УДК 004.4`2 + 004.62
DOI https://doi.org/10.32782/IT/2024-3-14

Юрій МИРОНОВ
аспірант кафедри інформаційних технології та комп'ютерної інженерії, Національний технічний університет «Дніпровська політехніка», просп. Дмитра Яворницького, 19, м. Дніпро, Україна, 49005
***ORCID:** 0009-0006-0675-8033*
***Scopus Author ID:** 58765290900*

Леонід ЦВІРКУН
кандидат технічних наук, професор кафедри інформаційних технологій та комп'ютерної інженерії, Національний технічний університет «Дніпровська політехніка», просп. Дмитра Яворницького, 19, м. Дніпро, Україна, 49005
***ORCID:** 0000-0002-5568-5516*
***Scopus Author ID:** 57209003910*

Бібліографічний опис статті: Миронов, Ю., Цвіркун, Л. (2024). Аналіз методів структурної оптимізації процесів неперервної інтеграції. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 3, 133–139, doi: https://doi.org/10.32782/IT/2024-3-14

АНАЛІЗ МЕТОДІВ СТРУКТУРНОЇ ОПТИМІЗАЦІЇ ПРОЦЕСІВ НЕПЕРЕРВНОЇ ІНТЕГРАЦІЇ

Неперервна інтеграція є важливою практикою в сучасній розробці програмного забезпечення. Ця робота зосереджена на аналізі практичної ефективності застосування методів структурної оптимізації процесів неперервної інтеграції.

***Метою** цієї роботи є огляд і оцінка практичної ефективності різних методів структурної оптимізації, застосованих до процесів неперервної інтеграції в різноманітних програмних середовищах.*

***Методологія**: проведено систематичний огляд літератури із фокусом на публікаціях за останні п'ять років, щоб зафіксувати останні досягнення та нові тенденції в підвищенні ефективності процесів неперервної інтеграції. Для пошуку в академічних базах даних використовувалися такі ключові слова, як «неперервна інтеграція», «оптимізація неперервної інтеграції», «спрямований ациклічний граф», «пріоритизація завдань», «інкрементні збірки» та «паралельне тестування». Критерії огляду зосереджені на високоякісних досліджень, включаючи рецензовані статті в журналах, доповіді на конференціях і професійні публікації. Статті, які явно не зосереджувалися на підвищенні ефективності процесів неперервної інтеграції або не мали емпіричних доказів, були виключені з огляду. Крім того, було проведено експерименти для порівняльного аналізу ефективності та застосовності різних методів оптимізації для різних програмних рішень: S1 – застаріла багатомодульна монолітна система; S2 – веб-застосунок із сервісно-орієнтованою архітектурою; S3 – сучасний застосунок у хмарному середовищі; S4 – рішення «інфраструктура як код» зі складним багатоетапним процесом; S5 – рішення для автоматизації тестування, зосереджене на багатоетапних наскрізних тестах. Результати застосування кожного методу оптимізації були ретельно задокументовані, щоб підтвердити остаточні висновки. Аналіз ефективності базується на аналізі довжини критичного шляху процесу неперервної інтеграції у вигляді графу.*

***Наукова новизна** цього дослідження полягає у використанні методу порівняльного аналізу для оцінки ефективності застосування різних методів структурної оптимізації процесів неперервної інтеграції у складних програмних рішеннях у гібридних середовищах.*

***Результати** дослідження підкреслили неоднаковий вплив кожного методу структурної оптимізації на ефективність процесів неперервної інтеграції в різних рішеннях. Хоча загальна тенденція до більш ефективного виконання процесів очевидна в кожному випадку, ефективність кожного оцінюваного методу відрізняється від одного рішення до іншого. Схоже, що така варіація залежить від багатьох факторів, включаючи стек технологій, розмір кодової бази, внутрішні обмеження, а також розмір і складність процесу неперервної інтеграції. Виявлення цих факторів, а також оцінка методів структурної оптимізації в більшому масштабі для визначення кореляцій між цими факторами та ефективністю процесів неперервної інтеграції можуть стати питаннями майбутніх досліджень.*

***Ключові слова:** неперервна інтеграція, структурна оптимізація, розробка програмного забезпечення, обслуговування програмного забезпечення, оптимізація обчислень.*

