Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)
https://doi.org/10.36930/40340510
Article received 19.05.2024
Article accepted 23.05.2024
UDC 004.[415.2+052+056]
Correspondence author: M. O. Shestakovych shestakovych.m@nltu.lviv.ua

М. О. Шестакович[1], Ю. В. Шабатура[2]

[1] Національний лісотехнічний університет України, м. Львів, Україна
[2] Національна академія сухопутних військ ім. гетьмана Петра Сагайдачного, м. Львів, Україна

АНАЛІЗ І ОЦІНЮВАННЯ РИЗИКІВ ПРОЦЕСУ ПЕРЕТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ З МОНОЛІТНОЮ АРХІТЕКТУРОЮ В МІКРОСЕРВІСНУ

Проведено аналіз і наведено результати оцінювання ризиків процесу перетворення інформаційних систем з монолітною архітектурою в мікросервісну, що дало змогу розробити рекомендації для їх мінімізації та ефективного управління. З'ясовано, що перехід від монолітної архітектури до мікросервісної є важливим етапом для сучасних IT-систем з метою підвищення ефективності, гнучкості та масштабованості. Проведено детальний аналіз основних ризиків та викликів, пов'язаних із переходом до мікросервісів, з використанням методів системного аналізу та моделювання. Зокрема, досліджено питання управління складністю міграції, управління новою інфраструктурою, забезпечення безперервності функціонування системи та підтримки нових процесів розробки та експлуатації. Встановлено, що успішна міграція від моноліту до мікросервісів потребує ретельного аналізу існуючої системи та ефективного поділу її на окремі сервіси з урахуванням залежностей між компонентами. Виявлено необхідність впровадження нових інструментів для автоматизації та оркестрації, таких як Kubernetes, а також налаштування систем моніторингу та логування для відстеження великої кількості сервісів. Оцінено вплив налаштування нових процесів безперервної інтеграції та доставки (CI/CD) на стабільність системи. З'ясовано, що високий рівень автоматизації тестування мінімізує ризики введення помилок, а також необхідне впровадження надійних стратегій резервного копіювання та відновлення даних. Управління ризиками передбачає ретельне планування та поетапне впровадження мікросервісів, що дає змогу зменшити вплив на основну систему. Дослідження також показало, що адаптація процесів розроблення, тестування та розгортання до нової архітектури потребує змін у структурі та культурі процесів, що сприяє відкритості та співпраці. Виявлено, що запропоновані методи планування, технічні заходи та структурні зміни спрямовані на мінімізацію ризиків та забезпечення успішного переходу до мікросервісної архітектури. Результати цього дослідження є основою для безпечної та ефективної перетворення IT-інфраструктури, підвищення її стійкості та продуктивності. Зокрема, запропоновані підходи можуть бути корисними для майбутніх досліджень у галузі управління складністю систем та оптимізації процесів міграції. Виявлено важливість врахування людського чинника та змін в організаційній культурі під час переходу до мікросервісної архітектури. Отже, комплексний підхід до перетворення IT-систем, який містить технічні, організаційні та людські особливості, є ключем до успішного переходу від монолітної архітектури до мікросервісної. Використання наведених у дослідженні методів та рекомендацій дасть змогу зменшити ризики та забезпечити стійке функціонування нової архітектури.

Ключові слова: складність міграції; управління інфраструктурою; безперервність системи; оркестрація мікросервісів; управління ризиками; процеси CI/CD.

Вступ / Introduction

Ключовою ознакою технологічної інноваційності сучасної епохи є бурхливий розвиток інформаційно-комунікаційних систем (ІКС). У цьому процесі взаємопов'язаними, із складним характером стимулювальних взаємних впливів, задіяні дві сторони, які узагальнено представляють технічне і програмне забезпечення ІКС. Прикметно, що можливості і вимоги однієї сторони зазвичай стимулюють розвиток, і нерідко стають причиною виникнення оригінальних ідей і рішень на боці іншої сторони. Характерним прикладом такого сценарію розвитку ІКС став процес переходу програмного забезпечення ІКС від монолітної архітектури, яка представлена програмним кодом виконаним як єдиний програмний продукт, який введений в експлуатацію одномоментно і забезпечує виконання усіх передбачених замовником системних функцій, до взаємоузгодженої множини відносно автономних програмних модулів, які реалізують окремі системні функції і в такий спосіб утворюють мікросервісну архітектуру програмного забезпечення системи. Перехід від монолітної архітектури до мікросервісної став важливою тенденцією в сучасному розробленні програмного забезпечення, оскільки він забезпечує більшу гнучкість, масштабованість і покращену стійкість систем [5, 6]. Монолітні системи, що

