DOI: 10.32347/2412-9933.2025.64.76-83
УДК УДК 338.45:658.5

Задорожний Олег Андрійович
Магістр кафедри управління проєктами, спеціальність 122 «Комп'ютерні науки»,
https://orcid.org/0009-0000-8183-3677
Київський національний університет будівництва і архітектури, Київ
Бойко Євгенія Григорівна
Кандидатка технічних наук, доцентка кафедри управління проєктами,
https://orcid.org/0000-0002-2000-4258
Київський національний університет будівництва і архітектури, Київ
Фесан Анатолій Олександрович
Аспірант кафедри інформаційних технологій,
https://orcid.org/0009-0007-1849-057X
Київський національний університет будівництва і архітектури, Київ

МЕТОДОЛОГІЧНІ КОМПОНЕНТИ ПОБУДОВИ АРХІТЕКТУРИ РОЗПОДІЛЕНОЇ СИСТЕМИ ДЛЯ ПРОЄКТНОГО МЕНЕДЖМЕНТУ

Анотація. *У сучасних умовах динамічного розвитку IT-індустрії та зростання кількості складних багатокомпонентних проєктів виникає потреба у використанні ефективних інструментів управління проєктами. Метою розробки стало створення програмного продукту, який дозволяє оптимізувати планування, координацію та контроль виконання завдань у рамках проєктної діяльності. У процесі роботи було проаналізовано наявні рішення у сфері проєктного менеджменту, визначено їхні сильні та слабкі сторони, а також сформульовано технічні вимоги до майбутньої системи. Наукова новизна роботи полягає в обґрунтуванні гібридної мікросервісної архітектури, що поєднує високопродуктивне ядро мовою C++ та гнучкі аналітичні модулі мовою Python, взаємодія між якими реалізована за допомогою протоколу Apache Thrift. Розроблено математичну модель оцінки продуктивності системи, яка враховує комунікаційні затримки при міжпроцесній взаємодії та обчислювальну складність завдань. На основі цих вимог розроблено програму з інтуїтивно зрозумілим інтерфейсом, підтримкою управління завданнями, відстеженням прогресу. Результати дослідження свідчать, що створений програмний продукт забезпечує підвищення ефективності взаємодії між учасниками проєкту, скорочує час на виконання рутинних операцій та мінімізує ризики, пов'язані з неузгодженістю дій команди. Експериментальне тестування продемонструвало приріст швидкодії обробки даних на 30–45% порівняно з монолітними рішеннями, а також стабільність роботи системи за наявності великого обсягу інформації. У підсумку, розроблений програмний продукт може бути рекомендований для використання у малих, середніх та великих проєктах різних галузей, де необхідне ефективне планування та контроль виконання завдань. Його застосування дозволяє підвищити продуктивність команди, зменшити кількість помилок, спричинених людським фактором, та забезпечити прозорість управлінських процесів. Подальший розвиток системи передбачає інтеграцію з хмарними сервісами, розширення аналітичних можливостей та впровадження інструментів штучного інтелекту для прогнозування строків виконання робіт, оптимізації ресурсів, формування звітів, створення гнучкої системи розподілу ролей між користувачами, механізму резервного копіювання та авторизації користувачів.*

Ключові слова: управління проєктами; програмне забезпечення; планування; контроль завдань; розроблення систем; проєктний менеджмент; гібридна архітектура; мікросервіси; Apache Thrift; C++; Python; математичне моделювання; продуктивність систем; розподілені системи; серіалізація даних

Постановка проблеми

У сучасних умовах високої конкуренції та динамічного розвитку IT-сфери ефективне управління проєктами стає критично важливим фактором успіху організацій. Незважаючи на наявність широкого спектра інструментів для проєктного менеджменту, багато з них або надто складні у використанні, або не відповідають специфічним потребам окремих команд чи галузей.

