КОМП'ЮТЕРНІ НАУКИ

УДК 004.4 DOI: https://doi.org/10.32515/2664-262X.2024.10(41).1. 3-16

О.С. Улічев, канд. техн. наук
Центральноукраїнський національний технічний університет, м. Кропивницький, Україна
Л.М. Папіж, асп.
ПВНЗ "Європейський університет", м. Київ, Україна
О.А. Ревнюк, асп.
Тернопільський національний технічний університет імені Івана Пулюя, м. Тернопіль, Україна
e-mail: askin79@gmail.com, l.m.papizh@gmail.com, revo0708@gmail.com

Досягнення в тестуванні програмного забезпечення. Наукова перспектива

Тестування програмного забезпечення відіграє ключову роль у визначенні надійності, функціональності та безпеки систем. Стаття досліджує різні аспекти процесу, акцентуючи увагу на наукових принципах, що підвищують результативність процесу. Аналізуються сучасні методології, інструменти та техніки, висвітлюються останні досягнення. Досліджується вплив наукових принципів на тестування та процеси забезпечення якості, практичні наслідки досліджень через аналіз прикладів, їхня ефективність у покращенні якості та зниженні витрат. Мета статті - сприяти розумінню наукових засад тестування програм та його важливій ролі у розвитку майбутніх практик інженерії програмного забезпечення.

тестування програмного забезпечення, тестування зсув ліворуч, неперервне тестування, штучний інтелект та машинне навчання, автоматизація тестування, тестування зсув праворуч, управління тестовими даними

Постановка проблеми. У цій статті досліджуються різні аспекти інновацій у тестуванні ПЗ, включаючи новітні тенденції, виклики та стратегії для виходу вперед, у постійно змінній галузі розробки програмного забезпечення. Серед ключових моментів варто виділити:

- Важливість прийняття інноваційних практик тестування, таких як тестування "зсув ліворуч", НТ та тестування, що використовує ШІ, для підвищення точності, ефективності та надійності тестів.
- Загальні виклики в реалізації інноваційних практик тестування, такі як опір змінам, застарілі системи, недоліки у навичках та проблеми з конфіденційністю даних.
- Нові тенденції та технології, що формують майбутнє тестування ПЗ, включаючи ШІ та МН, НТ, тестування великих даних, тестування Інтернету речей та автоматизацію роботизованих процесів.
- Стратегії подолання викликів та випередження конкурентів, такі як постійне навчання, співпраця, пілотні проекти, гнучкі методології та орієнтація на клієнта.

У сучасному швидкозмінному та динамічному цифровому середовищі забезпечення якості програмного забезпечення (ПЗ) стає все більш критичним завданням. Традиційні методи тестування часто не встигають за темпами розвитку технологій та вимогами ринку, що призводить до затримок у випуску продуктів, підвищення витрат та зниження якості ПЗ. Основна проблема полягає в необхідності впровадження інноваційних практик тестування, які можуть забезпечити високу точність, ефективність та надійність тестів.

Ця проблема має важливе наукове та практичне значення. З наукової точки зору, дослідження нових методів та технологій тестування, таких як штучний інтелект (ШІ), машинне навчання (МН), тестування великих даних та Інтернету речей (IoT), відкриває нові горизонти для підвищення якості ПЗ. Практичне значення полягає у здатності організацій швидко адаптуватися до змін, знижувати витрати на розробку та тестування, а також забезпечувати високу якість продуктів, що відповідають потребам та очікуванням користувачів.

Впровадження інноваційних практик тестування також стикається з низкою викликів, таких як опір змінам з боку зацікавлених сторін, застарілі системи, недоліки у навичках та проблеми з конфіденційністю даних. Подолання цих викликів вимагає ефективної комунікації, освіти, інвестицій у нові інструменти та технології, а також постійного вдосконалення процесів тестування.

Таким чином, дослідження та впровадження інноваційних практик тестування є ключовим завданням для забезпечення конкурентоспроможності та успіху організацій у сучасному цифровому світі.

Аналіз останніх досліджень і публікацій. У статті розглядаються сучасні проблеми та інноваційні підходи до тестування програмного забезпечення (ПЗ), спираючись на численні дослідження та публікації. Основні аспекти, які висвітлюються в статті, включають тестування зі зсувом ліворуч, неперервне тестування, автоматизацію тестування, використання штучного інтелекту (ШІ) та машинного навчання (МН), а також тестування зі зсувом праворуч.

Тестування зі зсувом ліворуч у дослідженні Дональда Файрсміта [1] та Пета Фелана і Рене Веллса [2] підкреслюють важливість раннього тестування, що дозволяє виявляти дефекти на початкових етапах розробки ПЗ. Ці роботи акцентують увагу на економічних перевагах та підвищенні якості продукту завдяки ранньому виявленню дефектів. Максиміліано А. Маскероні та Емануель Ірразабаль [3] у своєму систематичному огляді літератури досліджують рішення для проблем тестування в умовах неперервної доставки. Вони підкреслюють важливість автоматизації тестування та інтеграції тестових процесів на кожному етапі розробки ПЗ. Альтай Атаман [4] також розглядає переваги та виклики неперервного тестування, акцентуючи на його ролі у забезпеченні якості та надійності ПЗ.Енріке Де Косс [10] та Леонардо Маріані з колегами [11] досліджують еволюцію автоматизації тестування та її центральну роль у забезпеченні якості ПЗ. Вони підкреслюють важливість використання сучасних інструментів та фреймворків для автоматизації тестування, таких як Selenium, Appium та Cypress. Ніко Крюгер [6] та Gradient Ascent [7] досліджують роль Штучного Інтелекту та Машинного Навчання у тестуванні ПЗ, підкреслюючи їх здатність покращувати процеси тестування та знижувати кількість помилок. Вони наводять приклади успішного використання ШІ та МН для автоматизації тестування та аналізу логів. Піюша Подутвар [13] та Парваті Пурушотаман [15] розглядають переваги тестування зі зсувом праворуч, яке акцентується на тестуванні після випуску ПЗ. Вони підкреслюють важливість отримання зворотного зв'язку від користувачів та виявлення проблем у реальних умовах використання.

Незважаючи на значний прогрес у дослідженнях та впровадженні інноваційних практик тестування, залишаються невирішені частини загальної проблеми, які потребують подальшого вивчення. Серед них: інтеграція нових технологій, таких як ШІ та МН, у існуючі процеси тестування. Подолання опору змінам з боку зацікавлених сторін та адаптація до нових методологій. Забезпечення конфіденційності та цілісності даних під час тестування. Розвиток навичок та знань у сфері інноваційних методів тестування серед фахівців. Ця стаття присвячена дослідженню та вирішенню цих

невирішених питань, пропонуючи нові підходи та стратегії для покращення якості та ефективності тестування ПЗ.

