Тернопільський національний технічний університет імені Івана Пулюя
Міністерство освіти і науки України

Кваліфікаційна наукова праця
на правах рукопису

НЕБЕСНИЙ РУСЛАН МИХАЙЛОВИЧ

УДК 005.8:004.9:[005.336.5:005.743-027.522]

ДИСЕРТАЦІЯ

РЕКОМЕНДАЦІЙНА СИСТЕМА ФОРМУВАННЯ КОМАНД ВИКОНАВЦІВ З ВІДПОВІДНИМИ ФАХОВИМИ КОМПЕТЕНТНОСТЯМИ

122 – Комп’ютерні науки
12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ Р. М. Небесний

Науковий керівник – Кунанець Наталія Едуардівна, доктор наук із соціальних комунікацій, професор

Тернопіль – 2023

АНОТАЦІЯ

Небесний Р.М. Рекомендаційна система формування команд виконавців з відповідними фаховими компетентностями. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки. Тернопільський національний технічний університет імені Івана Пулюя, Тернопіль, 2023.

Підготовка здійснювалась на кафедрі комп'ютерних наук Тернопільського національного технічного університету імені Івана Пулюя Міністерства освіти і науки України.

В умовах постійних технологічних трансформацій та зростання ринкової конкуренції, рекомендаційна система, зорієнтована на формування високофахових проєктних команд стає стратегічним інноваційним інструментом. Забезпечуючи точний відбір претендентів з відповідними фаховими компетентностями, вона сприяє оперативному реагуванню на виклики та сприяє успішному завершенню завдань, підвищуючи при цьому загальну продуктивність команди проєкту.

Метою дисертаційної роботи є розроблення методів і засобів реалізації процедур формування команд для успішного виконання ІТ проєктів. Для досягнення мети необхідно було проаналізувати підходи щодо формування команд виконавців для реалізації ІТ проєкту, розробити комплекс формалізмів для створення концептуальних моделей цілевизначального та рольового підходів, ізоморфної, експертної, колегіальної, розробницької, проектної, аналітичної, інтеграційної, інноваційної структур команд, розробити модель ізоморфної структури команди з використанням формалізмів теорії графів, розробити інформаційну технологію відбору претендентів з певними компетентностями для успішного виконання ІТ проєкту, розробити поведінкову модель команди проєкту як сукупний рух імітованої зграї, подаючи взаємодії членів команди за допомогою ройового алгоритму; побудувати архітектуру рекомендаційної системи відбору претендентів для формування ефективної команди розробників, розробити

рекомендаційну систему відбору претендентів в команду з певними компетентностями, що використовує гібридний метод генерування рекомендацій.

У вступі аргументовано актуальність проведення наукового дослідження, вказано на зв'язок роботи із науково-дослідними темами, сформульовано мету та визначено завдання, визначено об’єкт та предмет дослідження, подано опис методів, що використовувались для досягнення поставленої мети. Подано відомості щодо наукової новизни, практичного значення отриманих результатів та особистого внеску автора у розроблення нових моделей та інформаційних технологій, а також у реалізацію прикладного програмного забезпечення. Розкрито інформацію про апробацію та опублікування отриманих в процесі дослідження результатів та їх важливості для теорії та практики.

У першому розділі проаналізовано методи формування команд для виконання ІТ проєктів, які ґрунтуються на аналізі компетентностей та навичок претендентів, врахуванні їхнього попереднього фахового досвіду та успішних проєктних реалізацій. Обґрунтовано необхідність використання сучасних підходів в розробленні рекомендаційних систем, що передбачає застосування алгоритмів машинного навчання, методів та засобів штучного інтелекту. Зафіксовано необхідність оцінювання здібностей та врахування особистих якостей претендентів, що є ключовими елементами у процесах формування успішних команд для втілення ІТ проєктів. Відзначено, що для забезпечення ефективної взаємодії членів команди сучасні інноваційні підходи обов’язково включають процедури аналізу не лише їх технічних, а й комунікаційних навичок, що дозволяє максимально використовувати творчий потенціал учасників та досягати гарантованого успіху в реалізації сформованих завдань.

Проведений ґрунтовний аналіз проблемної галузі дозволив сформулювати постановку задач дослідження, які передбачають виконання аналізу існуючих підходів до формування команд; дослідити методи, які використовуються при розробленні рекомендаційних систем; проаналізувати параметри ІТ середовища, яке доцільно сформувати у «розумному місті»; дослідити підходи для залучення випускників шкіл до навчання за ІТ спеціальностями: мотиваційні чинники,

обрання закладу вищої освіти для здобуття фаху, формування навчального контенту, що є передумовою формування в майбутньому ефективних команд для реалізації ІТ проєктів; формалізувати ряд існуючих підходів до створення команд; обрати методи аналізу інформації та генерування рекомендацій; розробити інформаційну технологію підбору претендентів у команду виконавців з відповідними фаховими компетентностями та сформувати архітектуру рекомендаційної системи.

У другому розділі проведено аналіз ІТ ринку «розумного міста», що вимагає дослідження ряду аспектів, таких як: технологічні інновації, тренди, ключові гравці, можливості та виклики, збір та аналіз даних, аналіз технологічних та інноваційних трендів та інше. На основі відкритих даних про м. Тернопіль проаналізовано успішність випускників шкіл міста в контексті вступу в заклади вищої освіти на спеціальності ІТ галузі та мотиваційні чинники, які цьому сприяли. Запропоновано методи комплексного аналізу нахилів абітурієнтів до ІТ профілю. Обгрунтовано твердження, що формування мотиваційної платформи майбутнього ІТ фахівця є важливим суспільним завданням, вирішення якого сприяє формуванню позитивного ставлення до навчання та майбутньої успішної роботи в сфері ІТ. При цьому враховувались окремі умови часової невизначеності, оскільки переважна більшість факторів за природою мають довгостроковий вплив. Використання методології когнітивного моделювання дозволило провести аналіз сили та спрямованості зовнішнього впливу на об’єкт, визначенню провідних факторів для приведення його в цільовий стан. Передумовою для вибору базисних факторів став етап вербального соціально-комунікаційного моделювання середовища «розумного міста» та застосування методу експертних оцінок для визначення ваг факторів впливу. Після отримання мотиваційних стимулів щодо обрання фаху, абітурієнт приймає рішення щодо вибору вищого навчального закладу. Це рішення ухвалюється шляхом оцінювання та рейтингування доступних варіантів.

Запропоновано використовувати алгоритми, що базуються на методах машинного навчання для виявлення залежностей між змістом навчальних

дисциплін та описами вимог до наявних вакансій на ІТ ринку, оскільки системне узгодження цих чинників сприяє загалом здобуттю студентами компетентностей, які необхідні для успішного проходження ними відбору до складу команду ІТ проєкту.

У третьому розділі запропоновано оригінальні концептуальні моделі цілевизначального та рольового підходів до формування структури команд виконавців ІТ-проєкту, методи оптимального розподілу ролей, які вирішуються як задачі лінійного програмування або з використанням генетичних алгоритмів. Формалізація підходів до формування структури команди проєкту допомагає забезпечити збалансований та ефективний відбір виконавців для успішного виконання завдань проєкту.

Розроблено концептуальні моделі ізоморфної, експертної, колегіальної, розробницької, проєктної, аналітичної, інтеграційної та інноваційної структур команди проєкту. Модель ізоморфної структури команди подано за допомогою формалізмів теорії графів, де вузли подають ролі або конкретних учасників команди, а ребра - зв'язки або взаємодії між ними. Модель, заснована на рольовому підходу до формування команди побудовано з використанням методів лінійного програмування. Запропоновано підхід до розрахунку функції корисності кожного члена команди для пошуку оптимального розподілу ролей в команді. Подано метод формування команди проєкту з використанням генетичного алгоритму, а оптимального її складу та розподілу ролей у команді з використанням формалізмів нейронних мереж. Розроблено метод, який дозволяє максимізувати продуктивність команди з використанням ітераційного та кваліативного аналізу.

Запропоновано поведінкову модель команди як сукупний рух імітованої зграї, використовуючи процедури моделювання взаємодії членів команди за допомогою ройового алгоритму. Розроблено метод формування команди за аналогією зграї з використанням агентних моделей, де кожен член команди виступає як агент зі своїми особистісними характеристиками та правилами поведінки, моделюючи взаємодію між агентами за допомогою графової агентної моделі, де ребра графа представляють комунікаційні зв'язки, а вузли - агентів.

У четвертому розділі проаналізовано доцільність використання портфельного управління людськими ресурсами проєктних команд, визначено які аспекти такого управління можна реалізувати, використовуючи рекомендаційну систему. Запропоновано використовувати метод аналізу ієрархій та експертного оцінювання для побудови ієрархії претендентів в команду та концептуальну модель процедури управління людськими ресурсами в команді, яка покладена в основу розробленої рекомендаційної системи.

На основі проведеного аналізу підходів до формування високофахових команд для реалізації ІТ проєктів сформовані первинні та вторинні вимоги до рекомендаційної системи для підбору членів команди проєкту з відповідними фаховими компетентностями. Проаналізовано широкий спектр засобів побудови рекомендаційної системи, її функціонал та обґрунтовано системні переваги. Результатом практичного втілення є розроблення мультиплатформної рекомендаційної системи, яка може функціонувати в різних операційних середовищах на базі широкого спектру пристроїв.

Наукова новизна отриманих результатів:

вперше запропоновано:

комплекс формалізмів, який покладено в основу процесів створення концептуальних моделей цілевизначального та рольового підходів, ізоморфної, експертної, колегіальної, розробницької, проектної, аналітичної, інтеграційної, інноваційної структур команди, що дозволило спростити процедури побудови ефективних команд виконавців ІТ проектів, які формуються на платформі рекомендаційної системи;

поведінкова модель команди проєкту як сукупний рух імітованої зграї, подаючи взаємодії членів команди за допомогою ройового алгоритму, що дозволило розробити метод формування команди з використанням агентного підходу, що забезпечило проведення серії симуляційних процесів для формування ефективних команд для проєктів;

удосконалено

модель ізоморфної структури команди з використанням формалізмів теорії графів, що дозволило візуалізувати процеси побудови команд проєктів та формування оптимальної структури команди виконавців проєктів;

метод формування команди з використанням генетичного алгоритму та оптимального розподілу ролей у команді з використанням нейронних мереж, що дозволило здійснювати пошук рішення щодо оптимального складу команди, базуючись на певній популяції претендентів;

архітектуру рекомендаційної системи відбору претендентів з заданою системою компетентностей для створення ефективної команди розробників;

інформаційну технологію відбору претендентів до складу команди з певними компетентностями для успішного виконання IT проекту.

Результати дисертаційних досліджень опубліковані та апробовані в 16 наукових працях, в тому числі 5 статтях у фаховий виданнях України, одна з яких індексується в Web of Science та в 11-ти матеріалах наукових конференцій з яких у виданнях, що індексуються в міжнародних наукометричних базах 5 -в Scopus та 2 - в Web of Science.

Ключові слова: модель, проєкт, рекомендаційні системи, команда, компетентності, людські ресурси, інформаційні технології, управління проєктами, smart city, генетичний алгоритм, нейронні мережі, машинне навчання, ройові алгоритми, software, productivity.

