https://doi.org/10.31891/2219-9365-2025-84-11
УДК 004.414.4:005.94

ГУДАКОВ Денис
Київський національний університет імені Тараса Шевченка
https://orcid.org/0009-0000-3003-842X
e-mail: denysgud@gmail.com

АРХІТЕКТУРА ІНСТРУМЕНТІВ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ: ПАТЕРНИ ІНТЕГРАЦІЇ ТА КЕРУВАННЯ ЗНАННЯМИ

У статті досліджується проблематика фрагментації знань у сучасних процесах розробки програмного забезпечення, де використання великої кількості спеціалізованих інструментів призводить до ізоляції даних. Автором проаналізовано фундаментальну архітектурну невідповідність між операційними системами управління завданнями (наприклад, Jira) та системами баз знань (наприклад, Confluence). Розглянуто три основні архітектурні парадигми інтеграції: уніфіковані платформи (GitLab), тісно пов'язані екосистеми (Atlassian) та взаємопов'язані екосистеми (Microsoft 365). Визначено роль технічних патернів інтеграції (API-центричних, подієво-орієнтованих та опосередкованих) як основних каналів для забезпечення «потоку знань» та трансформації ефемерних артефактів у довготривалі знання.

Ключові слова: архітектура інструментів управління ІТ-проєктами, інтеграція програмних систем, управління знаннями, патерни інтеграції, подієво-орієнтовані моделі, інформаційні технології підтримки проєктного менеджменту, оптимізація процесів розробки програмного забезпечення.

GUDAKOV Denys
Kyiv National University named after Taras Shevchenko

ARCHITECTURE OF IT PROJECT MANAGEMENT TOOLS: PATTERNS OF INTEGRATION AND KNOWLEDGE MANAGEMENT

The article focuses on the paradox within modern IT project management. It examines how, despite software development being an inherently knowledge-intensive process with an unprecedented abundance of productivity tools, critical project knowledge remains fragmented and isolated. The research addresses how organizations' simultaneous use of version control systems, bug trackers, wikis, and communication platforms causes valuable data - such as documentation, source code, and error reports - to reside in disconnected repositories. This study argues that the integration architecture uniting these disparate tools is a fundamental mechanism that actively shapes the efficiency of knowledge management processes. The paper identifies a core architectural mismatch between tool stacks and necessary knowledge flows, drawing a fundamental distinction between two classes of tools: Task Management Systems (e.g., Jira), which focus on the operational and ephemeral state of the project, and Knowledge Base Systems (e.g., Confluence), designed for the long-term structuring of permanent knowledge. The key challenge lies in bridging the temporal gap between these tools to transform ephemeral artifacts into reflective, long-term knowledge artifacts.

The research analyzes three primary architectural paradigms that address this integration issue such as unified platforms (e.g., GitLab), tightly coupled ecosystems (e.g., Atlassian) and interconnected ecosystems (e.g., Microsoft 365). The study further evaluates the technical integration patterns that function as channels for this "knowledge flow," including API-Centric Integration, Event-Driven Architecture and Middleware Integration.

The research concludes that integration architecture is not merely passive infrastructure but an active, formative system governing the knowledge lifecycle. The choice between unified, coupled, or interconnected paradigms dictates an organization's ability to automatically capture, semantically enrich, and efficiently retrieve critical project knowledge. Future research directions include the impact of semantic integration, the role of Artificial Intelligence (LLMs) in active knowledge synthesis, and the "democratization of integration" through Low-Code/No-Code platforms.

Keywords: Architecture of IT project management tools, software system integration, knowledge management, integration patterns, event-driven models, information technologies for project management support, optimization of software development processes.

Стаття надійшла до редакції / Received 22.09.2025
Прийнята до друку / Accepted 27.10.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

В основі сучасної розробки програмних продуктів лежить ефективна акумуляція та використання інтелектуальних активів. Проте, в практиці управління ІТ-проєктами виникла суттєва суперечність, яка полягає у тому, що попри наявність широкого арсеналу цифрових інструментів для підвищення продуктивності, накопичений досвід та критично важливі проєктні дані залишаються дезінтегрованими та розпорошеними.

Складність заключається в тому, що сучасний інструментальний стек проєкту, який неминуче включає окремі платформи для коду, відстеження завдань, документації та комунікації, на практиці часто функціонує як набір ізольованих "силосів". Це створює глибокі інформаційні розриви: документація втрачає зв'язок із реальним станом коду, а управлінські рішення приймаються без повного контексту виконання. Як наслідок, втрачається наскрізна прозорість життєвого циклу розробки.