Yurii MYRONOV
Postgraduate Student at the Department of Information Technology and Computer Engineering, Dnipro University of Technology, 19, Dmytra Yavornytskoho Ave., Dnipro, Ukraine, 49005
***ORCID:** 0009-0006-0675-8033*
***Scopus Author ID:** 58765290900*

Leonid TSVIRKUN
PhD, Professor at the Department of Information Technology and Computer Engineering, Dnipro University of Technology, 19, Dmytra Yavornytskoho Ave., Dnipro, Ukraine, 49005
***ORCID:** 0000-0002-5568-5516*
***Scopus Author ID:** 57209003910*

To cite this article: Myronov, Yu., Tsvirkun, L. (2024). Analiz metodiv strukturnoi optymizatsii protsesiv neperervnoi intehratsii [Analysis of continuous integration structural optimization methods]. *Information Technology: Computer Science, Software Engineering and Cyber Security,* 3, 133–139, doi: https://doi.org/10.32782/IT/2024-3-14

ANALYSIS OF STRUCTURAL OPTIMIZATION METHODS FOR CONTINUOUS INTEGRATION PIPELINES

Continuous Integration (CI) is an essential practice in modern software development. This paper focuses on analyzing the practical efficiency of applying structural optimization techniques to continuous integration pipelines.

***The objective** of this work is to review and evaluate the practical efficiency of various structural optimization techniques applied to continuous integration pipelines in diverse software environments.*

***Methodology:** A comprehensive literature review was conducted, focusing on publications from the past ten years to capture recent advancements and emerging trends in CI efficiency improvements. The systematic search of academic databases used keywords such as «Continuous Integration», «CI optimization», «direct acyclic graph», «tasks prioritization», «incremental builds», and «parallel testing». Inclusion criteria ensured the selection of high-quality studies, including peer-reviewed journal articles, conference papers, and industry reports. Articles that did not explicitly focus on CI efficiency improvements or lacked empirical evidence were excluded from the review. Additionally, an experiment was conducted to perform a comparative analysis of the effectiveness and applicability of various optimization techniques across different software solutions: S1 – A legacy multi-module monolithic system; S2 – A web-based application with service-oriented architecture; S3 – A modern cloud-native application; S4 – An Infrastructure as a Code solution with a complex multi-staging pipeline; S5 – A test automation solution focused on multi-staged end-to-end tests. The results of applying each optimization technique were thoroughly documented to support final conclusions. The effectiveness analysis is based on the critical path length analysis of a directed acyclic graph representation of the Continuous Integration pipeline.*

***The novelty** of this research lies in using comparative analysis to evaluate the efficiency of applying various structural optimization techniques to complex software solutions in hybrid environments.*

***The results** of the research highlighted the uneven impact of each structural optimization technique on Continuous Integration pipeline efficiency across the different solutions. While an overall trend toward more effective pipeline execution is evident in each case, the effectiveness of each evaluated method varies from one solution to another. This variation appears to depend on multiple factors, including the technology stack, solution size, inherent limitations, and the size and complexity of the Continuous Integration pipeline. Identifying these factors, as well as evaluating structural optimization techniques on a larger scale to determine correlations between these factors and pipeline efficiency, could be a focus for future research.*

***Key words:** Continuous Integration, structural optimization, software development, software maintenance, computational optimization.*

Актуальність проблеми. Неперервна інтеграція стала фундаментальною практикою в розробці сучасного програмного забезпечення, полегшуючи систематичне тестування та інтеграцію змін програмного коду. Спочатку розроблені для вирішення проблем, пов'язаних з інтеграцією програмного коду, який змінюється одночасно багатьма інженерами, процеси неперервної інтеграції постійно змінюються і відповідно до вимог сучасних середовищ розробки, які характеризуються великими складними кодовими базами, над якими працюють розподілені команди фахівців. Основними об'єктами сучасних робіт у напрямку підвищення ефективності процесів неперервної інтеграції є прискорення зворотного зв'язку та ефективне управління ресурсами (Jin, Servant, 2021), що особливо стає важливим із зростанням популярності хмарних обчислень – підвищення ефективності процесів неперервної інтеграції

