УДК 004.8:004.85 https://doi.org/10.31713/vt3202517

Воробйов В. С. [1; ORCID ID: 0009-0003-2894-4070],
студент
Зубик Л. В. [1; ORCID ID: 0000-0002-2087-5379],
доцент,
Зубик Я. Я. [2; ORCID ID: 0000-0002-0802-3552],
старший викладач

[1]Київський національний університет імені Тараса Шевченка, м. Київ
[2]Національний університет водного господарства та природокористування, м. Рівне

МОДЕЛЬ ІНТЕЛЕКТУАЛЬНОГО РОЗПОДІЛУ ЗАВДАНЬ У КОМАНДАХ ПРОГРАМНОЇ РОЗРОБКИ НА ОСНОВІ НЕЙРОМЕРЕЖЕВИХ АЛГОРИТМІВ

У цій статті представлено модель інтелектуального розподілу завдань для команди розробників, побудовану на основі нейронних мереж. Інформація про наявні навички, поточну зайнятість та попередню ефективність окремих членів команди автоматично враховуються для прийняття рішень про призначення нових завдань. Архітектура інтелектуальної системи модульна, для клієнтської частини системи розподілу завдань, побудованої на основі запропонованої моделі, пропонується використання ASP.Core, для машинного навчання – Python API. Нейронна модель була навчена на історичних даних про виконання задач у командах розробників, а також вимогах до проєкту. Отримані результати підтверджують, що запропонована модель суттєво підвищує ефективність розподілу завдань у порівнянні з традиційним ручним підходом. Модель дозволяє більш рівномірно розподіляти робоче навантаження на окремих членів команди та мінімізує зусилля, затрачені на вироблення командних рішень. Продуктивність команди підвищується за рахунок зростання швидкості виконання задач, зменшення перевантаження окремих розробників, врахування компетенції та попереднього досвіду виконавців, покращення прогнозування дедлайнів, зниження ризику затримок або невиконання поставлених завдань.

***Ключові слова*:** система; інтелектуальна система; проєктування систем; модель; нейронні мережі; машинне навчання.

Аналіз досліджень. Традиційний розподіл завдань керівником проєкту вручну переважно породжував нерівномірність завантаження виконавців, що знижувало ефективність роботи команди в цілому. Залучення останнім часом штучного інтелекту для

 ISSN 2306-5478

автоматизації призначення завдань дозволяє істотно підвищити продуктивність команд.

Такі сучасні рішення, як Asana використовують штучний інтелект для автоматизації типових завдань управління проєктами. Asana Intelligence можна використовувати для автоматичного створення підзадач з описів завдань або коментарів, прагнучи розподілити завдання рівномірно між окремими членами команди та максимально уникнути їх перевантаження [4].

Подібні функції розроблені в Jira, де алгоритми штучного інтелекту забезпечують автоматизацію управління завданнями та прогнозування затримок, а також розподілу ресурсів. Усі ці можливості підтримують своєчасне внесення змін до плану та підвищення загальної продуктивності команди [5].

У джерелах наведено переваги використання штучного інтелекту в управлінні завданнями. Автоматизація монотонних завдань призводить до звільнення ресурсів для менеджерів та розробників і можливість роботи над більш складними завданнями. Здатність штучного інтелекту обробляти великі масиви даних допомагає виявляти нові закономірності та тенденції, а отже, приймати більш обґрунтовані рішення. Також автори демонструють використання нейронних мереж для більш точного прогнозування ризиків та оптимального розподілу ресурсів, делегуючи функції управління проєктами інтелектуальним агентам [5].

Однак, попри згадані переваги, існують деякі труднощі щодо реалізації ідей. Найбільш вагомою проблемою адаптації таких систем є вартість їх впровадження, яка може бути дуже високою. Це істотна перешкода для малих компаній. Також потрібен час для адаптації та навчання персоналу новим робочим процесам. Необхідно враховувати також етичні питання, включаючи прозорість алгоритмів та уникнення потенційної упередженості в процесі ухвалення рішень.

Постановка проблеми. Хоча інтерес до рішень для управління завданнями на основі штучного інтелекту зростає, жодне комерційно доступне рішення не є одночасно функціональним, простим у використанні та економічно ефективним. Більшість відомих рішень розроблені для корпоративного сегмента і потребують тонкого налаштування та ручного управління. Тому завжди корисно мати власне рішення для малих та середніх команд розробників. Рішення, яке автоматизує процес розподілу завдань, може бути побудоване

на нейромережевих алгоритмах. При цьому воно буде простим у використанні та доступним для непрофесійних користувачів.