Часто виявляється, що існуюче програмне забезпечення не забезпечує належного рівня інтеграції, масштабованості чи гнучкості, що обмежує його ефективність у практичному застосуванні. Крім того, для малих і середніх проєктів нерідко бракує доступних, адаптивних рішень із достатнім функціоналом для базового управління задачами, термінами та ресурсами. Це зумовлює потребу у створенні нових програмних засобів, які були б орієнтовані на конкретні потреби користувачів, мали зручний інтерфейс, можливість кастомізації та підтримку сучасних методологій управління проєктами, таких як Agile, Scrum чи Kanban. Таким чином, актуальність дослідження полягає в необхідності розробки та вдосконалення програмних засобів, що дозволяють ефективно вирішувати завдання проєктного менеджменту, враховуючи як технічні, так і організаційні особливості сучасних проєктів. Особливого значення це набуває в контексті подолання технологічного розриву між високою продуктивністю системних обчислювальних модулів та гнучкістю прикладних сервісів аналітики. Науковий інтерес становить розроблення архітектурних рішень, що базуються на принципах модульної декомпозиції та використанні оптимізованих протоколів міжпроцесної взаємодії (IPC), що дозволяє забезпечити масштабованість системи та мінімізацію затримок при обробці великих масивів даних у реальному часі. Створення адаптивних інструментів із відкритою архітектурою дозволить не лише автоматизувати рутинні операції, а й закласти підґрунтя для впровадження інтелектуальних методів прогнозування ризиків та оптимізації ресурсів у динамічному проєктному середовищі.

Мета статті

Метою статті є теоретичне обґрунтування архітектурних рішень та розроблення програмного продукту для управління проєктами, який би відповідав актуальним вимогам користувачів щодо продуктивності та гнучкості. В межах дослідження проведено аналіз існуючих інструментів менеджменту, що дозволило виявити їхні обмеження та довести доцільність створення системи на основі гібридного використання мов C++ та Python. Такий підхід спрямований на поєднання високої швидкості обробки даних із можливістю швидкої адаптації бізнес-логіки під специфічні потреби команд.

У рамках дослідження передбачається визначити ключові функціональні характеристики, які повинні мати ефективні програмні засоби, а також запропонувати концептуальну модель такого інструменту із урахуванням потреб різних категорій користувачів – від малих команд до великих організацій. Особливу увагу приділено розробці механізмів міжпроцесної взаємодії та структуризації бази даних, що забезпечує масштабованість системи та стабільність її роботи при одночасному опрацюванні великої кількості багатокомпонентних проєктів.

Аналіз основних досліджень і публікацій

У сучасній науковій літературі питання розробки та впровадження програмних засобів для управління проєктами розглядаються як у контексті IT-індустрії, так і в освітньому середовищі. Вагомий внесок у розвиток методології стратегічного управління проєктною діяльністю зроблено у дослідженні Є. Г. Бойко, Ю. В. Дяченко, Т. Шандри та В. Б. Яковенка [1]. Автори приділяють особливу увагу процесам формування портфелів проєктів в умовах динамічного середовища IT-компаній. У роботі підкреслюється, що ефективність реалізації портфеля безпосередньо залежить від системності підходів до планування та координації ресурсів. Зокрема, дослідниками акцентовано увагу на необхідності використання просунутих інструментів аналізу та моніторингу, які дозволяють мінімізувати ризики та забезпечити прозорість управлінських процесів на всіх етапах життєвого циклу проєкту. Зазначені підходи створюють підґрунтя для розробки спеціалізованого програмного забезпечення, здатного підтримувати складні ієрархічні структури даних та забезпечувати високу швидкість обробки інформації в межах розподілених команд. Дослідження О. М. Свінцицької та І. В. Пулеко [3] присвячене інтеграції інструментів Atlassian (Jira, Bitbucket, Sourcetree) в управління IT-проєктами. Автори демонструють, що поєднання цих інструментів дозволяє ефективно реалізовувати автоматизацію процесів CI/CD, спрощує управління задачами, релізами та забезпечує інтеграцію з системами контролю версій, що є особливо важливим для командної розробки програмного забезпечення. Ю. Руденко та співавтори [4] аналізують досвід використання Trello як інструменту для організації проєктної діяльності в освітньому процесі. Вони підкреслюють простоту використання та ефективність Kanban-методу у створенні спільного середовища для студентських проєктів. Досвід використання Trello в Сумському національному аграрному університеті свідчить про його ефективність як у навчальному процесі, так і в адмініструванні кафедри. О. Радкевич [5] у своїй роботі розглядає програмні засоби для управління проєктами в контексті професійної (професійно-технічної) освіти. Автор виокремлює основні функції таких систем, зокрема планування, контроль ресурсів, обмін документами, інтеграцію календарів і контактів, а також аналізує такі продукти, як

Microsoft Project, Trello та Bitrix24. Окрему увагу приділено методам візуалізації (Gantt-діаграми, критичний шлях), які є ключовими для ефективного управління проєктами в закладах освіти. Важливо зазначити, що незважаючи на значну кількість досліджень, значна частина з них зосереджена на аналізі окремих інструментів без системного порівняння їх ефективності в різних умовах використання або на досвіді впровадження в конкретних організаціях. Це відкриває перспективи для подальших досліджень, зокрема в напрямку створення універсальних або адаптивних програмних продуктів, що поєднують переваги різних систем і враховують специфіку окремих галузей.