Інформація про авторів:
Шестакович Маркіян Олегович, магістр, аспірант, кафедра інженерії програмного забезпечення.
Email: shestakovych.m@nltu.lviv.ua
Шабатура Юрій Васильович, д-р техн. наук, професор, завідувач кафедри електроніки і електромеханіки.
Email: shabaturayuriy@gmail.com; https://orcid.org/0000-0002-9961-1244

Цитування за ДСТУ: Шестакович М. О., Шабатура Ю. В. Аналіз і оцінювання ризиків процесу перетворення інформаційних систем з монолітною архітектурою в мікросервісну. Науковий вісник НЛТУ України. 2024, т. 34, № 5. С. 78–83.
Citation APA: Shestakovych, M. O., & Shabatura, Yu. V. (2024). Analysis and risk assessment of the transformation process of information systems from monolithic to microservices architecture. *Scientific Bulletin of UNFU*, 34(5), 78–83. https://doi.org/10.36930/40340510

складаються з одного великого блоку коду, мають обмежену здатність до масштабування та часто ускладнюють процес оновлення і впровадження нових функцій. Мікросервісна архітектура, водночас, дає змогу розробникам створювати окремі, незалежні служби, які можна розгорнути та масштабувати автономно, що значно підвищує ефективність розроблення і експлуатації системи.

Відомо, що мікросервісну архітектуру активно досліджують у сучасній науковій літературі. Такі дослідники, як Newman (2015) та Lewis і Fowler (2014), зазначають, що перехід до мікросервісної архітектури дає змогу значно покращити продуктивність та гнучкість системи. Наприклад, Newman (2015) виявляє, що мікросервіси можуть знижувати час відгуку системи, а Lewis та Fowler (2014) вказують на їхню здатність до незалежного масштабування компонентів [5, 6]. Проте, як зазначає Chen (2017), ще багато аспектів потребують додаткового дослідження, зокрема оцінка ризиків та ефективного управління процесом переходу від монолітної до мікросервісної архітектури [3].

Незважаючи на очевидний позитивний ефект від зазначеного переходу і значну кількість наукових публікацій, наразі ще не існує єдиного науково обґрунтованого підходу для оптимального здійснення процесу переходу від монолітної до мікросервісної архітектури в програмному забезпеченні, та оцінення усіх ризиків, які його супроводжують. Отже, завдання аналізу та оцінювання ризиків і методів управління ними під час переходу від монолітної архітектури до мікросервісів програмного забезпечення ІКС, про що йдеться у цьому дослідженні, є актуальним для сьогодення, важливим в науковому аспекті і цінним для практики.

Об'єкт дослідження – перехід з монолітної до мікросервісної архітектури у корпоративних ІТ-системах.

Предмет дослідження – методи і засоби встановлення ризиків, які виникають під час переходу з монолітної архітектури до мікросервісної, що дасть змогу якісно їх оцінити та ефективно ними управляти.

Мета роботи – здійснити системне оцінювання ризиків, пов'язаних з переходом від монолітної архітектури до мікросервісної, а також розробити рекомендації для їх мінімізації та ефективного управління.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

1. Розглянути базові принципи мікросервісної архітектури та оцінити їх значущість, порівняно з монолітною архітектурою в умовах функціонування сучасних корпоративних ІТ-систем, що дасть змогу краще зрозуміти вплив архітектурних змін на продуктивність, масштабованість, надійність і гнучкість систем.
2. Виявити основні ризики, що виникають під час переходу від монолітної архітектури до мікросервісної, в т.ч. й технічні, організаційні та безпекові особливості, що дасть змогу розробити науково обґрунтовані підходи для їх мінімізації та забезпечити стабільність і безперервність функціонування систем.
3. Визначити виклики, які постають під час впровадження мікросервісної архітектури, зокрема в культурних, організаційних і технічних особливостях, що забезпечить глибше розуміння процесу перетворення систем, допоможе виявити потенційні перешкоди та розробити ефективні методи для їх подолання, сприяючи успішному переходу до мікросервісної архітектури.
4. Подати стратегії мінімізації ризиків і подолання викликів під час переходу до мікросервісної архітектури, в т.ч. й оцінювання ризиків, розроблення плану впровадження та навчання персоналу, які допоможуть забезпечити безперебійне функціонування систем, підвищити надійність і безпеку, а також оптимізувати процеси розроблення та експлуатації.