Мета дослідження – розробити модель інтелектуальної системи управління завданнями, яка оптимізує розподіл робіт між учасниками команди на основі аналізу їхніх професійних навичок, поточної зайнятості та історичних даних про виконання ними типових завдань.

Архітектура системи. Модульна, масштабована, підтримувана архітектура стає дедалі більш затребуваним архітектурним шаблоном у світі розробки програмного забезпечення. У такому проєкті поєднуються дві парадигми: шаблон чистої архітектури та патерн MVC з використанням ASP.NET Core для основної програми, а для розумного розподілу ресурсів використовується Python API.

Чиста архітектура, як її визначив Роберт К. Мартін [6], вкладає системи в шари, із залежностями, спрямованими всередину, до бізнесу в центрі. Таким чином, гарантується ізоляція між проблемами високого рівня (такими як бізнес-правила) від проблем низького рівня, пов'язаних з фреймворками та зовнішніми сервісами.

Основні шари:

1. Домен: де знаходиться більшість бізнес-об'єктів та правил.

2. Застосування: описує варіанти використання та розташовується між доменом та інфраструктурою.

3. Інфраструктура: займається зовнішніми речами, такими як доступ до бази даних або сторонні сервіси.

4. Презентація: пропонує інтерфейс користувача (наприклад, вебпредставлення або контролери API).

Це, разом із використанням MVC, забезпечує розділення завдань, де контролер керує вхідними HTTP-запитами, моделі зберігають бізнес-логіку та дані програми, а представлення піклуються про надання відповіді користувачеві.

Базою цієї архітектури є ASP.Core, який повинен обробляти HTTP-запити та орієнтуватися на взаємодію з користувачем. Для спеціалізованих завдань, наприклад, для призначення завдань на основі машинного навчання, існує окремий сервіс, написаний на Python.

Такий поділ гарантує, що система може використовувати високу ефективність та масштабованість ASP. В основу системи покладено вебвзаємодії, які вона забезпечує, а також можливості машинного навчання та штучного інтелекту, що пропонуються

 ISSN 2306-5478

Python. Дві системи взаємодіють за принципом клієнт/сервер. Додаток NET Core є клієнтом, а сервіс Python – сервером. Він базується на протоколі HTTP, а обмін даними відбувається у форматі JSON (рис. 1).