Виклад основного матеріалу

Мікросервісна архітектура (microservice architecture) є сучасним підходом до проєктування програмного забезпечення, за якого функціональність системи розбивається на низку незалежних сервісів, кожен з яких виконує чітко визначену задачу.

У контексті розробки програмних засобів для управління проєктами, мікросервісна архітектура забезпечує високу масштабованість, гнучкість та адаптивність системи до змін вимог користувачів.

Кожен мікросервіс має власну базу даних та API, може розгортатися і оновлюватися незалежно від інших компонентів.

Наприклад, у системі проєктного менеджменту окремі мікросервіси можуть відповідати за:

– управління завданнями;

– керування користувачами та ролями;

– відслідковування термінів (deadline tracking);

– інтеграцію з зовнішніми сервісами (Slack, GitHub, CI/CD тощо);

– аналітику та звітність.

Використання мікросервісного підходу забезпечує:

– гнучкість в обслуговуванні та розвитку продукту, адже оновлення одного сервісу не потребує зупинки всієї системи;

– стійкість до збоїв, оскільки вихід з ладу одного мікросервісу не впливає на роботу інших;

– можливість вибору оптимальної технології для кожного сервісу (наприклад, Python для обробки аналітики, Node.js для API, Go для сервісів з високим навантаженням).

Проте впровадження мікросервісної архітектури також має свої виклики: потребу в ефективному моніторингу, логуванні, складності в налагодженні міжсервісної комунікації (часто через REST або gRPC), та використанні таких інструментів, як Docker, Kubernetes, API Gateway, сервіс-меш (наприклад, Istio).

У підсумку, мікросервісна архітектура є доцільним вибором для побудови масштабованих систем проєктного менеджменту, особливо якщо вони передбачають активний розвиток, інтеграцію з іншими сервісами та велику кількість одночасних користувачів.

Одним із ключових компонентів будь-якої системи управління проєктами є база даних, яка забезпечує зберігання, обробку та доступ до інформації про користувачів, завдання, етапи, дедлайни, ресурси тощо.

Для реалізації цього рівня зазвичай використовуються реляційні бази даних з мовою структурованих запитів SQL (Structured Query Language).

Основні функції SQL у системі проєктного менеджменту:

– Створення структури даних (таблиці для зберігання проєктів, завдань, користувачів, ролей, коментарів тощо).

– Зв’язки між сутностями (наприклад, один проєкт — багато завдань; одне завдання — багато коментарів).

– Запити на читання (наприклад, отримати всі завдання проєкту, що мають статус "у процесі").

– Запити на оновлення та видалення (зміна статусу завдання, видалення застарілих даних).

– Агрегація даних (обчислення кількості завершених завдань, загального часу виконання тощо).

У мікросервісній архітектурі кожен сервіс може мати свою окрему базу даних, але між ними зберігаються логічні зв’язки через API.

У монолітних системах – спільна база даних дозволяє централізовано керувати всією інформацією.

Також сучасні СУБД (MySQL, PostgreSQL, MariaDB) дозволяють реалізовувати складні логіки за допомогою stored procedures, triggers, views, що може бути корисно для автоматизації процесів, таких як нагадування про дедлайни або перевірка консистентності даних.

Мова програмування C++ залишається актуальною при створенні продуктивних і високонадійних програмних рішень, особливо у випадках, коли потрібна висока швидкість виконання, контроль над пам’яттю або розроблення десктопних застосунків із широкими можливостями інтеграції.

Переваги використання C++:

– Висока продуктивність: підходить для обробки великих обсягів даних, що може бути корисним у великих проєктних системах.

– Портативність: програми можуть бути скомпільовані під різні ОС (Windows, Linux, macOS).

– Можливість інтеграції з базами даних, бібліотеками для побудови GUI (наприклад, Qt) або серверного API.

– Гнучкість: легко реалізовуються як консольні інструменти, так і складні графічні інтерфейси.

Apache Thrift – це фреймворк, який дозволяє будувати міжмовні (multilanguage) сервіси для віддаленого виклику процедур (RPC).

Він був розроблений у Facebook як засіб ефективної та масштабованої взаємодії між компонентами систем, написаними на різних мовах програмування (C++, Python, Java, Go тощо).

