ПЕДАГОГІЧНА АКАДЕМІЯ:
НАУКОВІ ЗАПИСКИ

**Теорія і методика професійної освіти**

**УДК 378.147:004.4**

**DOI https://doi.org/10.5281/zenodo.19250709**

# Застосування технологій спільної розробки програмного забезпечення в умовах проєктного навчання майбутніх ІТ-фахівців

**Контуганський Павло Валерійович,**
аспірант кафедри інформатики і кібернетики,
Мелітопольський державний педагогічний університет
імені Богдана Хмельницького,
м. Мелітополь, Україна, https//orcid.org/0009-0000-3173-8508

**Прийнято: 11.03.2026 | Опубліковано: 27.03.2026**

***Анотація.*** *Стаття присвячена порівняльному аналізу підходів до впровадження технологій спільної розробки програмного забезпечення у систему проєктного навчання майбутніх ІТ-фахівців. Розглянуто симбіоз індустріальних стандартів Git-flow та Agile у поєднанні з дидактичними вимогами закладів вищої освіти.*

*У ході дослідження застосовано ряд наукових методів, серед яких теоретичний аналіз наукових публікацій та компаративний аналіз для класифікації існуючих моделей колаборативного навчання. Метод моделювання дозволяє сформулювати рекомендації для створення етапної системи переходу від індивідуального до колективного програмування.*

*У рамках дослідження вдалося встановити, що найбільш ефективною є гібридна технологія регулювання спільної розробки, яка є симбіозом технічного контролю та педагогічного інструментарію. Додатково розроблено багатоетапну систему критеріїв оцінювання командної діяльності студентів,*

*що включає об'єктивні метрики систем контролю версій. Особливу увагу приділено ролі педагога в освітньому процесі, що в рамках спільної розробки трансформувалася від трансляції знань до впровадження складних багатокомпонентних рішень. Імплементація запропонованої системи оцінювання процесу спільної розробки уможливлює високий рівень об'єктивності та прозорості освітнього процесу.*

*Таким чином, мінімізується проблема нерівномірного розподілу навантаження в групах студентів незалежно від рівня знань кожного. Практична значущість роботи полягає у розробці ряду прикладних рекомендацій, що зменшують когнітивний дисонанс студентів під час переходу від індивідуальних завдань до професійної колаборації робочого процесу. У подальшому слід звернути увагу на автоматизацію контролю навчальних репозиторіїв з використанням генеративних систем штучного інтелекту.*

***Ключові слова:*** *колаборативне навчання, системи контролю версій, Git-flow, Agile-фреймворки, IT-освіта, фахова компетентність.*

## Application of collaborative software development technologies in project-based learning for future IT specialists

**Pavlo Kontuhanskyi,**
Postgraduate Student of the Department of Informatics and Cybernetics,
Bogdan Khmelnitsky Melitopol State Pedagogical University,
Melitopol, Ukraine, https://orcid.org/0009-0000-3173-8508

***Abstract.*** *The article is devoted to a comparative analysis of approaches to implementing collaborative software development technologies within the project-based learning system for future IT specialists. The symbiosis of the Git-flow and Agile*

*industry standards in combination with the didactic requirements of higher education institutions is considered.*

*During the study, a number of scientific methods were applied, including theoretical analysis of scientific publications and comparative analysis for the classification of existing models of collaborative learning. The modelling method allowed us to formulate recommendations for creating a staged system of transition from individual to collective programming.*

*The study established that the most effective is the hybrid technology of joint development regulation, which is a symbiosis of technical control and pedagogical tools. Additionally, a multi-stage system of criteria for evaluating students' team activities was developed, which includes objective metrics of version control systems. Particular attention is paid to the role of the teacher in the educational process, which within the framework of joint development has transformed from knowledge translation to the implementation of complex multi-component solutions. The implementation of the proposed system for assessing the joint development process enables a high level of objectivity and transparency of the educational process.*

*Thus, the problem of uneven distribution of the load in groups of students is minimized, regardless of the level of knowledge of each. The practical significance of the work lies in the development of a number of applied recommendations that reduce the cognitive dissonance of students during the transition from individual tasks to professional collaboration of the workflow. In the future, attention should be paid to the automation of control of educational repositories using generative artificial intelligence systems.*

***Keywords:*** *collaborative learning, version control systems, Git-flow, Agile frameworks, IT education, professional competence.*

**Постановка проблеми.** Сьогодні розвиток IT-технологій можна охарактеризувати зміною акцентів від розробки та публікації окремих програм

та компонентів до імплементації складних багатокомпонентних рішень [1]. Таким чином змінюються і вимоги до професійних навичок IT-спеціаліста. Фахова компетентність тепер включає не лише знання мов програмування, але й навички роботи в колаборативному середовищі на основі Version Control Systems, CI/CD, Agile-фреймворки тощо [2].