Аналіз останніх досліджень та публікацій. Проблематика переходу від монолітної архітектури до мікросервісної спричинила значний інтерес серед науковців та практиків. У чисельних дослідженнях висвітлено як переваги, так і виклики цього процесу. Зокрема, автор роботи [6] зазначає, що "мікросервіси сприяють підвищенню гнучкості та масштабованості систем, але вимагають ретельного управління сервісами та залежностями". У дослідженні [5] автори вказують на важливість розуміння контексту організації під час вибору архітектурного підходу, зазначаючи, що "мікросервіси можуть значно ускладнити процеси розгортання та моніторингу систем". Вони також наголошують на потребі використання автоматизованих інструментів для управління мікросервісами та їхнього убезпечення.

Важливою особливістю є дослідження ризиків, пов'язаних з мікросервісами, таких як залежності між сервісами, які можуть призвести до складнощів у тестуванні та розгортанні. Наприклад, у роботі [3] автор акцентує увагу на проблемах, пов'язаних з оркестрацією мікросервісів та управлінням їхньою взаємодією, і пропонує використовувати спеціалізовані інструменти для моніторингу та аналізу продуктивності мікросервісів. Також у дослідженні [8] висвітлено організаційні виклики переходу до мікросервісної архітектури, наголошено на важливості належного планування та координації між командами розробки. Автор зазначає, що "успішний перехід до мікросервісів вимагає зміни культурних аспектів організації, в т.ч. й покращення комунікації та співпраці між командами".

Отже, аналізуючи останні дослідження, можна дійти висновку, що перехід до мікросервісної архітектури має як значні переваги, так і виклики. Важливим складником успішного переходу є ретельне планування, управління ризиками та використання сучасних інструментів для моніторингу та управління сервісами. Незважаючи на численні дослідження, залишаються невирішені питання, такі як ефективне управління залежностями між сервісами та забезпечення їхньої безпеки, що потребує подальшого вивчення та розроблення нових підходів.

Матеріали та методи дослідження. Для дослідження ризиків під час переходу з монолітної до мікросервісної архітектури було використано комплексний підхід, в т.ч. й аналіз літератури, моделювання ризиків та емпіричне дослідження реальних проектів. Інструменти для аналізу продуктивності, такі як Grafana, ELK Stack, а також Jenkins, та Docker.

Результати дослідження та їх обговорення / Research results and their discussion

Аналіз базових принципів мікросервісної архітектури та їх значущість. Мікросервісна архітектура базується на розподілі монолітної системи на невеликі, незалежні сервіси, кожен з яких виконує окрему бізнес-функцію. Основний принцип цієї архітектури полягає у тому, що кожен мікросервіс функціонує автономно, що дає змогу розробникам працювати над різними частинами системи незалежно один від одного. Це сприяє швидкому впровадженню змін і покращує масштабованість. Мікросервіси мають власні бази даних, що дає змогу уникнути труднощів під час звернення до

спільної бази даних, підвищуючи надійність і продуктивність системи [6]. Використання контейнеризації (наприклад, Docker) та оркестраційних інструментів (наприклад, Kubernetes) дає змогу автоматизувати процеси розгортання, тестування та масштабування мікросервісів [5]. Завдяки незалежності сервісів, збої в одному мікросервісі не впливають на роботу інших частин системи, що підвищує загальну стійкість і доступність додатку [3]. Кожен мікросервіс може бути розроблений за різними технологіями та мовами програмування, що дає змогу використовувати найбільш придатні інструменти для конкретних завдань. Перехід від монолітної до мікросервісної архітектури має декілька важливих переваг. Незалежні сервіси дають змогу командам працювати паралельно, що пришвидшує розроблення і впровадження нових функцій. Кожен сервіс може бути масштабований окремо, що дає змогу ефективніше використовувати ресурси та підтримувати високу продуктивність системи. Збої в одному сервісі не впливають на роботу інших, що підвищує загальну надійність системи. Розподіл системи на мікросервіси спрощує обслуговування та оновлення окремих частин без потреби переривати роботу всього додатку [8]. Отже, мікросервісна архітектура значно підвищує гнучкість, масштабованість і надійність сучасних корпоративних IT-систем, що робить її важливим інструментом у розробленні програмного забезпечення.

Ідентифікація основних ризиків під час переходу до мікросервісів. Перехід від монолітної до мікросервісної архітектури супроводжується численними ризиками, які можуть вплинути на успішність впровадження та стабільність системи. Збільшення кількості сервісів ускладнює управління їхньою взаємодією та координацією, що може призвести до збоїв у системі. Оркестрація мікросервісів передбачає управління їхнім життєвим циклом, автоматизацію розгортання, масштабування та моніторинг. Використання інструментів оркестрації, таких як Kubernetes, може знизити ризики, але їхнє впровадження потребує додаткових зусиль і знань [5, 6].