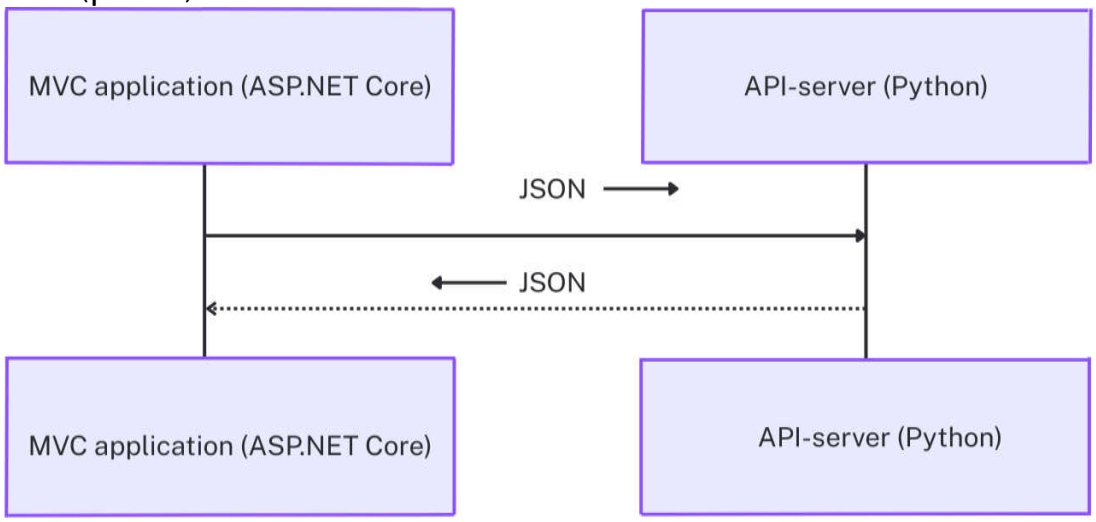

Рис. 1. Обмін даними JSON між ASP.NET Core та Python API

Як видно з рис. 1, застосунок ASP.NET Core створює корисне навантаження JSON, яке представляє завдання та потенційних співробітників, яким його можна призначити, і надсилає його до Python API за допомогою HTTP-клієнта, обгорнутого Refit. Служба Python обробляє запит за допомогою нейронної мережі, ранжує найбільш релевантних виконавців і надає відповідь. Після отримання цієї інформації, додаток .NET оновлює стан системи та пов'язує вибраного співробітника із завданням.

Опис взаємодії компонентів. Програма дозволяє користувачам легко створювати команди, призначати співробітників та делегувати завдання. Взаємодія під час запуску призначення завдання включає:

1. Дані про завдання та членів команди агрегуються в застосунку.
2. Застосунок надсилає сформований JSON-запит до Python API.
3. Служба Python обробляє запит за допомогою логіки машинного навчання та повертає найкращий результат.
4. Застосунок оновлює списки призначень завдань.

Описана архітектура пропонує високий рівень модульності, що дозволяє швидко модифікувати та замінювати деталі без порушення роботи всієї системи. Крім того, використання ASP.NET Core 8 з модульною архітектурою гарантує безпечний, ефективний та масштабований процес розробки [8].

Формування набору даних. Збір даних. Набір даних для навчання нейронної моделі призначення завдань було зібрано на основі інформації з загальнодоступних джерел, а саме – дані зібрані з онлайн-платформ GitHub та LinkedIn. Компетенції розробників отримано з аналізів публічних репозиторіїв, що підтвердили Олівейра та Соуза, а профілі LinkedIn використано для збагачення метаданих, таких як навички, професійний досвід та рекомендації колег. Однак ці джерела мали обмеження, зокрема, щодо повноти даних та потенційної упередженості самозвітності.

Попередня обробка та кодування. Отриманий набір даних був попередньо оброблений відповідно до архітектури нейронної мережі. Категоричні ознаки були закодовані один раз, щоб уникнути оманливих класифікаторів з порядковими даними, як запропонував Герон [10]. MinMaxScaler використано для нормалізації всіх безперервних ознак, включаючи складність завдання, терміни, робоче навантаження та продуктивність розробника, до значень від 0 до 1. Цей вибір був мотивований спостереженням Іоффе та Сегеді [10], що він не тільки зберігає початковий розподіл вхідних даних, але й покращує збіжність навчання. StandardScaler був пропущений, оскільки вхідні дані не розподілені нормально.

Розподіл для навчання та тестування. Кінцевий набір даних було розподілено на навчальну вибірку (80%) і тестову (20%). Це співвідношення забезпечило гарний компроміс між навчанням моделі та потужністю узагальнення, перевершуючи більш екстремальні поділи, такі як 70/30 або 60/40 на ранній валідації.

Проєктування та навчання моделі

Проєктування архітектури. Призначення завдання розробнику було змодельовано за допомогою глибокої нейронної мережі прямого зв'язку. Архітектура мала вхідний шар, який приймав попередньо оброблені вектори ознак, що фіксували інформацію як про завдання, так і про розробника, а потім три приховані шари, що використовували функцію активації ReLU. Ця глибина була обрана для балансу виразності моделі з можливістю узагальнення: хоча глибші архітектури були можливі, трьох прихованих шарів було достатньо для фіксації нелінійних взаємодій між характеристиками завдання та профілями розробника без збільшення ризику перенавчання. Останній шар використовував сигмоїдну активацію для створення скалярної Рейтинг сумісності, який показує, наскільки добре розробник підходить для виконання певного завдання.

 ISSN 2306-5478

Оптимізація та функція втрат. Оптимізатор Adam був використаний для навчання моделі завдяки його здатності швидко сходитися на розріджених градієнтах та самостійно налаштовувати швидкість навчання. На відміну від середньоквадратичної помилки (MSE), функція втрат була обрана як середня абсолютна помилка (MAE), враховуючи, що вона не чутлива до випадків випадіння, що забезпечує більшу інтерпретованість щодо ефективності прогнозування точності оцінки на основі регресії (Goodfellow, Bengio, & Courville, 2016) [12].