На актуальність роботи впливає об'єктивна суперечність сучасної IT-сфери. Доки роботодавці вимагають від майбутніх спеціалістів високий рівень навичок командної роботи та розуміння інструментарію DevOps, вища освіта залишається сфокусованою на індивідуальних навчальних завданнях [3]. Вкрай важливо популяризувати проєктне навчання (Project-Based Learning), адже саме воно найбільш точно імітує реальний виробничий цикл розробки програмного забезпечення на всіх його етапах. В університетах впровадити повноцінні технології спільної розробки досить важко, адже для цього потрібна адаптація професійного інструментарію GitHub чи Jira. Окрім того постає питання, яким чином зберегти об'єктивність в оцінюванні ролі кожного учасника проєкту у розробці спільного програмного коду [4]. Таким чином змінюється й роль самого викладача: з лектора він перетворюється на архітектора колективного навчального проєкту.

У такому контексті вкрай важливо проаналізувати методичні засади імплементації інструментарію спільної розробки в умови проєктного навчання та їх роль в технічній підготовці студентів.

**Аналіз останніх досліджень і публікацій.** У процесі детального аналізу тематичного наукового дискурсу вдалося виявити, що питання імплементації технологій спільної розробки в освітнє середовище потребує додаткового вивчення. Основоположним залишається теза T. Wu щодо необхідності застосування систем контролю версій (Git) у рамках проєктного навчання [5]. На думку науковців, це дозволяє суттєво підвищити рівень критичного мислення та креативності студентів. З цією думкою погоджується і N. Pongsakdi,

наголошуючи на зміні функцій хмарних репозиторіїв [6]. На його думку, тепер вони зі сховища коду перетворилися на інструмент саморефлексії майбутнього програміста. E. Skantz-Åberg також підтверджує це положення, зазначаючи, що завдяки використанню технологій спільної розробки окрема робота студента стає видимою для усієї команди [7].

Разом з тим, не всі дослідники вважають використання такого інструментарію цілком виправданим. На противагу заявам щодо автоматичного підвищення ефективності, A. Badia та ін. стверджують, що активне застосування Agile-методологій Scrum чи, наприклад, Kanban без прив'язки до ефективного фасилітаторського досвіду викладача нераціональне [8]. Воно може спричинити хаотизацію навчального процесу та погіршення технічного боку написання коду. Часткове підтвердження цієї думки знаходимо і в науковій праці S. Ferino, який акцентує увагу читачів на відсутності перспектив успіху проєкту, якщо він не буде підкріплений чітко вибудуваною педагогічною стратегією «м'якого супроводу» [9].

Аналізуючи український контекст заданої проблеми, слід звернути увагу на наукові праці Р. Васьків [10] та К. Палагути [11]. У їхніх роботах хмарна інфраструктура демонструється у якості базисного елементу для спільної розробки програмних забезпечень. Ця позиція знаходить своє підтвердження у дослідженнях Н. Долгової [12] та Т. Грибик [13]. Автори доводять, що цифрова трансформація університету можлива лише за умови відмови від локальних завдань: натомість студенти мають зосередитись на вивченні розподілених екосистем. Такий підхід дійсно є вкрай важливий, але окремі автори не розділяють загального захоплення колаборативними інструментами. Так, В. Боковець висловлює побоювання щодо зміщення акцентів у навчанні. Він зазначає, що першочерговою проблемою залишається адаптивність освітнього контенту до швидких змін в IT-індустрії [14]. При цьому важливо зосереджуватися не лише на технічному боці освіти, але й на управлінській

складовій у спільній розробці. Саме цей аспект нерідко залишається поза увагою при підготовці майбутніх спеціалістів [15].