має на меті не тільки підвищення ефективності використання обчислювальних ресурсів, але й зниження операційних витрат і скорочення часу випуску нових версій програмного забезпечення. За своєю суттю, процеси неперервної інтеграції є набором взаємопов'язаних завдань обробки програмного коду, які оркеструються (конфігуруються, координуються, впроваджуються, виконуються на керуються) системою неперервної інтеграції в обчислювальному середовищі (Burdiuzha, 2023). Методи підвищення ефективності процесів неперервної інтеграції націлені на кожний з аспектів цих процесів, їх умовно можна поділити на такі категорії: методи оптимізації обчислювального середовища, методи оптимізації кодової бази, методи оптимізації засобів обробки початкового коду, методи оптимізації системи управління процесами, та методи структурної оптимізації (Tsvirkun, Myronov, 2023). Ця робота акцентує увагу на дослідженні методів структурної оптимізації процесів неперервної інтеграції, тобто на методах, які впливають на порядок виконання завдань неперервної інтеграції та їхні зв'язки з метою зменшення загального часу для виконання процесу неперервної інтеграції, безвідносно стану початкового коду, засобів обробки чи обчислювального середовища.

Аналіз останніх досліджень і публікацій. Дослідження методів підвищення ефективності процесів неперервної інтеграції, які можна віднести до категорії методів структурної оптимізації, стосується чотирьох основних напрямів: методу розпаралелювання завдань неперервної інтеграції, методу пріоритизації завдань, методу оптимізації взаємозв'язків завдань шляхом структурування їх у вигляді спрямованого ациклічного графу та методу умовного виконання завдань (Jin, Servant, 2021). Найбільш широко досліджується методи розпаралелювання та умовного виконання завдань, в останньому слід виділити такий напрямок дослідження як використання засобів машинного навчання та штучного інтелекту для розпізнавання змін початкового коду та інтелектуальної побудови переліку завдань неперервної інтеграції (Jin, Servant, 2023). У порівнянні з тим, кількість наукових досліджень методів оптимізації черги завдань та побудови зв'язків у вигляді спрямованого ациклічного графу є незначною (Stahl, 2017). Проаналізовані роботи зосереджені на визначенні впливу застосованих методів на процеси неперервної інтеграції, порівнюючи розроблені методи з аналогічними, однак не містять систематичних порівняльних досліджень з методами інших напрямів.

Мета дослідження. Робота має на меті дослідити вплив сучасних практики підвищення ефективності процесів неперервної інтеграції шляхом оптимізації структури завдань процесів та їхніх взаємозв'язків, на загальний час виконання процесів. Завдання дослідження – визначити ключові методи структурної оптимізації процесів неперервної інтеграції, та порівняти ефективність їх застосування у різноманітному середовищі програмного забезпечення.

Виклад основного матеріалу дослідження. Методи структурної оптимізації є однією з ключових категорій методів підвищення ефективності процесів неперервної інтеграції, і зосереджений на оптимізацію внутрішньої структури процесів та взаємозв'язків між задачами. Як інші методи підвищення ефективності процесів неперервної інтеграції, вони мають на меті пришвидшення зворотного зв'язку із розробниками програмного забезпечення, зменшення часу на виконання завдань обробки даних, а також зменшення кількості обчислювальних програмних ресурсів для виконання завдань.

Найбільш широко застосовуваним методом структурної оптимізації процесів неперервної інтеграції є *метод розпаралелювання завдань*. Цей метод спрямований, в першу чергу, на підвищення ефективності найбільш ресуроємного етапу – кроку тестування, шляхом одночасного виконання кількох завдань.

У традиційних системах неперервної інтеграції завдання тестування часто виконуються послідовно, що створює обмеження продуктивності процесів, особливо в проєктах із великою кількістю тестів. Паралельне тестування вирішує цю проблему, коли тести розподіляються між кількома процесорами або обчислювальними вузлами, що зменшує загальний час, який необхідний для виконання тестів, і прискорює цикл зворотного зв'язку.

Впровадження розпаралеленого тестування передбачає як налаштування програмних засобів тестування, так і зміну структури завдань неперервної інтеграції для одночасного виконання тестів, гарантуючи, що кожен тест ізольований від інших, щоб запобігти перешкодам і хибним результатам. Дослідження паралельного тестування постійно демонструють його ефективність у скороченні часу, яке необхідне для етапу тестування, залежно від складності та розміру набору тестів (Fallahzadeh та ін., 2023).

