_with_SAST_DAST_and_SCA_Tools (дата звернення: 20.04.2026).
10. arXiv. Collaborative Application Security Testing for DevSecOps. URL: https://arxiv.org/pdf/2211.06953 (дата звернення: 26.04.2026).

Цимбал Сергій Олегович – студент групи 4ПІ-24б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: cimbal859@gmail.com
Коваленко Олена Олексіївна – доцент кафедри програмного забезпечення Вінницького національного технічного університету, м. Вінниця, e-mail: kovalenka88@gmail.com

Tsymbal Serhii O. – Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: cimbal859@gmail.com
Kovalenko Olena O. – Associate Professor of the Department of Software, Vinnytsia National Technical University, Vinnytsia, e-mail: kovalenka88@gmail.com