ABSTRACT

Nebesnyi R.M. Recommendation system for the teams formation of performers with relevant professional competences. – Qualifying scientific study on the rights as a manuscript.

Dissertation for obtaining the scientific degree of Doctor of Philosophy in specialty 122 – Computer Science. Ternopil Ivan Puluj National Technical University, Ternopil, 2023.

The training was conducted on the basis of the Department of Computer Sciences of the Ternopil Ivan Puluj National Technical University of the Ministry of Education and Science of Ukraine

In the conditions of constant technological transformations and the growth of market competition, the recommendation system, focused on the formation of highly specialized project teams, becomes a strategic innovation tool. By ensuring the accurate selection of applicants with relevant professional competencies, it facilitates prompt response to challenges and facilitates the successful completion of tasks, while increasing the overall productivity of the project team.

The purpose of the dissertation is to develop methods and means of implementing team formation procedures for the successful implementation of IT projects. To achieve the goal, it was necessary to analyze the approaches to the formation of teams of performers for the implementation of the IT project, to develop a complex of formalisms for the creation of conceptual models of goal-setting and role-based approaches, isomorphic, expert, collegial, development, project, analytical, integration, innovative team structures, to develop a model of isomorphic structure teams using the formalisms of graph theory, to develop an information technology for selecting applicants with certain competencies for the successful implementation of an IT project, to develop a behavioral model of the project team as a collective movement of a simulated swarm, presenting the interactions of team members using a swarm algorithm; to build the architecture of the recommendation system for the selection of applicants for the formation of an effective team of developers, to develop a recommendation system for

the selection of applicants into a team with certain competencies, which uses a hybrid method of generating recommendations.

In the introduction, the relevance of conducting scientific research is argued, the connection of the work with scientific research topics is indicated, the goal and task are defined, the object and subject of the research are established, a description of the methods used to achieve the goal is given. Information on scientific novelty, the practical significance of the obtained results, and the author's personal contribution to the development of new models and information technologies, as well as to the implementation of application software, is provided. Information about the approbation and publication of the results obtained in the research process and their importance for theory and practice is disclosed.

The first section analyzes the methods of forming teams for the implementation of IT projects, which are based on the analysis of the competencies and skills of the applicants, taking into account their previous professional experience and successful project implementations. The need to use modern approaches in the development of recommendation systems, which involves the use of machine learning algorithms, methods and means of artificial intelligence, is substantiated. The need to assess the abilities and take into account the personal qualities of the applicants, which are key elements in the processes of forming successful teams for the implementation of IT projects, was recorded. It was noted that to ensure effective interaction of team members, modern innovative approaches necessarily include procedures for analyzing not only their technical, but also communicative skills, which allows to maximize the creative potential of participants and achieve guaranteed success in the implementation of established tasks.

The conducted thorough analysis of the problem area made it possible to formulate research objectives that involve the analysis of existing approaches to team formation; to investigate the methods used in the development of recommendation systems; analyze the parameters of the IT environment that should be formed in a "smart city"; to investigate approaches to attract school graduates to study in IT specialties: motivational factors, choosing a higher education institution to acquire a specialty, formation of educational content, which is a prerequisite for the formation of effective teams for the

implementation of IT projects in the future; formalize a number of existing approaches to creating teams; choose methods of information analysis and generation of recommendations; to develop information technology for the selection of applicants for the team of performers with the appropriate professional competences and to form the architecture of the recommendation system.

In the second chapter, an IT analysis of the "smart city" market is carried out, which requires the study of a number of aspects, such as: technological innovations, trends, key players, opportunities and challenges, data collection and analysis, analysis of technological and innovation trends, etc. On the basis of open data about the city of Ternopil, the success of graduates of the city's schools in the context of admission to higher education institutions in the IT field and the motivational factors that contributed to this were analyzed. Methods of comprehensive analysis of applicants' inclinations towards the IT profile are proposed. The statement that the formation of the motivational platform of the future IT specialist is an important social task, the solution of which contributes to the formation of a positive attitude to education and future successful work in the field of IT, is substantiated. At the same time, certain conditions of temporal uncertainty were taken into account, since the vast majority of factors by nature have a long-term impact. The use of the cognitive modeling methodology made it possible to analyze the strength and direction of the external influence on the object, to determine the leading factors for bringing it to the target state. The stage of verbal social-communication modeling of the "smart city" environment and the application of the method of expert evaluations to determine the weights of influencing factors became the prerequisite for the selection of basic factors. After receiving motivational incentives for choosing a major, the applicant makes a decision about choosing a higher educational institution. This decision is made by evaluating and ranking the available options.

It is proposed to use algorithms based on machine learning methods to identify dependencies between the content of educational disciplines and descriptions of requirements for available vacancies on the IT market, since the systematic coordination of these factors contributes to the overall acquisition by students of the competencies necessary for their successful selection to the IT project team .

In the third section, original conceptual models of goal-setting and role-based approaches to the formation of the structure of teams of IT project executors, methods of optimal distribution of roles, which are solved as problems of linear programming or using genetic algorithms, were proposed. The formalization of approaches to the formation of the project team structure helps to ensure a balanced and effective selection of executors for the successful completion of project tasks.

Conceptual models of isomorphic, expert, collegial, developmental, project, analytical, integrative and innovative structures of the project team were developed. The model of the isomorphic structure of the team is presented using the formalisms of graph theory, where nodes represent roles or specific members of the team, and edges represent connections or interactions between them. The model based on the role-based approach to team formation was built using linear programming methods. An approach to calculating the utility function of each team member is proposed to find the optimal distribution of roles in the team. The method of forming the project team using the genetic algorithm, and its optimal composition and distribution of roles in the team using the formalisms of neural networks were presented. A method has been developed that allows to maximize team productivity using iterative and qualitative analysis.

A behavioral model of the team is proposed as a collective movement of a simulated flock, using procedures for modeling the interaction of team members using the swarm algorithm. A team formation method based on the analogy of a flock using agent models has been developed, where each team member acts as an agent with his personal characteristics and rules of behavior, modeling the interaction between agents using a graph agent model, where the edges of the graph represent communication links, and the nodes represent agents.

The fourth chapter analyzes the expediency of using portfolio management of human resources of project teams, determines which aspects of such management can be implemented using a recommendation system. It is proposed to use the method of analysis of hierarchies and expert evaluation to build a hierarchy of applicants to the team and a conceptual model of the human resources management procedure in the team, which is the basis of the developed recommendation system.

On the basis of the analysis of approaches to the formation of highly specialized teams for the implementation of IT projects, the primary and secondary requirements for the recommendation system for the selection of project team members with appropriate professional competences were formed. A wide range of means of building a recommendation system, its functionality, and system advantages are substantiated. The result of practical implementation is the development of a multi-platform recommendation system that can function in various operating environments based on a wide range of devices.

Scientific novelty of the obtained results:

first proposed:

a complex of formalisms, which is the basis of the processes of creating conceptual models of goal-setting and role-based approaches, isomorphic, expert, collegial, developmental, project, analytical, integrative, and innovative team structures, which made it possible to simplify the procedures for building effective teams of IT project executors, which are formed on the platform of recommendation systems;

a behavioral model of the project team as a collective movement of a simulated flock, presenting the interactions of team members using a swarm algorithm, which allowed to develop a method of team formation using an agent approach, which ensured a series of simulation processes for the formation of effective teams for projects;

improved

a model of the isomorphic structure of the team using the formalisms of graph theory, which made it possible to visualize the processes of building project teams and the formation of the optimal structure of the team of project executors;

the method of team formation using a genetic algorithm and the optimal distribution of roles in the team using neural networks, which made it possible to find a solution for the optimal composition of the team, based on a certain population of applicants;

the architecture of the recommendation system for the selection of applicants with a given system of competencies to create an effective team of developers;

information technology for the selection of applicants for the team with certain competencies for the successful implementation of the IT project.

The results of the dissertation research were published and approbated in 16 scientific papers, including 5 articles in specialized publications of Ukraine, one of which is indexed in Web of Sciece and in 11 materials of scientific conferences, of which in publications indexed in international scientometric databases, 5 - in Scopus and 2 - in Web of Sciece.

Keywords: model, project, recommendation systems, team, competences, human resources, information technologies, project management, smart city, genetic algorithm, neural networks, machine learning, swarm algorithms, software, productivity.

ПЕРЕЛІК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Kunanets N. E., Nazaruk M. V., Nebesnyi R. M., and Pasichnyk V. V. Information technology of personalized choice of profession in smart cities, *ITLT*, vol. 65, no. 3, pp. 277–290, Jul. 2018. https://doi.org/10.33407/itlt.v65i3.2172 (Web of Science (ESCI), USA)
2. Kunanets Natalia, Pasichnyk Volodymyr, Nebesnyi Ruslan, Nazaruk Mariia Аналіз вибору ІТ спеціальностей учнями випускних класів на прикладі м. Тернополя *Вісник Національного університету" Львівська політехніка". Інформаційні системи та мережі* 2019. №6. С. 79 - 89 https://doi.org/10.23939/sisn2019.02.079 (Index Copernicus)
3. Pasichnyk , V., Kunanets , N., Artemenko , O., Fedorka , P., & Nebesnyi , R. Using mobile crowd sensing for social distancing real-time navigation. *Управління розвитком складних систем*. 2021. №47. С. 57–62. https://doi.org/10.32347/2412-9933.2021.47.57-62 (Index Copernicus)
4. Калинич Ю., Білак Ю.Ю., Небесний Р., Федорка П. Аналіз процесів формування симуляцій з використанням графічного процесора *Вісник Національного університету" Львівська політехніка". Інформаційні системи та мережі*. 2022. №11. С.110-126 https://doi.org/10.23939/sisn2022.11.110 (Index Copernicus)
5. Кунанець Н. Е., Небесний Р. М., Мацюк О. В. Особливості формування цілей соціальних та соціокомунікаційних складових у проектах "Розумних міст" *Вісник Національного університету "Львівська політехніка". Серія : Інформаційні системи та мережі*. 2016. Випю 854. С. 257-274. - Режим доступу: http://nbuv.gov.ua/UJRN/VNULPICM_2016_854_26 (Index Copernicus)

Наукові праці, які засвідчують апробацію матеріалів дисертації:

1. Nebesnyi R., Kunanets N., Vaskiv R. and Veretennikova N. Formation of an IT Project Team in the Context of PMBOK Requirements *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, LVIV,

Ukraine, 2021, pp. 431-436, https://doi.org/10.1109/CSIT52700.2021.9648612. (Index Scopus)

2. Nebesnyi R., Pasichnyk V., Kunanets N., Veretennikova N. and Kunanets O. Formation of IT Project Implementation Team. *2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT)*, Zbarazh, Ukraine, 2020, pp. 203-206, https://doi.org/10.1109/CSIT49958.2020.9322005. (Index Scopus)
3. Matsyuk, O., Nazaruk, M., Turbal, Y., Veretennikova, N., Nebesnyi, R. Information Analysis of Procedures for Choosing a Future Specialty. *Advances in Intelligent Systems and Computing III. CSIT* 2018. vol 871. Springer, Cham. https://doi.org/10.1007/978-3-030-01069-0_26 (Index Scopus)
4. Pasichnyk V., Nazaruk M., Kunanets N., Veretennikova N. and Nebesnyi R. Information Analysis of Procedures for Choosing a Future Specialty Using Cognitive Cards. *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 2018, pp. 215-220, https://doi.org/10.1109/STC-CSIT.2018.8526626 (Index Scopus)
5. Pankiv Y., Kunanets N., Artemenko O., Veretennikova N. and Nebesnyi R. Project of an Intelligent Recommender System for Parking Vehicles in Smart Cities. *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, LVIV, Ukraine, 2021, pp. 419-422, doi: 10.1109/CSIT52700.2021.9648687 (Index Scopus)
6. Дуда О.М., Кунанець Н.Е., Липак Г.І., Мацюк О.В., Небесний Р.М., Пасічник В.В. Консолідація інформаційних ресурсів соціокомунікаційного середовища в проектах “Розумне місто” *System Analysis and Information Technologies 18-th International Conference SAIT 2016* Kyiv, Ukraine, May 30 – June 2, 2016 p.214 ISBN 978-966-2748-83-3
7. Кунанець Н. Небесний Р. Людський ресурс “розумного міста” та відкриті дані *Матеріали V науково-технічної конференції „Інформаційні моделі, системи та технології“*, 1-2 лютого 2018 року. Т. : ТНТУ, 2018. С. 41–42. (Секція 2. Інформаційні системи).

