УДК 681.3.07

О. В. Прус; В. П. Майданюк, канд. тех. наук, доц.;
І. Р. Арсенюк, канд. тех. наук, доц.

АНАЛІЗ ІНСТРУМЕНТІВ УПРАВЛІННЯ БАГАТОПРОЄКТНИМИ СЕРЕДОВИЩАМИ: ОПТИМІЗАЦІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У статті розглянуто підхід щодо підвищення ефективності управління багатопроєктними середовищами у сфері розробки програмного забезпечення. Основна увага приділяється детальному аналізу та порівнянню сучасних інструментів управління монорепозиторіями, зокрема, Lerna, Yarn Workspaces, Bazel, Rush, Pants, Bit та Nx. Для оцінювання їхньої функціональності, продуктивності, масштабованості, сумісності з різними технологічними стеками, а також вивчення їх впливу на загальну ефективність розробки додатків використано багатокритеріальний аналіз. У контексті підвищення ефективності управління багатопроєктними середовищами у сфері розробки програмного забезпечення, де ресурси містять, зокрема, час та зусилля програмістів, технічну інфраструктуру та фінансові засоби, у той час коли кілька проєктів конкурують за ці обмежені ресурси, виникає нагальна потреба у прийнятті складних рішень щодо пріоритизації та розподілу. Тому було висунуто концепцію застосування комплексного багатокритеріального аналізу для оцінювання інструментів управління монорепозиторіями. Такий підхід дозволяє кількісно оцінити та порівняти різні інструменти на основі заздалегідь визначених критеріїв з використанням формули корисності. При цьому проведено роботу щодо збирання та оцінювання даних критеріїв у контексті розробки програмних засобів у багатопроєктних середовищах. Це дозволило не лише кількісно оцінити різні інструменти на основі заздалегідь визначених критеріїв та їх ваг, а й дослідити переваги та обмеження кожного з них. Це дослідження дозволяє виявити найбільш ефективний варіант для підвищення продуктивності та оптимізації процесів управління проєктами, забезпечуючи розробників програмного забезпечення необхідною інформацією. Однак важливо враховувати і той факт, що вибір конкретного інструменту повинен бути обумовлений конкретними потребами та контекстом кожного окремого проєкту. Тому результати цього дослідження слід розглядати як підказку, а не як абсолютно однозначне рішення.

Ключові слова: *багатопроєктне середовище, управління монорепозиторіями, Lerna, Yarn Workspaces, Bazel, Rush, Pants, Bit, Nx, багатокритеріальний аналіз, оптимізація розробки програмного забезпечення, гнучкість розробки програмного забезпечення, масштабування проєктів.*

Вступ

У сфері розробки програмного забезпечення (ПЗ), особливо у контексті багатопроєктних середовищ, виникає низка унікальних викликів. Багатопроєктне середовище — це сценарій, де програмісти та команди розробників працюють над декількома проєктами одночасно, що часто призводить до перетинання задач і ресурсів [1].

Однією з ключових задач є розподіл ресурсів. У контексті багатопроєктного управління, ресурси включають час та зусилля програмістів, технічну інфраструктуру та фінансові засоби. Коли кілька проєктів конкурують за ці обмежені ресурси, виникає потреба у прийнятті складних рішень щодо пріоритизації та розподілу, що може призвести до затримок та збільшення витрат [2].

Інша важлива задача – координація роботи між проєктами. У багатопроєктному середовищі, синхронізація розкладів, узгодження інтересів та ефективна комунікація між командами є вирішальними для досягнення цілісності та ефективності у реалізації проєктів [3].

Останньою, але не менш важливою, є задача забезпечення якості ПЗ. У ситуації, де увага команд розділена між численними проєктами, існує високий ризик погіршення якості

окремих проєктів внаслідок недостатніх ресурсів або обмеженого часу [1].

Таким чином, необхідність вивчення та реалізації ефективних інструментів та методів управління у багатопроєктному середовищі є досить актуальною та вимагає глибокого аналізу та осмислення з точки зору програмістів і розробників ПЗ.