Таким чином, актуальним науково-практичним завданням стає подолання цього структурного розриву. Ключова проблема лежить не в площині вибору окремих програмних продуктів, а у сфері забезпечення їх когерентної системної взаємодії. Дослідження та проєктування архітектури, що забезпечує ефективне сполучення цих інструментів, є вирішальним. Саме така архітектурна основа дозволить перетворити окремі масиви даних на зв'язну систему знань, забезпечуючи їх вільну циркуляцію та повторне використання в рамках IT-проєкту.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Наукові праці зарубіжних та вітчизняних авторів засвідчують, що управління знаннями є предметом активних досліджень. Так, роботи [3,4] описують фундаментальні цикли управління знаннями та фактори, що впливають на їх успішність. У контексті розробки програмного забезпечення, дослідження [1] аналізує управління знаннєвими ресурсами у гнучких (Agile) методологіях. На практичному рівні, публікації [2] (на прикладі Atlassian) підтверджують проблему розриву між інструментами (Jira, Confluence) та необхідність їх глибокої інтеграції. Суміжні дослідження [5] розглядають застосування новітніх технологій, як-от штучного інтелекту, в управлінні IT-проєктами.

Водночас, незважаючи на наявні напрацювання, залишається недостатньо дослідженою саме архітектурна складова. Існуючі роботи зосереджені або на загальних процесах управління знаннями [3, 4], або на специфіці гнучких методологій [1], або пропонують точкові рішення для конкретних інструментів [2]. Проте, бракує системного аналізу того, як фундаментальний вибір архітектури інтеграції безпосередньо формує або обмежує "потік знань" між інструментами.

Таким чином, дана стаття спрямована на заповнення цієї прогалини, розглядаючи патерни інтеграції не як пасивну інфраструктуру, а як активний механізм реалізації циклу управління знаннями в IT-проєктах.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою статті є дослідження архітектури інтеграції інструментів управління IT-проєктами як активної системи управління знаннями, а не лише пасивної інфраструктури.

Для досягнення поставленої мети у роботі вирішуються такі завдання:

- провести аналіз проблематики фрагментації проєктних знань, що виникає через архітектурну невідповідність між стеками інструментів та необхідними потоками знань;
- визначити фундаментальний розрив між операційними системами управління завданнями та системами баз знань, а також архітектурні виклики, пов'язані з цим;
- порівняти основні архітектурні парадигми з точки зору їхнього впливу на управління знаннями;
- систематизувати ключові технічні патерни інтеграції та дослідити їхню роль як каналів для потоку знань;
- встановити відповідність між категоріями інструментів, патернами інтеграції та артефактами знань, що створюються у життєвому циклі IT-проєкту.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Сучасна розробка програмного забезпечення є за своєю суттю процесом, що інтенсивно використовує знання. Управління IT-проєктами в цьому середовищі дедалі більше стикається з парадоксом: незважаючи на безпрецедентну кількість інструментів, призначених для підвищення продуктивності, критично важливі проєктні знання залишаються фрагментованими та ізольованими. Організації одночасно використовують системи контролю версій, баг-трекери, вікі та платформи для спілкування. В результаті, цінні дані, такі як документація, вихідний код та звіти про помилки, опиняються у окремих репозиторіях і залишаються від'єднаними один від одного.

Таким чином, архітектура інтеграції, яка об'єднує ці розрізнені інструменти управління проєктами, стає фундаментальним механізмом, що активно формує та визначає ефективність процесів управління знаннями в рамках проєкту [1]. Вибір між API-центричним, подієво-орієнтованим, опосередкованим чи уніфікованим архітектурним підходом безпосередньо диктує швидкість, відстежуваність, доступність та якість кодифікації проєктних знань. Технічні патерни інтеграції функціонують як основні канали для "потоку знань" (knowledge flow), перетворюючи пасивні сховища даних на активну, взаємопов'язану систему знань. Таким чином, центральна проблема полягає не в дефіциті інструментів чи даних, а в архітектурній невідповідності між інструментальними стеками та необхідними потоками знань.

В основі цієї проблеми лежить фундаментальна різниця між двома класами інструментів. Перша категорія - це системи управління завданнями (наприклад, Jira), які фокусуються на операційному стані проєкту та активно використовуються наразі. Якщо вони ізольовані, виникає "ефект чорної скриньки", коли рівень планування втрачає зв'язок з рівнем виконання. Друга категорія - це системи баз знань (наприклад, Confluence), мета яких - довгострокове зберігання та структурування знань. Вони фіксують дані проєкту, орієнтуючись на перманентні знання. Таким чином, виникає розрив у часі: інструменти завдань орієнтовані

на теперішній час, а бази знань - на постійний. Ключовим архітектурним викликом є інтеграція, яка має стати мостом через цей розрив. Вона повинна забезпечити трансформацію темпоральності знань - перетворити ефемерний артефакт (наприклад, "завдання закрито") у довготривалий, рефлексивний артефакт (наприклад, "накопичено знання" або "документацію оновлено"). Яскравим прикладом є екосистема Atlassian, де Jira (завдання) та Confluence (знання) були свідомо створені як доповнюючі, а їхня глибока інтеграція дозволяє перетворювати оперативні дані на стійкі артефакти знань [2].

