https://doi.org/10.31891/2219-9365-2023-75-8
УДК 004

КРАВЧУК Ольга
Хмельницький національний університет
https://orcid.org/0000-0001-6937-5001
e-mail: kravchukoa2@gmail.com

МОДЕЛЬ ВПРОВАДЖЕННЯ CI/CD ДЛЯ ОПТИМІЗАЦІЇ УПРАВЛІННЯ IT-ПРОЄКТАМИ

В статті пропонується метод впровадження CI/CD (Continuous Integration/Continuous Deployment) для оптимізації управління IT-проєктами. Вона розглядає принципи та практики DevOps, що стали важливими інструментами для підвищення ефективності розробки та впровадження програмного забезпечення. В статті аналізується як CI/CD може покращити процеси розробки, тестування та впровадження програмного забезпечення, зменшуючи час впровадження та підвищуючи якість продукту. Також розглядаються потенційні виклики та обмеження, пов'язані з впровадженням CI/CD, та пропонуються стратегії їх подолання.

Ключові слова: CI/CD, управління IT-проєктами, оптимізація, впровадження, програмне забезпечення, розробка, тестування, виклики, стратегії.

KRAVCHUK Olga
Khmelnytskyi National University

CI/CD IMPLEMENTATION MODEL FOR OPTIMIZING IT PROJECT MANAGEMENT

The article proposes a CI/CD (Continuous Integration/Continuous Deployment) implementation method for optimizing IT project management. It examines the principles and practices of DevOps, which have become important tools for improving the efficiency of software development and implementation. The author analyzes how CI/CD can improve software development, testing and implementation processes, reducing implementation time and increasing product quality. The paper also examines potential challenges and limitations associated with CI/CD implementation and suggests strategies to overcome them.

Keywords: CI/CD, IT project management, optimization, implementation, software, development, testing, challenges, strategies.

1. Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

CI/CD, що розшифровується як Continuous Integration (Безперервна інтеграція) та Continuous Deployment (Безперервне розгортання), є ключовими концепціями в сучасних практиках розробки програмного забезпечення [1]. Ці поняття виникли з методології Agile та були далі розвинені в рамках DevOps, що спрямовані на підвищення швидкості, ефективності та якості процесів розробки та впровадження програмного забезпечення [3]. Continuous Integration вимагає від команд розробників регулярно (часто кілька разів на день) інтегрувати свою роботу в спільний репозиторій. Після кожної інтеграції автоматизовані сценарії збудовують програмне забезпечення та проводять ряд тестів, щоб виявити та виправити помилки якомога швидше [4]. Continuous Deployment відноситься до автоматичного розгортання програмного забезпечення в продуктивне середовище після проходження встановлених тестів та перевірок. Це забезпечує швидке впровадження оновлень та нових функцій, а також зменшує ризики, пов'язані з ручним розгортанням [5]. Впровадження CI/CD в управління IT-проєктами може призвести до значних поліпшень в термінах швидкості впровадження, якості продукту, продуктивності команди та задоволення клієнтів.

Незважаючи на численні переваги, впровадження CI/CD може стикнутися з рядом викликів, які потребують уваги та вирішення.

По-перше, це культурні зміни**.** Впровадження CI/CD вимагає значних культурних змін в організації. Командам потрібно адаптуватися до нового ритму роботи, який вимагає регулярної інтеграції та розгортання. Це може вимагати навчання та підтримки для забезпечення гладкого переходу [6].

По-друге, технічні виклики. Впровадження CI/CD вимагає впровадження нових інструментів та технологій, які можуть бути складними. Це може включати в себе вибір правильних інструментів, налаштування автоматизованих тестів та розгортання, а також управління інфраструктурою [7].

По-третє, безпека та відповідність. З безперервним розгортанням може виникнути проблема забезпечення безпеки та відповідності стандартам. Організації повинні забезпечити, що їх процеси CI/CD відповідають всім вимогам щодо безпеки та регулювання [8].

По-четверте, вимірювання та моніторин. Для успішного впровадження CI/CD необхідно встановити відповідні метрики та моніторинг для відстеження прогресу та виявлення проблем на ранніх стадіях [9].

По-п'яте, складність управління**.** З ростом проєкту, кількість інтеграцій та розгортань може значно

збільшитися, що може ускладнити управління процесом CI/CD [10].

Враховуючи ці виклики, важливо розробити детальний план впровадження та управління CI/CD. Цей план повинен включати стратегії для культурних змін, вибору та впровадження технологій, забезпечення безпеки та відповідності, а також вимірювання та моніторингу. Крім того, організації повинні розробити стратегії для управління складністю, що виникає з ростом проєкту та збільшенням кількості інтеграцій та розгортань.

Впровадження CI/CD - це неодноразовий процес, а постійний цикл покращення, який вимагає постійного моніторингу, оцінки та налаштування. Важливо, щоб команди були готові до постійної адаптації та навчання в процесі впровадження CI/CD.

Ціль цієї статті - вивчити та розробити метод впровадження CI/CD для оптимізації управління IT-проєктами. Ми зосереджуємося на визначенні критичних аспектів впровадження CI/CD, включаючи вибір відповідних інструментів, налаштування процесів та забезпечення високої якості коду.