Аналіз останніх досліджень і публікацій

У сучасному світі програмування, особливо у контексті багатопроєктних середовищ, важливо ознайомитися з останніми дослідженнями та трендами, щоб адекватно відповідати на виклики, які вони ставлять перед розробниками ПЗ. Нижче розглянемо ключові дослідження у цій галузі, з урахуванням їхніх переваг та потенційних недоліків.

У роботі [4] автор акцентує на важливості дотримання чистоти коду. Однак, ця робота може бути сприйнята як надто теоретична та не завжди практично придатна для конкретних викликів, які стоять перед розробниками в сучасних багатопроєктних системах.

Автори роботи [5] наголошують на неперервній доставці як ключовому елементі під час розробки ПЗ. Проте, їхній підхід може бути складним для імплементації у певних середовищах, де інфраструктура та процеси ще не повністю підготовлені для такого роду автоматизації.

У роботі [6] автор зосереджується на модульності у Web-дизайні. Однак, наведена теорія може не враховувати складнощі інтеграції та управління залежностями, які часто виникають у багатомодульних проєктах.

Автор роботи [7] досліджує використання мережевих технологій. Однак, ця робота може мати обмежене застосування у контексті загальних багатопроєктних середовищ, де потрібен більш комплексний підхід до управління проєктами.

У роботі [8] автори висвітлюють складнощі управління багатопроєктними середовищами, але можуть не враховувати специфічні технологічні виклики, пов'язані з розробкою ПЗ, такі як забезпечення сумісності та інтеграції різних платформ та інструментів.

Ці дослідження надають важливі напрямки для розуміння та вдосконалення процесів у багатопроєктних середовищах, однак важливо також враховувати потенційні виклики та обмеження, які вони можуть ставити перед розробниками.

Виділення не вирішених раніше частин загальної задачі

У контексті сучасних багатопроєктних середовищ програмування, одним з ключових викликів залишається вибір та оптимізація інструментів управління монорепозиторіями. Хоча існує багато розроблених рішень, таких як Lerna, Yarn Workspaces, Bazel, Rush, Pants, Bit, Nx та інших, не всі аспекти їх використання, інтеграції та оптимізації були ретельно проаналізовані та порівняні у наукових дослідженнях [9]. Актуальність вибору та оптимізації інструментів управління монорепозиторіями полягає у тому, що вибір правильного інструменту для управління монорепозиторієм може значно вплинути на продуктивність розробки, ефективність командної роботи та спроможність проєкту адаптуватися до змінних вимог. Ці інструменти варіюються за такими параметрами, як підтримка різноманітних мов програмування, інтеграція з наявними CI/CD (Continuous Integration and Continuous Delivery/Deployment) пайплайнами (автоматизована послідовність дій, яка дозволяє максимально ефективно інтегрувати, тестувати та доставляти оновлення ПЗ), зручність управління залежностями, та ефективність у масштабних проєктах [10].

Враховуючи це, стає очевидною потреба у детальному порівняльному аналізі цих інструментів. Такий аналіз повинен містити оцінку їх функціональності, продуктивності, масштабованості, сумісності з різними технологічними стеками, а також вивчення їх впливу на загальну ефективність розробки у багатопроєктних середовищах. Це не лише допоможе розробникам у прийнятті обґрунтованих рішень щодо вибору інструментів, а й сприятиме загальному прогресу у практиках управління монорепозиторіями у галузі розробки ПЗ [11].

Основною метою статті є підвищення ефективності розробки та спрощення процесів управління проєктами за рахунок проведення багатокритеріального аналізу щодо вибору найбільш прийнятного інструменту для управління монорепозиторіями, виходячи з конкретних потреб масштабованості, швидкості розгортання проєкту, гнучкості інтеграції, а також технічних вимог щодо чистоти коду, надійності та продуктивності. Мета досягається шляхом проведення детального аналізу та порівняння сучасних інструментів управління монорепозиторіями, таких як Lerna, Yarn Workspaces, Bazel, Rush, Pants, Bit та Nx у контексті визначення їхніх переваг, недоліків та оптимальних сценаріїв використання.

Основна частина

Забезпечення ефективності управління програмним кодом, покращення координації між проєктними командами та оптимізації процесів інтеграції та розгортання ПЗ досягається за рахунок використання інструментів управління монорепозиторіями. Серед сучасних та широко відомих у колах професійних розробників можна виділити нижче наведені інструменти.