Існують три основні архітектурні парадигми, що вирішують цю проблему. По-перше, це уніфікована платформа, як-от GitLab, що пропонує єдиний інтерфейс для зменшення фрагментації інструментів, що дозволяє пов'язувати код, завдання та документацію простими посиланнями. По-друге, це тісно пов'язана екосистема, як Atlassian. Вона складається з незалежних програмних продуктів, що вимагають ретельного налаштування. Двонаправлена інтеграція (наприклад, посилання з Jira на Confluence і назад) перетворює окремі артефакти на взаємно обізнані, створюючи єдине джерело для документації, яка динамічно пов'язана з живим станом завдань. По-третє, це взаємопов’язана екосистема, як Microsoft 365. Це навмисно фрагментований набір спеціалізованих інструментів: Azure DevOps, Microsoft Planner, Microsoft Teams тощо. З’єднуючим елементом цієї системи служить єдиний API-ендпоінт - Microsoft Graph. Інтеграція відбувається не на рівні вбудованих функцій, а на рівні API, де Teams виступає як агрегатор. Наприклад, додаток Azure Boards надсилає сповіщення в канали Teams через сервісні хуки. Цей підхід гнучкий, але перекладає тягар синтезу знань на інтегратора.

Вибір архітектурної парадигми диктує доступні технічні патерни інтеграції, які і є каналами для знань. API-центрична інтеграція - це синхронний патерн "запит-відповідь", який зазвичай реалізується через REST API. Це Pull-модель, що ідеально підходить для забезпечення відстежуваності артефактів знань. Наприклад, BI-система може запитати (Pull) дані про статус завдань з Planner API. Подієво-орієнтована архітектура (EDA) - це асинхронний патерн опублікувати/підписатися (publish/subscribe), що часто реалізується через вебхуки (Webhooks). Це push-модель, що є основою для управління знаннями в реальному часі. Вона забезпечує повний цикл автоматизованого потоку знань у DevOps (рис.1):

1) розробник виконує git push;
2) ця подія миттєво запускає вебхук;
3) вебхук активує конвеєр CI/CD для збірки;
4) якщо збірка зазнає невдачі, система CI/CD генерує подію збою;
5) інший вебхук ловить цю подію і миттєво публікує сповіщення про збій у канал Slack;
6) це сповіщення стає артефактом знання, що спонукає до створення звіту про помилку (завдання).

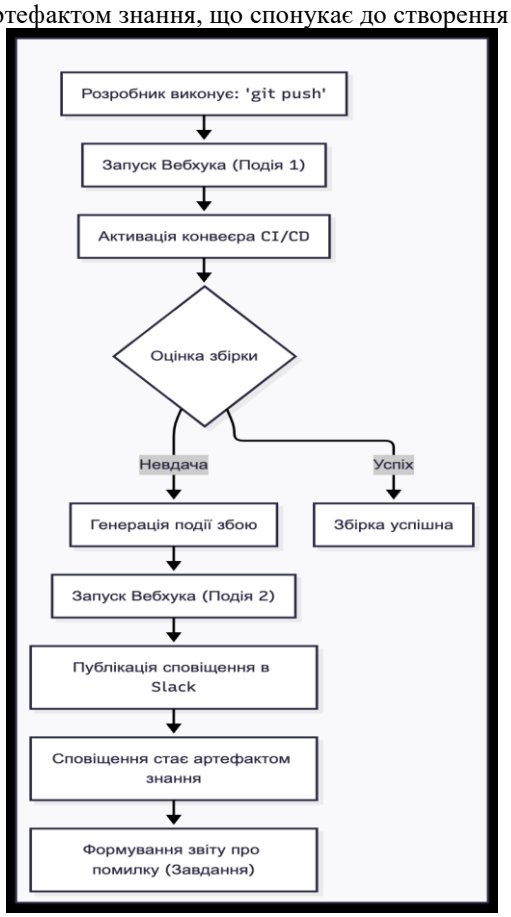

Рис. 1. Алгоритм автоматизованого потоку знань

Нарешті, опосередкована інтеграція (Middleware або ESB) вводить проміжний шар, щоб уникнути "спагеті-архітектури". ESB діє як централізований комунікаційний шар, що бере на себе трансформацію даних та маршрутизацію повідомлень. У контексті управління знаннями, Middleware виконує функцію семантичного медіатора або перекладача. Наприклад, спеціальний додаток може синхронізувати інциденти між Jira Service Management та ServiceNow, коректно перекладаючи артефакти знань, такі як Comments, Description та Attachments, між двома системами.