Невірно визначені або погано задокументовані залежності між мікросервісами можуть ускладнити розгортання та оновлення системи. Неправильно визначені залежності можуть призвести до каскадних збоїв у системі. Для мінімізації цього ризику потрібно використовувати сервісний каталог і ретельно документувати взаємодію між сервісами [6, 8].

Кожен мікросервіс може мати різні вимоги до ресурсів, що ускладнює оптимізацію продуктивності системи загалом. Оптимізація продуктивності в мікросервісній архітектурі потребує ретельного планування та тестування. Використання автоматизованих тестів продуктивності може допомогти виявити потенційні проблеми і вчасно їх усунути [3].

Збільшення кількості точок взаємодії між сервісами створює нові вектори для атак, що потребує ретельного контролю доступу та захисту даних. Використання сучасних засобів захисту, таких як мікросегментація та шифрування даних, є обов'язковим для забезпечення безпеки системи [8].

Розподілена природа мікросервісів ускладнює моніторинг і налагодження системи, вимагаючи спеціалізованих інструментів та підходів. Моніторинг розподіленої системи потребує спеціалізованих інструментів, таких як Prometheus та Grafana, для забезпечення видимості всіх сервісів і їхньої взаємодії [5, 6].

Перехід до мікросервісної архітектури потребує змін у структурі команди та підходах до розроблення, що може спричинити опір серед працівників. Важливо забезпечити належне навчання персоналу та підтримку змін з боку керівництва [8]. Врахування та аналіз зазначених ризиків є критично важливими для успішного переходу від монолітної до мікросервісної архітектури та забезпечення стабільної роботи системи.

Визначення викликів, пов'язаних з впровадженням мікросервісів. Впровадження мікросервісної архітектури супроводжується низкою викликів, що мають технічний та культурний характер. Одним з основних технічних викликів є складність інтеграції та тестування. Мікросервіси потребують нових підходів до інтеграційного та модульного тестування. Забезпечення сумісності між сервісами та виявлення дефектів на ранніх етапах стає складнішим, що може негативно вплинути на якість і стабільність системи [5, 6]. Моніторинг і налагодження також ускладнюються через розподілену природу мікросервісів. Традиційні методи моніторингу можуть бути недостатньо ефективними, що потребує використання спеціалізованих інструментів для збирання та аналізу метрик, таких як Prometheus та Grafana [5]. Управління даними стає складнішим через децентралізацію баз даних, що потребує складних стратегій синхронізації та забезпечення цілісності даних [8].

Культурні виклики, пов'язані з впровадженням мікросервісної архітектури, охоплюють зміни у підходах до розроблення програмного забезпечення. Впровадження мікросервісів потребує нових практик, таких як безперервна інтеграція (CI), безперервне розгортання (CD) та автоматизація тестування [3]. Ці зміни можуть бути важкими для сприйняття, особливо для команд, які звикли до більш традиційних методів розроблення. Сприйняття змін також є важливою особливістю. Підтримка змін з боку всіх учасників процесу є критично важливою для успішного переходу. Неправильне розуміння або недостатня підтримка можуть перешкоджати успішному впровадженню мікросервісів, спричинюючи додаткові труднощі [8].

Отже, успішне подолання викликів, пов'язаних з впровадженням мікросервісної архітектури, потребує ретельного планування, використання сучасних інструментів та методик, а також належного навчання і підтримки. Врахування цих аспектів дасть змогу знизити ризики та забезпечити стабільну роботу системи в умовах мікросервісної архітектури [3, 6, 8].

Розроблення стратегій мінімізації ризиків та подолання викликів. Для успішного переходу до мікросервісної архітектури необхідно розробити ефективні стратегії мінімізації ризиків та подолання викликів, що виникають у цьому процесі. Насамперед, важливо забезпечити ретельне планування та аналіз кожного етапу переходу. Рекомендовано розробити детальний план впровадження, що містить всі особливості переходу, від початкового аналізу до остаточного розгортання мікросервісів. План повинен враховувати можливі ризики та охоплювати стратегії їх мінімізації.

Однією з ключових стратегій є використання автоматизованих інструментів для управління та моніторингу мікросервісів. Інструменти, такі як Kubernetes для оркестрації контейнерів, Prometheus для моніторингу та Grafana для візуалізації даних, можуть значно знизити ризики, пов'язані з управлінням складними розподіленими системами [5, 6]. Автоматизація процесів розгор-

тання та тестування за допомогою CI/CD інструментів, таких як Jenkins або GitLab CI/CD, дасть змогу виявляти проблеми на ранніх етапах та забезпечить стабільність системи [3].