Система збирання та проєктування Lerna призначена для оптимізації роботи з проєктами, що містять кілька пакетів, спрощує процеси управління залежностями та розподілу пакетів, але відома своєю складністю у налаштуванні проєктів та має досить серйозні обмеження у використанні. Наприклад, Lerna управляє кожним пакетом у монорепозиторії окремо. Це означає, що для великих репозиторіїв з багатьма пакетами, кожен пакет потребує індивідуального аналізу, оновлення залежностей, інсталяції, та збирання і цей процес є досить тривалим. Крім того, якщо у різних пакетах використовуються подібні залежності, Lerna може не ефективно кешувати ці залежності між пакетами, змушуючи кожен пакет здійснювати окремі запити на встановлення залежностей [12].

Робоче середовище Yarn Workspaces дозволяє розробникам ефективно керувати залежностями у монорепозиторіях. Підтримує спільне використання пакетів і спрощує процеси інсталяції та оновлення, однак може вимагати додаткових зусиль для інтеграції з іншими інструментами [13].

Система збирання та тестування Bazel. Цей інструмент від Google підходить для масштабних монорепозиторіїв з високими продуктивністю та ефективністю. Bazel оптимізує процес збирання, використовуючи кешування та інкрементальне збирання. Це означає, що Bazel збирає тільки ті частини проєкту, які були змінені, а не увесь проєкт заново, що значно скорочує час збирання. Завдяки своїм алгоритмам кешування та паралелізації, Bazel здатний швидко обробляти великі об'єми даних, що забезпечує швидкі збирання та тестування. Однак не всі інструменти та мови програмування можуть бути легко інтегровані з Bazel, що може обмежувати його використання у деяких проєктах. Також Bazel має складну криву навчання та вимагає значних ресурсів для налаштування та підтримки, порівняно з аналогами [14].

Пакет для збирання та тестування Rush пропонує потужні можливості для управління залежностями та побудови монорепозиторіїв. Він оптимізує управління залежностями між пакетами, дозволяючи легко інсталювати, оновлювати та керувати залежностями для усіх пакетів у репозиторії. Проте використання пакету може бути складним у проєктах, де вже використовуються інші інструменти управління залежностями. Інтеграція Rush в наявні проєкти, особливо великі та складні, вимагає часу та може потребувати значних змін у структурі проєкту [15].

Система автоматизації збирання та управління залежностями Pants зорієнтована, у першу чергу, на високі швидкість та ефективність, особливо для великих кодових баз. Вона підтримує багато мов та платформ, забезпечуючи гнучкість для різноманітних проєктів. Дозволяє виконувати задачі компіляції та тестування паралельно, скорочуючи загальний час збирання. Легко розширюється та адаптується до специфічних потреб проєкту. Однак, порівняно з іншими інструментами, Pants може мати менш розгалужену спільноту

користувачів та обмежені ресурси для навчання та підтримки. До того ж, ефективна робота Pants у великих проєктах може вимагати значних обчислювальних ресурсів [16].

Пакет управління та повторного використання компонентів коду Bit зосереджений на перевикористанні компонентів та модульності. Він дозволяє легко масштабувати проєкти, додавати нові компоненти та адаптуватися до змін без переробки усього проєкту, забезпечує інтеграцію з хмарними сервісами та спільнотами, сприяючи співпраці та обміну компонентами. Але цей пакет не може бути ідеальним для дуже великих систем через потенційні обмеження в управлінні залежностями. До прикладу, Bit сильно залежить від своєї власної платформи для управління компонентами, що може створити залежність від конкретного сервісу та обмежити гнучкість у виборі технологічних рішень [17].

Система автоматизації збирання та управління залежностями Nx від Narwhal Technologies є потужним інструментом для управління монорепозиторіями з високою продуктивністю, особливо для Angular- та React-проєктів. Вона використовує інкрементальне збирання та кешування, що дозволяє підвищити швидкість розробки та зменшити час збирання. Підтримує інтеграцію з різними інструментами розробки та мовами програмування, забезпечуючи гнучкість у виборі технологій. Проте, для деяких специфічних сценаріїв розробки Nx може не надавати достатньої гнучкості або можливостей, що потребується в управлінні проєктами або процесах (наприклад, Nx має сильну інтеграцію з JavaScript- та TypeScript-системами, і якщо проєкт базується на інших мовах програмування або технологічних стеках, використання Nx може бути менш ефективним або навіть складним) [18].