Навчання моделі. Модель навчалася протягом 40 епох, використовуючи мініпакет розміром 8. На етапі навчання дотримувалися показників MAE для навчання та валідації. На рис. 2 показано динаміку навчання (синя лінія – це MAE для навчання, а червона лінія – для валідації).

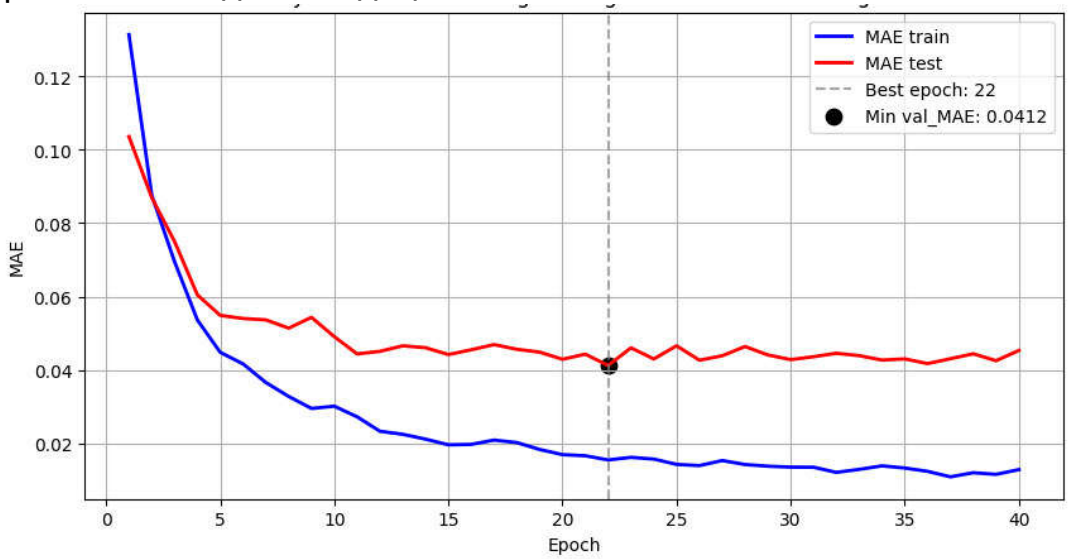

Рис. 2. Еволюція MAE у процесі навчання

На початку епох обидві криві демонструють високі значення MAE, що відображає високі помилки прогнозування. Під час навчання MAE на навчальному наборі швидко зменшується в процесі навчання, що відображає здатність моделі до навчання. Крива валідації знижується менш круто, що вказує на повільне перенавчання моделі.

У ці епохи (близько 15–20) MAE зменшується повільніше, і можна спостерігати розрив між помилкою навчання та валідації. Це ілюструє перенавчання, коли модель починає запам'ятовувати шаблони, які базуються лише на навчальних даних, замість того, щоб узагальнювати їх. Найнижче значення MAE на валідаційному наборі

було досягнуто після епохи 22 (найкраща модель) (пунктирна лінія на рис. 2) зі значенням 0,0412.

Оцінка узагальнення. Розбіжність у кривих навчання та валідації на рис. 2 також є симптомом. Більший/менший розрив вказує на більш серйозне перенавчання/краще узагальнення. Тут також різниця знаходиться в допустимому діапазоні, показуючи, що модель добре узагальнюється на нові приклади.

Порівняння прогнозованих та фактичних значень. Для додаткової оцінки ефективності прогнозування було використано діаграму розсіювання (рис. 3) для порівняння фактичних значень сумісності (вісь x) та прогнозованих значень (вісь y) моделлю. Червона пунктирна лінія представляє ідеальну лінію ідентичності (прогнозоване = фактичне).

З кривої видно, що більшість точок близькі до опорної лінії, що вказує на якісний прогноз. Тим не менш, існує певна дисперсія, особливо для вищих відповідностей. Це може бути викликано недостатньою кількістю навчальних вибірок у цьому інтервалі або властивим обмеженням форми нейронної мережі.