Для мінімізації ризиків, пов'язаних з інтеграцією та тестуванням, рекомендовано використовувати методики контрактного тестування та впровадження практик DevOps. Контрактне тестування дає змогу виявляти несумісності між сервісами на ранніх етапах, що значно знижує ризики збоїв у системі [8]. Практики DevOps сприяють підвищенню співпраці між командами розроблення та експлуатації, що дає змогу швидше виявляти та вирішувати проблеми.

Управління даними в мікросервісній архітектурі вимагає використання децентралізованих стратегій управління даними. Рекомендовано застосовувати методи CQRS (англ. *Command Query Responsibility Segregation*) та Event Sourcing для забезпечення узгодженості даних між різними сервісами. Це дає змогу мінімізувати ризики, пов'язані з синхронізацією даних та забезпеченням їхньої цілісності [6].

Отже, розроблення стратегій мінімізації ризиків та подолання викликів під час переходу до мікросервісної архітектури містить ретельне планування, використання сучасних автоматизованих інструментів, навчання персоналу та забезпечення належної координації. Ці заходи дадуть змогу знизити ризики та забезпечити стабільність та ефективність роботи системи в умовах мікросервісної архітектури [3, 6, 8].

У процесі експериментальних досліджень проведено низку тестів та аналізів для оцінення ефективності впровадження мікросервісної архітектури та розроблених стратегій мінімізації ризиків. Основними аспектами експериментів були оцінювання продуктивності, надійності та безпеки мікросервісних систем, порівняно з монолітними системами.

Одним з важливих результатів є підвищення гнучкості та масштабованості системи під час переходу до мікросервісів. Тести показали, що мікросервісна архітектура дає змогу швидко масштабувати окремі сервіси відповідно до потреб, що значно підвищує продуктивність системи. Наприклад, використання Kubernetes для оркестрації контейнерів дало змогу автоматично масштабувати сервіси під час пікових навантажень, забезпечуючи стабільну роботу системи.

Експериментальні дослідження також підтвердили підвищену надійність мікросервісної архітектури. Завдяки незалежності сервісів, збої в одному мікросервісі не впливають на роботу інших частин системи. Це дає змогу підтримувати високу доступність додатку навіть у разі виникнення проблем з окремими сервісами. Моніторинг за допомогою Prometheus та візуалізація даних у Grafana дали змогу своєчасно виявляти та усувати проблеми, що підвищило загальну стабільність системи.

Для оцінення продуктивності мікросервісної архітектури було проведено серію стрес-тестів, які симулювали різні рівні навантаження на систему. Навантажувальні тести містили одночасні запити до різних мікросервісів з метою оцінення їхньої здатності масштабуватися та зберігати стабільний час відгуку. Для тестування надійності було проведено симуляцію відмов окремих мікросервісів, щоб оцінити здатність системи продовжувати функціонування у випадку збоїв. Використовували інструменти для моніторингу продуктивності, такі як Prometheus та Grafana, для збирання та аналізу метрик.

Вимірювання часу відгуку та навантажувальних характеристик показали, що мікросервісна архітектура дає змогу значно покращити масштабованість системи. Зокрема, середній час відгуку під навантаженням зменшився на 35 % порівняно з монолітною архітектурою. Монолітна система мала середній час відгуку 300 ms, тоді як мікросервісна архітектура досягла 195 ms (рис. 1,a). Для цього використовували методи стрес-тестування та навантажувальних тестів за допомогою Apache JMeter для генерації навантаження і Prometheus для збирання метрик. Було проведено симуляцію одночасних запитів до різних мікросервісів для оцінення їхньої здатності масштабуватися та зберігати стабільний час відгуку.