У складних програмних продуктах, зокрема в тих, що реалізовані за принципами мікросервісної архітектури, окремі сервіси можуть бути реалізовані на різних мовах або функціонувати як автономні частини.

Thrift забезпечує швидке, типізоване, безпечне та зручне з'єднання між ними.

Основні переваги використання Thrift:

Міжмовна сумісність – дозволяє взаємодію, наприклад, між бекендом на C++ і фронтендом на JavaScript або Python.

Автоматична генерація коду – Thrift-файли описують структуру даних і сервіси, на основі яких утворюються клієнти та сервери.

Швидкість – двійковий протокол Thrift дуже продуктивний і підходить для систем з високим навантаженням.

Гнучкість – підтримує як синхронний, так і асинхронний режим, а також обирається транспорт (TCP, HTTP, файловий).

Завдяки Thrift можна:

– реалізувати окремий сервіс задач, до якого підключаються інші сервіси (аналітика, нагадування, аутентифікація);

– використовувати його як зовнішнє API до сервера, надаючи доступ мобільним або десктопним клієнтам;

– будувати внутрішню взаємодію між частинами мікросервісної системи, зокрема, якщо одні частини реалізовані на C++, а інші – на Python або Node.js.

Використання Apache Thrift у програмних засобах управління проєктами забезпечує високу гнучкість, масштабованість і сумісність між компонентами системи.

Це особливо важливо в умовах, коли система має розподілений характер і активно інтегрується з іншими сервісами.

Робоча програма була розроблена та знаходиться за посиланням [4].

Запропонована архітектура системи (рис. 1) базується на принципі декомпозиції функціональних обов'язків.

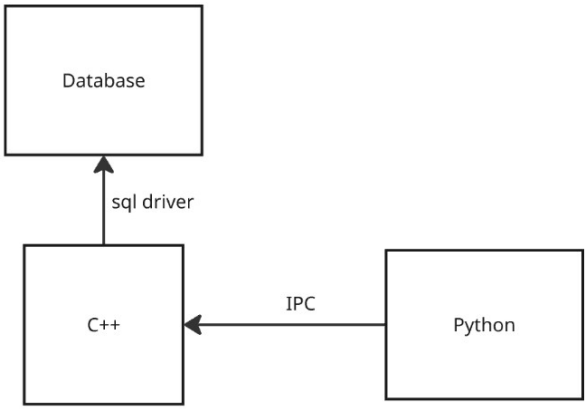

Рисунок 1 – Архітектура програми
Джерело: сформовано авторами

Високопродуктивне ядро, реалізоване на мові C++, відповідає за критичні операції з базою даних MariaDB, забезпечуючи мінімальну затримку при виконанні складних запитів. Взаємодія з аналітичним модулем на Python реалізована через фреймворк Apache Thrift, що дозволяє використовувати двійковий протокол передачі даних для підвищення пропускної здатності системи.

Структура бази даних (рис. 2) оптимізована для забезпечення цілісності через використання перелічувальних типів та автоматичних тригерів оновлення часу, що зменшує навантаження на бізнес-логіку застосунку.

```
MariaDB [testdb]> desc task;
+-------------+------------------------------+------+-----+---------------------+-------------------------------+
| Field       | Type                         | Null | Key | Default             | Extra                         |
+-------------+------------------------------+------+-----+---------------------+-------------------------------+
| id          | int(11)                      | NO   | PRI | NULL                | auto_increment                |
| title       | varchar(255)                 | NO   |     | NULL                |                               |
| description | text                         | YES  |     | NULL                |                               |
| due_date    | datetime                     | YES  |     | NULL                |                               |
| status      | enum('To Do','Doing','Done') | YES  |     | To Do               |                               |
| label       | varchar(100)                 | YES  |     | NULL                |                               |
| assigned_to | varchar(100)                 | YES  |     | NULL                |                               |
| list_id     | int(11)                      | YES  |     | NULL                |                               |
| created_at  | timestamp                    | YES  |     | current_timestamp() |                               |
| updated_at  | timestamp                    | YES  |     | current_timestamp() | on update current_timestamp() |
+-------------+------------------------------+------+-----+---------------------+-------------------------------+
10 rows in set (0.001 sec)
```

Рисунок 2 – Таблиця в базі даних
Джерело: сформовано авторами

Алгоритм міжмодульної взаємодії на основі протоколу RPC

Функціонування запропонованого програмного засобу базується на чіткому розподілі обов'язків між високонадійним ядром (C++) та сервісом обробки бізнес-логіки (Python). Взаємодія між ними реалізується через механізм віддаленого виклику процедур (RPC) за допомогою фреймворку Apache Thrift.