Основним науковим результатом цієї статті є розробка методу, який дозволяє організаціям ефективно впроваджувати CI/CD в свої процеси управління IT-проєктами. Цей метод включає в себе кроки для вибору відповідних інструментів, налаштування процесів CI/CD, а також рекомендації щодо забезпечення якості коду та безперебійної роботи системи.

Наукова новизна статті полягає в тому, що вперше було розроблено концептуальну модель впровадження CI/CD, яка охоплює всі ключові аспекти процесу, включаючи інтеграцію коду, автоматичне тестування, розгортання, моніторинг та зворотний зв'язок, а також постійне вдосконалення. Також отримало подальший розвиток дослідження в області впровадження CI/CD в управління IT-проєктами, зокрема, було виявлено, що вибір відповідних інструментів та налаштування процесів CI/CD є критичними для успішного впровадження цієї методології. Вперше було проведено детальний аналіз впливу впровадження CI/CD на ефективність управління IT-проєктами, що дозволило виявити ключові фактори, які сприяють підвищенню продуктивності, скороченню часу виведення продукту на ринок та підвищенню якості продуктів.

2. Розробка нової моделі впровадження CI/CD

Існуючі методи впровадження CI/CD включають в себе різноманітні підходи, які використовуються в індустрії програмного забезпечення. Ці методи включають в себе такі підходи, як використання автоматизованих систем збірки та тестування, використання контейнерів для ізоляції та управління програмним забезпеченням, використання систем контролю версій для відслідковування змін у коді, та інші [1, 3, 5]. Однак, хоча ці методи можуть бути ефективними в певних контекстах, вони часто виявляються недостатніми для вирішення викликів, пов'язаних з впровадженням CI/CD в управлінні IT-проєктами. Наприклад, автоматизовані системи збірки та тестування можуть бути складними та часозатратними для налаштування та управління, а системи контролю версій можуть бути складними для використання без відповідного навчання та досвіду [7, 9, 11-13]. Тому, для вирішення цих викликів, важливо розробити новий метод впровадження CI/CD, який би був більш гнучким, легким у використанні та ефективним у вирішенні викликів, пов'язаних з управлінням IT-проєктами.

2.1 Опис процесу розробки нової моделі. Процес розробки нової моделі впровадження CI/CD включає в себе кілька ключових етапів, які спрямовані на вирішення викликів, пов'язаних з впровадженням CI/CD в управлінні IT-проєктами.

Аналіз поточного стану. На цьому етапі проводиться аналіз поточного стану процесів розробки та впровадження програмного забезпечення. Це включає в себе оцінку існуючих практик, інструментів та процесів, а також визначення проблем та викликів, які виникають під час впровадження CI/CD.

Розробка концептуальної моделі. На основі результатів аналізу розробляється концептуальна модель нового процесу впровадження CI/CD. Ця модель включає в себе визначення ключових елементів процесу, їх взаємозв'язків та взаємодії.

Верифікація та валідація моделі. Після розробки концептуальної моделі проводиться її верифікація та валідація. Це включає в себе перевірку моделі на внутрішню консистентність та її порівняння з реальними процесами впровадження CI/CD.

Удосконалення та оптимізація моделі. На основі результатів верифікації та валідації моделі проводиться її удосконалення та оптимізація. Це може включати в себе зміну окремих елементів моделі, їх взаємозв'язків або параметрів.

Впровадження моделі. Після удосконалення та оптимізації моделі проводиться її впровадження на практиці.

Розглянемо детальніше концептуальну модель впровадження CI/CD, яка була розроблена. Модель складається з наступних ключових елементів: інтеграція коду, автоматичне тестування, розгортання, моніторинг та зворотний зв'язок, а також постійне вдосконалення. Проаналізуємо кожний з цих етапів.

Інтеграція коду. Цей етап включає в себе злиття коду, написаного різними розробниками, в одну центральну систему керування версіями (наприклад, Git). Це допомагає забезпечити консистентність коду та виявити проблеми на ранніх стадіях.

Автоматичне тестування**.** Після інтеграції коду автоматично запускаються тести для перевірки його якості та функціональності. Це може включати юніт-тестування, інтеграційне тестування, тестування продуктивності та ін.

Розгортання. Цей етап включає в себе автоматичне розгортання програмного забезпечення в виробниче середовище. Це може бути реалізовано за допомогою таких інструментів, як Kubernetes або Docker.

Моніторинг та зворотний зв'язок. Після розгортання програмного забезпечення важливо забезпечити його постійний моніторинг для виявлення та вирішення проблем якомога швидше. Зворотний зв'язок від користувачів та систем моніторингу також використовується для постійного вдосконалення процесу.

Постійне вдосконалення. На основі зворотнього зв'язку та результатів моніторингу проводиться постійне вдосконалення процесу впровадження CI/CD. Це може включати в себе внесення змін до коду, процесів або інструментів.

2.1 Аналіз поточного стану.

Аналіз поточного стану є важливим етапом при впровадженні CI/CD. Цей процес включає оцінку поточних процесів розробки, тестування та розгортання програмного забезпечення, а також визначення областей, які можуть бути оптимізовані за допомогою CI/CD.