Виклад основного матеріалу. Тестування ПЗ охоплює діяльності для оцінки якості та продуктивності програмних продуктів. Це включає тестові випробування, сценарії та скрипти для виявлення дефектів, перевірки функціональності та відповідності специфікаціям. Ретельне тестування дозволяє виявити та вирішити проблеми на ранніх етапах, зменшуючи ймовірність помилок у виробничих середовищах. Значення тестування ПЗ виходить за межі виявлення дефектів; це фундаментальний аспект забезпечення якості, включаючи зручність використання, безпеку та продуктивність. Практики тестування допомагають забезпечити відповідність ПЗ функціональним і нефункціональним вимогам. У сучасному ринковому середовищі, де користувацький досвід вирішальний, роль тестування ПЗ у створенні якісного ПЗ не можна недооцінювати.

Тестування зі зсувом ліворуч. Останнім часом відбулася зміна парадигми в практиках тестування ПЗ на користь підходу "зсув ліворуч". Цей метод передбачає переміщення тестувальних дій на ранні етапи життєвого циклу розробки ПЗ, зазвичай на початкові етапи процесу розробки[1].Підхід підкреслює важливість раннього тестування, такого як модульне тестування, статичний аналіз коду та рецензії коду, на відміну від відкладання тестування до пізніших етапів.

Тестування "зсув ліворуч" ґрунтується на принципі "тестування на ранніх етапах, тестування часто". Переміщення тестувальних дій на початок життєвого циклу розробки дозволяє організаціям виявляти та виправляти дефекти та вразливості меншими коштами та швидше. Основні принципи включають: раннє виявлення дефектів, що знижує ризик їх потрапляння в експлуатаційні середовища продукції. Тестування за типом "зсув ліворуч" забезпечує постійний зворотний зв'язок між командами розробки та тестування, дозволяючи швидші ітерації та вирішення проблем. Розробники отримують негайний зворотний зв'язок, що дозволяє їм вносити корективи в реальному часі. Це сприяє співпраці та комунікації між розробниками, тестувальниками та іншими зацікавленими сторонами протягом всього процесу розробки. Раннє залучення тестувальних команд забезпечує пріоритетність якості з самого початку. Автоматизація дозволяє швидко виконувати тести та аналіз коду, покращуючи ефективність та надійність результатів.

Раннє виявлення та запобігання дефектам є основою стратегії "зсув ліворуч" і приносить значні переваги організаціям. Виявлення та виправлення дефектів на ранніх етапах значно знижує витрати. Дослідження показують [2], що витрати на виправлення дефектів зростають експоненційно через життєвий цикл розробки ПЗ, тому раннє виявлення більш вигідне з економічної точки зору. Це прискорює цикл розробки та скорочує час до введення продукту на ринок, дозволяючи швидше реагувати на ринкові вимоги. Раннє виявлення дефектів підвищує якість та надійність продукту, що забезпечує позитивний досвід користувача, збільшує задоволеність та лояльність клієнтів. Організації, активно усуваючи дефекти, здобувають довіру клієнтів і підсилюють свою репутацію на ринку.

Комерційні організації ефективно застосовують методології тестування "зсув ліворуч" та отримують значні покращення якості ПЗ, ефективність та задоволення клієнтів[2]. Сценарії компаній демонструють ефективність тестування "зсув ліворуч" у різних галузях. Організації відзначили зниження дефектності через виявлення та виправлення проблем на ранніх етапах розробки. Це дозволило прискорити випуски продуктів та швидше виводити їх на ринок, отримуючи конкурентну перевагу. Раннє залучення тестувальних команд покращило співпрацю між розробниками і тестувальниками, сприяючи кращому розумінню цілей щодо якості. Організації

повідомляли про покращення якості продукту, надійності та задоволення користувачів через раннє виявлення та попередження дефектів. Ці приклади підтверджують важливість та актуальність тестування "зсув ліворуч" у сучасних практиках розробки ПЗ.

Неперервне тестування (НТ) є ключовим аспектом сучасних практик розробки ПЗ, що забезпечує якість та надійність програмних продуктів протягом усього життєвого циклу ПЗ [3]. Воно включає автоматизацію тестування та інтеграцію на кожному етапі, від початкової розробки до впровадження та підтримки.

НТ ґрунтується на принципі вчасного, частого і паралельного тестування з розробкою [4]. Воно включає модульне, інтеграційне, функціональне, продуктивності та безпеки тестування. Автоматизація тестування і включення його в процес неперервної інтеграції/неперервного впровадження (НІ/НВ) дозволяє виявляти дефекти на ранніх етапах, забезпечуючи швидкий зворотний зв'язок. НТ інтегроване з іншими DevOps практиками, такими як НІ та НВ, і підкреслює співпрацю між командами, забезпечуючи швидші ітерації, коротші цикли випуску та вищу якість ПЗ.

Інструменти, фреймворки та методології для впровадження НТ. Для впровадження практик неперервного тестування доступні численні інструменти, фреймворки та методології [5].

Таблиця 1 - Інструменти автоматизації процесів тестування

Категорія інструментів/практик тестування	**Опис та призначення інструментів/практик**
Фреймворки автоматизації тестування	Інструменти, такі як Selenium, Appium та Cypress, часто використовуються для автоматизації функціонального тестування WEB та мобільних додатків. Ці фреймворки дозволяють тестувальникам писати автоматизовані тестові скрипти, які можна виконувати повторно та з очікуваними надійними результатами на різних середовищах та платформах.
Інструменти для тестування продуктивності	Інструменти, такі як JMeter, Load Runner та Gatling, використовуються для тестування продуктивності, що дозволяє організаціям оцінити масштабованість, реактивність та стабільність їх додатків за різних навантажень.
Інструменти для тестування API	Інструменти, такі як Postman, Soap UI та REST Assured, сприяють автоматизованому тестуванню API, дозволяючи організаціям перевіряти функціональність, надійність та безпеку своїх API точок доступу.
Контейнеризація та оркестрація	Технології, такі як Docker та Kubernetes, дозволяють організаціям створювати та керувати ізольованими тестовими середовищами за допомогою контейнерів. Контейнеризація забезпечує більшу гнучкість, масштабованість та послідовність в тестових середовищах, сприяючи швидшому та надійному тестуванню.
Практики тестування "зсув ліворуч"	Постійне тестування часто доповнюється практиками "зсув ліворуч", такими як статичний аналіз коду, рецензування коду та модульне тестування. Ці практики допомагають виявляти дефекти та вразливості на ранніх етапах розробки, зменшуючи ймовірність проблем у експлуатаційних середовищах.