8. Кунанець Н.Е., Небесний Р.М., Мацюк О.В., Пасічник В.В. Соціокомунікаційна складова у портфелях проектів «Розумних міст». *Управління проектами: стан та перспективи : матеріали XII Міжнародної науково-практичної конференції.* Миколаїв : НУК, 13-16 вересня 2016, ст. 84-85 https://nuos.edu.ua/wp-content/uploads/2021/12/Konf_Upravlenie_Proekt.pdf
9. Кунанець Н., Кунанець О., Небесний Р., Пасічник В. Освітня соціокомунікаційна складова у портфелях проектів «Розумних міст»: досвід Великобританії. *Proceedings of the tenth international scientific-practical conference «Internet-Education-Science» (IES-2016)*, Vinnytsia, 11-14 October, 2016. Vinnytsia : VNTU, 2016. С. 192-194.
10. Кунанець Н., Пасічник В., Небесний Р. Інформаційні технології прогнозування розвитку освітнього середовища «розумного міста» *Proceedings of the tenth international scientific-practical conference «Internet-Education-Science» (IES-2016)*. Vinnytsia, 11-14 October, 2016. Vinnytsia : VNTU, 2016. С. 188-189.
11. Мартинюк С. В., Небесний Р. М. Розробка функціонуючої структури програмного консолідованого ресурсу *Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“*, 16-17 листопада 2017 року. Т. : ТНТУ, 2017. Том 2. С. 112–113. (Комп’ютерно-інформаційні технології та системи зв’язку).

Наукові праці, які додатково відображають наукові результати дисертації:

1. Кормило І. В., Небесний Р. М. Побудова інформаційної технології візуалізації інформаційних ресурсів *Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“*, 16-17 листопада 2017 року. Т. : ТНТУ, 2017. Том 2. С. 96–97. (Комп’ютерно-інформаційні технології та системи зв’язку).

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UML – (Unified Modeling Language) уніфікована мова графічного представлення та об’єктного моделювання в області розробки програмного забезпечення парадигми об’єктно-орієнтованого програмування.

Алгоритм – набір інструкцій, які описують порядок виконання дій.

ІТ – інформаційні технології.

ОС – операційна система.

ПЗ – програмне забезпечення.

ПС – програмна система, комплекс програмного забезпечення.

ЗМІСТ

ВСТУП

Актуальність теми.

В умовах постійних технологічних трансформацій та зростання ринкової конкуренції, виникає потреба вирішувати ряд специфічних задач оперативного підбору нових високофахових членів в команду проєкту, при цьому враховуючи психологічну сумісність, що дозволяє уникнути конфліктів і стресів при реалізації проєкту. При цьому виникає потреба пошуку, накопичення, опрацювання даних про вимоги до ролей в команді, характеристики претендентів. Для забезпечення процедур ефективного опрацювання цих даних необхідні інформаційні технології пошуку та порівняння даних у реальному часі з використанням нових і вдосконалених методів, моделей. Ефективне виконання процесів опрацювання даних потребує розроблення нових моделей та методів, які дозволили б формувати певний тип команди відповідно до потреб проєкту.

Забезпечення таких вимог можливе при розробленні рекомендаційної системи, зорієнтованої на формування високофахових проєктних команд, яка має стати стратегічним інноваційним інструментом. Забезпечуючи точний відбір претендентів з відповідними фаховими компетентностями, вона сприяє оперативному реагуванню на виклики та сприяє успішному завершенню завдань, підвищуючи при цьому загальну продуктивність команди проєкту. Рекомендаційна система повинна допомагати автоматизувати та оптимізувати процес відбору претендентів, що призводить до більш ефективного та швидкого відбору команди, враховуючи при цьому індивідуальні навички, досвід та властивості претендентів, що сприяє створенню більш збалансованих та компетентних команд. Застосування рекомендаційних систем може допомогти уникнути людських помилок або підвищити об'єктивність при виборі претендентів на певні ролі в команді. Оптимальний підбір працівників з врахуванням їхніх індивідуальних особливостей та вмінь може позитивно позначитися на роботі команди та сприяти більшій задоволеності від роботи. Рекомендаційна система може допомагати адаптувати склад команди до змін у проекті, дозволяючи швидко замінювати або додавати нових працівників в залежності від потреб. Автоматизований відбір претендентів

за допомогою рекомендаційної системи дозволяє зекономити час та ресурси, які раніше витрачались на ручний відбір та оцінку кандидатів. Оптимальний підбір може допомогти створити команду з різноманітними навичками та підходами, що сприяє інноваціям та творчому розвитку проєкту.

Загалом, розроблення рекомендаційної системи для підбору претендентів є актуальним, оскільки вона вирішує практичні завдання з покращення управління персоналом та оптимізації роботи команд в галузі ІТ.

Зв'язок роботи з науковими програмами, планами, темами.

Дисертаційне дослідження проводилось згідно з планами науково-дослідних робіт Науково-дослідної лабораторії «Розумне місто Тернопіль», зокрема напряму розроблення інформаційних технологій прогнозування розвитку освітнього середовища Smart City. Крім того в рамках ІІ етапу теми «Комплекс моделей формування та розвитку соціокомунікаційного середовища міста» (2017–2019 рр., державний реєстраційний № 0117U002240) запропоновано засоби та методи моделювання соціокомунікаційних процесів; «Класи інформаційних технологій в проектах «Розумне місто» » (2017–2019 рр., державний реєстраційний № 0117U002241) дисертантом проаналізовано технології, які дозволяють аналізувати людські ресурси та розроблено функціональні моделі соціокомунікаційного середовища «розумного міста».

Мета і завдання дослідження. Метою дисертаційної роботи є розроблення методів і засобів реалізації процедур формування команд для успішного виконання ІТ проектів.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати підходи щодо формування команд виконавців для реалізації ІТ проєкту,
- розробити комплекс формалізмів для створення концептуальних моделей цілевизначального та рольового підходів, ізоморфної, експертної, колегіальної, розробницької, проектної, аналітичної, інтеграційної, інноваційної структур команд,

- створити модель ізоморфної структури команди з використанням формалізмів теорії графів,
- побудувати інформаційну технологію відбору претендентів з певними компетентностями для успішного виконання ІТ проєкту,
- розробити поведінкову модель команди проєкту як сукупний рух імітованої зграї, подаючи взаємодії членів команди за допомогою ройового алгоритму;
- побудувати архітектуру рекомендаційної системи відбору претендентів для формування ефективної команди розробників,
- розробити рекомендаційну систему відбору претендентів в команду з певними компетентностями, що використовує гібридний метод генерування рекомендацій.

Об'єктом дослідження є процеси аналізу та оцінювання компетентностей претендента для команди реалізації ІТ проекту.

Предметом дослідження є методи та засоби вибору претендентів до команди ІТ проекту.

Методи дослідження: Для розв'язання поставлених в дисертаційній роботі завдань використані наступні методи: теорії рекомендаційних систем для побудови моделей архітектури, методи класифікації для здійснення розподілу компетентностей, системний підхід, об'єктно-орієнтовані методи аналізу та синтезу програмних продуктів, уніфіковану мову моделювання UML (Unified Modeling Language), елементи теорії графів, методи ройового алгоритму, машинного навчання, генетичний алгоритм.

Наукова новизна одержаних результатів. В результаті проведення досліджень одержано такі нові результати:

вперше запропоновано:

- комплекс формалізмів, який покладено в основу процесів створення концептуальних моделей цілевизначального та рольового підходів, ізоморфної, експертної, колегіальної, розробницької, проектної, аналітичної, інтеграційної, інноваційної структур команди, що дозволило спростити процедури побудови

ефективних команд виконавців ІТ проектів, які формуються на платформі рекомендаційної системи;

- поведінкова модель команди проєкту як сукупний рух імітованої зграї, подаючи взаємодії членів команди за допомогою ройового алгоритму, що дозволило розробити метод формування команди з використанням агентного підходу, що забезпечило проведення серії симуляційних процесів для формування ефективних команд для проєктів;

удосконалено

- модель ізоморфної структури команди з використанням формалізмів теорії графів, що дозволило візуалізувати процеси побудови команд проєктів та формування оптимальної структури команди виконавців проєктів;

- метод формування команди з використанням генетичного алгоритму та оптимального розподілу ролей у команді з використанням нейронних мереж, що дозволило здійснювати пошук рішення щодо оптимального складу команди, базуючись на певній популяції претендентів;

- архітектуру рекомендаційної системи відбору претендентів з заданою системою компетентностей, що сприяло створенню ефективної команди розробників;

- інформаційну технологію відбору претендентів до складу команди з певними компетентностями для успішного виконання ІТ проекту, що дозволило автоматизувати та оптимізувати процедуру підбору складу команди.

Практична значущість результатів дослідження –

Полягає у розробленні інформаційної технології відбору претендентів до складу команди з певними компетентностями для успішного виконання ІТ проекту та рекомендаційної системи відбору претендентів в команду з певними компетентностями, що використовує гібридний метод генерування рекомендацій.

Особистий внесок здобувача. Усі наукові результати дисертаційної роботи отримані автором самостійно.

Апробація результатів дисертації.

Результати наукових досліджень неодноразово обговорювалися на міжнародних та всеукраїнських науково-технічних конференціях, зокрема на: [78] 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), LVIV, 2021 (Index Scopus); [45] 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), LVIV, 2021 (Index Scopus); [79] 2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT), Zbarazh, 2020 (Index Scopus); [75] Advances in Intelligent Systems and Computing III: Selected Papers from the International Conference on Computer Science and Information Technologies, CSIT 2018, September 11-14, Lviv (Index Scopus); [72] 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, 2018 (Index Scopus); [57] Матеріали науково-технічної конференції "Інформаційні моделі, системи та технології", 1-2 лютого 2018 ТНТУ, 2018; [86] Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій", 16-17 листопада 2017 ТНТУ; [68] System Analysis and Information Technologies 18-th International Conference SAIT 2016 Kyiv, May 30 – June 2 2016; [8] Управління проектами: стан та перспективи : матеріали XII Міжнародної науково-практичної конференції Миколаїв НУК, 13-16 вересня 2016; [58] Proceedings of the tenth international scientific-practical conference «Internet-Education-Science» (IES-2016), Vinnytsia, 11-14 October VNTU; [74] Proceedings of the tenth international scientific-practical conference «Internet-Education-Science» (IES-2016), Vinnytsia, 11-14 October VNTU.