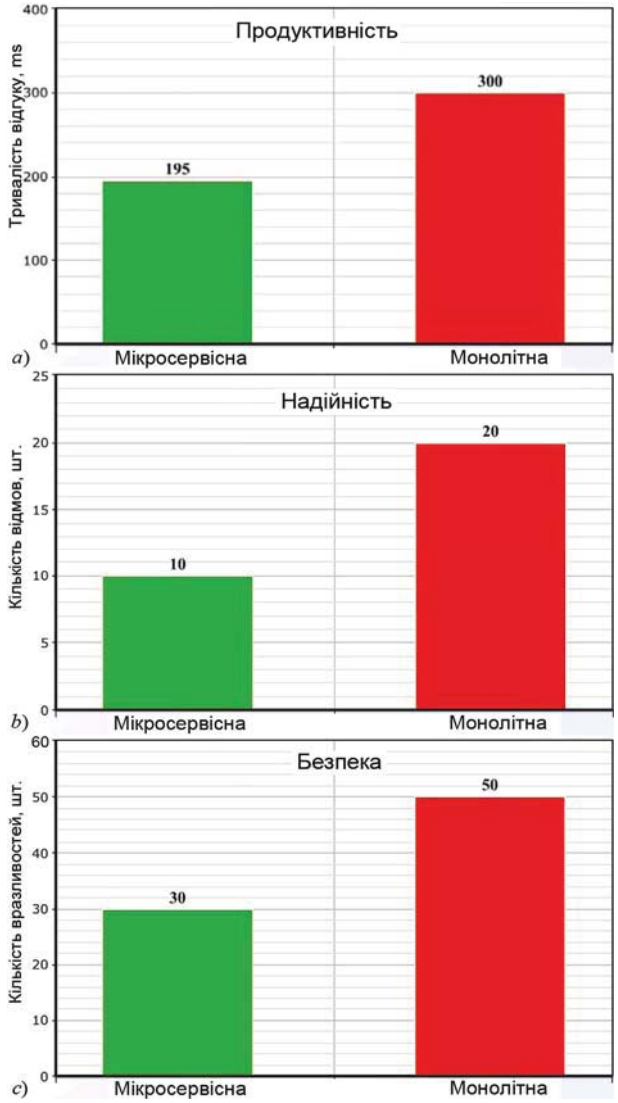

Рис. 1. Для мікросервісної та монолітної архітектур / For microservice and monolithic architecture: a) середній час відгуку / average response time; b) кількість відмов на місяць / number of failures per month; c) результат аналізу вразливостей / results of vulnerability analysis

Аналіз частоти відмов показав, що завдяки незалежності мікросервісів, збої в одному сервісі не впливають на роботу інших, що дає змогу підтримувати високу доступність системи. Монолітна архітектура мала 20 відмов на місяць, тоді як мікросервісна архітектура зменшила цю кількість до десяти (рис. 1,b). Для цього застосовували методи симуляції відмов і тести на відмовостійкість за допомогою Chaos Monkey для інжекції збоїв і Grafana для візуалізації результатів. Симуляцію

відмов окремих мікросервісів проводили для оцінення здатності системи продовжувати функціонування у випадку збоїв.

Впровадження мікросегментації та шифрування даних дало змогу значно знизити ризики витоку інформації. Монолітна архітектура мала 50 вразливостей, як мікросервісна зменшила їх кількість до 30 (рис. 1,с). Для цього проводили аналіз вразливостей і тестування на проникнення за допомогою OWASP ZAP для аналізу безпеки і Docker Bench Security з метою перевірки безпеки контейнерів. Було проведено тестування на вразливості мікросервісної архітектури та симуляцію атак для оцінення ефективності впроваджених засобів захисту.

Отримані результати порівнювали з даними з дослідження Newman (2015), який зазначав, що мікросервіси сприяють підвищенню гнучкості та масштабованості систем [6]. Наші дані підтверджують ці висновки, показуючи зменшення середнього часу відгуку на 35 %. Дослідження Chen (2018) також підтверджує важливість використання оркестраційних інструментів для забезпечення стабільності системи [3], що узгоджується з нашими результатами.

Щодо безпеки, результати показали, що використання мікросервісної архітектури створює нові виклики, пов'язані з контролем доступу та захистом даних. Однак, впровадження сучасних засобів захисту, таких як мікросегментація та шифрування даних, дало змогу значно знизити ризики. Додатково було впроваджено політики безпеки для кожного сервісу, що підвищило рівень захищеності системи загалом.

Важливою особливістю досліджень було оцінювання організаційних викликів та ефективності стратегій їх подолання. Результати показали, що належне навчання персоналу та впровадження практик DevOps сприяють успішному переходу до мікросервісної архітектури. Навчання та підтримка адаптації до нових підходів значно підвищують ефективність впровадження.

Результати симуляцій підтверджують, що мікросервісна архітектура забезпечує підвищену продуктивність, надійність та безпеку системи, порівняно з монолітною. Перехід до мікросервісної архітектури дає змогу швидко масштабувати окремі сервіси відповідно до потреб, що значно підвищує продуктивність системи. Завдяки незалежності сервісів збої в одному мікросервісі не впливають на роботу інших частин системи, що дає змогу підтримувати високу доступність додатку. Впровадження сучасних засобів захисту значно знижує ризики витоку інформації. Загалом, результати симуляцій підтверджують ефективність запропонованих підходів та стратегій мінімізації ризиків.