Джерело: розроблено авторами

Штучний інтелект та машинне навчанняце технології, що революціонізують галузь тестування ПЗ, пропонуючи інноваційні рішення для покращення процесів тестування та результатів [6]. У цьому розділі ми досліджуємо застосування ШІ та МН в тестуванні ПЗ, переваги рішень з тестування, що використовують ШІ, а також реальні випадки, що демонструють їх ефективність.

Технології ШІ та МН мають широкий спектр застосувань в тестуванні ПЗ. Алгоритми можуть автоматично генерувати тестові сценарії на основі вимог, аналізу коду та історичних даних [7], прискорюючи процес та забезпечуючи комплексне покриття. Моделі ШІ можуть аналізувати зміни в коді, результати тестування та інші дані для прогнозування дефектів. Це дозволяє пріоритезувати зусилля на високоризикові області. Алгоритми ШІ пріоритезують та планують тести, оптимізуючи ресурси та скорочуючи час виходу на ринок. Техніки ШІ можуть виявляти аномалії та відхилення, дозволяючи виявляти дефекти на ранніх стадіях. Вони аналізують журнали та системні дані, виявляючи шаблони, тенденції та аномалії, що вказують на проблеми в ПЗ [8].

Переваги ШІ у тестуванні ПЗ та оптимізації коду включають наступні аспекти: ефективність (автоматизація генерації тестів, прогнозування дефектів і оптимізація виконання тестів зменшують повторювані завдання, підвищуючи ефективність і продуктивність); точність (алгоритми ШІ аналізують великі обсяги даних і виявляють шаблони, пропущені людьми, покращуючи точність тестування); масштабованість (ШІ рішення масштабуються для складних систем і великих проектів, дозволяючи комплексне тестування); зниження витрат (автоматизація тестування і оптимізація ресурсів знижують витрати та скорочують час виходу на ринок, надаючи конкурентну перевагу); покращення якості (ШІ рішення виявляють дефекти на ранніх етапах, підвищуючи якість ПЗ і задоволення клієнтів).

Наведені переваги підкреслюють ефективність технологій ШІ та МН у вдосконаленні процесів тестування та результатів, демонструючи їх потенціал для революціонізації галузі тестування ПЗ. Приймаючи рішення щодо використання тестування, заснованого на ШІ, організації можуть досягти швидших, надійніших і вищої якості випусків ПЗ, що в кінцевому підсумку приносить їхнім клієнтам задоволення якістю [9].

Автоматизація тестування є невід'ємною частиною сучасних практик, покращуючи ефективність, точність і надійність процесів тестування. У цій частині досліджується еволюція автоматизації, її роль у підвищенні ефективності тестування ПЗ, найкращі практики для впровадження автоматизованих тестових наборів, а також виклики та аспекти реалізації [10].

Автоматизація тестування значно еволюціонувала від ручних методів до складних фреймворків та інструментів. Спочатку вона зосереджувалася на автоматизації повторюваних і часоємнісних ручних завдань, таких як регресійне та оглядове тестування. З розвитком технологій та практик Agile і DevOps автоматизація охопила різні типи тестування, включаючи функціональне, тестування продуктивності та API. Роль автоматизації тестування у вдосконаленні ефективності тестування ПЗ має багатогранний характер [11]:

- Швидкість та точність: автоматизовані скрипти виконують тести швидше та точніше за ручне тестування, що прискорює цикл зворотного зв'язку та зменшує час виходу на ринок.
- Повторюваність та послідовність: автоматизовані тести зменшують ризик людських помилок та забезпечують послідовність результатів.
- Збільшення покриття тестування: автоматизація дозволяє виконувати більше тестів за менший час, підвищуючи впевненість у якості ПЗ.

• Регресійне тестування: автоматизація регресійного тестування швидко виявляє та усуває дефекти від змін у коді, зберігаючи функціональність під час розробки.

Загалом, автоматизація тестування важлива для оптимізації процесів, зменшення ручної праці та підвищення ефективності тестування в проектах розробки ПЗ. Успішне впровадження та підтримка автоматизованих тестових наборів вимагають дотримання найкращих практик[12]:

• Проектування та планування тестів: розробка стратегії тестування та пріоритезація сценаріїв на основі ризику та впливу на бізнес.

• Вибір і проектування інструментів та фреймворків: вибір відповідних інструментів для автоматизації тестування на основі вимог проекту та експертизи.

• Проектування та виконання тестових сценаріїв: написання чітких тестових скриптів, використовуючи моделі сторінки та тестування на основі даних. Пріоритезація сценаріїв на основі частоти, складності та критичності.

• Постійна інтеграція та постачання: інтеграція автоматизованих тестів у процес НІ/НВ для неперервного тестування та зворотнього зв'язку.

• Моніторинг та підтримка: регулярне оновлення автоматизованих тестових скриптів для врахування змін у ПЗ. Механізми моніторингу та сповіщення для своєчасного виявлення проблем.

Слід враховувати виклики та аспекти при впровадженні та реалізації автоматизації тестування. Незважаючи на багато переваг, впровадження та реалізація автоматизації тестування стикаються з декількома викликами:

• Початкові витрати: інструменти, інфраструктура та навчання можуть бути дорогими, особливо для малих та середніх підприємств.

• Недостатність навичок: автоматизація потребує спеціалізованих навичок, що може бути проблемою для організацій у наборі та збереженні кваліфікованих інженерів.

• Витрати на підтримку: підтримка автоматизованих тестів потребує постійних зусиль для оновлення та адаптації до змін у тестованому ПЗ.

• Обмеження покриття тестування: Автоматизація може не охоплювати всі сценарії, особливо ті, що вимагають людського судження.

• Вибір і сумісність інструментів: Вибір інструментів залежить від вимог проекту та технологічного стеку, важлива сумісність з існуючою інфраструктурою.

Хоча автоматизація тестування надає значні переваги щодо ефективності, точності та надійності, організації повинні ретельно розглянути виклики та аспекти впровадження для максимізації її ефективності та повернення від інвестицій. Дотримуючись найкращих практик та вирішуючи виклики, організації можуть підвищити якість та ефективність своїх проєктів.

Тестування за типом “зсув праворуч”[13]та дослідницьке тестування[14] є інноваційними підходами, спрямованими на дослідження і підтвердження функціональності ПЗ в реальних середовищах. У цьому розділі досліджуються визначення та цілі цих підходів, переваги тестування у виробничих середовищах та стратегії їх включення до процесів забезпечення якості.