Також результати дисертаційних досліджень доповідалося на наукових семінарах кафедри компютерних наук ТНТУ ім. І. Пулюя (2020–2023 рр.)

Публікації. За результатами виконаних досліджень опубліковані та апробовані в 16 наукових працях, в тому числі 5 статтях у фаховий виданнях України, одна з яких індексується в Web of Science та в 11-ти матеріалах наукових

конференцій з яких у виданнях, що індексуються в міжнародних наукометричних базах 5 -в Scopus та 2 - в Web of Science.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел з 86 найменувань та 4 додатків, загальний обсяг – 253 сторінки, основний текст складає 186 сторінок, 36 рисунків, 8 таблиць.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗРОБЛЕННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ ПРОЕКТАМИ

1.1 Методи формування команд для виконання ІТ проектів

ІТ-сфера в Україні — одна з найдинамічніших і найперспективніших галузей в містах. В українській ІТ-індустрії працює понад 190 000 фахівців як подано у звіті IT Ukraine Association. Українські фахівці створюють, просувають й продають ІТ-продукти на глобальних ринках. При цьому, у ІТ галузі останніми роками набуває особливої актуальності використання методології управління проектами, що регламентується PMBOK [1], в контексті необхідності створення ефективної команди та налагодження комунікації між її членами.

Вперше термін «команда» почав застосовуватися в спортивній галузі, і розглядався як складова частина в тайм-менеджменту. В сучасному суспільстві поняття команди значно розширило множину предметних областей, в яких використовується. Дослідження суті команди та її типів приваблює науковців. Автори окреслюють переваги, які отримають від процесу командотворення, а також наголошують на проблемах, які виникають при командній роботі. Дослідники намагаються окреслити відмінні риси команди, особливості командної роботи та визначити ознаки ефективної команди[2].

У дослідників не виникає сумнівів, що команди можуть бути дуже ефективними, коли їх члени працюють разом. І багато експертів з бізнесу визнали, що робота в команді, хоч і складна у досягненні та підтримці, може бути дуже ефективною. Основний акцент, що ставиться у бізнесі на концепції колективної роботи та міркуваннях щодо такої співпраці, все ще відносно новий. Хоча лідери ІТ галузі досить довго проповідують роботу в команді, уявлення про співпрацю людей ніколи не було настільки популярним, як сьогодні.

Питання формування команд та їх характерні риси розглядалися в працях Chesbrough [3], Peterson [4]. Існують декілька принципів формування проєктних

команд для управління створення ІТ продукту: створення власних груп (група від замовника і група від виконавця) або створення єдиної команди [5].

Є декілька класичних підходів до впровадження проєктного підходу [6], а для його реалізації створюється команда. Формування команд щодо забезпечення їх функціональності може базуватися на різних засадах, які залежать від конкретної ситуації, цілей та особливостей проєкту. Ось чотири основних підходи до формування команд:

Функціональний підхід [7]: У цьому підході команда формується шляхом об'єднання спеціалістів, які мають спільні функціональні обов'язки або експертизу. Кожен член команди призначається для виконання конкретних завдань відповідно до його фахових знань і навичок. Цей підхід дозволяє досягати ефективності в різних областях діяльності та забезпечувати високу технічну компетентність.

Міжфункціональний підхід [8]. У цьому підході команда формується з представників різних функціональних областей або відділів організації. Такий підхід сприяє вирішенню комплексних завдань та співпраці між різними частинами організації. Члени команди можуть приносити різні погляди та ідеї, що сприяє інноваційності.

Проектний підхід [9]. У цьому підході команда формується специфічно для виконання певного проекту або завдання. Команда обирається на основі потреб проекту та включає фахівців з різних областей, які мають необхідні навички для досягнення цілей проекту. Після завершення проекту команда може розформовуватися.

Крос-функціональний підхід [10]. У цьому підході команда формується з представників різних функціональних областей та спеціалістів з різних компетенцій, необхідних для вирішення конкретної проблеми або завдання. Цей підхід підкреслює важливість комбінації різних знань та підходів для досягнення інтегрованих рішень.

Кожен з цих підходів має свої переваги та обмеження, і вибір конкретного підходу залежить від контексту, цілей та особливостей конкретної ситуації. Для команд у сфері інформаційних технологій притаманний підхід, який об’єднує всі

чотири види формування команди, саме так вдається виділити рольовий склад проєкту, визначити їхні функціональні зони, зони відповідальності та положення в проєкті для досягнення поставлених цілей. Для формування ІТ команди важливо при формуванні проєктної команди виділити існуючі проблеми [11], які необхідно вирішити, і рівень невизначеності на початок проєкту. Якщо необхідно сформувати команду для виконання проєкту, що реалізується силами кількох організацій, то належить вивчити потреби в рівні компетенцій, професійні характеристики, психологічний портрет, а також оцінити ролі учасників, їхню кількість і професійний рівень. Виокремлюють п’ять дій (етапів) для формування команди проєкту [12, 13]:

1. Аналіз потреб та завдань. Перший етап полягає у ретельному аналізі завдань, які необхідно виконати командою, та визначенні потрібних компетентностей. Розглядається детально, які навички, знання та досвід потрібні для успішного виконання завдань. Цей етап включає в себе оцінку того, які завдання повинна виконувати команда та які компетентності їй потрібні для успішного виконання цих завдань. На цьому етапі відбувається визначення завдань команди, зокрема з’ясовується, які конкретні завдання потрібно виконувати команді в контексті потреби організації, проекту чи задачі, які вони мають вирішити. Це може включати рутинні завдання, стратегічні проекти, впровадження нових рішень тощо. Далі відбувається визначення компетентностей виконавців. Для кожного завдання визначаються необхідні компетентності, навички, знання та досвід необхідні для ефективного виконання кожного завдання. Це може бути технічна експертиза, м'які навички, керівництво, спілкування та інше.

2. Оцінка потрібних ресурсів. Проводиться аналіз наявних у організації ресурсів. З’ясовується компетентності співробітників в контексті проєкту. Визначається можливість використання існуючих внутрішніх або зовнішніх ресурсів для виконання завдань. Відбувається створення матриці компетентностей [14], у якій для кожного завдання вказані необхідні компетентності. Це допомагає краще розуміти, які вимоги до навичок є для різних завдань. Визначаються додаткові компетентності з врахуванням можливість доповнити команду людьми,

які мають унікальні компетентності і можуть доповнити поточний склад команди. Це може збільшити варіативність підходів та рішень.

3. Планування навчання та розвитку. Визначається, які знання та навички необхідні членам команди для вирішення недостатності у компетентностях. Створюється план розвитку, який допомагає підготувати команду до виконання завдань. Аналіз потреб та завдань проєкту допоможе вам розробити чітку картину того, яка команда потрібна для досягнення поставлених цілей, і встановити фундамент для наступних кроків у формуванні команди.

4. Підбір членів команди. На цьому етапі важливо підібрати людей з відповідними компетентностями та вміннями, з врахуванням не лише технічних навичок, але й міжособистісних якостей, здатностей співпрацювати та відкритості до навчання. Підбір членів команди є важливим етапом у процесі формування команди. Від правильного вибору членів команди залежить ефективність її роботи та досягнення поставлених цілей. Враховуючи аналіз потреб та завдань команди, створюється детальний профіль кожного члена команди.

5. Формування командної культури [15,16]. Командна культура передбачає створення позитивної та сприяючої робочої атмосфери. Важливо забезпечити взаємодію, спільні цілі та цінності. Пропагування відкритості, співпраці та взаєморозуміння сприяє формуванню командної динаміки. Формування командної культури є ключовим аспектом створення команди, що ефективно працює. Культура визначає спосіб спілкування членів команди, подолання викликів. Формування командної культури - це процес, який потребує часу та зусиль, але він сприяє ефективності та згуртованості команди.

Для створення позитивної командної культури необхідно:

- Визначення завдань – відбувається декомпозиція робіт проєкту, визначається чорновий варіант плану проєкту, приблизне розуміння етапів роботи і завдань на ньому.

- Розуміння вимог проекту [17]: Перш за все досягається чіткого розуміння вимог та обсяг проекту. Це допомагає зорієнтуватися, які компетентності та навички потрібні для команди, визначитись які ролі (наприклад, проектний

менеджер, розробник, тестувальник тощо) будуть необхідні для успішного виконання проекту.

- Визначення умов і ресурсів – виявлення місць і умов роботи команди, розподіл ресурсів, визначення рамок.

- Визначення цінностей [18,19]. Встановлюються цінності, які визначають основні принципи командної культури, серед яких дотримання взаєморозуміння, відкритості, співпраці, інноваційності тощо.

- Створення місії та візії. Розробляється місія та візія роботи команди. Місія визначає, яку проблему або завдання команда прагне вирішити, а візія - які результати вона прагне досягнути.

- Залучення всіх членів до управління [20]. До обговорення та формування цінностей, місії та візії роботи команди залучаються всі її члени. Це допомагає відчуттю власної участі в прийнятті рішень та відповідальності за команду.

- Формування рольової моделі команди – за виділеними на першому етапі завданнями формується розуміння того, якими навичками та знаннями повинні володіти потенційні члени команди, в якій кількості і з якою кваліфікацією. Вибір моделі, за якою буде відбуватися взаємодія команди. Кожен член команди повинен знати свої ролі, обов'язки та відповідальність на них покладену. Це допомагає уникнути дублювання завдань, покращує координацію та підвищує продуктивність роботи. Встановлення ролей та відповідальності в команді є важливим кроком для забезпечення чіткої організації та координації робіт. Це допомагає уникнути дублювання зусиль, покращує співпрацю та забезпечує високий рівень продуктивності. Відбувається визначення ключових ролей, які необхідні для виконання завдань команди. Створюється опис обов'язків для кожної ролі, який надає чітке уявлення про те, які завдання, функції та сфери відповідальності належать певній ролі. На основі профілів та компетентностей [21] кожного члена команди, призначаються їх на певні ролі, при цьому враховується, які навички і досвід підходять для кожної з них. Ролі та обов'язки можуть змінюватися з часом в залежності від потреб команди та проекту. Важливо підтримувати гнучкість та готовність до змін. Кожен член команди повинен розуміти свої завдання та внесок

у загальний успіх проєкту. Ролі та сфери відповідальності періодично переглядаються, визначається їх відповідність потребам команди та її цілям. Встановлення ролей та сфери відповідальності сприяє зосередженню на конкретних завданнях та допомагає кожному члену команди розуміти, як вони вносять свій внесок у спільний успіх.