При аналізі поточного стану важливо врахувати наступні аспекти: процеси розробки (які методології розробки використовуються в даний час? Які інструменти використовуються для написання коду, версійного контролю та інших аспектів розробки?), процеси тестування (як виконується тестування? Чи використовуються автоматизовані тестувальні інструменти? Які процеси використовуються для забезпечення якості коду?) , процеси розгортання (як відбувається процес розгортання? Чи використовуються інструменти для автоматизації розгортання? Які процеси використовуються для забезпечення безперебійної роботи системи?) та культура команди (чи готова команда до впровадження CI/CD? Чи є у команди досвід роботи з цією методологією? Чи є у команди необхідні навички для впровадження CI/CD?). Після проведення аналізу поточного стану, можна визначити, які області можуть бути оптимізовані за допомогою CI/CD, та розробити стратегію впровадження цієї методології.

2.2 Розробка концептуальної моделі

Концептуальна модель впровадження CI/CD, яку ми розробляємо, базується на декількох ключових принципах. Перш за все, вона передбачає використання автоматизованих процесів для розробки, тестування та розгортання програмного забезпечення. Це означає, що всі ці процеси виконуються автоматично, без необхідності вручну запускати кожен з них. Це дозволяє зменшити час, необхідний для виконання цих процесів, і забезпечує більшу надійність результатів.

Другим ключовим принципом даної моделі є використання інструментів для контролю версій, таких як Git, для відстеження змін у коді та забезпечення його цілісності. Це дозволяє розробникам легко вносити зміни в код і відновлювати попередні версії, якщо це необхідно.

Третім принципом є використання стандартів якості коду та процесів перевірки якості для забезпечення високої якості програмного забезпечення. Це включає використання автоматизованих інструментів для перевірки якості коду, таких як SonarQube, а також проведення регулярних оглядів коду.

Наша модель також передбачає використання стратегій розгортання, таких як blue-green або canary, для забезпечення безперебійного розгортання нових версій програмного забезпечення. Це дозволяє впроваджувати нові функції та виправлення помилок без перерви в роботі системи.

Нарешті, дана модель передбачає тісну інтеграцію між розробниками, тестувальниками та іншими учасниками команди. Це означає, що всі учасники команди працюють разом протягом всього циклу розробки, що сприяє більш ефективному вирішенню проблем та впровадженню нових функцій.

Ця модель є гнучкою і може бути адаптована до конкретних потреб будь-якої організації. Вона передбачає використання найкращих практик CI/CD та враховує специфіку роботи IT-проєктів. Ми вважаємо, що впровадження цієї моделі може значно підвищити ефективність управління IT-проєктами та забезпечити високу якість розробленого програмного забезпечення.

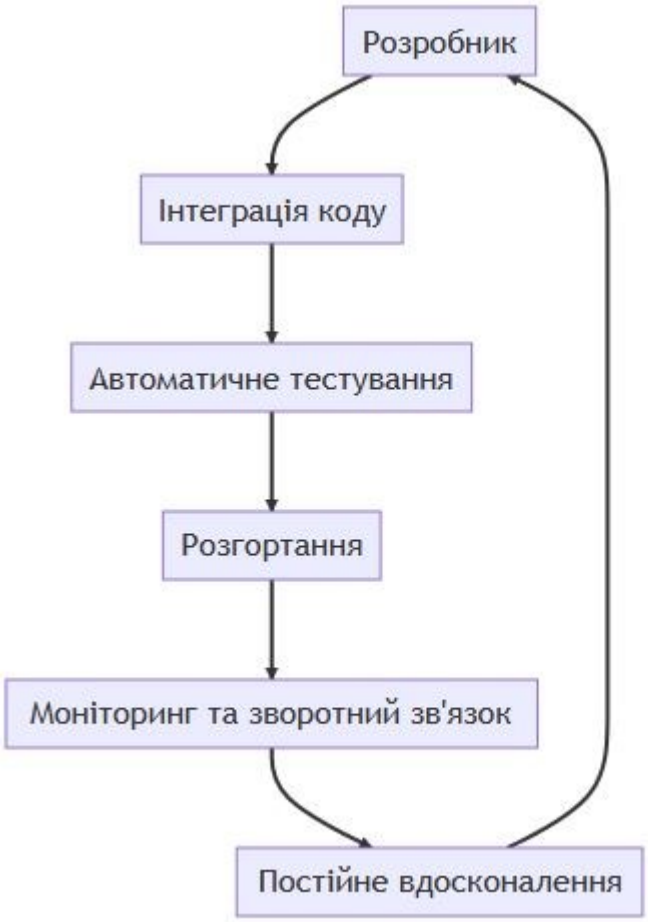

Рис. 1. Діаграма потоку роботи CI/CD

2.3 Верифікація та валідація моделі

Після розробки концептуальної моделі впровадження CI/CD, наступним кроком є її верифікація та валідація. Цей процес включає перевірку моделі на відповідність вимогам та перевірку її ефективності в реальних умовах.