**Виділення невирішених раніше частин загальної проблеми.** Попри переважну згоду науковців у питанні важливості впливу проєктного навчання на формування фахових компетентностей, питання контролю досі викликає низку протиріч. Зокрема постає питання, як зберегти об'єктивність у оцінці активності студента в колективних проєктах GitHub. Крім того, існуючі моделі колаборативного навчання загалом працюють для уже сформованих команд, тоді як момент переходу від індивідуального написання коду до об'єднання у професійні колаборації залишається поза методологічним контролем. Саме визначення ключових технологій спільної розробки програмного забезпечення та оптимальної етапності їх впровадження є релевантним питанням для вивчення.

**Формулювання цілей статті (постановка завдання).** Метою дослідження є порівняльний аналіз основних методик імплементації технологій спільної розробки програмного забезпечення у систему проєктного навчання майбутніх ІТ-фахівців. За основу взято симбіоз індустріальних стандартів (Git-flow, Agile) та дидактичних вимог закладів вищої освіти. На основі встановленої мети визначено ряд завдань:

1. Провести порівняльний аналіз методик впровадження технологій спільної розробки ПЗ.
2. Запропонувати систему критеріїв оцінювання особистісного внеску студента в командний проєкт.
3. Розробити базові рекомендації для педагогів щодо спрощення етапу переходу від індивідуального написання коду до об'єднання у професійні колаборації для самих студентів.

**Виклад основного матеріалу дослідження.** Аби досягнути поставленої мети, було використано комплексний науковий підхід із залученням

 ISSN: 2786-9458

порівняльного аналізу та контент-аналізу вже існуючих викладацьких стратегій. Першочергово було проаналізовано релевантні наукові публікації, що було опубліковані упродовж останніх 7 років. Чимало з них фокусуються не лише на теоретичній базі використання технологій спільної розробки програмного забезпечення, але й на вивченні практичних кейсів університетів світу. Основними критеріями відбору методик для порівняння стали:

1. Технологічна відповідність щодо використання профільного інструментарію (Git, GitHub/GitLab, Jira/Trello, Docker).
2. Дидактична інтеграція технологій спільної розробки в існуючу кредитну-модульну систему навчання.
3. Перевірка масштабованості на основі малих (3–5 осіб) і великих проєктів.
4. Наявність механізмів зворотного зв'язку для аналізу прогресу студента.

Таким чином, для дослідження було обрано три базові моделі імплементації. Серед них інструментально-орієнтована (технічне виконання Git), процесно-керована (робота з Agile-фреймворками) та екосистемна (створення прототипу виробничого циклу в хмарному середовищі).

*Порівняльний аналіз методик впровадження технологій.* Ретельний аналіз обраних підходів свідчить, що на вибір методики в першу чергу впливає етап навчання та спеціалізація здобувачів освіти. При цьому у якості ключової проблеми все одно залишається пошук балансу між «техноцентричним» та «людиноцентричним» підходами.

Методика Git-Centric є однією з найбільш часто використовуваних в українських ЗВО завдяки простоті імплементації. Разом з тим ключовим недоліком залишається управлінський аспект, про який згадував В. Осадчий. Студенти в рамках цього підходу використовують репозиторій як «хмарну флешку». При цьому вони не залучають механізми *Pull Requests* або *Code Review*.

**Таблиця 1**

*Порівняльна характеристика основних методик впровадження спільної розробки*

| **Критерій порівняння** | Модель «Git-Centric» (Технічна) | Модель «Agile-Educational» (Процесна) | Модель «DevOps-Ecosystem» (Інтегрована) |
|---|---|---|---|
| **Основний інструментарій** | VCS (Git), GitHub Desktop. | Kanban-board, Scrum, Trello/Jira. | CI/CD pipelines, Docker, Cloud IDE. |
| **Роль викладача** | Технічний консультант. | Скрам-майстер / Фасилітатор. | Архітектор рішень / Product Owner. |
| **Ключовий акцент** | Чистота коду та історія комітів. | Командна комунікація, швидкість ітерацій. | Автоматизація, розгортання та життєвий цикл. |
| **Складність імплементації** | Низька (потребує лише знання Git). | Середня (потребує зміни графіку занять). | Висока (потребує значних серверних ресурсів). |
| **Ризики** | Студенти працюють ізольовано, зливають код в останній момент. | Надмірна увага до мітингів на шкоду якості коду. | Складність освоєння інструментів затіняє освітню мету. |

На противагу методиці Git-Centric, модель Agile-Educational (Морзе та О. Буйницька) фокусується на розподіленні роботи. При цьому технологія спільної розробки з мети перетворюється на ефективний засіб досягнення результату.