- Створення умов для комунікації та відкритості [22]. У середовищі проєкту створюється атмосфера відкритості та довіри, де члени команди можуть вільно висловлювати свої думки та ідеї. Важливо налаштувати членів команди активно спілкуватися та ділитися інформацією. Разом з членами команди розробляються норми та правила взаємодії. Це можуть бути регулярні зустрічі, спільні вирішення конфліктів, взаємна підтримка тощо. Відбувається формування позитивного середовища проєкту [23], сприятливе для роботи та спілкування. Розглядаються можливості для розвитку м’яких навичок. Навчається членів команди м'яким навичкам [24], таким як комунікація, управління, співпраця та робота в групі. Організовуються заходи із відзначення досягнень, успіхів команди. Це може бути внутрішнє визнання, нагороди або інші форми заохочень. Визначається характер взаємодії, встановлюється правила, як ролі будуть взаємодіяти між собою, яка інформація має передаватися. Разом з тим, забезпечується відкрита комунікація членів команди, не зважаючи на розподіл ролей.

- Навчання та розвиток [25]. Забезпечується навчання та можливості розвитку для членів команди, що сприятиме підвищенню рівня компетентностей та підтримці корпоративної культури. Розробляється плану розвитку [26] членів команди, які мають бажання розвивати свої навички, який допоможе їм зростати у команді проєкту.

- Розвиток команди [27]. Команда повинна постійно розвиватися. Організовуються навчання, тренінги та взаємний обмін знаннями. Підтримка особистісного та професійного зростання членів команди сприяє вдосконаленню результатів. Розвиток команди - це процес неперервного покращення навичок, співпраці та результатів, які досягає команда. Цей процес включає навчання, тренінги, підтримку та забезпечення умов для особистого та колективного росту.

Для підвищення результативності навчання проводиться оцінка потреб, розглядається, які навички та компетентності потребують розвитку в команді, оцінюється, які знання можуть підвищити ефективність її роботи. Для їх здобуття організовується навчання та навчальні програми для команди. Це може бути внутрішнє навчання, залучення зовнішніх експертів, відвідування конференцій та тренінгів. Використовуються рольові ігри та симуляції для вирішення реальних сценаріїв, що допомагає команді вдосконалити навички та прийняття рішень. Створюються можливості для менторства та коучингу. Досвідчені члени команди допомагають менш досвідченим у розвитку навичок та вирішенні викликів. Формулюються задачі, які стимулюють розвиток. Це можуть бути проекти, які вимагають нових навичок, або завдання, що сприяють творчому мисленню. Створюється платформа для обміну знаннями та досвідом всередині команди. Забезпечується підтримка для індивідуальний розвиток кожного члена команди. аналізуються їхні цілі, амбіції та надається підтримка для їхнього досягнення. Спільно з членами команди визначаються цілі розвитку та розробляється план для досягнення цих цілей. Регулярно відбувається оцінювання прогресу розвитку команди, перевіряється, чи досягаються цілі, які були встановлені для покращення. Розвиток команди розглядається як постійний процес, який допомагає забезпечити високу ефективність її роботи та досягнення цілей проєкту.

- Наслідування прикладу лідера [28]. Лідер команди має бути прикладом для її членів щодо дотримання цінностей та культури. Саме такий підхід до підтримки та дотримання цінностей впливає на всю команду.

Ці етапи можуть бути доповнені або адаптовані залежно від конкретної ситуації та потреб проєкту, але ця загальна структура допоможе вам створити ефективну та спільнодіючу команду. Формування команди – вивчення потенційних кандидатів на виділені ролі: функціональні вимоги, психологічний портрет тощо. Команда може формуватися з існуючого штату або шукають нових кандидатів, відбувається створення сприятливих міжособистісних відносин, згуртування, взаємодія, навчання. Для успішного виконання проєкту необхідно сформувати

команду, що максимально відповідатиме поняттю «ефективна команда». Існують характерні властивості та особливості, якими можна описати ефективну команду:

1. Професійна ефективність – командна націленість на досягнення кінцевого результату проєкту, проявляє творчий підхід та ініціативу при виконанні робіт проєкту.

2. Організаційно-психологічна ефективність – створення мікроклімату, в якому присутня неформальна атмосфера, взаєморозуміння, члени команди прислухаються один до одного, вільно висловлюють свої думки, ідеї, проблеми, конфліктні ситуації швидко вирішуються, суперечки виникають з приводу ідей і методів, а не особистостей, рішення приймають на підставі обговорення і узгодження, а не за думкою більшості.

Робота над формуванням проєктної команди (в ІТ-команді) не закінчується після затвердженого складу команди, протягом усього проєктного циклу ведуться роботи зі складом команди: уточняється опис змістів робіт, зон відповідальності, взаємодії, склад тощо.

Визначення навичок та досвіду передбачає оцінювання навичок, досвіду та знань потенційних членів команди. При формуванні команди, варто забезпечити наявність різних компетенцій, що відповідають потребам проекту. Наприклад, команда може потребувати фахівця з програмування, тестувальника, дизайнера, аналітика тощо.

Співбесіди та оцінка передбачає проведення інтерв'ю та співбесіди з потенційними членами команди. Це допоможе вам підібрати тих людей, які володіють необхідними знаннями, комунікаційними навичками та мотивацією. Оцінка референцій може також допомогти вам отримати додаткову інформацію про кандидатів.

Комунікація та співпраця є ключовими аспектами успішної команди, оскільки необхідно впевнитись, що кожен член команди розуміє мету та цілі проекту, а також їх власні ролі та відповідальності.

Розподіл завдань та відповідальності. Чітко визначте завдання та відповідальність для кожного члена команди.

Мотивація та підтримка. Підтримання та мотивування своєї команди протягом усього проекту. Визнання досягнення та успіхи команди, створення сприятливу робочу атмосферу, що допомагає вирішувати проблеми, які виникають по ходу проекту.

Постійне удосконалення. Забезпечення можливості для навчання та розвитку команди. Організовуються тренінги, семінари або інші форми навчання, щоб підтримувати і покращувати навички членів команди.

Ці методи допомагають формувати ефективні команди для виконання IT проектів. При цьому враховується, що успіх залежить не лише від технічних навичок, але й від здатності команди працювати разом, спілкуватися та вирішувати проблеми.

В індустрії IT і проектному управлінні було проведено багато досліджень та опубліковано різні підходи та методи формування команд для виконання IT проектів. Ось декілька загальноприйнятих практик:

Роль-орієнтований підхід. Цей підхід передбачає визначення ролей та відповідальності для кожного члена команди. Різні ролі можуть включати проектних менеджерів, розробників програмного забезпечення, тестувальників, аналітиків тощо. Відповідно до цього підходу, вибір та розподіл ролей в команді здійснюється на підставі компетенцій, навичок та досвіду кожного учасника.

Компетенції та навички. При формуванні команди враховуються необхідні компетенції та навички для успішного виконання проекту. Це може включати технічні навички, знання певних мов програмування або платформ, а також здатність до комунікації, співпраці та розв'язання проблем.

Розподіл завдань. Важливо чітко визначити завдання та відповідальність для кожного члена команди. Кожен учасник має знати, що очікується від нього та це вписується у загальну мету проекту. Розподіл завдань допомагає забезпечити ефективну організацію роботи та уникнути дублювання зусиль.

Комунікація та співпраця. Успішна команда потребує ефективної комунікації та співпраці між учасниками. Важливо створити сприятливу атмосферу, де

учасники можуть відкрито обговорювати проблеми, ділитися ідеями та спільно працювати над вирішенням завдань.

Мотивація та підтримка. Важливим аспектом формування ефективної команди є мотивація та підтримка учасників. Це може включати визнання досягнень, надання можливостей для професійного розвитку, створення робочих умов, які сприяють продуктивності та задоволенню від роботи.

Досвід та навчання: Формування команди може базуватися на досвіді та навчанні. Команди можуть складатися з досвідчених фахівців, які мають попередній успіх у подібних проектах, або включати молодих талантів з бажанням навчитися та розвиватися.

Загалом, формування ефективних команд для виконання IT проектів вимагає збалансованого підходу, урахування потреб проекту та взаємодії між учасниками. Команда повинна мати необхідні компетенції, досвід та вміння співпрацювати, щоб досягти успіху в реалізації проекту.

Роль-орієнтований підхід до формування команд для виконання IT проектів отримав значну увагу і дослідження у галузі проектного управління. Напрацювання включають концепції та практики, спрямовані на ефективну організацію ролей і відповідальності в команді. Ось деякі з цих напрацювань:

Ролева матриця Белбіна. Белбін (Belbin) [29] розробив ролеву матрицю, яка ідентифікує декілька ключових ролей, що необхідні для ефективної команди. Ці ролі включають координатора, спонсора, творця ідей, реалізатора, дослідника ресурсів тощо. Матриця Белбіна надає уявлення про типові ролі, які можуть бути виконані різними людьми і може бути використана для більш раціонального розподілу ролей в команді.

Рольовий аналіз. Рольовий аналіз [30] дозволяє докладно проаналізувати потреби проекту та визначити необхідні ролі в команді. Цей аналіз може включати ідентифікацію ключових завдань, визначення компетенцій та відповідальності для кожної ролі, а також визначення взаємозв'язків та комунікації між ролями.

Agile-підходи. Agile-методології, такі як Scrum та Kanban, також підтримують роль-орієнтований підхід. Вони пропонують конкретні ролі, такі як

Scrum Master, Product Owner та розробник, і визначають їх відповідальності та взаємодію в рамках ітераційного розвитку. Це допомагає забезпечити чітку рольову структуру та розподіл обов'язків.

Командна ефективність. Існує також велика кількість досліджень, спрямованих на вивчення та розуміння чинників, які впливають на ефективність команд. Це дозволяє краще розуміти, які ролі, динаміка комунікації та співпраця між ролями можуть сприяти успіху команди.

Розвиток ролей. Деякі організації активно займаються розвитком ролей у команді. Це може включати тренінги, менторство, навчання та практичні вправи для покращення навичок та компетенцій конкретних ролей. Це сприяє зростанню професійного рівня та здатності команди до досягнення поставлених цілей.

Ці напрацювання допомагають організаціям впроваджувати роль-орієнтований підхід до формування команд для виконання IT проектів. Вони дозволяють більш ефективно визначати ролі, розподіляти відповідальність та забезпечувати взаємодію між членами команди, що сприяє досягненню кращих результатів у проекті.

Компетенції та навички грають важливу роль у формуванні ефективних команд для виконання IT проектів. Ось деякі загальні компетенції та навички, які можуть бути важливими для учасників команди:

Технічні навички. Це включає знання та досвід у використанні конкретних технологій, мов програмування, фреймворків, баз даних тощо. Команда повинна мати людей з різними технічними навичками, щоб впоратися з різними аспектами проекту.

Аналітичні навички. Важливо мати членів команди, які володіють здатністю аналізувати проблеми, розуміти бізнес-вимоги та визначати оптимальні рішення. Це може включати здатність до розробки стратегій, аналізу даних, оцінки ризиків та прийняття рішень.

Комунікаційні навички. Учасники команди повинні володіти сильними комунікаційними навичками для ефективного спілкування з іншими членами команди, керівництвом, клієнтами та іншими зацікавленими сторонами. Це

включає вміння слухати, висловлювати свої думки чітко та конкретно, а також вміння працювати в команді.