Тестування за типом “зсув праворуч” акцентується на тестуванні після випуску ПЗ. Це розширює тестувальні дії на післявипускову фазу, дозволяючи отримувати зворотний зв'язок в реальному часі та підтверджувати функціональність ПЗ в реальних умовах. Основна мета - виявлення та усунення проблем і дефектів, які виникають в середовищах використання.

Дослідницьке тестування акцентується на дослідженні, відкритті та навчанні під час виконання тестових сценаріїв. На відміну від сценарійних підходів, воно

ґрунтується на інтуїції, креативності та знаннях, щоб виявляти дефекти та оцінювати поведінку ПЗ. Мета дослідницького тестування — виявлення несподіваної поведінки, проблем з використанням та граничних випадків, що не включені в заздалегідь визначені сценарії, покращуючи обсяг та ефективність тестування.

Користь в тестуванні в середовищах використання та вивченні функціональності ПЗ

- Перевірка в реальних умовах: тестування дозволяє організаціям перевірити функціональність ПЗ в реальних умовах, надаючи уявлення про роботу ПЗ.
- Виявлення проблем на ранніх стадіях: “зсув праворуч” тестування виявляє та усуває проблеми, які з'являються лише в середовищах використання, як обмеження продуктивності та сумісність.
- Зворотний зв'язок від користувачів: тестування сприяє отриманню зворотного зв'язку від користувачів, що дозволяє організаціям зрозуміти їхню поведінку, вподобання та проблеми, враховуючи це у майбутніх ітераціях ПЗ.
- Постійне вдосконалення: дослідницьке тестування заохочує креативне мислення та виявлення несподіваних проблем, що призводить до покращення якості ПЗ.

Стратегії включення “зсув праворуч” тестування та дослідницького тестування в процеси забезпечення якості передбачають:

- Постійний моніторинг: впровадження механізмів моніторингу та сповіщення для відстеження продуктивності та поведінки ПЗ в реальних умовах. Використання інструментів для виявлення аномалій та проблем.
- A/B-тестування: використання A/B-тестування для порівняння продуктивності різних версій ПЗ. Отримання зворотного зв'язку від користувачів та аналіз метрик для оцінки змін.
- Зворотний зв'язок від користувачів: створення каналів для збору відгуків від користувачів, залучення зворотного зв'язку в процеси забезпечення якості для пріоритизації тестування.
- Міжфункціональна співпраця: співпраця між командами розробки, тестування, операцій та управління продуктом. Впровадження культури навчання та вдосконалення в організації.

Включаючи “зсув праворуч” та дослідницьке тестування в процеси якості, організації покращують валідацію функціональності ПЗ, отримують зворотний зв'язок від користувачів і постійно вдосконалюють якість та зручність ПЗ. Ці інноваційні підходи доповнюють традиційні методи тестування, дозволяючи створювати високоякісне ПЗ, яке відповідає потребам та очікуванням користувачів [15].

Методології DevOps та Agile змінили парадигму розробки ПЗ та тестування, акцентуючи на співпраці, гнучкості та постійному вдосконаленні. У цьому розділі надається огляд методологій DevOps та Agile та їх вплив на тестування ПЗ, розглядається колаборативний підхід до тестування в цих середовищах, а також інструменти та методики для інтеграції забезпечення якості в робочі процеси. Методології DevOps та Agile значно впливають на процеси тестування програмного забезпечення, змінюючи підходи до розробки, інтеграції та доставки продуктів[16]. Agile Методологія – ітеративний підхід до розробки ПЗ, що акцентує співпрацю, гнучкість та зворотний зв'язок від клієнта. Agile методології, такі як Scrum та Kanban, розбивають цикли на невеликі інкременти (спринти або ітерації), дозволяючи командам швидко постачати ПЗ та адаптуватися до змін. Agile тестування фокусується на постійному тестуванні з акцентом на співпраці. DevOps Методологія – культурний та організаційний рух, що акцентує співпрацю, автоматизацію та неперервне впровадження. DevOps руйнує бар'єри між командами розробки та операцій, дозволяючи швидше постачати ПЗ за допомогою автоматизації, НІ та НВ. DevOps тестування інтегрує тестувальні процеси у розробку та впровадження, дозволяючи

командам швидко постачати високоякісне ПЗ. Вплив методологійDevOps та Agile на тестування ПЗ включає акцент на співпрацю та комунікацію між командами. Підходи "зсув ліворуч" та “зсув праворуч” інтегрують тестування на початкових та кінцевих етапах розробки. Автоматизація тестувальних активностей, як-от модульне, інтеграційне та регресійне тестування, підтримує практики НІ та НВ. Методи тестування Agile, як-от розробка на основі тестування, поведінки та дослідницьке тестування, забезпечують всебічне покриття та постачання високоякісного ПЗ.

У середовищах DevOps та Agile тестування є спільним зусиллям розробників, тестувальників, інженерів з операцій та інших зацікавлених сторін. Основні принципи включають крос-функціональні команди для тісної співпраці, культуру спільної відповідальності за якість, постійний зворотний зв'язок через зустрічі та ретроспективи, а також інвестиції в інструменти автоматизації, які підтримують інтеграцію протягом усього життєвого циклу розробки та тестування, дозволяючи автоматизувати повторювані завдання. Деякі інструменти та методики, що підтримують інтеграцію контролю якості у робочі процеси DevOps та Agile, представлено на рис. 1.

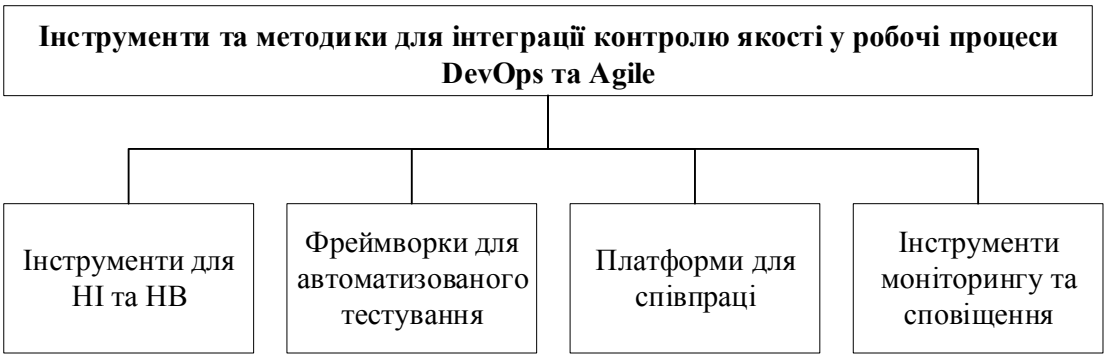

Рисунок 1 – Інструменти контролю якості

Джерело: розроблено авторами

Інструменти, такі як Jenkins, Travis та GitLab, автоматизують процеси збирання, тестування та розгортання ПЗ, дозволяючи командам швидко та надійно постачати ПЗ. Використання фреймворків, таких як Selenium, Cypress та JUnit, для автоматизації тестування та підтримки постійного тестування. Платформи співпраці, такі як Jira, Slack та Confluence, сприяють комунікації, координації та обміну знаннями. Інструменти моніторингу та сповіщення, такі як Prometheus, Grafana та NewRelic, відстежують продуктивність, виявляють аномалії та повідомляють про потенційні проблеми. Використання цих інструментів дозволяє інтегрувати контроль якості в процеси DevOps та Agile, забезпечуючи швидку та надійну доставку високоякісного ПЗ, сприяючи культурі співпраці, вдосконалення та інновацій.