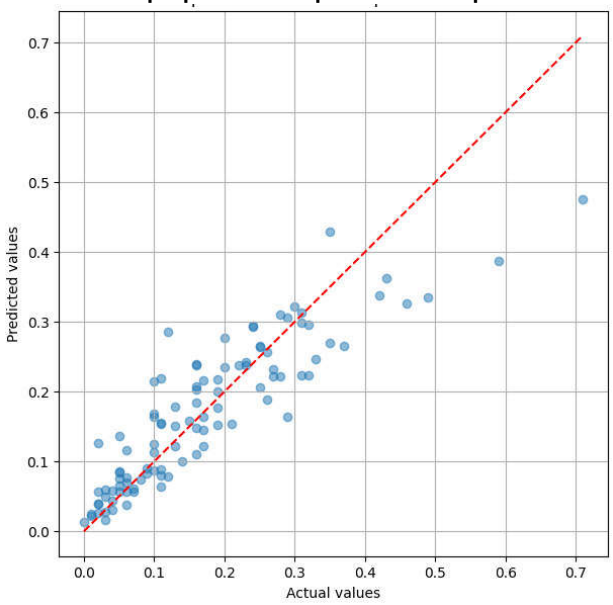

Рис. 3. Порівняння дійсних і прогнозованих значень

Аналіз розподілу помилок. Гістограма помилок моделі представлена на рис. 4. Як видно, переважна більшість значень таких відхилень скупчується навколо нуля, що свідчить про надійність моделі. Більшість діаграм розсіювання приблизно нормальні, навіть якщо є невелике зміщення. Середнє значення

 ISSN 2306-5478

діаграми розсіювання (0,0017) вказує на відсутність систематичного зміщення в моделі.

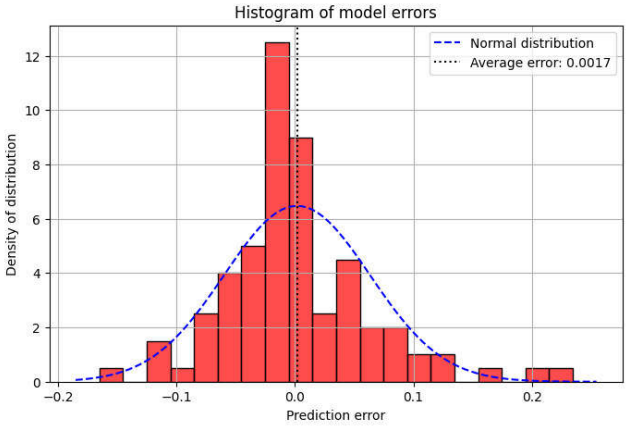

Рис. 4. Гістограма помилок моделі

Числова оцінка. Результати порівняння реальних та прогнозованих значень зведених даних машинного перекладу на перших 10 прикладах тестового набору наведені у таблиці. Загальна оцінка точності моделі показала, що вона є прийнятною лише з деякими варіаціями, які не роблять систему неактуальною у виробничому середовищі.

Таблиця

Фактичні та прогнозовані бали сумісності розробника завдань (перші 10 зразків)

Зразок	1	2	3	4	5	6	7	8	9	10
Фактичний бал	0,24	0,28	0,23	0,08	0,13	0,19	0,15	0,02	0,33	0,14
Прогнозований бал	0,2945	0,2662	0,2397	0,1053	0,1511	0,1766	0,154	0,0347	0,3244	0,1175

Алгоритм розподілу завдань у системі. Система зчитує JSON-файл, що містить завдання та список розробників. Ці дані надсилаються на API-сервер з моделлю нейронної мережі (TensorFlow/Keras), який аналізує кандидатів і повертає найкращу пропозицію для розробника. Рекомендуються підходи, подібні до вищезазначеного. Аль-Фрайхат та ін. (2024) описують нейронний підхід, який досягає точності до 96,7% у пошуку відповідності розробників та завдань [12].

Вхідні дані JSON перевіряються та очищуються, а потім нормалізуються для забезпечення однорідності. Атрибути, пов'язані із завданням, наприклад, тип, складність, необхідні технології, час,

тощо, та атрибути, пов'язані з розробником, наприклад, досвід, набір навичок, робоче навантаження, історія завдання тощо, витягуються як вхідні вектори для моделі. Модель, у свою чергу, виводить AssignmentScore для кожного розробника та вибирає того, хто отримав найвищий бал. Вихідні дані повертаються у вигляді відповіді API та зв'язуються із системою управління завданнями.

Оцінювання та результати

Налаштування тестування. Було створено спеціальний тестовий набір даних для перевірки продуктивності моделі призначення завдань. У ньому присутні зразки завдань і параметри розробників на вибір. Кожен профіль розробника вміщував відповідну інформацію про навички, поточне навантаження завданнями і темп роботи.