Порівняльний аналіз інструментів управління монорепозиторіями наведено у таблиці 1.

Використовуючи дані з порівняльного аналізу інструментів для управління монорепозиторіями, наведених у роботах [21 – 22], а також уточнену інформацію за допомогою досліджень, поданих у роботах [12 – 18] нами запропоновано критеріальний аналіз різних інструментів, результати якого наведено у таблиці 2. Цей аналіз охоплює широкий спектр параметрів, від ефективності обробки залежностей до аспектів інтеграції з іншими інструментами розробки. Значна увага приділяється також оцінюванню масштабованості, гнучкості конфігурації та вартісним параметрам кожного інструменту. Такий підхід дозволяє забезпечити всебічне розуміння переваг та обмежень кожного рішення, сприяючи оптимізації вибору інструментів для ефективного управління багатопроєктними середовищами.

Для оцінювання інструментів управління монорепозиторіями обрано метод багатокритеріального аналізу, заснований на концепціях Вілларда Лероя Кейні та Говарда Райфа [19]. Застосування цієї методології дозволяє кількісно оцінити і порівняти різні інструменти на основі заздалегідь визначених критеріїв з використанням формули корисності:

$$U_i = \sum_{i=1}^{n} x_{il} \times P_l \tag{1}$$

де U_i представляє корисність i-го інструменту; x_{il} – відображає оцінку за l-тим критерієм; i = 1, …, 7 – кількість інструментів. l = 1, …, 10 – кількість критеріїв, а P_i – являє собою вагу l-го критерію.

Визначимо корисність кожного інструменту за допомогою формули (1):

U_1 = 0.7×1.0+0.6×0.9+0.8×0.8+0.7×0.7+0.8×0.6+0.7×0.9+0.6×0.5+0.7×0.6+0.6×0.5+0.7×0.4 = 4.51.

U_2 = 0.8×1.0+0.7×0.9+0.9×0.8+0.8×0.7+0.7×0.6+0.7×0.9+0.7×0.5+0.8×0.6+0.7×0.5+0.8×0.4 = 4.95.

U_3 = 0.9×1.0+0.9×0.9+0.6×0.8+0.6×0.7+0.5×0.6+0.9×0.9+0.8×0.5+0.6×0.6+0.8×0.5+0.6×0.4 = 4.83.

U_4 = 0.8×1.0+0.8×0.9+0.7×0.8+0.7×0.7+0.6×0.6+0.8×0.9+0.7×0.5+0.7×0.6+0.7×0.5+0.7×0.4 = 4.79.

$U_5 = 0.7\times1.0+0.9\times0.9+0.6\times0.8+0.6\times0.7+0.7\times0.6+0.9\times0.9+0.7\times0.5+0.6\times0.6+0.6\times0.5+0.6\times0.4 = 4.75.$

$U_6 = 0.6\times1.0+0.6\times0.9+0.7\times0.8+0.7\times0.7+0.8\times0.6+0.6\times0.9+0.6\times0.5+0.8\times0.6+0.7\times0.5+0.8\times0.4 = 4.61.$

$U_7 = 0.8\times1.0+0.8\times0.9+0.8\times0.8+0.8\times0.7+0.7\times0.6+0.8\times0.9+0.8\times0.5+0.9\times0.6+0.8\times0.5+0.7\times0.4 = 5.03.$