Менеджмент проектів. У команді можуть бути члени, які володіють навичками управління проектами, такими як планування, встановлення пріоритетів, управління ресурсами та виконання завдань в рамках обмежень проекту. Це важливо для забезпечення успішної реалізації проекту.

Проблемно-орієнтоване мислення. Члени команди повинні мати здатність розпізнавати проблеми, шукати творчі рішення та працювати над вирішенням складних завдань. Це включає гнучкість у мисленні та здатність до інновацій.

Співпраця та робота в команді. Учасники команди повинні вміти ефективно працювати разом, співпрацювати, ділитися ідеями та ресурсами. Важливо мати членів команди, які є колаборативними, вміють підтримувати інших та вирішувати конфлікти.

Навички управління часом. Це включає здатність до планування робочого часу, встановлення пріоритетів та здійснення ефективного управління часом для досягнення цілей проекту в зазначені терміни.

Це лише кілька загальних прикладів компетенцій та навичок, які можуть бути важливими для команди, що працює над IT проектом. Конкретні компетенції та навички будуть залежати від специфіки проекту та його вимог.

Метод розподілу завдань є важливим аспектом управління командою та виконання IT проектів. Цей метод передбачає розподіл завдань та відповідальності між членами команди з метою ефективного виконання проекту. Основні кроки методу розподілу завдань включають наступне:

Визначення завдань [31] Перший крок - це ідентифікація всіх завдань, які необхідно виконати для успішного завершення проекту. Розбийте проект на окремі етапи, задачі та підзадачі. Це дозволить зрозуміти всі складові проекту та його вимоги.

Аналіз компетенцій та навичок. Аналізується навички, досвід та компетенції кожного члена команди. Обирається людей, які мають потрібні знання та навички

для виконання кожного завдання. Важливо врахувати потенційні обмеження, такі як наявність ресурсів та графік роботи.

Встановлення пріоритетів. Визначається важливість та пріоритетність кожного завдання. Розглядається терміни виконання, залежності між завданнями та ризики. Встановлюються пріоритети, щоб забезпечити вчасне та успішне виконання критичних завдань.

Розподіл завдань. На основі аналізу завдань та компетенцій команди, розподіляються завдання між учасниками команди. Забезпечується умови, щоб кожен отримав відповідальність за конкретні завдання в рамках своїх компетентностей. Визначається, хто буде відповідальний за кожне завдання, а також встановлюються взаємозв'язки та залежності між завданнями.

Контроль та спілкування. Після розподілу завдань важливо забезпечити контроль та спілкування між учасниками команди. Проводиться регулярне оновлення стану завдань, відстежується прогрес та вирішується проблеми, які виникають. Забезпечується відкрита комунікацію, де члени команди можуть ділитися інформацією та звертатися за допомогою, якщо потрібно.

Підтримка та коригування. В процесі виконання проекту можуть виникати зміни або проблеми, які вимагають коригування розподілу завдань. Необхідно при потребі здійснювати перерозподіл завдань або вносити зміни в план, щоб забезпечити успішне виконання проекту.

1.2 Сучасні підходи до розроблення рекомендаційних систем

Рекомендаційні системи (RS) - це тип інформаційних систем, які використовуються для надання рекомендацій користувачам. Сучасні підходи до розроблення RS можна розділити на два основні типи:

- Фільтрація на основі схожості. Цей тип RS рекомендує користувачам об'єкти, схожі на ті, які вони вже оцінювали або якими цікавилися.
- Фільтрація на основі прогнозування. Цей тип RS рекомендує користувачам об'єкти, які, ймовірно, їм сподобаються, на основі їхніх інтересів та поведінки.

Фільтрація на основі схожості [32] є одним з найпростіших і найбільш поширених підходів до розроблення RS. Він ґрунтується на припущенні, що користувачам, які подобаються схожі об'єкти, також можуть подобатися інші схожі об'єкти. Для реалізації фільтрації на основі схожості необхідно визначити метрику схожості між об'єктами. Метрика схожості може бути заснована на різних факторах, таких як схожість характеристик об'єктів, схожість оцінок користувачів об'єктів або схожість поведінки користувачів щодо об'єктів. Для реалізації фільтрації на основі схожості необхідно визначити метрику схожості між об'єктами. Метрика схожості - це функція, яка приймає два об'єкти як вхідні дані та повертає значення, яке визначає ступінь схожості між цими об'єктами.

Існує багато різних метрик схожості, які можна використовувати для реалізації фільтрації на основі схожості. Одні з найпоширеніших метрик схожості включають:

- Коефіцієнт кореляції: Цей коефіцієнт вимірює ступінь лінійної кореляції між двома векторами даних.
- Коефіцієнт схожості Дженелена: Цей коефіцієнт вимірює ступінь схожості між двома векторами даних, ґрунтуючись на їхніх найближчих сусідах.
- Метод косинусної відстані: Цей метод вимірює ступінь схожості між двома векторами даних, ґрунтуючись на косинусі кута між цими векторами.

Вибір метрики схожості залежить від конкретних потреб застосування.

Фільтрація на основі прогнозування [33] є більш складним підходом до розроблення RS. Він ґрунтується на припущенні, що можна прогнозувати, які об'єкти сподобаються користувачам, на основі їхніх інтересів та поведінки. Для реалізації фільтрації на основі прогнозування необхідно навчити модель прогнозування. Модель прогнозування може бути заснована на різних алгоритмах машинного навчання, таких як логістична регресія, нейронні мережі або дерева рішень.

Крім фільтрації на основі схожості та прогнозування, існують також інші підходи до розроблення RS. Одним з таких підходів є рекомендаційні системи на основі спільнот. Рекомендаційні системи на основі спільнот рекомендують користувачам об'єкти, які подобаються іншим користувачам із їхньої спільноти. Іншим підходом є рекомендаційні системи на основі контексту. Рекомендаційні системи на основі контексту рекомендують користувачам об'єкти, які є релевантними для їхнього контексту. Наприклад, рекомендаційні системи на основі контексту можуть рекомендувати користувачам товари, які вони можуть купити в магазині, або контент, який вони можуть дивитися в певний час доби.

Вибір підходу до розроблення RS залежить від конкретних потреб застосування. Фільтрація на основі схожості є хорошим вибором для застосувань, де важливо рекомендувати користувачам об'єкти, які вони вже знають і люблять. Фільтрація на основі прогнозування є хорошим вибором для застосувань, де важливо рекомендувати користувачам нові об'єкти, які їм можуть сподобатися. Рекомендаційні системи на основі спільнот та контексту можуть бути хорошим вибором для застосувань, де важливо враховувати додаткові фактори, такі як спільнота користувача або контекст його використання.

Сучасні тенденції в розробленні RS включають:

- Використання великих даних: RS все частіше використовують великі набори даних для навчання моделей прогнозування.
- Використання штучного інтелекту: RS все частіше використовують штучний інтелект для підвищення точності рекомендацій.
- Використання мобільних пристроїв: RS все частіше розробляються для мобільних пристроїв.

Ці тенденції дозволяють розробникам RS створювати більш точні та персоналізовані рекомендації.

Фільтрування на основі популярності [34] - це простий і ефективний метод рекомендацій, який рекомендує користувачам об'єкти, які є популярними серед інших користувачів. Для реалізації фільтрування на основі популярності необхідно визначити метрику популярності. Метрика популярності - це функція, яка приймає

об'єкт як вхідні дані та повертає значення, яке визначає ступінь популярності цього об'єкта. Існує багато різних метрик популярності, які можна використовувати для реалізації фільтрування на основі популярності. Одні з найпоширеніших метрик популярності включають:

- Кількість оцінок: Ця метрика вимірює кількість користувачів, які оцінили об'єкт.
- Середній рейтинг: Ця метрика вимірює середній рейтинг, який користувачі дали об'єкту.
- Частка користувачів, які оцінили об'єкт: Ця метрика вимірює частку користувачів, які оцінили об'єкт.

Вибір метрики популярності залежить від конкретних потреб застосування. Наприклад, для рекомендації товарів, які користувачі, ймовірно, куплять, можна використовувати кількість оцінок, яка вимірює, скільки користувачів цікавляться цими товарами. Для рекомендації контенту, який користувачі, ймовірно, подивляться, можна використовувати середній рейтинг, який вимірює, наскільки користувачі люблять цей контент.

Для фільтрування на основі популярності можна використовувати різні формули. Одні з найпоширеніших формул включають:

- Кількість оцінок. Ця формула повертає кількість користувачів, які оцінили об'єкт.

$$\text{popularity}(\text{object}) = \text{count}(\text{users who rated object}) \quad (1.1)$$

- Середній рейтинг. Ця формула повертає середній рейтинг, який користувачі дали об'єкту.

$$\text{popularity}(\text{object}) = \text{mean}(\text{ratings of object}) \quad (1.2)$$

- Частка користувачів, які оцінили об'єкт. Ця формула повертає частку користувачів, які оцінили об'єкт.

$$\text{popularity}(\text{object}) = \text{count}(\text{users who rated object}) / \text{count}(\text{all users}) \quad (1.3)$$

Вибір формули залежить від конкретних потреб застосування. Для рекомендації контенту, який користувачі, ймовірно, подивляться, можна використовувати середній рейтинг, який вимірює, наскільки користувачі люблять

цей контент. Після того, як ми визначили цю умову, ми можемо використовувати її для відбору найбільш популярних веб-сайтів. Наприклад, ми можемо відсортувати всі веб-сайти за середнім рейтингом і рекомендувати користувачам веб-сайти з найвищим рейтингом [127].

Розрахунок проводиться за формулою:

$$J_i = \frac{V}{V_i}, \tag{1.4}$$

де:

J_i – затребуваність об'єкта i; V_i – кількість лайків, які отримав об'єкт i; V – загальна кількість лайків для всіх об'єктів.

З використанням формули аналізується затребуваність кожного об'єкта з врахуванням кількості лайків для об'єкту або взаємодій з користувачами. Об'єкти з більшими значеннями J_i вважаються затребуваними і підлягають пропонуванню користувачам. Проте метод не враховує індивідуальні схильності користувачів і пропонує найпопулярніші об'єкти.

Фільтрування на основі спільності [35]- це метод рекомендацій, який рекомендує користувачам об'єкти, які подобаються іншим користувачам, подібним до них. Для реалізації фільтрування на основі спільності необхідно визначити метрику схожості між користувачами. Метрика схожості - це функція, яка приймає двох користувачів як вхідні дані та повертає значення, яке визначає ступінь схожості між цими користувачами. Існує багато різних метрик схожості, які можна використовувати для реалізації фільтрування на основі спільності. Одні з найпоширеніших метрик схожості включають:

- *Коефіцієнт кореляції*. Цей коефіцієнт вимірює ступінь лінійної кореляції між двома векторами даних.
- *Коефіцієнт схожості Дженелена*. Цей коефіцієнт вимірює ступінь схожості між двома векторами даних, ґрунтуючись на їхніх найближчих сусідах.
- *Метод косинусної відстані*. Цей метод вимірює ступінь схожості між двома векторами даних, ґрунтуючись на косинусі кута між цими векторами.