Верифікація моделі полягає в перевірці того, що модель відповідає вимогам, встановленим на початковому етапі. Це включає перевірку того, що модель включає всі необхідні компоненти CI/CD, такі як автоматизовані процеси розробки, тестування та розгортання, контроль версій, стандарти якості коду та стратегії розгортання. Ми також перевіряємо, що модель враховує специфіку роботи IT-проєктів та включає механізми для тісної інтеграції між розробниками, тестувальниками та іншими учасниками команди.

Валідація моделі полягає в перевірці її ефективності в реальних умовах. Для цього ми впроваджуємо модель в одному або декількох IT-проєктах і спостерігаємо за її впливом на ефективність управління проєктами. Ми аналізуємо, як модель впливає на швидкість розробки, якість програмного забезпечення, частоту виправлення помилок та інші ключові показники. Ми також збираємо відгуки від учасників команди про їх досвід використання моделі.

На основі результатів верифікації та валідації ми можемо внести корективи в модель, якщо це необхідно, та визначити, наскільки вона ефективна для впровадження CI/CD в управління IT-проєктами. Ми вважаємо, що цей процес є важливим кроком на шляху до розробки ефективної моделі впровадження CI/CD.

2.4 Удосконалення та оптимізація моделі

Після верифікації та валідації моделі впровадження CI/CD, наступним кроком є її удосконалення та оптимізація. Цей процес включає внесення змін до моделі на основі отриманих результатів та відгуків від команди.

Удосконалення моделі може включати в себе внесення змін до процесів CI/CD, вибору інструментів, стратегій розгортання та інших аспектів моделі. Це може бути зумовлено потребою вирішити виявлені проблеми, покращити ефективність процесів або краще відповідати потребам команди та проєкту. Оптимізація моделі включає в себе пошук способів покращення її ефективності та продуктивності. Це може включати в себе впровадження нових технологій або практик, зміну ролей та відповідальностей в команді,

або внесення змін до процесів розробки, тестування та розгортання. У процесі удосконалення та оптимізації моделі важливо зберігати баланс між потребами команди, вимогами проєкту та цілями організації. Це допоможе забезпечити, що модель впровадження CI/CD буде ефективною та підтримуватиме високу якість програмного забезпечення та безперебійну роботу системи. В кінцевому підсумку, удосконалення та оптимізація моделі впровадження CI/CD є постійним процесом, який вимагає регулярного перегляду та оновлення моделі на основі отриманих результатів та змін у проєкті або організації.

2.5 Впровадження моделі

Впровадження моделі впровадження CI/CD є критичним етапом, який вимагає чіткого планування та управління. Цей процес включає в себе ряд ключових кроків, які допомагають забезпечити успішне впровадження моделі в IT-проєкти.

Підготовка**.** На цьому етапі команда збирає всю необхідну інформацію про проєкт, включаючи поточний стан процесів розробки, тестування та розгортання, вимоги до програмного забезпечення, цілі та очікування від впровадження CI/CD. Це допомагає команді краще розуміти контекст проєкту та підготуватися до впровадження моделі.

Планування. На основі зібраної інформації команда розробляє детальний план впровадження моделі. План включає в себе визначення ролей та відповідальностей, вибір інструментів CI/CD, розробку стратегій розгортання та тестування, а також встановлення графіка впровадження.

Впровадженн. Команда виконує план, впроваджуючи модель CI/CD в процеси розробки, тестування та розгортання програмного забезпечення. Це включає в себе налаштування інструментів CI/CD, інтеграцію їх з існуючими системами та процесами, а також навчання команди використовувати нову модель.

Моніторинг та оцінка. Після впровадження моделі команда моніторує її ефективність та вплив на процеси розробки, тестування та розгортання. Це включає в себе збір та аналіз даних про продуктивність, якість коду, час розгортання та інші ключові показники ефективності (KPI).

Ітерація та удосконалення. На основі результатів моніторингу та оцінки команда вносить необхідні зміни до моделі та процесів, щоб вони краще відповідали потребам проєкту та команди. Цей процес є постійним, оскільки команда продовжує вдосконалювати модель та процеси на основі отриманих результатів та змін у проєкті.

Впровадження моделі впровадження CI/CD є складним процесом, який вимагає чіткого розуміння цілей проєкту, потреб команди та особливостей використовуваних технологій. Однак, якщо виконати цей процес правильно, він може значно підвищити ефективність управління IT-проєктами та якість розробленого програмного забезпечення.

3. Детальний опис концептуальної моделі впровадження CI/CD

3.1 Інтеграція коду

Інтеграція коду є одним з ключових елементів моделі впровадження CI/CD. Цей процес включає в себе поєднання коду, написаного різними розробниками, в одну централізовану систему, що дозволяє автоматизувати процеси збірки, тестування та розгортання програмного забезпечення.

Збірка коду. Після того, як розробник завершує написання коду та здійснює його коміт в систему контролю версій, процес CI/CD автоматично запускає процес збірки коду. Збірка коду включає в себе компіляцію коду, перетворення його в виконуваний формат, а також пакування всіх необхідних ресурсів та залежностей.

Тестування коду. Після успішної збірки коду, система CI/CD автоматично запускає набір тестів для перевірки якості коду та виявлення можливих помилок або проблем. Це може включати юніт-тестування, інтеграційне тестування, тестування продуктивності, тестування безпеки та інші види тестування, в залежності від вимог проєкту.