$$max U_l = U_7$$

Таблиця 1

Порівняльний аналіз інструментів управління монорепозиторіями

Інструмент	Переваги	Недоліки	Оптимальний сценарій використання
Lerna	Ефективне управління залежностями між пакетами. Автоматизація процесів публікації пакетів. Здатність керувати версіонуванням у складних проєктах.	Складний процес настроювання та обслуговування. Потенційні ускладнення при масштабуванні проєкту. Вимагає певного рівня технічного розуміння для ефективного використання.	Проєкти з кількома пакетами.
Yarn Workspaces	Централізоване управління залежностями. Підтримка спільного використання коду між проєктами. Оптимізація процесів інсталяції та оновлення пакетів.	Обмежена підтримка не-JavaScript мов програмування. Складності в інтеграції з деякими CI/CD інструментами. Може вимагати додаткового часу на налаштування у складних проєктах.	Монорепозиторії з акцентом на JavaScript.
Bazel	Висока продуктивність і ефективність збирання. Підтримка різноманітних мов програмування та платформ. Масштабування для великих та складних проєктів.	Висока крива навчання для нових користувачів. Складність у конфігурації та підтримці. Може бути перевантажуючим для малих чи середніх проєктів.	Великі монорепозиторії, корпоративний рівень.
Rush	Ефективне управління залежностями та збиранням проєктів. Інтеграція з широким спектром CI/CD інструментів. Підтримка ізоляції проєктів для запобігання конфліктів.	Потреба у глибокому розумінні конфігурації. Можливі складності з інтеграцією в наявні робочі процеси. Може вимагати значних ресурсів для великих проєктів.	Багатоплатформні монорепозиторії.
Pants	Оптимізована швидкість збирання для великих кодових баз. Підтримка різноманітних мов та фреймворків. Гнучкість у налаштуванні та використанні.	Складність у первинному налаштуванні та управлінні. Потенційні виклики при інтеграції з іншими системами. Може бути складним для розуміння для новачків.	Проєкти зі складною структурою та великою кодовою базою.
Bit	Зосереджений на реюзабельності та модульності компонентів. Підтримка декомпозиції та спільного використання коду між проєктами. Інтуїтивний інтерфейс та зручність управління.	Обмеження під час роботи з дуже великими монорепозиторіями. Може вимагати додаткового часу на інтеграцію з існуючими проєктами. Певні обмеження в управлінні складними залежностями.	Проєкти з потребою у високій реюзабельності компонентів.
Nx	Високі продуктивність і ефективність, особливо для Angular та React проєктів. Вбудовані можливості для масштабування та оптимізації проєктів. Підтримка модернізованих робочих процесів та інструментів.	Може вимагати часу для навчання та адаптації, особливо для новачків. Потенційні виклики при інтеграції з іншими мовами та фреймворками. Вимоги до ресурсів для ефективного використання у великих проєктах.	Проєкти на Angular та React, середні та великі монорепозиторії.

Таблиця 2

Оцінки інструментів управління монорепозиторіями

№	*l*-ті критерії, *l* = 1,*l*	Lerna	Yarn Workspaces	Bazel	Rush	Pants	Bit	Nx	вага *l*-го критерію
		1	2	3	4	5	6	7	
1	Продуктивність та ефективність	0,7	0,8	0,9	0,8	0,7	0,6	0,8	1
2	Масштабованість	0,6	0,7	0,9	0,8	0,9	0,6	0,8	0,9
3	Спрощення управління залежностями	0,8	0,9	0,6	0,7	0,6	0,7	0,8	0,8
4	Інтеграція з іншими інструментами	0,7	0,8	0,6	0,7	0,6	0,7	0,8	0,7
5	Гнучкість конфігурації	0,8	0,7	0,5	0,6	0,7	0,8	0,7	0,6
6	Безпека та надійність	0,7	0,7	0,9	0,8	0,9	0,6	0,8	0,9
7	Підтримка різноманітності мов та фреймворків	0,6	0,7	0,8	0,7	0,7	0,6	0,8	0,5
8	Інтуїтивність інтерфейсу та зручність використання	0,7	0,8	0,6	0,7	0,6	0,8	0,9	0,6
9	Підтримка спільноти та документації	0,6	0,7	0,8	0,7	0,6	0,7	0,8	0,5
10	Вартість та ліцензування	0,7	0,8	0,6	0,7	0,6	0,8	0,7	0,4

Отже, на основі проведених розрахунків корисності для різних інструментів управління монорепозиторіями, можна зробити висновок, що найкращим інструментом згідно з визначеними критеріями та їх вагами є Nx **(**U_7**)**. Цей інструмент демонструє найвищу загальну корисність зі значенням 5.03, що вказує на його ефективність у багатьох ключових аспектах, включаючи продуктивність, масштабованість, інтеграцію з іншими інструментами, а також підтримку спільноти (ком'юніті) та документації.