У контексті зіставлення вдається відстежити, що найбільшу ефективність проявив гібридний підхід *Git-flow* (сувора модель розгалуження) з

використанням коротких спринтів Agile. Завдяки такому поєднанню дозволяє структурувати технічну роботу та забезпечити видимість внеску кожного, що є логічним містком до нашого наступного завдання.

*Система критеріїв оцінювання особистісного внеску студента.* Основна проблема для ефективного впровадження проєктного навчання у закладах вищої освіти – це так звана «проблема вільного вершника» (free-rider problem). Це явище передбачає активне включення лідерів у командну роботу, тоді як решта учасників за менші зусилля отримують ту ж саму оцінку. Аби гарантувати об'єктивність оцінювання, рекомендується використовувати багаторівневу матрицю, в основі якої лежить об'єктивна метрика контролю версій і суб'єктивна оцінка команди. У якості системи оцінювання використовується Matrix of Student Contribution та наступні параметри:

1. Співвідношення доданих/видалених рядків коду – виконаний обсяг роботи аналізується з допомогою показників рефакторингу.
2. Частота та плановість комітів – важливо проаналізувати планомірність виконання завдань упродовж усього навчального періоду (модуля/семестру/академічного року).
3. Кількість закритих завдань – до уваги взято прямий зв'язок із таск-менеджером Jira/GitHub Projects. Знову ж показники враховуються за увесь навчальний період, а не лише за декілька днів перед здачею роботи.
4. Якість рецензій коду – важливо проаналізувати, наскільки якісно та часто студент рецензував код своїх колег, чи були його зауваження доречними та вдумливими. Завдяки цьому показнику можна проаналізувати рівень критичного мислення виконавця.
5. Наявність конфліктів при злитті коду *(Merge Conflicts)* – цей показник демонструє, чи узгоджено частину конкретного виконавця із загальною архітектурою проєкту.


ISSN: 2786-9458

6. Колаборативний коефіцієнт (Soft Skills) – колеги по команді можуть анонімно оцінювати навички одне одного з точки зору якості комунікації, відповідальності та працездатності.

**Таблиця 2**

*Пропонована структура підсумкового балу за проєкт*

| **Компонент оцінки** | **Вага в загальному балі** | **Інструмент перевірки** |
|---|---|---|
| Індивідуальний код (функціональність) | 40 % | Тести, демонстрація ПЗ. |
| Культура роботи з Git (коміти, гілки, PR) | 20 % | GitHub Insights, GitKraken Boards. |
| Якість рецензування (Code Review) | 15 % | Історія коментарів у Pull Requests. |
| Управління завданнями (Agile-активність) | 15 % | Trello/Jira Activity Log. |
| Командна оцінка (Soft Skills) | 10 % | Google Forms (Peer Review). |

Як демонструє таблиця, використання такої системи робить процес оцінювання достатньо прозорим та чітким. Якщо студент № 1 написав 80 % коду, але при цьому не брав участі в обговореннях, не узгоджував загальну архітектуру проєкту і не перевіряв коди інших учасників, його бал не буде максимальним. Таким чином, ключова мета спільної розробки – це ефективна комунікація з метою підвищення загальної якості роботи, а не лише індивідуальне кодування.

*Рекомендації для педагогів щодо переходу від індивідуального до колаборативного програмування.* Транзиторний етап від індивідуальних завдань до колективних проєктів – це доволі стресовий період («когнітивний шок») для

студентів. Аби його пом'якшити та імплементувати процес спільної розробки, слід враховувати наступні рекомендації:

1. Створення «Пісочниці» – перед об'єднанням студентів у загальну команду слід провести цикл лабораторних робіт з навмисним провокуванням конфліктів коду. Студенти мають одночасно редагувати один файл, щоб навчитися вирішувати ситуації з суперечністю коду в симуляційному середовищі. Це мінімізує страх зіпсувати спільний код в реальних умовах.

2. Рольова детермінація – далі викладач має розподілити ролі кожного члена команди (Frontend, Backend, QA, DevOps, Scrum Master). Вкрай важливо змінювати ці ролі між учасниками команди принаймні раз на місяць. Таким чином, кожен студент зможе приміряти на себе різні ролі та збагнути взаємозалежність компонентів системи.