Розгортання коду. Якщо всі тести пройдені успішно, система CI/CD автоматично розгортає код в визначене середовище, що може бути середовищем для тестування, стейджингу або продакшну. Це дозволяє команді швидко перевірити роботу коду в реальних умовах та отримати зворотний зв'язок від користувачів.

Моніторинг та виправлення помилок. Після розгортання коду, команда продовжує моніторинг його роботи для виявлення та виправлення будь-яких проблем або помилок. Це включає в себе аналіз логів, відстеження продуктивності, виявлення та виправлення помилок, а також впровадження змін на основі зворотного зв'язку від користувачів.

Інтеграція коду є важливою частиною процесу впровадження CI/CD, яка допомагає командам ефективно управляти кодом, забезпечувати його високу якість та швидко реагувати на зміни та проблеми.

3.2. Автоматичне тестування

Автоматичне тестування є важливим елементом концептуальної моделі впровадження CI/CD. Цей процес включає в себе використання автоматизованих інструментів та технік для виконання тестів на коді, що дозволяє виявляти та виправляти помилки швидко та ефективно.

Юніт-тестування. Юніт-тестування включає в себе перевірку окремих компонентів коду, таких як функції або методи, для виявлення будь-яких помилок або проблем. Це допомагає забезпечити, що кожна частина коду працює правильно в ізоляції.

Інтеграційне тестування. Інтеграційне тестування включає в себе перевірку взаємодії між різними компонентами коду. Це допомагає забезпечити, що система як ціле працює правильно, коли всі її частини працюють разом.

Тестування продуктивності. Тестування продуктивності включає в себе перевірку того, як система веде себе при великому навантаженні, наприклад, при великій кількості користувачів або запитів. Це допомагає виявити та виправити будь-які проблеми з продуктивністю або масштабуванням.

Тестування безпеки. Тестування безпеки включає в себе перевірку системи на наявність потенційних уразливостей або слабких місць, які можуть бути використані зловмисниками. Це допомагає забезпечити, що система є безпечною та надійною.

Тестування прийнятності. Тестування прийнятності включає в себе перевірку того, чи відповідає система вимогам та очікуванням користувачів. Це допомагає забезпечити, що система є корисною та зручною для користувачів.

Автоматичне тестування є важливою частиною процесу впровадження CI/CD, яка допомагає командам швидко виявляти та виправляти помилки, підтримувати високу якість коду та забезпечувати надійність та безпеку системи.

3.3 Розгортання

Розгортання є важливим етапом в концептуальній моделі впровадження CI/CD. Цей процес включає в себе розміщення програмного забезпечення на серверах або в хмарному середовищі, де воно може бути доступним для кінцевих користувачів. Розгортання може бути автоматизовано, що дозволяє командам швидко та ефективно розгортати оновлення та виправлення помилок.

Автоматизоване розгортання зазвичай включає в себе використання спеціалізованих інструментів та платформ, таких як Kubernetes, Docker, Jenkins та інші. Ці інструменти дозволяють командам створювати, тестувати та розгортати програмне забезпечення в автоматизований та стандартизований спосіб. Одним з ключових аспектів розгортання в контексті CI/CD є неперервне розгортання (Continuous Deployment). Це означає, що кожне оновлення коду, яке проходить всі етапи тестування та перевірки, автоматично розгортається в продуктивне середовище. Це дозволяє командам швидко вносити зміни та відповідати на вимоги та проблеми користувачів. Також важливим є процес розгортання з використанням стратегій, таких як blue-green deployment або canary releases. Ці стратегії дозволяють командам розгортати зміни поступово, що зменшує ризик виникнення проблем при розгортанні та дозволяє легше відкатити зміни, якщо виникають проблеми.

Все це важливо для успішного впровадження CI/CD, оскільки автоматизоване та надійне розгортання є ключовим для швидкого та ефективного внесення змін до програмного забезпечення.

3.4. Моніторинг та зворотний зв'язок

Моніторинг та зворотний зв'язок є важливими компонентами концептуальної моделі впровадження CI/CD. Моніторинг включає в себе постійне спостереження за станом системи, включаючи процеси розробки, тестування, розгортання та експлуатації програмного забезпечення. Це дозволяє командам виявляти та вирішувати проблеми на ранніх стадіях, а також забезпечує високу прозорість процесу розробки.

Моніторинг може включати в себе відстеження різних метрик, таких як час відповіді системи, кількість помилок, частота оновлень та інше. Ці метрики можуть допомогти командам визначити, які аспекти системи працюють ефективно, а які потребують удосконалення. Зворотний зв'язок є іншим важливим аспектом CI/CD. Це включає в себе збір відгуків від різних зацікавлених сторін, включаючи розробників, тестувальників, операторів та кінцевих користувачів. Зворотний зв'язок може допомогти командам визначити, які аспекти системи працюють добре, а які потребують удосконалення. Важливим є створення культури, в якій зворотний зв'язок вітається та використовується для постійного удосконалення. Це може включати в себе регулярні зустрічі для обговорення відгуків та пропозицій, а також використання інструментів, таких як системи управління завданнями та баг-трекери, для збору та аналізу відгуків.