Скорочення часу призводять до швидшого зворотного зв'язку для розробників, що дозволяє швидше ідентифікувати та вирішувати проблеми. Відповідно до експериментальних

досліджень, особливу ефективність цей метод демонструє у великомасштабних і складних проєктах S1 та S5, у яких довге тестування може затримати випуск нових версій та збільшити витрати на розробку (таблиця 1). Однак впровадження паралельного тестування також пов'язане з ризиком отримання помилкових результатів (Bavand, 2021). При впровадженні методу розпаралелювання завдань тестування важливо проаналізувати потенційні особливості виконуваних тестів на предмет внутрішніх залежностей та доступу до спільних ресурсів.

Метод пріоритизації завдань в процесах неперервної інтеграції є найменш популярною практикою підвищення ефективності процесів, зокрема, через відсутність підтримки у багатьох системах неперервної інтеграції. Тим не менш, цей метод дозволяє забезпечувати більшу ефективність процесів шляхом керування чергою завдань. Спираючись на принципи теорії масового обслуговування, керування чергами в неперервної інтеграції полягає у встановленні пріоритетів завдань збірки, тестування і розгортання програмного забезпечення з метою мінімізації обмежень продуктивності процесів, а також оптимізації використання ресурсів. У теорії масового обслуговування завдання часто організовуються в пріоритетні черги, де важливіші завдання обробляються раніше за інші. Ця концепція безпосередньо застосовна до процесів неперервної інтеграції, де керування порядком і пріоритетом завдань може значно вплинути на загальну ефективність циклу розробки.

Наприклад, високопріоритетні збірки або критичні зміни можна розміщувати на початку черги, забезпечуючи їхню миттєву обробку, тоді як менш критичні завдання ставляться в чергу за ними. Іншим важливим аспектом керування чергами неперервної інтеграції є динамічне керування чергами, яке передбачає налаштування пріоритету завдань у режимі реального часу на основі поточних умов, таких як доступність ресурсів або терміновість конкретної збірки. Наприклад, якщо важливе впровадження нового коду затримується через обмеження ресурсів, система неперервної інтеграції може динамічно перерозподіляти ресурси або коригувати пріоритети завдань для вирішення проблеми, таким чином зберігаючи потік процесу неперервної інтеграції. Дослідження застосування теорії масового обслуговування в процесах неперервної інтеграції продемонстрували, що ефективне керування чергами може призвести до незначного покращення часу використання ресурсів і загальної ефективності неперервної інтеграції (Wang та ін., 2020)

Впроваджуючи пріоритетні черги, системи неперервної інтеграції можуть гарантувати, що найважливіші завдання вирішуються першими, зменшуючи ймовірність виникнення вузьких місць і забезпечуючи більш передбачуваний і надійний процес обробки даних. Поставлений експеримент показав, що впровадження цього методу дозволяє скоротити середній час виконання процесів неперервної інтеграції до 20% (таблиця 1), особливо у масштабних проєктах на декілька інженерних команд, що працюють в умовах обмежених ресурсів.

Ще одним непопулярним методом оптимізації процесів неперервної інтеграції є метод *організації завдань у вигляді спрямованого ациклічного графу*. Цей метод пропонує формалізований підхід для структурування складних процесів обробки даних. У моделі спрямованого ациклічного графу кожен вузол представляє завдання неперервної інтеграції, тоді як ребра між вузлами означають залежності між цими завданнями. Визначальною особливістю методу є гарантія того, що завдання неперервної інтеграції не залежать циклічно один від одного, і таким чином виключаються взаємоблокування або можливість появи нескінченних циклів.

Цей метод оптимізація за своєю суттю підходить для керування залежностями між

Таблиця 1

Експериментальне порівняння ефективності застосування методів структурної оптимізації процесів неперервної інтеграції

Метод структурної оптимізації	Оптимізація часу виконання процесів				
	S1	**S2**	**S3**	**S4**	**S5**
розпаралелювання завдань[1]	0,8n	0,9n	n	N/A[2]	0.8n
пріоритизація завдань	1,1	1,0	1,0	1,0	1,2
організація завдань CAГ	1,7	1,6	1,0	1,2	2,0
умовне виконання завдань	1,4	1,7	1,8	1,2	85,0

[1] n-ступінь паралелізації, n=>2.