3. Використання шаблонів – мінімізувати технічний поріг входу також можна, якщо викладач підготує базову структуру репозиторію з уже налаштованими правилами лінтування (ESLint, Prettier) та CI/CD. Таким чином, фокус студентів не зміщуватиметься з логіки колаборації на конфігурацію середовища.

4. Фасилітація через «Pull Request Culture» – робота викладача не полягає у особистій перевірці коду в кожному репозиторії. Він має коментувати та контролювати загальне обговорення у Pull Requests. У разі виникнення суперечностей щодо технічного інструментарію вирішення того чи іншого питання, викладач виконує роль медіатора та стимулює студентів до вирішення проблеми самостійно.

5. Поступове ускладнення – починати спільну розробку можна з парного програмування, коли перший студент займається написанням коду, а другий виконує роль штурмана. Наступний крок передбачає вже об'єднання двох пар у квартет. Така спрощена поступовість дозволяє органічно вибудувати

навички комунікації та не кваплячись наблизитись до головного командного проєкту.

Результати проведеного аналізу та розроблені методичні рекомендації дозволяють глибше поглянути на трансформацію ролі викладача та студента в умовах цифровізації освіти. Порівняння наших висновків із напрацюваннями інших дослідників виявляє як спільні, так і дискусійні моменти.

Перш за все, система критеріїв оцінювання персонального результату кожного зі студентів з використанням метрики GitHub співвідноситься з дослідженнями F. Caena [16] і B. Gopal [17]. Так, автори стверджують, що саме використання систем контролю версій у спільній розробці програмного забезпечення дозволяє розвинути критичне мислення здобувачів освіти. У цьому дослідженні вдалося визначити, що студент дійсно змушений більш відповідально підходити до виконання покладених на нього завдань, якщо він розуміє, що викладач контролює усі його «commit» та «pull request». Додатково йому потрібно створювати рецензії на коди інших учасників, брати активну участь у обговоренні коду та його узгодженні у загальній архітектурі проєкту. Фактично це унеможливлює ситуації, коли уся робота виконується в останню ніч.

У такому ж контексті розвиває свою думку С. Конюхов, говорячи про перетворення хмарних репозиторіїв на інструмент саморефлексії [18]. У нашому ж дослідженні залучена матриця внеску (Matrix of Student Contribution) також є фактичним підтвердженням професійного зростання студента на кожному з етапів виконання командної роботи. Кожна ітерація в репозиторії перетворюється на «цифровий слід». Цей феномен детально описав у своїй роботі S. Ferino. Він стверджував, що саме використання хмарних репозиторіїв дає змогу всій команді бачити індивідуальну роботу кожного учасника, співвідносити вагу такого внеску [9]. Вважаємо, що такий формат взаємодії додатково допомагає захистити команду від академічного нечестя та пасивності,

 ISSN: 2786-9458

мінімізуючи ситуації отримання несправедливих оцінок за бездіяльність у груповому проєкті.

Розроблені нами рекомендації щодо використання готових шаблонів та CI/CD також знаходять своє підтвердження у працях M. Zarour і M. Akour [19]. Як і зазначені автори, ми змогли визначити, що імплементувати аспекти цифрової трансформації в освітній процес вищої школи фактично неможливо без відмови від локальних IDE. Наші результати демонструють, що саме хмарна інфраструктура створює безперебійну динаміку процесу програмування, яка необхідна для реалізації складних проєктів.

Разом з тим, наше дослідження частково суперечить позиції G. Gudmundsdottir [20]. Автор припускає, що активне використання Agile-методологій без належного контролю та використання традиційного інструментарію може спровокувати хаотизацію навчання. У той же час наші результати демонструють, що хаос виникає не через самі методики, а через відмову від початкового «етапу пісочниці», розглянутого у рекомендаціях. Якщо структура проєкту заздалегідь спланована, а студенти мали змогу натренуватися вирішувати симуляційні конфлікти кодів, ризик хаотизації нівелюється. Таким чином, теза щодо нераціональності використання Agile у великих групах невиправдана. Ця технологія цілком виправдана в умовах регулярного моніторингу та жорсткої технічної перевірки (linting, автоматичні тести) на початкових етапах.