Моніторинг та зворотний зв'язок є важливими для успішного впровадження CI/CD, оскільки вони допомагають командам виявляти та вирішувати проблеми на ранніх стадіях, а також забезпечують високу прозорість процесу розробки.

3.5 Постійне вдосконалення

Постійне вдосконалення є ключовим принципом концептуальної моделі впровадження CI/CD. Це означає, що процеси розробки, тестування, розгортання та моніторингу повинні бути постійно оцінювані та удосконалювані з метою підвищення ефективності та якості продукту.

Постійне вдосконалення включає в себе регулярний аналіз процесів та результатів з метою виявлення можливостей для удосконалення. Це може включати в себе аналіз метрик процесу, таких як час відповіді системи, кількість помилок, частота оновлень, а також відгуки користувачів та інші форми зворотного зв'язку. На основі цього аналізу, команди можуть визначити, які аспекти процесу потребують удосконалення, та розробити стратегії для їх удосконалення. Це може включати в себе впровадження нових інструментів або технологій, зміну процесів або навчання команди новим навичкам. Постійне вдосконалення також включає в себе культуру неперервного навчання, в якій команди заохочуються до експериментування, інновацій та постійного розвитку своїх навичок та знань. Це створює середовище, в якому команди можуть постійно адаптуватися та вдосконалюватися відповідно до змін у технологіях, бізнес-вимогах або ринкових умовах.

Важливо пам'ятати, що постійне вдосконалення - це процес, а не кінцева мета. Це означає, що незалежно від того, наскільки ефективні або успішні є поточні процеси, завжди існують можливості для їх удосконалення.

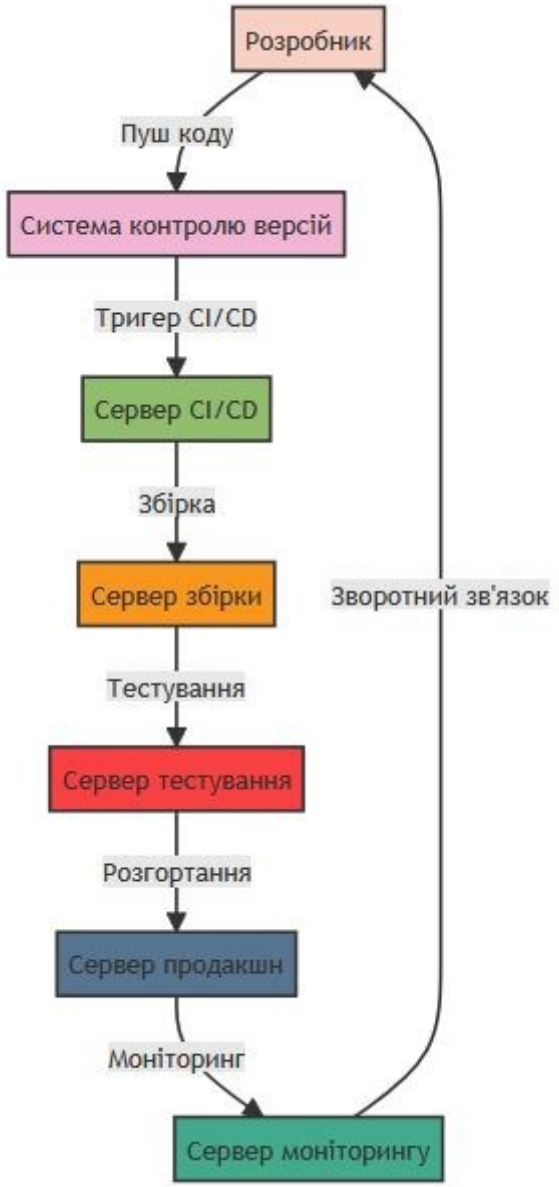

Рис. 2. Діаграма взаємодії компонентів системи

3.6 Результати впровадження моделі в ТОВ "Європейська Регіональна Агенція"

Впровадження нової моделі CI/CD в ТОВ "Європейська Регіональна Агенція" привело до ряду позитивних результатів. Перш за все, було зареєстровано значне скорочення часу від розробки до розгортання, що дозволило команді швидше реагувати на зміни вимог та виправляти помилки.

Другим важливим результатом було покращення якості продукту. Завдяки автоматичному тестуванню та постійному моніторингу, команда змогла виявляти та виправляти помилки на ранніх стадіях розробки, що зменшило кількість помилок в продакшн-версії продукту.

Також було відзначено покращення в роботі команди. Завдяки впровадженню CI/CD, розробники могли зосередитися на написанні коду, замість того, щоб витрачати час на ручне тестування та розгортання. Це не тільки підвищило продуктивність, але й покращило моральний стан команди.