[2] Через специфіку проєкту, застосування даного методу не є можливим.

завданнями неперервної інтеграції, які виникають у складних великих процесах. У класичних процесах неперервної інтеграції завдання одного етапу виконуються лише після завершення усіх завдань попереднього етапу. Тому застосування цього методу дозволяє системам неперервної інтеграції порушувати послідовність виконання етапів і виконувати завдання у встановленому порядку, гарантуючи, що завдання виконуються лише тоді, коли задовольняються їхні передумови.

Такий підхід підвищує ефективність процесу неперервної інтеграції, уникаючи непотрібних затримок і оптимізуючи порядок виконання завдань.

Як показують дослідження (Zheng та ін., 2024) та проведений експеримент (таблиця 1), організація завдань неперервної інтеграції у вигляді графів особливо ефективна у великих і складних проєктах із величезною кількістю залежностей, які обмежують продуктивність завдань обробки даних. Репрезентація цих залежностей у графічному виді (як правило, у вигляді діаграм Санкея) дозволяє проводити поглиблений аналіз цих місць і критичного шляху з метою впровадження інших методів оптимізації процесів неперервної інтеграції.

Однак ефективність цього методу залежить від точного визначення залежностей між завданнями і забезпечення того, що ці залежності контрольовані. У середовищах із дуже мінливими або непередбачуваними залежностями, застосування цього методу може спричиняти затримки в обробці даних або неефективне використання обчислювальних ресурсів (Lyu та ін., 2024). Незважаючи на це, за правильного впровадження, метод організації завдань у вигляді спрямованого ациклічного графу залишається потужним інструментом для підвищення ефективності процесів неперервної інтеграції.

Під методом *умовного виконання завдань* неперервної інтеграції розуміється вибірковий запуск завдань обробки даних на основі попередньо визначених правил або інтелектуального розпізнавання контексту змінених даних. Ця практика спрямована на оптимізацію використання ресурсів і скорочення часу процесу неперервної інтеграції шляхом виконання лише необхідних завдань, а не повторного запуску всього процесу для кожної зміни початкового коду. Традиційні системи неперервної інтеграції використовують заздалегідь визначені правила, такі як зміна файлів або ключові слова у змінюваних даних, для визначення переліку завдань для виконання.

Наприклад, тести можуть запускатися лише тоді, коли є зміни у файлах в певному каталозі, що зменшує час на виконання тестів. Хоча застосування методу у його традиційному виді, на основі правил, підвищує ефективність процесів неперервної інтеграції (таблиця 1), ручні налаштування вимагають постійної підтримки з боку інженерів, які повинні регулярно оновлювати правила в міру розвитку проєкту, щоб гарантувати відповідність правил найновішим змінам. У великомасштабних проєктах зі складними залежностями між задачами, впровадження методу у його традиційному варіанті може бути надто коштовним.

З метою подолання традиційних обмежень цього методу досліджуються інноваційні підходи, які використовують машинне навчання та штучний інтелект для прогнозування та інтелектуального визначення переліку завдань неперервної інтеграції на основі змін початкових даних. Одними з найбільш популярних підходів є SmartBuildSkip, BuildFast і HybridCISave. Підхід SmartBuildSkip базується на аналізі певних параметрів проєкту та внесених змін, беручи до уваги як нетехнічні показники, наприклад, робочий день, кількість інженерів і вік даних, так і аналізуючи конкретні зміни, внесені до початкового коду, що дозволяє ідентифікувати зміни які навряд чи створять нові проблеми, таким чином економлячи час і обчислювальні ресурси (Jin, Servant, 2020). Метод BuildFast використовує подібний підхід, але також звертає увагу на історичні результати завдань інтеграції, щоб вирішити, чи потрібна збірка, що значно підвищило якість прогнозування необхідних завдань порівняно із SmartBuildSkip (Chen та ін., 2020). HybridCISave, який має найновітніший підхід до умовного виконання завдань неперервної інтеграції, поєднує підходи на основі правил і прогнозування для гібридного рішення для оптимізації процесів збірки початкового коду і тестування. Наприклад, HybridCISave може спочатку визначати набір завдань за допомогою попередньо визначених правил, а потім уточнювати рішення щодо їх виконання за допомогою моделей машинного навчання, які передбачають ймовірність необхідності виконання завдання на основі останніх змін. Цей гібридний підхід врівноважує надійність конфігурацій на основі правил із гнучкістю прогнозних моделей, пропонуючи потужний інструмент для підвищення ефективності процесів неперервної інтеграції (Jin, Servant, 2023).