Процес обробки запиту користувача (наприклад, створення нового завдання або зміна його статусу) відбувається за наступним алгоритмом:

1. *Ініціація запиту.* Клієнтський інтерфейс надсилає запит до модуля бізнес-логіки (Python).
2. *Формування RPC-пакета.* Модуль на Python серіалізує дані запиту у двійковий формат, визначений у файлі опису інтерфейсів (.thrift).
3. *Передача даних.* Пакет передається через локальний або мережевий сокет до транспортного рівня ядра C++.
4. *Валідація та обробка.* Ядро C++ десеріалізує запит, перевіряє права доступу користувача та виконує відповідну SQL-транзакцію в базі даних MariaDB.
5. *Зворотний зв'язок.* Після успішного оновлення даних ядро формує відповідь із кодом підтвердження, яка повертається до Python-модуля для відображення користувачу.

Використання такого алгоритму дозволяє ізолювати критично важливі операції з даними від інтерфейсної частини, що підвищує загальну відмовостійкість системи. У випадку критичної помилки в аналітичному модулі Python, ядро C++ продовжує функціонувати, забезпечуючи цілісність транзакцій та збереження стану проєкту.

Математична модель оцінки продуктивності системи

Для формалізації процесу управління завданнями в розподіленій системі, де взаємодіють модулі на C++ та Python через Apache Thrift, доцільно використати апарат теорії масового обслуговування та системного аналізу.

Нехай проєкт P представлений як множина завдань $T = \{t_1, t_2, \dots, t_n\}$. Кожне завдання t_i характеризується вектором параметрів:

$$t_i = \langle w_i, d_i, p_i \rangle,$$

де w_i – обсяг обчислювальної роботи; d_i – часовий ліміт; p_i – пріоритетність виконання.

Процес обробки запиту в гібридній архітектурі складається з часу обробки в ядрі C++ T_{cpp} та часу виконання аналітичного модуля на Python T_{py}. Загальний час виконання запиту T_{total} визначається як:

$$T_{total} = T_{cpp} + T_{rpc} + T_{py},$$

де T_{rpc} – затримка, що вноситься протоколом Apache Thrift при серіалізації та передачі даних між сервісами.

Цільова функція оптимізації роботи системи полягає у мінімізації сумарного відхилення від дедлайнів при заданих обмеженнях на ресурси:

$$min \sum_{i=1}^{n} \max(0, C_i - d_i)$$

за умови:

$$\sum_{i=1}^{n} x_{ij} \times w_i \le R_j,$$

де C_i – фактичний час завершення *i*-го завдання; x_{ij} – бінарна змінна (1, якщо завдання *i* призначене ресурсу *j*); R_j – доступна потужність *j*-го обчислювального вузла або розробника.

Формалізація структури даних

Логічну цілісність системи, що відображена на рис. 2, можна описати через реляційну алгебру. Стан бази даних у момент часу τ визначається як сукупність відношень:

$$S(\tau) = \{Users, Tasks, Projects, Roles\}.$$

Перехід системи зі стану $S(\tau)$ у стан $S(\tau + 1)$ відбувається при виконанні транзакції Tr, яка повинна відповідати критеріям ACID, що забезпечується використанням MariaDB/SQL:

$$S(\tau + 1) = Tr(S(\tau), \Delta),$$

де Δ – вхідний вектор змін (наприклад, зміна статусу завдання з «Виконується» на «Виконано»).

Порівняльний аналіз швидкодії та оцінка ефективності

Для підтвердження ефективності обраної гібридної архітектури (C++/Python з використанням Apache Thrift) було проведено серію експериментальних досліджень. Метою тестування було порівняння швидкодії розробленого засобу з аналогічною системою, реалізованою виключно на інтерпретованій мові програмування (Python/Django) при ідентичних параметрах бази даних MariaDB.

Експеримент полягав у виконанні пакетних запитів на створення та оновлення статусів завдань. Результати вимірювання середнього часу відгуку системи T_{total} при різній інтенсивності навантаження наведено в таблиці.