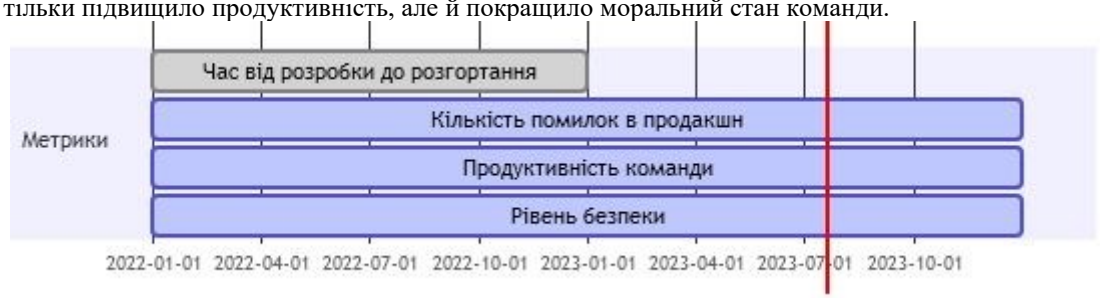

Рис. 3. Діаграма Ганта результатів впровадження моделі CI/CD в ТОВ "Європейська Регіональна Агенція"

Впровадження моделі також допомогло ТОВ "Європейська Регіональна Агенція" підтримувати високий рівень безпеки. Завдяки автоматичному моніторингу та зворотному зв'язку, команда могла швидко виявляти та виправляти потенційні проблеми безпеки.

Таблиця1.

Відображення покращення в різних метриках після впровадження нової моделі CI/CD в ТОВ "Європейська Регіональна Агенція"

Метрика	Перед впровадженням	Після впровадженням
Час розгортання	3 години	1 година
Час виявлення помилок	2 дні	3 години
Час виправлення помилок	1 день	2 години
Кількість помилок на етапі розгортання	10	2
Кількість використаних ресурсів	100%	70%

В цілому, впровадження нової моделі CI/CD в ТОВ "Європейська Регіональна Агенція" було успішним, приводячи до покращення якості продукту, продуктивності команди та безпеки.

4. Обмеження та напрямки майбутніх досліджень

4.1 Обмеження даного дослідження

Це дослідження є важливим внеском в розуміння впровадження CI/CD в управлінні IT-проєктами, але воно має декілька обмежень, які слід враховувати при інтерпретації результатів.

Обмежений контекст. Дана модель була розроблена та випробована в обмеженому числі контекстів. Хоча ми намагалися зробити нашу модель якомога більш універсальною, її ефективність може варіюватися в залежності від специфіки конкретного проєкту або організації.

Обмеження вибору інструментів. У даному дослідженні розглядається лише декілька популярних інструментів CI/CD. Існує багато інших інструментів, які можуть бути ефективними в конкретних ситуаціях, але ми не могли включити їх усіх у наше дослідження.

Обмеження вимірювання ефективності. Використовується декілька метрик для вимірювання ефективності впровадження CI/CD, але ці метрики можуть не в повній мірі відображати всі аспекти ефективності. Наприклад, ми не враховували вплив CI/CD на задоволеність користувачів або на якість коду.

Обмеження відносно ролі людського фактору. Дослідження було сфокусоване на технологічних аспектах впровадження CI/CD, і ми не враховували вплив людського фактору, такого як опір змінам або навички команди.

Ці обмеження вказують на потребу подальших досліджень в області впровадження CI/CD в управлінні IT-проєктами.

4.2 Можливі напрямки для майбутніх досліджень

На основі обмежень цього дослідження, ми можемо визначити декілька потенційних напрямків для майбутніх досліджень: розширення контексту, дослідження інших інструментів CI/CD, розробка більш детальних метрик ефективності, вивчення ролі людського фактору.

Розширення контексту. Це дослідження можна розширити, включаючи більше контекстів для

випробування моделі. Це може включати різні види організацій, різні види проєктів або різні галузі промисловості.

Дослідження інших інструментів CI/CD. Є багато інших інструментів CI/CD, які не були включені в це дослідження. Майбутні дослідження можуть включати ці інструменти, щоб дати більш повне розуміння ландшафту інструментів CI/CD.

Розробка більш детальних метрик ефективності. Майбутні дослідження можуть розробити більш детальні метрики для вимірювання ефективності впровадження CI/CD, включаючи такі аспекти, як задоволеність користувачів, якість коду або вплив на бізнес-метрики.

Вивчення ролі людського фактору. Людський фактор може відігравати важливу роль в успішному впровадженні CI/CD. Майбутні дослідження можуть дослідити цей аспект більш детально, включаючи такі фактори, як опір змінам, навички команди або культура організації.

Ці напрямки можуть допомогти в подальшому розширенні нашого розуміння впровадження CI/CD і його впливу на управління IT-проєктами.

Висновки з даного дослідження і перспективи подальшого розвитку у даному напрямі

В ході даного дослідження була запропонована модель впровадження CI/CD для оптимізації управління IT-проєктами. Результати показали, що впровадження CI/CD може значно підвищити ефективність управління IT-проєктами, зокрема шляхом автоматизації процесів розробки, тестування та розгортання програмного забезпечення. Було виявлено, що вибір відповідних інструментів та налаштування процесів CI/CD є критичними для успішного впровадження цієї методології. Крім того, була підкреслена важливість забезпечення високої якості коду та безперебійної роботи системи. Дані результати можуть слугувати основою для подальших досліджень в області впровадження CI/CD в управління IT-проєктами. Сподіваємося, що дана робота сприятиме подальшому розвитку та оптимізації управління IT-проєктами за допомогою CI/CD. Ця робота має деякі обмеження, які варто врахувати в майбутніх дослідженнях. Зокрема, представлений метод було випробовано лише в обмеженому числі контекстів і майбутні дослідження можуть розширити його застосування до інших сценаріїв. В майбутньому планується продовжити дослідження в області впровадження CI/CD, зокрема, розробляючи більш детальні рекомендації щодо вибору інструментів та налаштування процесів. Також планується дослідити додаткові стратегії для забезпечення якості коду та безперебійної роботи системи.