Управління тестовими даними є важливим для точності, надійності та ефективності тестування ПЗ. Основні причини включають: ефективне управління даними, що забезпечує всебічне покриття тестами різних функцій програми. Високоякісні тестові дані забезпечують точність результатів, мінімізуючи хибні позитивні та негативні результати, і дозволяють виявляти справжні дефекти. Добре організовані дані прискорюють цикли тестування, зменшують час на підготовку та підтримку. Управління даними забезпечує цілісність та послідовність тестових середовищ, запобігаючи втратам даних та несанкціонованому доступу. Це допомагає організаціям дотримуватися нормативних вимог, таких як GDPR, HIPAA та PCI DSS, шляхом впровадження заходів конфіденційності та безпеки даних.

Техніки генерації, маскування та управління тестовими даними [17]включають синтетичну генерацію даних, де штучні набори даних створюються на основі

попередньо визначених правил, а також екстракцію даних з виробничих середовищ або зовнішніх джерел. Часто застосовуються техніки анонімізації та приховування для захисту конфіденційної інформації під час генерації. Маскування даних передбачає заміну конфіденційної або чутливої інформації в наборах тестових даних фіктивними або перемішаними значеннями зі збереженням структури та цілісності даних. Для ефективного маскування чутливих елементів даних використовуються техніки псевдонімізації, шифрування та токенізації. Підмножини даних передбачають видобуток підмножини важливих даних з виробничних баз даних для створення менших, репрезентативних наборів даних для цілей тестування. Техніки підмножини даних забезпечують копіювання або реплікацію лише необхідних даних в тестові середовища, що зменшує вимоги до сховища та покращує продуктивність. Інструменти профілювання даних аналізують та характеризують набори тестових даних для ідентифікації шаблонів, аномалій та проблем якості даних. Техніки виявлення даних допомагають тестувальникам ефективно знаходити та витягувати відповідні дані з розподілених або гетерогенних джерел даних. Практики керування версіями та життєвим циклом тестових даних відстежують зміни в наборах тестових даних з плином часу, що дозволяє тестувальникам повертатися до попередніх версій, перевіряти зміни в даних та ефективно керувати залежностями даних.

Аспекти, які забезпечують конфіденційність та безпеку даних в тестових середовищах [18] включають: використання маскування та шифрування для захисту чутливої інформації (особисті дані, фінансові дані, інтелектуальна власність) від несанкціонованого доступу. Застосування політики контролю доступу на основі ролей для обмеження доступу до даних залежно від ролей користувачів. Реалізація механізмів аутентифікації та авторизації для доступу до чутливих даних лише авторизованими користувачами. Встановлення політик управління даними для збору, зберігання, використання та видалення тестових даних відповідно до нормативних актів. Регулярні аудити та оцінки для дотримання політик конфіденційності та безпеки. Використання протоколів та алгоритмів шифрування для безпечної передачі даних між середовищами. Реалізація механізмів безпечного передавання файлів (SFTP, HTTPS) для захисту даних під час транспортування. Маскування даних для захисту в невиробничих середовищах (розробка, тестування, підготовка). Шляхом врахування цих аспектів та впровадження передових практик управління тестовими даними, організації можуть забезпечити точність, ефективність та безпеку своїх процесів тестування ПЗ, захищаючи важливу інформацію та відповідаючи вимогам законодавства.

Виклики та майбутні тенденціїу впровадженні інноваційних практик тестування часто супроводжується своєрідними труднощами [18]. Опір змінам від зацікавлених сторін і команд може ускладнити прийняття. Подолання опору вимагає ефективної комунікації, освіти та демонстрації цінності через пілотні проекти. Старі системи та застарілі інструменти можуть не легко приймати інновації. Модернізація систем, інвестиції в нові інструменти та поступова відмова від старих процесів допомагають подолати ці виклики. Отримання навичок у нових технологіях, як-от ШІ та автоматизація тестування, може бути викликом. Надання навчання, наставництва та професійного розвитку допомагає заповнити прогалини в навичках. Інтеграція нових інструментів з існуючими середовищами може викликати проблеми. Забезпечення сумісності та інтеграції вимагає планування та співпраці. Управління чутливими даними, забезпечення конфіденційності та відповідність вимогам регуляторів, таких як GDPR та HIPAA, становить виклик. Впровадження маскування, шифрування та контролю доступу є ключовим для захисту інформації.

Нові тенденції та технології, що формують майбутнє тестування ПЗ представлено на рис.2.

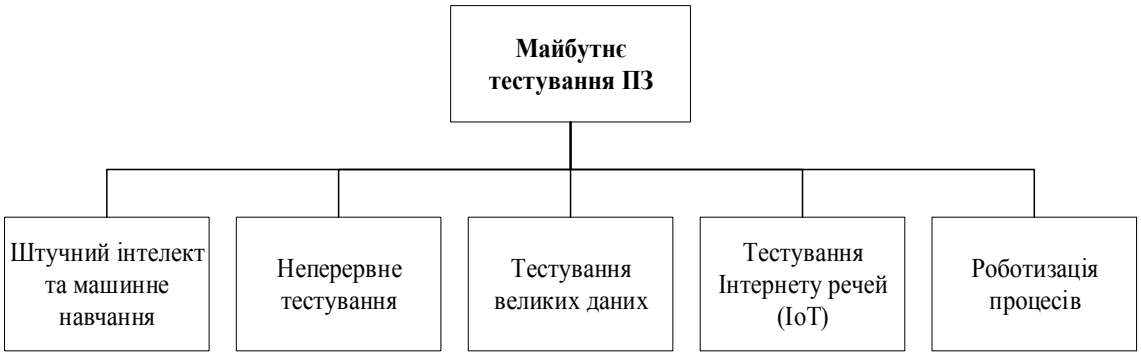

Рисунок 2 – Подальші напрямки розвитку тестування

Джерело: розроблено авторами

Кілька нових тенденцій та технологій перетворюють простір тестування ПЗ.ШІ та МН революціонізують тестування ПЗ, автоматизуючи генерацію тестових випадків, передбачаючи дефекти та оптимізуючи процеси тестування. Інструменти тестування, які працюють на базі ШІ, можуть аналізувати величезні обсяги тестових даних, виявляти закономірності та надавати корисні висновки тестувальникам.