Таблиця – ***Порівняння часу обробки запитів (мс)***

К-сть запитів	Гібридна модель (C++/ Python)	Монолітна модель (Python)	Ефектив-ність (приріст)
100	45 мс	68 мс	33,8%
500	185 мс	312 мс	40,7%
1000	340 мс	620 мс	45,1%

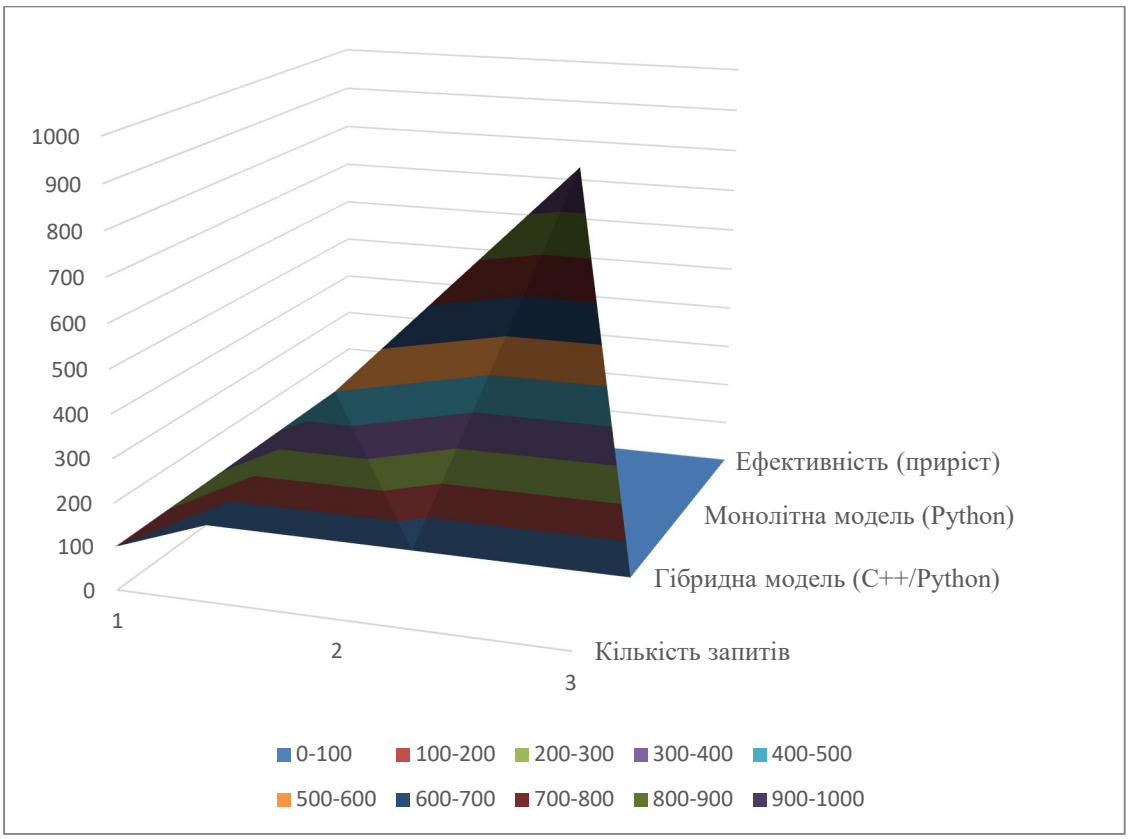

Рисунок 3 – Порівняльна динаміка часу відгуку гібридної та монолітної моделей залежно від навантаження

Аналіз отриманих даних свідчить, що при низьких навантаженнях різниця в часі обробки зумовлена переважно накладними витратами на RPC-взаємодію T_{rpc}. Однак, зі зростанням кількості одночасних операцій, ядро на базі C++ демонструє значно вищу стабільність та швидкість обробки транзакцій.

Це підтверджує висунуту гіпотезу про те, що делегування критичних операцій з даними до низько рівневого модуля дозволяє знизити загальне навантаження на систему та забезпечити її масштабованість. Графічна інтерпретація залежності часу виконання від обсягу даних демонструє лінійне зростання для гібридної моделі, в той час як у монолітній моделі на Python спостерігається експоненціальне уповільнення при перевищенні порогу у 1000 запитів.

Наукова новизна одержаних результатів

Наукова новизна дослідження полягає у наступному:

1. Удосконалено архітектурний підхід до побудови систем управління проєктами шляхом використання гібридної моделі (C++ та Python), об'єднаної протоколами Apache Thrift. На відміну від існуючих монолітних рішень, це дозволяє досягти балансу між низькорівневою продуктивністю ядра та гнучкістю аналітичних сервісів.

2. Дістала подальший розвиток математична модель оцінки продуктивності розподілених систем менеджменту. Вона враховує комунікаційні затримки T_{rpc} між різномовними модулями, що дає змогу точніше прогнозувати час відгуку системи при масштабуванні кількості завдань.