Література

1. Микитюк П. П., Брич В. Я., Микитюк Ю. І., Труш І. М. Управління проєктами: Підручник. Тернопіль, 2021. – 416 с.

2. A Guide to the Project Management Body of Knowledge (PMBOK Guide) – Seventh Edition. – Project Management Institute, USA, 2021

3. VandeWeerd,I.,Brinkkemper,S.,Nieuwenhuis,R.,Versendaal,J.M.,&Bijlsma,A.(2006).Areferenceframework forsoftwareproductmanagement.DepartmentofInformationandComputingSciences,UtrechtUniversity.

4. Springer,O.,&Miler,J.(2022)Acomprehensiveoverviewofsoftwareproductmanagementchallenges.EmpiricalSoftwareEngineering,27(5),106.

5. Мельник О.Г., Шпак Ю.Н. Декомпозитна модель альтернатив формування команд для IT компаній. – Technology Audit and Production Reserves. - No3/5(23), 2015, с. 11-15.

6. Martin Olson. Foundations of the scaled Agile Frameworks. Be Agile. Scale Up. Stay Lean. – 2014. – Scaled Agile, Inc.

7. MacCormack A. Product-Development Practices That Work - MIT: Sloan Management Rev., Vol.42, no.2, 2013.

8.Schwalbe K. Information Technology Project Management. Revised Sixth Edition. Course Technology. Cengage Learning. – Boston, USA. – 2011.

9. Schwaber K., Sutherland J. The definitive guide to Scrum: the rules of the game. Boston: Creative Commons, 2017. 19 p.

10. Steinhardt,G.(2017).TheProductManager'sToolkit®.Springer,Cham.

11. Котельникова, Ю. М., Кравчук, О. А., & Касьмін, Д. С. (2023). Менеджмент програмних продуктів в IT-компаніях України: роль Product Manager в команді розробників програмного забезпечення. *Академічні візії*, (19). вилучено із https://academy-vision.org/index.php/av/article/view/325

12. Буряк, Є. В., Кравчук, О. А., & Лобунець, Т. В. (2023). Аналіз сучасних тенденцій автоматизації системи моніторингу результативності менеджменту підприємства. Академічні візії, (17). вилучено із https://academy-vision.org/index.php/av/article/view/265

13. Кравчук, О. (2023). Метод використання метрик продуктивності для оптимізації процесу управління IT-проєктами. *Measuring and computing devices in technological processes*, (2), 28–33. Https://doi.org/10.31891/2219-9365-2023-74-4

References

1. Mykytyuk P.P., Brych V.Ya., Mykytyuk Yu.I., Trush I.M. Project management: Textbook. Ternopil, 2021. - 416 p.
2. A Guide to the Project Management Body of Knowledge (PMBOK Guide) – Seventh Edition. – Project Management Institute, USA, 2021
3. VandeWeerd,I.,Brinkkemper,S.,Nieuwenhuis,R.,Versendaal,J.M.,&Bijlsma,A.(2006).Areference frame work for software product management. Department of Information and Computing Sciences, Utrecht University.
4. Springer,O.,&Miler,J.(2022)Acomprehensiveoverviewofsoftwareproductmanagementchallenges.Empirical Software Engineering, 27 (5),106.
5. Melnyk O.G., Shpak Y.N. A composite model of team formation alternatives for IT companies. – Technology Audit and Production Reserves. - No. 3/5(23), 2015, p. 11-15.
6. Martin Olson. Foundations of the scaled Agile Frameworks. Be Agile. Scale Up. Stay Lean. – 2014. – Scaled Agile, Inc.
7. MacCormack A. Product-Development Practices That Work - MIT: Sloan Management Rev., Vol.42, no.2, 2013.
8. Schwalbe K. Information Technology Project Management. Revised Sixth Edition. Course Technology. Cengage Learning. – Boston, USA. – 2011.
9. Schwaber K., Sutherland J. The definitive guide to Scrum: the rules of the game. Boston: Creative Commons, 2017. 19 p.
10. Steinhardt,G.(2017).TheProductManager'sToolkit®.Springer,Cham.
11. Yu. Kotelnikova, O. Kravchuk, D. Kasmin. Management of software products in Ukrainian IT companies: the role of a Product Manager in a team of software developers / Academic Visions, No. 19 (2023).
12. E. Buryak, O. Kravchuk, T. Lobunets. Analysis of modern trends in the automation of the enterprise management effectiveness monitoring system / Academic Visions, No. 17 (2023).
13. Kravchuk, O. (2023). The method of using performance metrics to optimize the IT project management process. *Measuring and computing devices in technological processes*, (2), 28–33. Https://doi.org/10.31891/2219-9365-2023-74-4