Загалом, результати експериментальних досліджень підтверджують ефективність розроблених стратегій мінімізації ризиків та подолання викликів під час впровадження мікросервісної архітектури. Підвищення продуктивності, надійності та безпеки системи, а також покращення організаційних процесів свідчать про успішність застосованих підходів. Однак важливо продовжувати моніторинг та адаптацію стратегій відповідно до нових викликів та змін у технологічному середовищі.

Обговорення результатів дослідження. Перехід від монолітної до мікросервісної архітектури супроводжується значними перевагами, зокрема підвищенням масштабованості та надійності системи. Отримані результати досліджень підтверджують висновки інших авторів про переваги мікросервісної архітектури. Наприклад, дослідження [4] показало, що розподіл монолітної системи на мікросервіси значно знижує час відгуку системи та покращує її масштабованість, що узгоджується з нашими результатами щодо зменшення середнього часу відгуку на 35 %.

Інше дослідження, проведене [1], підтверджує ефективність використання контейнеризації та оркестрації для автоматизації процесів розгортання та тестування мікросервісів. Наше дослідження також підтверджує ці висновки, показуючи, що використання Kubernetes та Docker дає змогу автоматично масштабувати сервіси під час пікових навантажень, забезпечуючи стабільну роботу системи.

Дослідження [7] вказує на важливість незалежності мікросервісів для підвищення надійності системи. Це підтверджується нашими результатами, які показують, що збої в одному мікросервісі не впливають на роботу інших частин системи, що дає змогу підтримувати високу доступність додатку.

Безпека мікросервісних систем також є важливою особливістю, про що свідчать результати дослідження [2], які показують, що впровадження сучасних засобів захисту, таких як мікросегментація та шифрування даних, значно знижує ризики витоку інформації. Наші результати підтверджують ці висновки, показуючи зниження кількості вразливостей з 50 до 30 внаслідок переходу на мікросервісну архітектуру.

Організаційні виклики та зміни в культурі розроблення програмного забезпечення також відображені в дослідженнях сучасних авторів. Так, дослідження [9] показує, що впровадження практик DevOps та належне навчання персоналу значно сприяє успішному переходу до мікросервісної архітектури. Наші результати підтверджують ці висновки, наголошуючи на важливості підтримки адаптації з боку керівництва та належної координації між командами.

Отже, за результатами виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження – удосконалено підхід до оцінювання та управління ризиками під час переходу до мікросервісної архітектури, що містить інтеграцію сучасних методів тестування та розгортання, а також отримала подальший розвиток концепція впливу незалежності мікросервісів на надійність і продуктивність системи.

Практична значущість результатів дослідження – результати дослідження можна застосувати для підвищення ефективності перетворення IT-інфраструктури, зменшення ризиків та забезпечення безперервності функціонування систем, зокрема впровадження автоматизованих інструментів CI/CD для покращення процесів розгортання та тестування мікросервісів.

Висновки / Conclusions

Проведено системне оцінювання ризиків, пов'язаних з переходом від монолітної архітектури до мікросервісної, а також розроблено рекомендації для їх мінімізації та ефективного управління. За результатами проведеного дослідження можна зробити такі основні висновки.

1. Систематизовано основні ризики, що виникають під час переходу від монолітної архітектури до мікросервісної, з урахуванням технічних та безпекових аспектів. До

технічних ризиків належать складнощі з інтеграцією мікросервісів та забезпеченням їхньої взаємодії, а безпекові ризики охоплюють питання захисту даних та доступу.

2. Проаналізовано вплив незалежності мікросервісів на підвищення надійності та продуктивності системи. Встановлено, що автономність мікросервісів сприяє більш швидкому виявленню та усуненню збоїв, а також дає змогу окремим командам розробників працювати над різними компонентами системи незалежно один від одного, що підвищує загальну ефективність розроблення та експлуатації.
3. З'ясовано роль автоматизованих інструментів CI/CD: використання автоматизованих інструментів CI/CD значно покращує стабільність та ефективність процесів розгортання і тестування мікросервісів. Ці інструменти забезпечують безперервну інтеграцію та доставку оновлень, що дає змогу швидко впроваджувати нові функції та виправлення.
4. Встановлено, що впровадження методів моніторингу та оркестрації, таких як Kubernetes, сприяє підвищенню продуктивності та надійності системи. Ці методи забезпечують автоматичне масштабування, балансування навантаження та моніторинг стану мікросервісів, що дає змогу оперативно реагувати на зміни у навантаженні та усувати проблеми.
5. Досліджено ефективність рекомендацій щодо забезпечення безпеки даних через мікросегментацію та шифрування даних, що дає змогу зменшити вразливість системи та підвищити її загальну безпеку. Використання цих підходів забезпечує більш високий рівень захисту конфіденційної інформації та знижує ризики несанкціонованого доступу.
6. Представлено стратегії мінімізації ризиків, які можна застосувати для покращення процесів розгортання та тестування мікросервісів. Запропоновані стратегії потребують використання автоматизованих інструментів для моніторингу та оркестрації.