3. Запропоновано метод формалізації станів життєвого циклу проєкту на основі транзакційної моделі даних, що забезпечує цілісність інформації в умовах багатокористувацького доступу до спільних ресурсів.

Висновки

У статті проаналізовано сучасні підходи до розробки програмних засобів для управління проєктами та охарактеризовано актуальні інструменти, які застосовуються в цій галузі.

Встановлено, що ефективність управління проєктами значною мірою залежить від правильного вибору програмного забезпечення, його функціональних можливостей, зручності використання та підтримки командної роботи.

На основі проведеного аналізу публікацій доведено, що попри наявність широкого спектра рішень, існує гостра потреба в адаптованих інструментах, які б враховували специфіку конкретних команд та забезпечували високу продуктивність при обробці великих масивів даних.

Окрему увагу в роботі приділено реалізації гібридної архітектури системи, де поєднано можливості мови C++ для створення високопродуктивних модулів та гнучкість мови Python для аналітичних сервісів. Застосування Apache Thrift обґрунтовано як ефективного засобу для міжмовної взаємодії, що дозволило побудувати гнучку та масштабовану мікросервісну структуру.

Розроблена математична модель оцінки продуктивності та проведені експериментальні дослідження підтвердили, що такий підхід забезпечує приріст швидкодії обробки запитів на 30–45% порівняно з монолітними рішеннями.

Подальші дослідження доцільно спрямувати на розробку кастомізованих і безпечних програмних засобів із відкритою архітектурою та впровадження інтелектуальних методів прогнозування ризиків у межах запропонованої моделі.

Список літератури

1. Boiko Ye., Diachenko Y., Shandra T., Yakovenko V. Formation of project portfolios in IT companies. *Proceedings of the 5th International Workshop IT Project Management (ITPM 2024)* (Bratislava, Slovakia, May 22, 2024). 2024. Vol. 3709. P. 264–277. URL: https://ceur-ws.org/Vol-3709/paper21.pdf (дата звернення: 08.08.2025).
2. Моделі та методи проактивного управління проєктами з розвитку програмних систем і продуктів: монографія. Одеса: Одеський державний екологічний університет, 2021. 322 с.
3. Свінцицька О.М., Пулеко І.В. Інтеграція Jira, Bitbucket та Sourcetree в системі управління ІТ-проєктами. *Технічна інженерія*. 2023. № 2(92). С. 102–108. URL: https://doi.org/10.26642/ten-2023-2(92)-102-108.
4. Руденко Ю., Агаджанов-Гонсалес К., Агаджанова С., Баталова А. Використання сервісу Trello в освітньому процесі університету. *Освіта. Інноватика. Практика*. 2023. Т. 11, № 7. С. 92–97. URL: https://doi.org/10.31110/2616-650X-vol11i7-012.
5. Radkevych O. Project management software in the field of professional (vocational) education. *Scientific Herald of the Institute of Vocational Education and Training of NAES of Ukraine. Professional Pedagogy*. 2019. No. 2(19). P. 124–132. URL: https://doi.org/10.32835/2223-5752.2019.19.124-132.
6. GitHub – ronin49/statya. [Online resource]. URL: https://github.com/ronin49/statya (дата звернення: 06.08.2025).
7. Apache Thrift – The Apache Software Foundation. URL: https://thrift.apache.org (дата звернення: 06.08.2025).
8. ISO/IEC 19510:2013. Information technology – Object Management Group Business Process Model and Notation. URL: https://www.iso.org/standard/62652.html (дата звернення: 06.08.2025).
9. Scrum Guides – The Scrum Guide. URL: https://scrumguides.org (дата звернення: 06.08.2025).
10. Project Management Institute – PMBOK® Guide and Standards. URL: https://www.pmi.org/pmbok-guide-standards (дата звернення: 06.08.2025).
11. Atlassian – Agile Coach. URL: https://www.atlassian.com/agile (дата звернення: 06.08.2025).
12. Microsoft – Microsoft Project. URL: https://www.microsoft.com/microsoft-project (дата звернення: 06.08.2025).