Окремої уваги заслуговує аспект адаптивності, що піднімає у своїй роботі A. Lund [21]. Вчений побоюється, що фокус на технічному забезпеченні може погіршити загальну якість освітнього середовища. На нашу думку, це побоювання виправдане лише частково, адже імплементація промислових стандартів, накшталт Git-flow, і є тим самим «освітнім контентом», що найкраще адаптується до умов IT-індустрії.

 ISSN: 2786-9458

Таким чином, запропонований нами підхід є симбіозом технологічного автоматизованого інструментарію та гуманістичної педагогіки з акцентом на фасилітарність. Також підтверджується ефективність хмарних технологій, описаних, але водночас увазі читачів пропонується окреслений інструментарій для вирішення проблем моніторингу та оцінювання, які раніше залишалися дискусійними в рамках тематичного наукового дискурсу.

**Висновки.** Проведене дослідження надало змогу вирішити актуальне наукове завдання, пов'язане з оптимізацією технічної підготовки майбутніх спеціалістів сфери ІТ. Це стало можливим завдяки імплементації технологій спільної розробки у систему проєктного навчання. Як продемонстрував порівняльний аналіз методик спільної розробки, найбільш ефективною та адаптивною в умовах вищої освіти є гібридна модель. Вона об'єднує суворе слідування технічному регламенту контролю версій Git-flow, а також методологію управління проєктами Agile. За умов використання інструментарію без додаткової методичної підтримки взаємодія зводиться до сухого виконання технічних завдань. Натомість залучення інтегрованого підходу дозволяє сформувати комплексну фахову компетентність (як «hard», так і «soft» skills).

Вдалося також сформулювати систему критеріїв оцінювання, завдяки чому вирішується питання об'єктивності оцінки знань окремих студентів у процесі спільної розробки. Запропонована «Матриця внеску» є симбіозом кількісних даних із логів репозиторію (частота комітів, обсяг коду) та якісних показників (результативність рецензування коду, командна взаємодія). Завдяки такому підходу нівелюються ризики нерівномірного розподілу навантаження, а також практично унеможливлюється академічна недоброчесність окремих студентів.

Зрештою вдалося сформулювати практичні рекомендації для науково-педагогічних працівників вищої школи. Вони націлені на спрощення умов адаптації студентів до спільної розробки програмного забезпечення в умовах масштабних проєктних завдань. Наголошено, що ключем до успіху в цьому

контексті є зміна ролі викладача. Він має перетворитися на архітектора проєкту, займаючись питаннями регуляції етапності виробничого циклу. Використання стратегії поступового ускладнення проєктних завдань мінімізує когнітивне перенавантаження студентів у колективах. Перспективи подальших досліджень сфокусовані на вивченні можливостей автоматизації процесу оцінювання студентського коду з використанням генеративних систем штучного інтелекту.

**Список використаних джерел**

1. Hawkridge D. New information technology in education. Routledge, 2022. 250 p. DOI: https://doi.org/10.4324/9781003312826

2. Erstad O., Kjällander S., Järvelä S. Facing the challenges of ’digital competence’ a Nordic agenda for curriculum development for the 21st century. *Nordic Journal of Digital Literacy*. 2021. Vol. 16, No. 2. P. 77–87. DOI: https://doi.org/10.18261/issn.1891-943x-2021-02-02

3. Chou Chih-Yuan. A DevOps-based Service-Learning Design in An Advanced IS-related Course. *Rising like a Phoenix: Emerging from the Pandemic and Reshaping Human Endeavors with Digital Technologies ICIS 2023*. 2023. Art. 8. URL: https://aisel.aisnet.org/icis2023/learnandiscurricula/learnandiscurricula/8

4. Radenković M., Popović S., Mitrović S. Project based learning for DevOps: School of Computing experiences. *E-Business Technologies Conference Proceedings*. 2022. Vol. 2, No. 1. P. 127–131. URL: https://www.ebt.rs/journals/index.php/conf-proc/article/view/133

5. Wu T. Digital project management: Rapid changes define new working environments. *Journal of Business Strategy*. 2021. DOI: https://doi.org/10.1108/JBS-03-2021-0047

6. Pongsakdi N., Kortelainen A., Veermans M. The impact of digital pedagogy training on in-service teachers’ attitudes towards digital technologies. *Education and*

*Information Technologies*. 2021. Vol. 26, No. 5. P. 5041–5054. DOI: https://doi.org/10.1007/s10639-021-10439-w