Дослідження показують, що такі інструменти, як HybridCISave, можуть зменшити кількість завдань неперервної інтеграції на 93%,

мінімально впливаючи на точність виявлення інтеграційних проблем. Ти не менш, слід мати на увазі, що точність прогнозів залежить від якості даних, які використовуються для навчання моделей, а неправильні прогнози можуть призвести до пропуску завдань, які насправді були необхідними. Гібридні підходи, такі як HybridCISave, зменшують цей ризик, поєднуючи узгодженість конфігурацій на основі правил із можливістю адаптації прогнозних моделей.

Висновки і перспективи подальших досліджень. У цій статті досліджено ряд сучасних методів оптимізації структури процесів неперервної інтеграції, кожен з яких спрямований на підвищення швидкості, ефективності та використання ресурсів робочих процесів розробки програмного забезпечення. Розглянуто як традиційні методи: паралельне виконання завдань, пріоритизація завдань та умовного виконання завдань, так і новітні методи інтелектуального визначення переліку завдань неперервної інтеграції.

Ці методи продемонстрували значні покращення ключових показників неперервної інтеграції, зокрема скорочення критичного шляху процесів і пришвидшення циклу зворотного зв'язку. Найбільш ефективними виявилися методи розпаралелювання завдань та умовного виконання завдань, вплив яких значною мірою перевищує вплив на швидкість виконання процесів неперервної інтеграції при використанні методів структуризації завдань у вигляді спрямованого ациклічного графу і пріоритизації завдань. Однак слід зазначити, що застосування цих методів не завжди є можливим, особливо у нетрадиційних проєктах програмного забезпечення, як-то проєкти, спрямовані на описання інфраструктурної конфігурації середовища.

Інтелектуальні підходи, хоч і багатообіцяючі, значною мірою залежать від якості навчальних даних і точності базових алгоритмів. Неправильна конфігурація може призвести до пропуску завдань, критичних для підтримки цілісності коду. Крім того, інтеграція із застарілими системами та забезпечення міжплатформної сумісності продовжують викликати занепокоєння, особливо для організацій, які переходять на хмарні середовища неперервної інтеграції.

Заглядаючи вперед, майбутні дослідження можуть зосередитися на вдосконаленні методів прогнозування необхідного переліку завдань неперервної інтеграції, впровадження цих методів для традиційних систем неперервної інтеграції, а також на більш ретельному і масштабному аналізі впливу методів пріоритизації завдань та організації їх у вигляді графів. З розвитком програмної інженерії впровадження передових методів підвищення ефективності неперервної інтеграції сприятиме підвищенню ефективності, масштабованості та гнучкості процесів розробки, що зрештою сприятиме швидшій і надійнішій доставці програмного забезпечення.

ЛІТЕРАТУРА:

1. Xianhao J., Servant F. What helped, and what did not? An Evaluation of the Strategies to Improve Continuous Integration. *43rd International Conference on Software Engineering*: Conference Proceedings., Madrid, 25–28 May 2021. P. 213–225.

2. Burdiuzha R. Building an effective CI/CD pipeline: A comprehensive guide. Medium.com. URL: https://gartsolutions.medium.com/building-an-effective-ci-cd-pipeline-a-comprehensive-guide-bb07343973b7 (date of access: 10.09.2024).

3. Tsvirkun L., Myronov Y. Challenges and Specificities of Adopting Continuous Integration within Scalable Cloud Environments. *IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*: Conference Proceedings, Lviv, 19–21 October 2023.

4. Xianhao J., Servant F. HybridCISave: A Combined Build and Test Selection Approach in Continuous Integration. *ACM Transactions on Software Engineering and Methodology*. 2023. Vol. 32, no. 4. P. 1–39.