Критерії призначення. Параметри, за якими модель оцінювала кожного розробника:

- Відповідність навичок – рівень, до якого розробник має необхідні навички для конкретного необхідного внеску.
- Швидкість виконання завдання – наскільки швидко розробник зазвичай завершує завдання.
- Поточне навантаження – існуюче навантаження розробника.

Ці параметри використовувалися для обчислення AssignmentScore для кожного розробника. Коли розробнику призначалося завдання, робота розробника над завданням обчислювалася за формулою:

$$Workload = \frac{\sqrt{EstimatedHours} \times Difficulty}{(DeadlineDays \times TaskCompetitionSpeed) + 1}, \quad (1)$$

де *Workload* – поточне навантаження,
Estimatehours – очікувана кількість годин,
Difficulty – складність завдання,
DeadlineDays – кількість днів для виконання завдання,
TaskCompetitionSpeed – швидкість завершення завдань.

Висновки

Запропоновано модель і розроблено інтелектуальну систему розподілу окремих завдань всередині команд розробників програмного забезпечення. Для налаштування інтелектуальної системи застосовано машинне навчання. Навчання нейронної мережі здійснювалося на основі даних, зібраних для аналітики з відкритих джерел: наявні навички, пропускна здатність, а також

 ISSN 2306-5478

продуктивність розробників, які впливають на те, як система розподіляє завдання більш справедливим та точним чином. Система побудована на чистій архітектурі та патерні MVC для досягнення модульності та масштабованості. Тестування моделі дозволило зробити висновки щодо її вищої ефективності порівняно з традиційними методами, за рахунок зменшення навантаження на команду та підвищення її продуктивності. Подальші дослідження передбачають застосування глибших нейронних моделей та підвищення інтерпретованості процесу прийняття рішень.

1. Shafiq S., at al. TaskAllocator: A Recommendation Approach for Role-based Tasks Allocation in Agile Software Development. 2021. URL: https://arxiv.org/pdf/2103.02330 (дата звернення: 10.05.2025). **2.** Al-Fraihat D., at al. Utilizing machine learning algorithms for task allocation in distributed agile software development. 2024. doi:10.1016/j.heliyon.2024.e39926 **3.** Sajid B., and Maya K. AI-powered Software Engineeging: Automating: Code Generation with Multi-Agent Systems. 2023. URL: https://www.researchgate.net/publication/388834604 (дата звернення: 10.05.2025). **4.** Ogunbukola M. The Impact of Artificial Intelligence on Project Management: Enhanciring Efficiency, Risk Mitigation, and Decision-Making in Complex Projects. 2024. URL: https://www.researchgate.net/publication/384266056 (дата звернення: 10.05.2025). **5.** Morris G. Automated task management system: AI-based intelligent workflow optimization. 2024. URL: https://www.tech.vernadskyjournals.in.ua/journals/2024/4_2024/12.pdf. (дата звернення: 10.05.2025). **6.** Martin R. C. Clean Architecture. A Craftman's Guide To Software Structure And Design. 2018. ISBN-13: 978-0-13-449416-6 **7.** Marcotte C. H. Architecting ASP.NET Core Applications: An atypical design patterns guide for .NET 8. ISBN 978-1-80512-338-5 **8.** Yan X. Web API Development with ASP.NET Core 8: Learn techniques, patterns, and tools for building high-performance, robust, and scalabre web APIs. ISBN 978-1804610954 **9.** Hussain F., at al. Machine Learning for Resource Management in Cellular and IoT Networks: Potentials, Current Solutions, and Open Challenges. 2020. URL: https://ieeexplore.ieee.org/ document/8951180 (дата звернення: 10.05.2025). **10.** Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2023. ISBN 978-1-098-12597-4 **11.** Sun J. at al. New Interpretations of Normalization Methods in Deep Learning. 2019. URL: https://www.researchgate.net/publication/339138686 (дата звернення: 10.05.2025). **12.** Goodfellow I., Bengio Y., Courville A. Deep Learning. 2016. URL: https://www.deeplearningbook.org/. (дата звернення: 10.05.2025). **13.** Neural networks for predicting developer performance. URL: https://www.cell.com/heliyon/fulltext/S2405-8440(24)15957-9. (дата звернення: 10.05.2025). **14.** Nielsen M. A. Neural Networks and Deep Learning. 2015. URL: http://neuralnetworksanddeeplearning.com/ (дата звернення: 10.05.2025). **15.** Fang Zh., at al. Efficient Task Allocation in Multi-Agent Systems Using Reinforcement Learning and Genetic Algorithm. 2025. URL: https://doi.org/10.3390/app15041905 **16.** Ali A., Ajaz S., and Daniyal S. M. Optimized Artificial Neural Network-based Approach for Task Scheduling in Cloud Computing. *Annual Methodological Archive Research Review*. 2025. Vol. 3, Issue 4(2025). URL: https://amresearchreview.com/index.php/Journal/article/view/101/146 (дата