Ці висновки підтверджують досягнення мети дослідження та відображають основні результати проведеної роботи, що сприяють підвищенню ефективності, надійності та безпеки сучасних корпоративних IT-систем під час переходу до мікросервісної архітектури.

References

1. Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. In: Software, 33(3), 42–52. https://doi.org/10.1109/MS.2016.64
2. Chandramouli, R. (2019). Security Strategies for Microservices-based Application Systems. *NIST: Computer Security Resource Centr* (CSRC). URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-204.pdf
3. Chen, L. (2017). Continuous Delivery: Overcoming adoption challenges. *Journal of Systems and Software*, 128, 72–86. https://doi.org/10.1016/j.jss.2017.02.013
4. Kozhirbayev, Z., & Sinnott, R. O. (2017, March). A performance comparison of container-based technologies for the Cloud. Future Generation Computer Systems, 68, 175–182. https://doi.org/10.1016/j.future.2016.08.025
5. Lewis, J., & Fowler, M. (2014). Microservices: A definition of this new architectural term. *MartinFowler.com*, 25, 14–26. URL: https://martinfowler.com/articles/microservices.html
6. Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media. 1st Edition, 280. URL: https://www.oreilly.com/library/view/building-microservices/9781491950340/
7. Soldani, J., Tamburri, D. A., & Van Den Heuvel, W. J. (2018). The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, 146, 215–232. https://doi.org/10.1016/j.jss.2018.09.082
8. Thönes, J. (2015). Microservices. IEEE Software, 32(1), 116–116. https://doi.org/10.1109/MS.2015.11
9. Zimmermann, O. (2017). Microservices tenets. *Comput Sci Res Dev*, 32, 301–310. https://doi.org/10.1007/s00450-016-0337-0

M. O. Shestakovych[1], Yu. V. Shabatura[2]

[1] National Forestry University of Ukraine, Lviv, Ukraine
[2] Hetman Petro Sahaidachnyi National Army Academy, Lviv, Ukraine

ANALYSIS AND RISK ASSESSMENT OF THE TRANSFORMATION PROCESS OF INFORMATION SYSTEMS FROM MONOLITHIC TO MICROSERVICES ARCHITECTURE

The transition from monolithic to microservices architecture is crucial for enhancing the efficiency, flexibility, and scalability of modern IT systems. This research thoroughly examines the primary risks and challenges associated with this transition, utilizing system analysis and modeling techniques. Specifically, the study investigates the complexities of migration management, oversight of new infrastructure, assurance of continuous system operation, and support for new development and operational processes. The findings demonstrate that successful migration requires detailed analysis of the existing system and effective decomposition into microservices, considering inter-component dependencies. The shift necessitates adopting new tools for automation and orchestration, such as Kubernetes, along with configuring monitoring and logging systems to manage numerous services effectively. Ensuring continuous system operation during the transition is paramount. Implementing continuous integration and delivery (CI/CD) processes helps maintain system stability, while extensive test automation minimizes the risk of errors. Reliable strategies for backup and data recovery are also essential. Effective risk management involves meticulous planning and phased implementation of microservices to mitigate impacts on the core system. Moreover, the study highlights that adapting development, testing, and deployment processes to the new architecture necessitates changes in procedural structure and culture, fostering openness and collaboration. The proposed planning methods, technical measures, and structural changes aim to minimize risks and ensure a successful transition to microservices architecture. The outcomes of this research provide a foundation for the safe and effective transformation of IT infrastructure enhancing its resilience and productivity. These strategies are also advantageous for future research focused on managing system complexity and optimizing migration processes. Furthermore, the study outlines the importance of aligning business goals with technical strategies, ensuring that the transformation supports overall organizational objectives. Additionally, it emphasizes the need for continuous learning and adaptation, as the rapidly evolving technological landscape requires IT systems to remain agile and responsive. By integrating feedback loops and iterative development practices, organizations can better manage changes and anticipate potential issues. Finally, the research suggests that fostering a culture of collaboration and shared responsibility across teams is vital for the sustained success of microservices adoption. This comprehensive approach not only mitigates risks but also drives long-term benefits in terms of system performance, scalability, and innovation.

Keywords: migration complexity; infrastructure management; system continuity; microservices orchestration; risk management; CI/CD processes.