Вибір метрики схожості залежить від конкретних потреб застосування. Наприклад, для рекомендації товарів, які користувачі, ймовірно, куплять, можна

використовувати коефіцієнт кореляції, який вимірює ступінь подібності між історіями покупок користувачів. Для рекомендації контенту, який користувачі, ймовірно, подивляться, можна використовувати коефіцієнт схожості Дженелена, який вимірює ступінь подібності між рейтингами контенту користувачами. Після того, як ми визначили метрику схожості, ми можемо використовувати її для визначення подібності між будь-якими двома користувачами.

Матриця користувачів і об'єктів (User-Item Matrix) [36] - це двовимірна матриця, в якій користувачі представлені рядками, а об'єкти - стовпцями. В клітинках матриці зберігається інформація про взаємодію користувачів з об'єктами. У методі фільтрування на основі спільності матриця користувачів і об'єктів використовується для обчислення схожості між користувачами. Для цього можна використовувати різні метрики схожості, такі як коефіцієнт кореляції, коефіцієнт схожості Дженелена або метод косинусної відстані.

Після того, як схожість між користувачами обчислена, можна використовувати її для ранжування об'єктів для користувача. Отже, матриця користувачів і об'єктів є важливим компонентом методу фільтрування на основі спільності. Вона використовується для обчислення схожості між користувачами, яка потім використовується для ранжування об'єктів для користувачів. Цей алгоритм можна використовувати для рекомендації будь-яких об'єктів, наприклад, товарів, контенту, людей тощо.

Математичні моделі прогнозування використовуються для оцінки майбутніх значень змінних на основі їхніх минулих значень. Вони використовуються в різних галузях, таких як бізнес, фінанси, наука, інженерія та медицина.

Ось деякі приклади того, як використовуються математичні моделі прогнозування:

- Бізнес. Математичні моделі прогнозування використовуються для прогнозування попиту на товари та послуги, прогнозування продажів, прогнозування прибутку та прогнозування ризиків.

- Фінанси. Математичні моделі прогнозування використовуються для прогнозування цін на акції, прогнозування курсів валют, прогнозування кредитного ризику та прогнозування ризику інвестицій.
- Наука. Математичні моделі прогнозування використовуються для прогнозування погоди, прогнозування землетрусів, прогнозування епідемій та прогнозування змін клімату.
- Інженерія. Математичні моделі прогнозування використовуються для прогнозування зносу обладнання, прогнозування відмов обладнання та прогнозування аварій.
- Медицина. Математичні моделі прогнозування використовуються для прогнозування ризику захворювань, прогнозування результатів лікування та прогнозування ризику смерті.

Математичні моделі прогнозування можна класифікувати за типом моделі та за типом даних, які використовуються для навчання моделі.

За типом моделі математичні моделі прогнозування можна поділити на такі категорії:

- Детерміновані моделі: Детерміновані моделі прогнозування дають точні прогнози, якщо дані, на яких вони навчалися, є точними. Детерміновані моделі зазвичай використовуються для прогнозування змінних, які залежать від фізичних законів.
- Стохастичні моделі: Стохастичні моделі прогнозування дають прогнози з певною ймовірністю. Стохастичні моделі зазвичай використовуються для прогнозування змінних, які залежать від випадкових факторів.

За типом даних, які використовуються для навчання моделі, математичні моделі прогнозування можна поділити на такі категорії:

- Моделі на основі статистичних даних: Моделі на основі статистичних даних використовують статистичні методи для навчання моделі. Статистичні методи використовуються для виявлення закономірностей у даних.
- Моделі на основі машинного навчання: Моделі на основі машинного навчання використовують методи машинного навчання для навчання моделі.

Методи машинного навчання використовуються для навчання моделі на основі прикладів.

Вибір типу математичної моделі прогнозування залежить від конкретних потреб застосування. Наприклад, якщо необхідно зробити точний прогноз, то слід використовувати детерміновану модель. Якщо необхідно зробити прогноз з певною ймовірністю, то слід використовувати стохастичну модель.

Ефективність математичної моделі прогнозування оцінюється за допомогою таких критеріїв:

- Точність. Точність моделі прогнозування визначається як відношення кількості правильних прогнозів до загальної кількості прогнозів.
- Достовірність. Достовірність моделі прогнозування визначається як ймовірність того, що правильний прогноз буде зроблений.
- Статистична значущість. Статистична значущість моделі прогнозування визначається як ймовірність того, що результати моделі не є випадковими.

Для поліпшення ефективності математичної моделі прогнозування можна використовувати такі методи:

- Попереднє опрацювання даних, яке включає в себе видалення шумів, виправлення помилок та вибір оптимальних характеристик для навчання моделі.
- Вибір моделі залежить від конкретних потреб застосування.
- Налаштування параметрів моделі включає в себе вибір оптимальних значень параметрів моделі.
- Перевірка моделі. Перевірка моделі включає в себе оцінку ефективності моделі на відкладеному наборі даних.

Математичні моделі прогнозування є потужним інструментом, який можна використовувати для прогнозування майбутніх значень змінних. Однак важливо правильно вибрати тип моделі та провести попередню обробку даних, щоб забезпечити ефективність моделі.

Метрики схожості використовуються для вимірювання подібності між двома об'єктами. Вони можуть використовуватися в різних областях, таких як машинне

навчання, штучний інтелект, рекомендаційні системи, біоінформатика та багато інших.

Ось деякі приклади того, як використовуються метрики схожості:

- Машинне навчання. Метрики схожості використовуються для навчання моделей машинного навчання, таких як кластерні алгоритми, алгоритми аналізу головних компонентів та алгоритми розпізнавання образів.
- Штучний інтелект. Метрики схожості використовуються для розробки штучних інтелектуальних агентів, таких як роботів, які можуть взаємодіяти з навколишнім середовищем.
- Рекомендаційні системи. Метрики схожості використовуються для рекомендування об'єктів користувачам на основі їхніх інтересів.
- Біоінформатика. Метрики схожості використовуються для порівняння генів, білків та інших біологічних об'єктів.

Існує багато різних метрик схожості. Одні з найпоширеніших метрик схожості включають:

- Коефіцієнт кореляції. Коефіцієнт кореляції вимірює ступінь лінійної кореляції між двома змінними.
- Коефіцієнт схожості Дженелена. Коефіцієнт схожості Дженелена вимірює ступінь подібності між двома наборами даних.
- Метод косинусної відстані. Метод косинусної відстані вимірює ступінь неподібності між двома векторами.

Вибір метрики схожості залежить від конкретних потреб застосування. Наприклад, якщо необхідно виміряти ступінь лінійної кореляції між двома змінними, то слід використовувати коефіцієнт кореляції. Якщо необхідно виміряти ступінь подібності між двома наборами даних, то слід використовувати коефіцієнт схожості Дженелена. Якщо необхідно виміряти ступінь неподібності між двома векторами, то слід використовувати метод косинусної відстані.

Метрики схожості можуть бути використані для вирішення різних завдань. Ось деякі приклади того, як можна використовувати метрики схожості:

- Кластеризація. Метрики схожості можна використовувати для кластеризації об'єктів на основі їхньої подібності.
- Аналіз головних компонентів. Метрики схожості можна використовувати для аналізу головних компонентів, щоб виділити основні тенденції в наборі даних.
- Розпізнавання образів. Метрики схожості можна використовувати для розпізнавання образів, порівнюючи нові зображення з відомими зображеннями.
- Рекомендаційні системи. Метрики схожості можна використовувати для рекомендування об'єктів користувачам на основі їхніх інтересів.

Метрики схожості є потужним інструментом, який можна використовувати для вирішення різних завдань у різних областях.

Функції оцінки (Scoring Functions) [37] використовуються у методі фільтрування на основі спільності для ранжування об'єктів для користувача. Вони визначають, наскільки імовірно, що користувачеві сподобається об'єкт.

Функції оцінки можуть використовувати різні фактори для визначення ймовірності того, що користувачеві сподобається об'єкт. Одні з найпоширеніших факторів включають:

- Схожість між користувачем і іншими користувачами, які оцінили об'єкт. Чим більш схожий користувач з іншими користувачами, які оцінили об'єкт, тим ймовірніше, що користувачеві теж сподобається об'єкт.
- Оцінки, які інші користувачі дали об'єкту. Чим вище оцінки, які інші користувачі дали об'єкту, тим ймовірніше, що користувачеві теж сподобається об'єкт.
- Відомості про об'єкт. Функції оцінки можуть використовувати відомості про об'єкт, такі як його жанр, рейтинг, опис тощо, для визначення того, наскільки імовірно, що користувачеві він сподобається.

Вибір функції оцінки залежить від конкретних потреб застосування. Наприклад, якщо необхідно рекомендувати популярні об'єкти, то можна використовувати функцію оцінки, яка враховує лише оцінки інших користувачів. Якщо необхідно рекомендувати об'єкти, які є подібними до об'єктів, які

користувачеві вже сподобалися, то можна використовувати функцію оцінки, яка враховує схожість між користувачем і іншими користувачами, які оцінили об'єкт.

Функції ваги (Weighting Functions) [38]використовуються у методі фільтрування на основі спільності для регулювання значення схожості між користувачами або об'єктами. Вони дозволяють надати більшу або меншу вагу певним факторам, які враховуються при обчисленні схожості. Функції ваги можуть використовувати різні фактори для регулювання значення схожості. Одні з найпоширеніших факторів включають:

- Кількість взаємодій між користувачами або об'єктами. Чим більше взаємодій між користувачами або об'єктами, тим більшу вагу можна надати схожості між ними.
- Новина взаємодій між користувачами або об'єктами. Чим новіша взаємодія між користувачами або об'єктами, тим більшу вагу можна надати схожості між ними.
- Важливість взаємодій між користувачами або об'єктами. Чим важливіші взаємодії між користувачами або об'єктами, тим більшу вагу можна надати схожості між ними.

Вибір функції ваги залежить від конкретних потреб застосування. Наприклад, якщо необхідно рекомендувати об'єкти, які є подібними до об'єктів, які користувачеві вже сподобалися, то можна використовувати функцію ваги, яка надає більшу вагу взаємодіям, які відбулися недавно. Якщо необхідно рекомендувати об'єкти, які є подібними до об'єктів, які сподобалися іншим користувачам, які мають схожі інтереси, то можна використовувати функцію ваги, яка надає більшу вагу взаємодіям, які відбулися з користувачами, які мають схожі інтереси.

Функції ваги є важливим компонентом методу фільтрування на основі спільності. Вони дозволяють регулювати значення схожості між користувачами або об'єктами, щоб надати більшу вагу певним факторам.

У методі фільтрування на основі спільності, сусіди - це користувачі або об'єкти, які є найбільш схожими до заданого користувача або об'єкта. Сусіди використовуються для ранжування об'єктів для користувача.

Існує кілька методів визначення сусідів у методі фільтрування на основі спільності. Одні з найпоширеніших методів включають:

- Метод найближчих сусідів: Цей метод визначає сусідів як користувачів або об'єкти, які мають найвищу схожість із заданим користувачем або об'єктом.
- Метод k-ближчих сусідів: Цей метод визначає сусідів як k користувачів або об'єктів, які мають найвищу схожість із заданим користувачем або об'єктом.
- Метод усіх сусідів: Цей метод визначає сусідами всіх користувачів або об'єктів, які мають хоча б деяку схожість із заданим користувачем або об'єктом.