Крім високої загальної корисності, інструмент Nx має значні переваги, які роблять його ідеальним вибором для управління монорепозиторіями. Серед ключових переваг Nx можна виділити такі:

1. Висока продуктивність: Nx оптимізований для швидкої роботи з великими кодовими базами, забезпечуючи високу продуктивність навіть у складних проєктах.

2. Гнучке управління залежностями: Nx пропонує розширені можливості для ефективного управління залежностями, що дозволяє легко керувати складними взаємозв'язками між різними частинами проєкту.

3. Інтеграція з популярними фреймворками та мовами: Nx підтримує широкий спектр мов програмування та фреймворків, особливо ефективний у роботі з Angular та React.

4. Масштабування проєктів: Nx є ідеальним вибором для масштабування проєктів, забезпечуючи гнучкість та ефективність під час роботи з великими та багатофункціональними проєктами.

5. Підтримка спільноти та документація: Nx має активну спільноту розробників та добре підтримувану документацію, що сприяє легкій інтеграції та використанню інструменту.

6. Легкість впровадження та використання: Nx пропонує інтуїтивно зрозумілий інтерфейс та простоту в налаштуванні, роблячи його доступним для розробників різного рівня кваліфікації.

Ці та інші характеристики роблять Nx інструментом, який відповідає потребам сучасних розробників та команд, шукаючих ефективні та гнучкі рішення для управління монорепозиторіями [20].

Висновки

У контексті підвищення ефективності управління багатопроєктними середовищами у сфері розробки ПЗ було висунуто нову концепцію, що полягає у застосуванні комплексного багатокритеріального аналізу для оцінювання інструментів управління монорепозиторіями. Це надає можливість здійснювати детальне кількісне оцінювання різних інструментів, враховуючи заздалегідь визначені критерії та їх ваги, ефективно висвітлюючи переваги та обмеження кожного з них. Запропонований підхід відрізняється від наявних, насамперед, комплексним та системним оцінюванням інструментів управління монорепозиторіями. На відміну від традиційних підходів, які можуть покладатися на суб'єктивне судження або частковий аналіз, ця концепція використовує детальний аналіз за багатьма критеріями, забезпечуючи кількісну та об'єктивну основу оцінювання інструментів. Отже, цей підхід дозволяє отримати збалансовану оцінку, враховуючи різні чинники, вирішальні для ефективності розробки ПЗ у багатопроєктних середовищах, що приводить до більш обґрунтованого прийняття рішень щодо вибору та використання відповідних інструментів.

Здійснене дослідження також продемонструвало, що інструмент Nx виокремлюється як найбільш ефективний варіант, забезпечуючи розробників ПЗ необхідною інформацією щодо підвищення продуктивності та оптимізації процесів управління проєктами.

Важливо зауважити, що у загальному плані вибір інструментів залежить від специфічних вимог та контексту кожного проєкту, а тому наведені результати слід розглядати як орієнтир, а не як однозначне рішення.

СПИСОК ЛІТЕРАТУРИ

1. Achieving success in large, complex software projects [Electronic resource] / M. Bloch, S. Blumberg, J. Laartz // McKinsey Digital. – 2014. – Access mode : https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/achieving-success-in-large-complex-software-projects.

2. Managing complex software projects [Electronic resource] / Atlassian. – Access mode : https://www.atlassian.com/team-playbook/plays/ludicrously-complex-software-projects.

3. Transfer learning : a comprehensive introduction [Electronic resource] / A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, M. Abdul Azim // Journal of Big Data. – 2022. – № 102 (2022). – Access mode : https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00652-w.

4. Clean Code : A Handbook of Agile Software Craftsmanship [Electronic resource] / C. Martin. // Prentice Hall. – 2008. – Access mode: https://github.com/jnguyen095/clean-code/blob/master/Clean.Code.A.Handbook.of.Agile.Software.Craftsmanship.pdf.

5. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation [Electronic resource] / D. Farley J. Humble // Addison-Wesley Professional. – 2010. – Access mode : https://github.com/aaquresh/CITraining/blob/master/Ebooks/Continuous%20Delivery%20-%20Reliable%20Software%20Releases%20Through%20Build,%20Test%20And%20Deployment%20Automation.pdf.