Практики НТ, такі як тестування на початкових етапах, тестування на післяринкових етапах та HI/HB, набирають популярності в Agileta DevOps середовищах. НТ дозволяє командам швидко перевіряти зміни, зменшувати ризики та забезпечувати якість ПЗ протягом усього життєвого циклу розробки.

З поширенням додатків та платформ великих даних тестування продуктивності, масштабованості та надійності систем великих даних стає новими викликами для тестувальників. Інструменти та методики тестування великих даних, такі як валідація даних, тестування продуктивності та розподілене тестування, є ключовими для забезпечення якості рішень в галузі великих даних.

Інтернет речей (IoT) створює унікальні виклики для тестування через складність, різноманіття та взаємопов'язаність пристроїв та екосистем IoT. Тестування IoT зосереджене на підтвердженні взаємодії пристроїв, безпеці, надійності та продуктивності в різних середовищах та випадках використання.

Технології роботизації процесів автоматизують повторювані завдання на основі правил, таких як регресійне тестування, введення даних та тестування графічного інтерфейсу користувача, звільняючи тестувальників від нудних завдань та дозволяючи їм зосередитися на складніших та творчих тестувальних діях. Інструменти технології роботизації можуть оптимізувати робочі процеси тестування, покращувати ефективність та зменшувати ручні зусилля.

Щоб подолати виклики та виходити вперед у галузі тестування ПЗ, організації можуть прийняти інноваційні стратегії подолання викликів та виходу вперед у галузі комп'ютерних наук.

Заохочення культури постійного навчання, експериментів та адаптації. Інвестиції в навчальні програми, сертифікаційні курси та обмін знаннями для підвищення кваліфікації співробітників. Сприяння співпраці між розробниками, тестувальниками, операторами та іншими зацікавленими сторонами. Створення крос-функціональних команд для вирішення проблем та обміну експертизами. Аналіз пілотних проектів та доказів концепцій для оцінки нових інструментів і технологій. Збирання відгуків, вдосконалення на основі отриманих уроків та масштабування успішних ініціатив. Використання гнучкого підходу до тестування, регулярний збір відгуків та постійне вдосконалення процесів. Розвиток експериментів та інновацій для забезпечення постійного вдосконалення. Першочергово враховувати потреби та

зворотний зв'язок клієнтів, інтегрувати його у процеси тестування для забезпечення відповідності ПЗ потребам користувачів.

Шляхом прийняття новітніх тенденцій, подолання викликів та впровадження ефективних стратегій, організації можуть розмістити себе таким чином, щоб процвітати у все більш динамічному та конкурентному середовищі, надаючи високоякісне ПЗ, яке відповідає зростаючим потребам та очікуванням клієнтів.

Висновки. Прийняття інновацій у тестуванні ПЗ є невід'ємною складовою для підвищення якості, ефективності та конкурентоспроможності в сучасному швидкозмінному та динамічному цифровому середовищі. Шляхом прийняття інноваційних практик тестування, організації можуть:

- Підвищення точності та надійності тестів: інноваційні практики тестування дозволяють організаціям виявляти дефекти на ранніх етапах, покращувати покриття тестування та постачати високоякісне ПЗ, що відповідає потребам і очікуванням користувачів.
- Прискорення виходу на ринок: НТ, автоматизація та гнучкі методології оптимізують процеси тестування, прискорюють вихід на ринок та дозволяють організаціям швидко реагувати на змінні вимоги клієнтів та ринкові тенденції.
- Зменшення витрат та ризиків: інноваційні практики тестування допомагають організаціям зменшити витрати на тестування, мінімізувати ризики та зменшити вплив дефектів та відмов на бізнес-операції, задоволеність клієнтів та репутацію бренду.
- Стимулювання постійного вдосконалення та інновацій: прийняття інновацій у тестуванні ПЗ сприяє створенню культури постійного вдосконалення, експериментування та інновацій всередині організацій. За допомогою постійного дослідження нових інструментів, технологій та методологій, організації можуть залишатися на передньому краї індустрії та стимулювати інновації в тестуванні та розробці ПЗ.

У статті досліджено важливість тестування програмного забезпечення в контексті інженерії програмного забезпечення, підкреслюючи ключову роль, яку відіграє тестування у визначенні надійності, функціональності та безпеки програмних систем. Проаналізовано сучасні методології, інструменти та техніки тестування, щоб висвітлити найновіші досягнення в цій області.

Тестування програмного забезпечення включає в себе різні дії, спрямовані на оцінку якості та продуктивності продуктів програмного забезпечення. Це включає систематичне виконання тестових випробувань, сценаріїв та скриптів для виявлення дефектів, перевірки функціональності та підтвердження відповідності специфікаціям.

Мета статті - надати глибоке розуміння змін у галузі тестування програмного забезпечення та їх впливу на якість та надійність програм. Ця мета досягалась шляхом узагальнення результатів академічних досліджень, розгляду тенденцій в галузі та оцінки думок експертів. Стаття має наукову значущість, оскільки вона надає цінний внесок у розуміння сучасних тенденцій та методологій в галузі тестування програмного забезпечення, що може слугувати основою для подальших досліджень та розробок.

Підсумовуючи, прийняття інновацій у тестуванні ПЗ є невід'ємною складовою для організацій, які прагнуть постачати високоякісне ПЗ ефективно та конкурентоспроможно в сучасній цифровій економіці. Шляхом прийняття новітніх тенденцій, подолання викликів та прийняття орієнтованого на клієнта підходу до тестування, організації можуть позиціонувати себе для успіху вусе складнішому та конкурентному ринковому середовищі.

Через постійне навчання, співпрацю та адаптацію, організації можуть використовувати потужність інновацій для підвищення якості, ефективності та інновацій в тестуванні ПЗ, в кінцевому підсумку постачаючи передові продукти та враження, які приносять задоволення клієнтам та забезпечують зростання бізнесу.

Список літератури

1 Donald Firesmith, «Four Types of Shift Left Testing,» Carnegie Mellon University, 15 March 2015. URL: https://insights.sei.cmu.edu/blog/four-types-of-shift-left-testing/. (дата звернення: 8 April 2024).

2 Pat Phelan та Renee Wells, «Adopting a "Shift Left" Strategy to Transform Technology Support Services,» 2020. URL: https://www.riministreet.com/wp-content/uploads/2020/07/Rimini-Street-Shift-Left-Strategy-Transforms-Tech-Support-Services-White-Paper.pdf. (дата звернення: 2 April 2024).