Ці патерни інтеграції безпосередньо впливають на життєвий цикл артефактів знань. Проєктні знання існують у явних, кодифікованих формах (артефактах), таких як Software Development Plan (план розробки), Bug tracking systems (звіти про помилки) або, що особливо важливо, Lessons Learned Database (база накопичених знань). Управління знаннями — це динамічний цикл, у якому архітектура інтеграції є активним виконавцем. Синтез компонентів таких, як категорії інструментів, патерни інтеграції та артефакти знань, у життєвому циклі IT-проєктів узагальнено у таблиці 1.

Таблиця 1.

Відповідність категорій інструментів, патернів інтеграції та артефактів знань

Категорія інструментів	Репрезентативні продукти	Патерни інтеграції	Артефакти знань	Ключові метрики
Вимоги та знання	Confluence, Notion, Google Workspace, Miro	API Gateway, Webhooks	PRD/BRD, ADR, глосарій, ретроспектива	Час узгодження PRD, швидкість онбордингу
Планування й трекінг	Jira/YouTrack/Azure Boards, Trello/Asana	Webhook-Oriented, Event-Driven	Backlog, story map, DoD	Стабільність WIP, точність оцінок
Код і збірки	GitHub, GitLab, Bitbucket; Jenkins, GitHub Actions, GitLab CI	API, Webhooks, GitOps	PR/MR, артефакти збірок, SBOM	Lead time, пропускна здатність PR/MR
Якість і безпека	Xray/Zephyr, TestRail; SAST/DAST, Cypress/Playwright	Quality Gates у pipeline	Тест-кейси, звіти про дефекти, політики	% регрес- автоматизації, частка невдалих змін
Доставка й експлуатація	Argo CD, Flux, Helm; Kubernetes; Terraform	GitOps, Event-Driven Ops	Маніфести, плейбуки, пост-мортеми	Частота деплоїв, MTTR, інциденти/місяць
Контроль і аналітика	Prometheus/Grafana, ELK/OpenSearch, Sentry; Power BI/Tableau	API ingestion, alert webhooks	Дашборди SLO/SLA, аналітика	p95/p99-латентність, error budget burn

ВИСНОВКИ З ДАНОГО ДОСЛУДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Отже, архітектура інтеграції інструментів управління IT-проєктами не є пасивною інфраструктурою, а слугує активною, формуючою системою управління знаннями. Вибір між уніфікованою (GitLab), тісно пов'язаною (Atlassian) або цілісною (Microsoft) архітектурною парадигмою має глибокі наслідки для здатності організації автоматично захоплювати, семантично збагачувати, надійно зберігати та ефективно поширювати критичні проєктні знання. Майбутні дослідження в цій галузі можуть бути зосереджені на впливі семантичної інтеграції та штучного інтелекту (зокрема, LLM) для активного синтезу знань, викликах "cloud-native" та гібридних архітектур, а також на феномені "демократизації інтеграції" через платформи Low-Code/No-Code.

References

1. Raquel Ouriques. (2023). Managing Knowledge Resources in Agile Software Development. Doctoral Dissertation in Software Engineering. Retrieved from: https://www.diva-portal.org/smash/get/diva2:1789238/FULLTEXT01.pdf
2. Sudlow, B. (2020, August 31). Link Jira issues to Confluence pages automatically. Atlassian. Retrieved from: https://www.atlassian.com/blog/confluence/link-jira-issues-to-confluence-pages-automatically.
3. Rabelo, J.D., & Conte, T.U. (2018). Influence Factors for Knowledge Management Initiatives - A Systematic Mapping Study. International Conference on Enterprise Information Systems. https://pdfs.semanticscholar.org/ab5b/c7600e86691b63921b264d58adc3778c0988.pdf
4. Dhanashree B. (2025, May 22). Mastering the Knowledge Management Cycle: A Complete Guide. Retrieved from: https://www.alltius.ai/glossary/knowledge-management-cycle
5. GUDAKOV Д., & KOLODINSKA Я. (2025). THE APPLICATION OF ARTIFICIAL INTELLIGENCE FOR MANAGING HUMANCENTRED PROJECT OF INFORMATION TECHNOLOGIES. *MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES*, (1), 122–129. https://doi.org/10.31891/2219-9365-2025-81-15