звернення: 10.05.2025). **17.** Kibiwot S. K. A Machine Learning model for task allocation. Faculty of Information Technology Strathmore University. 2020. URL: https://su-plus.strathmore.edu/server/api/core/bitstreams/09625ec1-a206-47bd-930f-dc3d0d96 98ec/content (дата звернення: 10.05.2025). **18.** Al-Fraihat D. at al. Utilizing machine learning algorithms for task allocation in distributed agile software development. 2024. URL: https://doi.org/10.1016/j.heliyon.2024.e39926 (дата звернення: 10.05.2025). **19.** Simonyan K., & Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014. URL: https://arxiv.org/pdf/1409.1556v5 (дата звернення: 10.05.2025). **20.** Sutton R. S. & Barto A. G. Reinforcement Learning: An Introduction. 2-nd ed. 2015. URL: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf (дата звернення: 10.05.2025). **21.** Martin R., C., & Martin M. Agile Principles, Patterns, and Practices in C#. 2006. ISBN-13 978-0131857254 **22.** Pressman R. S. Software Engineering: A Practitioner's Approach. 8th ed. 2014. URL: https://www.amazon.com/Software-Engineering-Practitioners-Roger-Pressman/dp/0078022126 (дата звернення: 10.05.2025). **23.** Taboada I., at al. Artificial Intelligence Enabled Project Management: A Systematic Literature Review. *Appl. Sci*. 2023. Vol. 13(8). P. 5014. URL: https://doi.org/10.3390 /app13085014

REFERENCES:

1. Shafiq S., at al. TaskAllocator: A Recommendation Approach for Role-based Tasks Allocation in Agile Software Development. 2021. URL: https://arxiv.org/pdf/2103.02330 (data zvernennia: 10.05.2025). **2.** Al-Fraihat D., at al. Utilizing machine learning algorithms for task allocation in distributed agile software development. 2024. doi:10.1016/j.heliyon.2024.e39926 **3.** Sajid B., and Maya K. AI-powered Software Engineeging: Automating: Code Generation with Multi-Agent Systems. 2023. URL: https://www.researchgate.net/publication/388834604 (data zvernennia: 10.05.2025). **4.** Ogunbukola M. The Impact of Artificial Intelligence on Project Management: Enhanciring Efficiency, Risk Mitigation, and Decision-Making in Complex Projects. 2024. URL: https://www.researchgate.net/publication/384266056 (data zvernennia: 10.05.2025). **5.** Morris G. Automated task management system: AI-based intelligent workflow optimization. 2024. URL: https://www.tech.vernadskyjournals.in.ua/journals/2024/4_2024/12.pdf. (data zvernennia: 10.05.2025). **6.** Martin R. C. Clean Architecture. A Craftman's Guide To Software Structure And Design. 2018. ISBN-13: 978-0-13-449416-6 **7.** Marcotte C. H. Architecting ASP.NET Core Applications: An atypical design patterns guide for .NET 8. ISBN 978-1-80512-338-5 **8.** Yan X. Web API Development with ASP.NET Core 8: Learn techniques, patterns, and tools for building high-performance, robust, and scalabre web APIs. ISBN 978-1804610954 **9.** Hussain F., at al. Machine Learning for Resource Management in Cellular and IoT Networks: Potentials, Current Solutions, and Open Challenges. 2020. URL: https://ieeexplore.ieee.org/ document/8951180 (data zvernennia: 10.05.2025). **10.** Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2023. ISBN 978-1-098-12597-4 **11.** Sun J. at al. New Interpretations of Normalization Methods in Deep Learning. 2019. URL: https://www.researchgate.net/publication/339138686 (data zvernennia: 10.05.2025). **12.** Goodfellow I., Bengio Y., Courville A. Deep Learning. 2016. URL:

 ISSN 2306-5478

https://www.deeplearningbook.org/. (data zvernennia: 10.05.2025). **13.** Neural networks for predicting developer performance. URL: https://www.cell.com/heliyon/fulltext/S2405-8440(24)15957-9. (data zvernennia: 10.05.2025). **14.** Nielsen M. A. Neural Networks and Deep Learning. 2015. URL: http://neuralnetworksanddeeplearning.com/ (data zvernennia: 10.05.2025). **15.** Fang Zh., at al. Efficient Task Allocation in Multi-Agent Systems Using Reinforcement Learning and Genetic Algorithm. 2025. URL: https://doi.org/10.3390/app15041905 **16.** Ali A., Ajaz S., and Daniyal S. M. Optimized Artificial Neural Network-based Approach for Task Scheduling in Cloud Computing. *Annual Methodological Archive Research Review*. 2025. Vol. 3, Issue 4(2025). URL: https://amresearchreview.com/index.php/Journal/article/view/101/146 (data zvernennia: 10.05.2025). **17.** Kibiwot S. K. A Machine Learning model for task allocation. Faculty of Information Technology Strathmore University. 2020. URL: https://su-plus.strathmore.edu/server/api/core/bitstreams/09625ec1-a206-47bd-930f-dc3d0d96 98ec/content (data zvernennia: 10.05.2025). **18.** Al-Fraihat D. at al. Utilizing machine learning algorithms for task allocation in distributed agile software development. 2024. URL: https://doi.org/10.1016/j.heliyon.2024.e39926 (data zvernennia: 10.05.2025). **19.** Simonyan K., & Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014. URL: https://arxiv.org/pdf/1409.1556v5 (data zvernennia: 10.05.2025). **20.** Sutton R. S. & Barto A. G. Reinforcement Learning: An Introduction. 2-nd ed. 2015. URL: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf (data zvernennia: 10.05.2025). **21.** Martin R., C., & Martin M. Agile Principles, Patterns, and Practices in C#. 2006. ISBN-13 978-0131857254 **22.** Pressman R. S. Software Engineering: A Practitioner's Approach. 8th ed. 2014. URL: https://www.amazon.com/Software-Engineering-Practitioners-Roger-Pressman/dp/0078022126 (data zvernennia: 10.05.2025). **23.** Taboada I., at al. Artificial Intelligence Enabled Project Management: A Systematic Literature Review. *Appl. Sci*. 2023. Vol. 13(8). P. 5014. URL: https://doi.org/10.3390 /app13085014

Vorobiov V. S. [1; ORCID ID: 0009-0003-2894-4070],
Senior Student
Zubyk L. V. [1; ORCID ID: 0000-0002-2087-5379],
Associate Professor
Zubyk Ya. Ya. [2; ORCID ID: 0000-0002-0802-3552],
Senior Lecturer

[1]Taras Shevchenko National University of Kyiv, Kyiv,
[2]National University of Water and Environmental Engineering, Rivne

MODEL OF INTELLIGENT TASK DISTRIBUTION IN SOFTWARE DEVELOPMENT TEAMS BASED ON NEURAL NETWORK ALGORITHMS

This article presents a model of intelligent task distribution for a development team, built on the basis of neural networks. Information about

the available skills, current employment and previous performance of individual team members are automatically taken into account when making decisions about assigning new tasks. The architecture of the intelligent system is modular, for the client part of the task distribution system built on the basis of the proposed model, it is proposed to use ASP.Core, for machine learning – Python API. The neural model was trained on historical data on the performance of tasks in development teams, as well as project requirements. The results obtained confirm that the proposed model significantly increases the efficiency of task distribution compared to the traditional manual approach. The model allows for a more even distribution of the workload among individual team members and minimizes the effort spent on developing team solutions. Team productivity increases due to an increase in the speed of task execution, a decrease in the overload of individual developers, taking into account the competence and previous experience of performers, improving deadline forecasting, reducing the risk of delays or failure to complete assigned tasks.

***Keywords*:** system; intelligent system; system design; model; neural networks; machine learning.

Отримано: 17 червня 2025 року
Прорецензовано: 02 вересня 2025 року
Прийнято до друку: 25 вересня 2025 року