3 Maximiliano A. Mascheroni та Emanuel Irrazábal, «Continuous Testing and Solutions for Testing Problems in Continuous Delivery: A Systematic Literature Review,» September 2018. URL: https://www.researchgate.net/publication/340396275_Continuous_Testing_and_Solutions_for_Testing_Problems_in_Continuous_Delivery_A_Systematic_Literature_Review. (дата звернення: 7 April 2024).

4 Altay Ataman, «Continuous Testing in 2024: Top 7 Benefits & Challenges,» January 2024. URL: https://research.aimultiple.com/continuous-testing/. (дата звернення: 5 April 2024).

5 Yogesh Solanki, «Top 21 API Automation Testing Tools that Make Automation Easy,» TestGrid, 11 March 2022. URL: https://testgrid.io/blog/api-automation-testing-tools/. (дата звернення: 5 April 2024).

6 Nico Krüger, «AI Testing and Machine Learning in Software Testing,» 24 August 2020. URL: https://www.perforce.com/blog/alm/ai-testing-and-machine-learning-software-testing. (дата звернення: 9 April 2024).

7 Gradient Ascent, «The Role of AI in Software Testing: Streamlining Processes and Reducing Errors,» 23 January 2024. URL: https://gradient-ascent.com/ai-in-software-testing/. (Дата звернення: 9 April 2024).

8 Qian Cheng, Amrita Saha, Wenzhuo Yang, Chenghao Liu, Doyen Sahoo та Steven Hoi, «LOGAI: A LIBRARY FOR LOG ANALYTICS AND INTELLIGENCE,» 31 January 2023. URL: https://arxiv.org/pdf/2301.13415.pdf. (дата звернення: 9 April 2024).

9 Abdullah A. Abonamah та Neda Abdelhamid, «Managerial insights for AI/ML implementation: a playbook for successful organizational integration,» 12 March 2024. URL: https://link.springer.com/article/10.1007/s44163-023-00100-5. (дата звернення: 7 April 2024).

10 Enrique DeCoss, «The Evolution of Automation Testing,» 5 March 2023. URL: https://www.accelq.com/blog/testing-evolution/. (дата звернення: 11 April 2024).

11 Leonardo Mariani, Dan Hao, Rajesh Subramanyan та Hong Zhu, «The central role of test automation in software quality assurance,» 10 July 2017. URL: https://link.springer.com/article/10.1007/s11219-017-9383-5. (дата звернення: 11 April 2024).

12 Tahanima Chowdhury, «Test Automation Best Practices For Better Testing In 2024,» 28 March 2024. URL: https://saucelabs.com/resources/blog/test-automation-best-practices-2024. (дата звернення: 11 April 2024).

13 Piyusha Podutwar, «Shift Right Testing: Know its Benefits, Types, and Tools,» 14 November 2023. URL: https://www.lambdatest.com/learning-hub/shift-right-testing. (ата звернення: 15 April 2024).

14 Ahmad Nauman Ghazi, Kai Petersen, Elizabeth Bjarnason та Per Runeson, «Exploratory Testing: One Size Doesn’t Fit All,» URL: https://arxiv.org/ftp/arxiv/papers/1704/1704.00537.pdf. (дата звернення: 15 April 2024).

15 Parvathy Purushothaman, «Beyond Deployment: Unveiling the Dynamics of ShiftRight Testing,» 19 February 2024. URL: https://ijcttjournal.org/2024/Volume-72%20Issue-2/IJCTT-V72I2P104.pdf. (дата звернення: 15 April 2024).

16 Thomas Hamilton, «Agile Vs. DevOps – Difference Between Them,» 17 February 2024. URL: https://www.guru99.com/agile-vs-devops.html. (дата звернення: 15 April 2024).

17 Shaishav Desai, «Protecting Data Integrity and Privacy in Testing,» 1 September 2023. URL: https://www.c-sharpcorner.com/article/protecting-data-integrity-and-privacy-in-testing/. (дата звернення: 17 April 2024).

18 Yash Bansal, «23 Software Testing Trends To Look Out For In 2024,» 28 February 2024. URL: https://www.lambdatest.com/blog/software-testing-trends/.(дата звернення: 17 April 2024).

References

1. Donald Firesmith, «Four Types of Shift Left Testing,» Carnegie Mellon University, 15 March 2015. Retrieved from https://insights.sei.cmu.edu/blog/four-types-of-shift-left-testing/ [in English].

2 Pat Phelan та Renee Wells, «Adopting a "Shift Left" Strategy to Transform Technology Support Services,» 2020. Retrieved from https://www.riministreet.com/wp-content/uploads/2020/07/Rimini-Street-Shift-Left-Strategy-Transforms-Tech-Support-Services-White-Paper.pdf [in English].

3 Maximiliano A. Mascheroni та Emanuel Irrazábal, «Continuous Testing and Solutions for Testing Problems in Continuous Delivery: A Systematic Literature Review,» September 2018. Retrieved from