5. Stahl D. Large scale continuous integration and delivery: Making great software better and faster: PhD thesis. Groningen, 2017. 257 p.

6. Fallahzadeh E., Bavand A. H., Rigby P. C. Accelerating Continuous Integration with Parallel Batch Testing. 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering: Conference Proceedings, San Francisco, 3–9 December 2023. New York.

7. Bavand A. H. The Impact of Parallel and Batch Testing in Continuous Integration Environments: Masters Thesis. Concordia University, 2021. 109 p.

8. Scalable build service system with smart scheduling service / K. Wang et al. *ISSTA 2020: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*: Conference Proceedings, 18–22 July 2020. New York, 2020. P. 456–462.

9. Zheng S., Adams B., Hassan A. E. Does using Bazel help speed up continuous integration builds?. *Empirical Software Engineering*. 2024. Vol. 29, no. 5.

10. Detecting Build Dependency Errors in Incremental Builds / J. Lyu et al. *ISSTA 2024: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*: Conference Proceedings, Vienna, 16–20 September 2024. New York, 2024. P. 1–12.

11. Xianhao J., Servant F. A cost-efficient approach to building in continuous integration. *ICSE '20: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*: Conference Proceedings, Seoul, 27 June – 19 July 2020. New York, 2020. P. 13–25.

12. BUILDFAST: History-Aware Build Outcome Prediction for Fast Feedback and Reduced Cost in Continuous Integration / B. Chen et al. *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*: Conference Proceedings, Melbourne, 21–25 September 2020. 2024. P. 42–53.

REFERENCES:

1. Jin, X., & Servant, F. (2021). What helped, and what did not? An Evaluation of the Strategies to Improve Continuous Integration. In Proceedings of the *43rd International Conference on Software Engineering* (pp. 213–225). IEEE Press.

2. Burdiuzha, R. (2023). Building an effective CI/CD pipeline: A comprehensive guide. Retrieved from: https://gartsolutions.medium.com/building-an-effective-ci-cd-pipeline-a-comprehensive-guide-bb07343973b7.

3. Tsvirkun, L., & Myronov, Y. (2023). Challenges and Specificities of Adopting Continuous Integration within Scalable Cloud Environments. 2023 IEEE *18th International Conference on Computer Science and Information Technologies (CSIT)*, 1–4. doi:10.1109/CSIT61576.2023.10324010.

4. Xianhao, J., Servant, F. (2023). HybridCISave: A Combined Build and Test Selection Approach in Continuous Integration. *ACM Transactions on Software Engineering and Methodology*, 32(4).

5. Stahl, D. (2017). Large scale continuous integration and delivery: Making great software better and faster. [Thesis fully internal (DIV), University of Groningen]. University of Groningen.

6. Fallahzadeh, E., Bavand, A. H., & Rigby, P. C. (2023, November). Accelerating Continuous Integration with Parallel Batch Testing. Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 55–67. doi:10.1145/3611643.3616255

7. Bavand, A. (2021) The Impact of Parallel and Batch Testing in Continuous Integration Environments. Masters thesis, Concordia University.

8. Wang, K., Tener, G., Gullapalli, V., Huang, X., Gad, A., & Rall, D. (2020). Scalable build service system with smart scheduling service. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 452–462). Association for Computing Machinery.

9. Zheng, Shenyu & Adams, Bram & Hassan, Ahmed E. (2024). Does using Bazel help speed up continuous integration builds?. *Empirical Software Engineering*. 29. 10.1007/s10664-024-10497-x.

10. Lyu, Jun & Li, Shanshan & Zhang, He & Zhang, Yang & Rong, Guoping & Rigger, Manuel. (2024). Detecting Build Dependency Errors in Incremental Builds. 1-12. 10.1145/3650212.3652105.

11. Jin, X., & Servant, F. (2020). A Cost-efficient Approach to Building in Continuous Integration. In 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE) (pp. 13-25).

12. Chen, B., Chen, L., Zhang, C., & Peng, X. (2020). BUILDFAST: History-Aware Build Outcome Prediction for Fast Feedback and Reduced Cost in Continuous Integration. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 42–53).