7. Skantz-Åberg E., Lantz-Andersson A., Lundin M., Williams P. Teachers' professional digital competence: an overview of conceptualisations in the literature. *Cogent Education*. 2022. Vol. 9, No. 1. Art. 2063224. DOI: https://doi.org/10.1080/2331186X.2022.2063224

8. Badia A., Meneses J., Sergi F., Sigalés C. Factors affecting school teachers' perceptions of the instructional benefits of educational digital media. *Electronic Journal of Educational Research, Assessment & Evaluation*. 2014. Vol. 21, No. 2. P. 1–11. DOI: https://doi.org/10.1016/j.sbspro.2014.05.063

9. Ferino S. et al. Overcoming Challenges in DevOps Education through Teaching Method. *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. Melbourne, Australia, 2023. P. 166–178. DOI: https://doi.org/10.1109/ICSE-SEET58685.2023.00022

10. Васьків Р., Веретеннікова Н. Інформаційні та комунікаційні інструменти ефективного функціонування розподілених проєктних команд. *Вісник Національного університету «Львівська політехніка». Серія: «Інформаційні системи та мережі»*. 2024. Вип. 15. С. 357–369. DOI: https://doi.org/10.23939/sisn2024.15.357

11. Палагута К., Савон О. Методи та інформаційні технології дистанційного навчання. *Кібербезпека: освіта, наука, техніка*. 2025. № 1(29). С. 287–297. DOI: https://doi.org/10.28925/2663-4023.2025.29.826

12. Долгова Н. Г., Шаповалова О. О., Солодовник Г. В. Система підтримки рішень у завданнях проектування кібербезпеки критичної інфраструктури. *Кібербезпека: освіта, наука, техніка*. 2025. № 1(29). С. 348–372. URL: https://repository.hneu.edu.ua/handle/123456789/37326

13. Грибик Т. Т. Особливості використання системи контролю версій при розробці освітніх проєктів з інженерії програмного забезпечення. *Інноваційна педагогіка*. 2024. Вип. 70, т. 1. С. 189–192. DOI: https://doi.org/10.32782/2663-6085/2024/70.1.40

14. Боковець В. В., Заяц О. М. Сучасні методи управління проєктами та їх особливості. *Інфраструктура ринку*. 2023. № 65. С. 61–66. DOI: https://doi.org/10.32843/infrastruct65-9

15. Задоя В. О., Чаркіна Т. Ю., Корольов Д. С. Цифрові інструменти та технології в управлінні науковими проєктами. *Review of Transport Economics and Management*. 2025. Вип. 12(28). С. 224–232. DOI: https://doi.org/10.15802/rtem2024/328104

16. Caena F., Redecker C. Aligning teacher competence frameworks to 21st century challenges: The case for the European digital competence framework for educators (Digcompedu). *European Journal of Education*. 2019. Vol. 54, No. 3. P. 356–369. DOI: https://doi.org/10.1111/ejed.12345

17. Gopal B., Cooper S. POGIL-like Learning in Undergraduate Software Testing and DevOps – A Pilot Study. *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1 (ITiCSE '22)*. New York, NY, USA : ACM, 2022. P. 484–490. DOI: https://doi.org/10.1145/3502718.3524776

18. Конюхов С. Аналіз іноземного досвіду застосування технологій навчання у співпраці для професійної підготовки майбутніх IT-фахівців у закладах вищої освіти. *Освітній дискурс*. 2025. № 48(1). С. 29–40. DOI: https://doi.org/10.28925/2312-5829/2025.1.3

19. Zarour M., Akour M., Alenezi M. Enhancing DevOps Engineering Education Through System-Based Learning Approach. *Open Education Studies*. 2024. Vol. 6, No. 1. Art. 20240012. DOI: https://doi.org/10.1515/edu-2024-0012

20. Gudmundsdottir G. B., Hatlevik O. Newly qualified teachers' professional digital competence: Implications for teacher education. *European Journal of Teacher Education*. 2018. Vol. 41, No. 2. P. 214–231. DOI: https://doi.org/10.1080/02619768.2017.1416085

21. Lund A., Aagaard T. Digitalization of teacher education: Are we prepared for epistemic change? *Nordic Journal of Comparative and International Education*. 2020. Vol. 4, No. 3. P. 56–71. DOI: https://doi.org/10.7577/njcie.3751