https://www.researchgate.net/publication/340396275_Continuous_Testing_and_Solutions_for_Testing_Problems_in_Continuous_Delivery_A_Systematic_Literature_Review [in English].
4 Altay Ataman, «Continuous Testing in 2024: Top 7 Benefits & Challenges,» January 2024. Retrieved from https://research.aimultiple.com/continuous-testing/. (дата звернення: 5 April 2024).
5 Yogesh Solanki, «Top 21 API Automation Testing Tools that Make Automation Easy,» TestGrid, 11 March 2022. Retrieved from https://testgrid.io/blog/api-automation-testing-tools/ [in English].
6 Nico Krüger, «AI Testing and Machine Learning in Software Testing,» 24 August 2020. Retrieved from https://www.perforce.com/blog/alm/ai-testing-and-machine-learning-software-testing [in English]..
7 Gradient Ascent, «The Role of AI in Software Testing: Streamlining Processes and Reducing Errors,» 23 January 2024. Retrieved from https://gradient-ascent.com/ai-in-software-testing/ [in English].
8 Qian Cheng, Amrita Saha, Wenzhuo Yang, Chenghao Liu, Doyen Sahoo та Steven Hoi, «LOGAI: A LIBRARY FOR LOG ANALYTICS AND INTELLIGENCE,» 31 January 2023. Retrieved from https://arxiv.org/pdf/2301.13415.pdf [in English].
9 Abdullah A. Abonamah та Neda Abdelhamid, «Managerial insights for AI/ML implementation: a playbook for successful organizational integration,» 12 March 2024. Retrieved from https://link.springer.com/article/10.1007/s44163-023-00100-5 [in English].
10 Enrique DeCoss, «The Evolution of Automation Testing,» 5 March 2023. Retrieved from https://www.accelq.com/blog/testing-evolution/ [in English].
11 Leonardo Mariani, Dan Hao, Rajesh Subramanyan та Hong Zhu, «The central role of test automation in software quality assurance,» 10 July 2017. Retrieved from https://link.springer.com/article/10.1007/s11219-017-9383-5 [in English].
12 Tahanima Chowdhury, «Test Automation Best Practices For Better Testing In 2024,» 28 March 2024. Retrieved from https://saucelabs.com/resources/blog/test-automation-best-practices-2024 [in English].
13 Piyusha Podutwar, «Shift Right Testing: Know its Benefits, Types, and Tools,» 14 November 2023. Retrieved from https://www.lambdatest.com/learning-hub/shift-right-testing [in English].
14 Ahmad Nauman Ghazi, Kai Petersen, Elizabeth Bjarnason та Per Runeson, «Exploratory Testing: One Size Doesn’t Fit All,» Retrieved from https://arxiv.org/ftp/arxiv/papers/1704/1704.00537.pdf [in English].
15 Parvathy Purushothaman, «Beyond Deployment: Unveiling the Dynamics of ShiftRight Testing,» 19 February 2024. Retrieved from https://ijcttjournal.org/2024/Volume-72%20Issue-2/IJCTT-V72I2P104.pdf [in English].
16 Thomas Hamilton, «Agile Vs. DevOps – Difference Between Them,» 17 February 2024. Retrieved from https://www.guru99.com/agile-vs-devops.html [in English].
17 Shaishav Desai, «Protecting Data Integrity and Privacy in Testing,» 1 September 2023. Retrieved from https://www.c-sharpcorner.com/article/protecting-data-integrity-and-privacy-in-testing/ [in English].
18 Yash Bansal, «23 Software Testing Trends To Look Out For In 2024,» 28 February 2024. Retrieved from https://www.lambdatest.com/blog/software-testing-trends/ [in English].

Oleksandr Ulichev, Ph.D tech. sci.
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine
Lyubomyr Papizh, post-graduate
Private Higher Education Establishment "European University", Kyiv, Ukraine
Oleksandr Revniuk, post-graduate
Ivan Pulyuy Ternopil National Technical University, Ternopil, Ukraine

Advancements in Software Testing: A Scientific Perspective

The article aims to explore various aspects of innovation in software testing, including the latest trends, challenges, and strategies for advancing in the ever-evolving field of software development. It emphasizes the importance of adopting innovative testing practices to enhance the accuracy, efficiency, and reliability of tests. Additionally, the article seeks to provide insights into how organizations can integrate these innovative practices into their existing workflows. By doing so, it aims to help organizations stay competitive and meet the growing demands of the digital market.

The article delves into different software testing activities, such as test trials, scenarios, and scripts designed to detect defects, verify functionality, and ensure compliance with specifications. It highlights the significance of thorough testing in identifying and addressing issues early, thereby reducing the likelihood of errors in production environments. The shift-left testing approach is examined in detail, advocating for early testing such as unit testing, static code analysis, and code reviews. This method promotes the principle of "test early, test often," enabling organizations to detect and fix defects more cost-effectively and swiftly. The article also discusses the challenges in implementing innovative testing practices, such as resistance to change, outdated systems, skill gaps, and data privacy issues. It presents new trends and technologies shaping the future of

software testing, including AI and ML, big data testing, IoT testing, and robotic process automation. Strategies for overcoming challenges and staying ahead of competitors, such as continuous learning, collaboration, pilot projects, agile methodologies, and customer focus, are also explored.

The article concludes that adopting innovative testing practices is essential for improving the quality, efficiency, and competitiveness of software in today's fast-paced and dynamic digital environment. Early defect detection is fundamental to the shift-left strategy, offering substantial benefits to organizations. Identifying and fixing defects early significantly reduces costs, accelerates development cycles, and shortens time-to-market, allowing for quicker responses to market demands. Early defect detection enhances product quality and reliability, leading to a positive user experience, increased customer satisfaction, and loyalty. Organizations that proactively address defects gain customer trust and strengthen their market reputation. Continuous testing is emphasized as a crucial component in modern software development practices, ensuring quality and reliability throughout the software lifecycle.

Software testing, Shift-Left Testing, Continuous Testing, AI and ML, Test Automation, Shift-Right Testing, Test Data Management

Одержано (Received) 16.09.2024 *Прорецензовано (Reviewed) 12.10.2024*
Прийнято до друку (Approved) 28.10.2024

УДК 004.4 DOI: https://doi.org/10.32515/2664-262X.2024.10(41).16-29

О.С. Улічев, канд. техн. наук, **О.П. Доренський,** доц., канд. техн. наук
Центральноукраїнський національний технічний університет м. Кропивницький, Україна
В.П. Кулагін, асп.
Приватний вищий навчальний заклад "Європейський університет", м. Київ, Україна
e-mail: askin79@gmail.com, dorensky@ukr.net, victor@kulagin.com.ua

Інноваційні рішення та переваги мікросервісної архітектури програмних продуктів

У цьому дослідженні розглядається мікросервісна архітектура, як сучасний підхід до розробки програмного забезпечення, який відповідає зростаючим вимогам бізнесу та технологій. Проведено порівняльний аналіз мікросервісної архітектури з іншими архітектурними стилями, такими як монолітна та сервіс-орієнтована архітектури. Розглянуто приклади впровадження мікросервісів в індустрії на прикладі технологічних гігантів. Висвітлено використання інструментів та технологій, зокрема контейнеризації та оркестрації. Проаналізовано стандарти та найкращі практики, що сприяють ефективному впровадженню мікросервісів. Результати дослідження сприятимуть глибшому розумінню мікросервісної архітектури.

мікросервісна архітектура, масштабованість, DevOps, контейнеризація, оркестрація, розподілені системи

Постановка проблеми. У сучасному світі інформаційних технологій традиційні монолітні архітектури програмного забезпечення стають перешкодою для швидкого розвитку та масштабування систем, що не відповідає зростаючим вимогам бізнесу та користувачів. Мікросервісна архітектура виникла як рішення цих проблем, пропонуючи розподіл додатка на незалежні сервіси, але її впровадження пов'язане з новими викликами, такими як складність управління розподіленими системами.

Аналіз останніх досліджень і публікацій. Мартін Фаулер та Джеймс Льюїс у своїй статті "Microservices" [1] формалізували концепцію мікросервісної архітектури. Вони підкреслили переваги цього підходу, зокрема підвищену гнучкість, масштабованість