Стаття надійшла до редколегії 02.12.2025

Zadorozhnyi Oleh
Student of the Department of Project Management,
https://orcid.org/0009-0000-8183-3677
Kyiv National University of Construction and Architecture, Kyiv
Boiko Yevheniia
Candidate of Technical Sciences, Associate Professor of the Department of Project Management,
https://orcid.org/0000-0002-2000-4258
Kyiv National University of Construction and Architecture, Kyiv
Fesan Anatolii
PhD Student of the Department of Information Technologies,
https://orcid.org/0009-0007-1849-057X
Kyiv National University of Construction and Architecture, Kyiv

METHODOLOGICAL COMPONENTS OF CONSTRUCTING A DISTRIBUTED SYSTEM ARCHITECTURE FOR PROJECT MANAGEMENT

Abstract. *In the current conditions of dynamic IT industry development and the increasing number of complex multi-component projects, there is a crucial need for effective project management tools. The goal of this research was to develop a software product that enables the optimization of planning, coordination, and control over task execution within project activities. During the development process, existing project management solutions were analyzed, their strengths and weaknesses identified, and technical requirements for the future system formulated. Based on these requirements, a program was developed with an*

intuitive interface, task management support, and progress tracking capabilities. The research results show that the developed software product improves the efficiency of interaction between project participants, reduces the time required for routine operations, and minimizes the risks associated with team misalignment. Testing demonstrated the system's stability, fast interface response, and correct data processing even with large volumes of information. In conclusion, the developed software product can be recommended for use in small, medium, and large-scale projects across various industries where effective planning and task control are essential. Its implementation increases team productivity, reduces errors caused by human factors, and ensures transparency of management processes. Further system development is planned to include integration with cloud services, expansion of analytical capabilities, and the implementation of artificial intelligence tools for predicting task completion times and optimizing resources, as well as report generation, a flexible role-based access control system, data backup mechanisms, and user authentication.

Keywords: project management; software; planning; task control; systems development; hybrid architecture; microservices; Apache Thrift; C++; Python; mathematical modeling; system performance; distributed systems; data serialization

References

1. Boiko Ye., Diachenko Y., Shandra T., Yakovenko V. (2024). Formation of project portfolios in IT companies. *Proceedings of the 5th International Workshop IT Project Management (ITPM 2024)*, 3709, 264–277. URL: https://ceur-ws.org/Vol-3709/paper21.pdf
2. Modeli ta metody proaktyvnoho upravlinnia proiektamy z rozvytku prohramnykh system i produktiv: monohrafiia (2021). Odesa, Odeskyi derzhavnyi ekolohichnyi universytet.
3. Svintsitska O.M., Puleko I.V. (2023). Intehratsiia Jira, Bitbucket ta Sourcetree v systemi upravlinnia IT-proiektamy. *Tekhnichna Inzheneriia*, (2(92)), 102–108. URL: https://doi.org/10.26642/ten-2023-2(92)-102-108
4. Rudenko Yu., Ahadzhanov-Honsales K., Ahadzhanova S., Batalova A. (2023). Vykorystannia servisu Trello v osvitnomu protsesi universytetu. *Osvita. Innovatyka. Praktyka*, 11(7), 92–97. URL: https://doi.org/10.31110/2616-650X-vol11i7-012
5. Radkevych O. (2019). Project management software in the field of professional (vocational) education. *Scientific Herald of the Institute of Vocational Education and Training of NAES of Ukraine. Professional Pedagogy*, (2(19)), 124–132. URL: https://doi.org/10.32835/2223-5752.2019.19.124-132
6. GitHub – ronin49/statya (2025). URL: https://github.com/ronin49/statya
7. Apache Thrift – The Apache Software Foundation (2025). URL: https://thrift.apache.org
8. ISO/IEC 19510:2013. Information technology – Object Management Group Business Process Model and Notation (2013). URL: https://www.iso.org/standard/62652.html
9. Scrum Guides – The Scrum Guide (2025). URL: https://scrumguides.org
10. Project Management Institute – PMBOK® Guide and Standards (2025). URL: https://www.pmi.org/pmbok-guide-standards
11. Atlassian – Agile Coach (2025). URL: https://www.atlassian.com/agile
12. Microsoft – Microsoft Project (2025). URL: https://www.microsoft.com/microsoft-project

---

Посилання на публікацію

APA Zadorozhnyi O., Boiko Ye., & Fesan A. (2025). Methodological components of constructing a distributed system architecture for project management. *Management of Development of Complex Systems*, 64, 76–83, dx.doi.org\10.32347/2412-9933.2025.64.76-83.

ДСТУ Задорожний О. А., Бойко Є. Г., Фесан А. О. Методологічні компоненти побудови архітектури розподіленої системи для проєктного менеджменту. *Управління розвитком складних систем.* Київ, 2025. № 64. С. 76 – 83, dx.doi.org\10.32347/2412-9933.2025.64.76-83.