6. Modular Web Design: Creating Reusable Components for User Experience Design [Electronic resource] / N. Curtis // Peachpit Press. – 2009. – Access mode : https://dokumen.pub/modular-web-design-creating-reusable-components-for-user-experience-design-and-documentation-9780321601353-0321601351-9780321638311-032163831x.html.

7. Yeping Z. The Interactive Design of Library Information Sharing in View of Network Communication Technology / Z. Yeping // Advanced Pattern Recognition Systems for Multimedia Data. – 2022. – Vol. 2022. – P. 1 – 14. – URL: https://doi.org/10.1155/2022/5342645.

8. Managing complex projects in multi-project environments [Electronic resource] / G. Hagan, D. Bower, N. Smith // Procs 27th Annual ARCOM Conference, 5 – 7 September 2011, Bristol, UK, Association of Researchers in Construction Management. – P. 787 – 796. Access mode : https://www.researchgate.net/publication/267423601_Managing_complex_projects_in_multi-project_environments/.

9. Monorepo in action for library maintenance [Electronic resource] / N. Nguen // Access mode: https://namnguyen.design/blog/2023-07-03-monorepo-in-action-for-library-maintenance-%F0%9F%92%9E.

10. Прус О. В. Використання графових нейронних мереж для автоматичної детекції залежностей між компонентами в монорепозиторіях / О. В. Прус, В. П. Майданюк // III Всеукраїнська науково-технічна конференція молодих вчених, аспірантів і студентів «Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації – 2023». – 2023. – С. 211 – 214.
11. Прус О. В. Аналіз основних принципів роботи з монорепозиторіями: особливості, переваги та недоліки / О. В. Прус, В. П. Майданюк // XVI міжнародна науково-практична конференція «Інформаційні технології та автоматизація – 2023». – 2023. – С. 267 – 270.
12. Documentation. Lerna [Electronic resource] : Access mode : https://lerna.js.org/.
13. Yarn Workspaces [Electronic resource] : Access mode : https://classic.yarnpkg.com/lang/en/docs/workspaces.
14. Bazel build system [Electronic resource] : Access mode : https://bazel.build/.
15. Rush: a scalable monorepo manager for the web [Electronic resource] : Access mode : https://rushjs.io/.
16. Pants 2: The ergonomic build system [Electronic resource] : Access mode : https://www.pantsbuild.org/.
17. Painless Monorepo Dependency Management with Bit [Electronic resource] / Z. Kochan // Access mode : https://bit.dev/blog/painless-monorepo-dependency-management-with-bit-l4f9fzyw/.
18. Nx: Smart, FastExtensibleBuild System. system [Electronic resource] : Access mode : https://nx.dev/.
19. Decisions with Multiple Objectives. Preferences and Value Tradeoffs / R. Keeney, R. Howard // Cambridge University Press. – 2014. – 592 p. – URL : https://doi.org/10.1017/CBO9781139174084.
20. Read B. 6 reasons why we chose Nx as our monorepo management tool [Electronic resource] – 2021 : Access mode : https://medium.com/purplebricks-digital/6-reasons-why-we-chose-nx-as-our-monorepo-management-tool-1fe5274a008e/.
21. Monorepo Part 3: The benchmark [Electronic resource] / R. Tahar // Medium. – 2022. – Access mode : https://engineering.combohr.com/part-3-the-benchmark-adb90f416151.
22. Monorepo vs Microrepo: How to Choose the Best Repository Structure for Your Code [Electronic resource] / K. Nirav // DEV. – 2023. – Access mode : https://dev.to/kanani_nirav/monorepo-vs-microrepo-how-to-choose-the-best-repository-structure-for-your-code-4pce.

Стаття надійшла до редакції 29.01.2024.
Стаття пройшла рецензування 06.03.2024.

Прус Олег Вікторович – аспірант кафедри програмного забезпечення.

Майданюк Володимир Павлович – канд. тех. наук, доцент кафедри програмного забезпечення.

Арсенюк Ігор Ростиславович – канд. тех. наук, доцент кафедри комп'ютерних наук, e-mail: igrosars@gmail.com.
Вінницький національний технічний університет.