Вибір методу визначення сусідів залежить від конкретних потреб застосування. Наприклад, якщо необхідно рекомендувати популярні об'єкти, то можна використовувати метод усіх сусідів. Якщо необхідно рекомендувати об'єкти, які є подібними до об'єктів, які користувачеві вже сподобалися, то можна використовувати метод найближчих сусідів або метод k-ближчих сусідів.

Методи визначення сусідів є важливим компонентом методу фільтрування на основі спільності. Вони дозволяють визначити користувачів або об'єкти, які є найбільш схожими до заданого користувача або об'єкта, щоб надати більш точні рекомендації.

Міра подібності користувачів (User-Based Similarity) [39] використовується у методі фільтрування на основі спільності для вимірювання подібності між двома користувачами. Подібність між користувачами визначається на основі їхніх взаємодій з об'єктами.

Існує кілька різних метрик подібності користувачів, які можна використовувати. Одні з найпоширеніших метрик включають:

- Коефіцієнт кореляції: Коефіцієнт кореляції вимірює ступінь лінійної кореляції між взаємодіями двох користувачів з об'єктами.
- Коефіцієнт схожості Дженелена: Коефіцієнт схожості Дженелена вимірює ступінь подібності між наборами взаємодій двох користувачів.

- Косинусна відстань: Косинусна відстань вимірює ступінь неподібності між наборами взаємодій двох користувачів.

Вибір метрики подібності користувачів залежить від конкретних потреб застосування. Наприклад, якщо необхідно рекомендувати популярні об'єкти, то можна використовувати коефіцієнт кореляції. Якщо необхідно рекомендувати об'єкти, які є подібними до об'єктів, які користувачеві вже сподобалися, то можна використовувати коефіцієнт схожості Дженелена або косинусну відстань.

Після того, як подібність між користувачами обчислена, вона використовується для ранжування об'єктів для користувача. Об'єкти, які мають найвищу схожість з користувачем, вважаються найбільш ймовірними рекомендаціями для цього користувача.

Міра подібності користувачів є важливим компонентом методу фільтрування на основі спільності. Вона дозволяє рекомендувати користувачам об'єкти, які є подібними до об'єктів, які сподобалися іншим користувачам, які мають схожі інтереси.

Міра подібності користувачів визначаються через коефіцієнти кореляції Пірсона (Pearson Correlation Coefficient) або косинусної схожості (Cosine Similarity):

Коефіцієнт кореляції Пірсона (Pearson Correlation Coefficient):

$$\text{sim}_{s,k} = \frac{\sum(rs,i - \bar{r}s)(rk,i - \bar{r}\text{k})}{\sqrt{\sum(rs,i - \bar{r}\text{s})^2 \sum(rk,i - \bar{r}k)^2}} \quad (1.5)$$

де: $\text{sim}_{s,k}$ – міра подібності між користувачами s і k; $r_{s,i}$ – оцінка, яку користувач u надав об'єкту i; $\bar{r}_s$ – середнє значення оцінок користувача u.

Коефіцієнт косинусної схожості (Cosine Similarity):

$$\text{sim}_{s,k} = \frac{\sum rs,i\, rk,i}{\sqrt{\sum rs,i^2 \sum(rk,i)^2}}, \quad (1.6)$$

де параметри та змінні визначені аналогічно.

Коефіцієнт схожості Дженелена (Jaccard Similarity) - це міра подібності між двома наборами даних. Він визначається як відношення кількості елементів, які є спільними для двох наборів, до загальної кількості елементів у двох наборах.

Формула для розрахунку коефіцієнта схожості Дженелена така:

$$similarity(A, B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (1.7)$$

де:

- A - перший набір даних
- B - другий набір даних
- $|A \cap B|$ - кількість елементів, які є спільними для наборів A і B
- $|A|$ - кількість елементів у наборі A
- $|B|$ - кількість елементів у наборі B

Наприклад, якщо у нас є два набори даних A і B, які містять такі елементи:

$A = \{1, 2, 3\}$

$B = \{2, 3, 4\}$

Тоді коефіцієнт схожості Дженелена між наборами A і B буде дорівнювати:

$$similarity(A, B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{2}{3 + 3 - 2} = \frac{2}{8} = 0.25 \quad (1.8)$$

Це означає, що 25% елементів наборів A і B є спільними.

Коефіцієнт схожості Дженелена можна використовувати для вимірювання подібності між двома користувачами на основі їхніх взаємодій з об'єктами. Наприклад, якщо у нас є база даних користувачів, які оцінювали фільми, то ми можемо використовувати коефіцієнт схожості Дженелена для вимірювання подібності між двома користувачами на основі фільмів, які вони оцінили.

Користувачі, які мають високу схожість за коефіцієнтом схожості Дженелена, вважаються схожими за інтересами. Таким чином, ми можемо рекомендувати користувачам об'єкти, які сподобалися іншим користувачам, які мають схожі інтереси.

Методи, що ґрунтуються на використанні змісту (Content-Based Methods), використовують інформацію про об'єкти для генерування рекомендацій. Ця інформація може включати такі характеристики об'єктів, як:

- Назва
- Опис
- Теги

- Категорії
- Рейтинги
- Відгуки

Методи, що грунтуються на використанні змісту, генерують рекомендації, використовуючи подібність між об'єктами. Об'єкти, які мають подібний контент, вважаються схожими. Таким чином, об'єкти, які сподобалися користувачеві в минулому, можуть бути рекомендовані користувачеві в майбутньому.

Існує кілька різних методів, які можна використовувати для генерування рекомендацій на основі контенту. Одні з найпоширеніших методів включають:

- Метод подібності. Цей метод генерує рекомендації, використовуючи міру подібності між об'єктами. Наприклад, можна використовувати коефіцієнт кореляції або косинусну відстань для вимірювання подібності між об'єктами.
- Метод кластеризації. Цей метод генерує рекомендації, групуючи об'єкти, які мають подібний контент. Об'єкти, які належать до однієї групи, вважаються схожими.
- Метод машинного навчання. Цей метод використовує машинне навчання для навчання моделі, яка може генерувати рекомендації на основі контенту об'єктів.

Методи, що грунтуються на використанні змісту, мають ряд переваг. Вони можуть бути ефективними для генерування рекомендацій для нових користувачів, які мають мало взаємодій з об'єктами. Вони також можуть бути ефективними для генерування рекомендацій для користувачів, які не хочуть ділитися своїми особистими даними.

Однак, методи, що грунтуються на використанні змісту, також мають ряд недоліків. Вони можуть бути не такими точними, як методи, що грунтуються на використанні спільності, оскільки вони не враховують індивідуальні інтереси користувачів. Крім того, вони можуть бути не такими ефективними для генерування рекомендацій для користувачів, які мають багато взаємодій з об'єктами.

Глибинне навчання (Deep Learning) [40] – це підхід до машинного навчання, який використовує багатошарові нейронні мережі для обробки даних. Нейронні мережі є моделлю, яка імітує роботу мозку людини. Вони складаються з вузлів, які називаються нейронами, які пов'язані між собою. Кожний нейрон обробляє сигнали, які надходять від інших нейронів, і виробляє сигнал, який передається іншим нейронам.

Глибинне навчання може використовуватися для вирішення широкого спектру завдань, включаючи:

- Рекомендаційні системи. Глибинне навчання може використовуватися для генерування рекомендацій для користувачів на основі їхніх взаємодій з об'єктами. Наприклад, глибинне навчання може використовуватися для рекомендування товарів користувачам, які відвідують веб-сайт покупок.
- Обробка природної мови. Глибинне навчання може використовуватися для обробки природної мови, наприклад, для перекладу текстів, розпізнавання мови та генерування тексту.
- Обробка зображень. Глибинне навчання може використовуватися для обробки зображень, наприклад, для розпізнавання об'єктів, виявлення лиць та створення фотореалістичних зображень.
- Обробка відео. Глибинне навчання може використовуватися для обробки відео, наприклад, для розпізнавання дій, виявлення об'єктів та створення штучного інтелекту.

Глибинне навчання є потужним інструментом, який може бути використаний для вирішення складних завдань. Однак, глибоке навчання також вимагає значних обчислювальних ресурсів і може бути складним для навчання.

В основі методу глибинного навчання лежить принцип навчання на прикладах. Для навчання моделі глибинного навчання використовуються набори даних, які містять приклади того, як вирішувати завдання. Модель навчається на цих прикладах, ідентифікуючи закономірності між вхідними даними та бажаними результатами.

Наприклад, для навчання моделі глибинного навчання для рекомендаційних систем можна використовувати набір даних, який містить інформацію про взаємодії користувачів з об'єктами. Модель навчиться на цих даних, ідентифікуючи закономірності між взаємодіями користувачів та тими об'єктами, які їм сподобалися. Після навчання модель може використовуватися для генерування рекомендацій для інших користувачів.

Метод глибинного навчання має ряд переваг. Він може бути більш точним, ніж інші методи машинного навчання, оскільки він може враховувати складні взаємозв'язки між даними. Крім того, глибинне навчання може використовуватися для вирішення завдань, які є складними або неможливими для вирішення іншими методами машинного навчання.

Однак, метод глибинного навчання також має ряд недоліків. Він може бути складним для навчання і вимагати значних обчислювальних ресурсів. Крім того, глибинне навчання може бути сприйнятливим до спотворень даних, оскільки він може навчитися на неправильних або нерепрезентативних даних [41].

Стохастичний градієнтний спуск (Stochastic Gradient Descent, SGD) [42] - це ітеративний метод оптимізації, який використовується для знаходження локального мінімуму функції. Він працює шляхом послідовного покращення оцінки функції шляхом руху в напрямку її градієнта.

SGD використовується для вирішення широкого спектру завдань, включаючи:

- Рекомендаційні системи: SGD можна використовувати для навчання моделей рекомендаційних систем, які можуть генерувати рекомендації для користувачів на основі їхніх взаємодій з об'єктами.
- Обробка природної мови: SGD можна використовувати для навчання моделей обробки природної мови, які можуть виконувати завдання, такі як переклад текстів, розпізнавання мови та генерування тексту.
- Обробка зображень: SGD можна використовувати для навчання моделей обробки зображень, які можуть виконувати завдання, такі як розпізнавання об'єктів, виявлення лиць та створення фотореалістичних зображень.

SGD має ряд переваг. Він є простим у реалізації та може бути ефективним для вирішення завдань з великою кількістю даних. Крім того, SGD є стійким до локальних мінімумів, що означає, що він, швидше за все, знайде глобальний мінімум функції, а не локальний мінімум.

Однак, SGD також має ряд недоліків. Він може бути повільним для досягнення локального мінімуму, особливо для функцій з великою кількістю локальних мінімумів. Крім того, SGD може бути нестабільним, що означає, що його оцінка функції може раптово змінюватися.

В основі методу стохастичного градієнтного спуску лежить принцип навчання на прикладах. Для навчання моделі SGD використовуються набори даних, які містять приклади того, як вирішувати завдання. Модель навчається на цих прикладах, коригуючи свої параметри в напрямку градієнта функції.

Наприклад, для навчання моделі SGD для рекомендаційних систем можна використовувати набір даних, який містить інформацію про взаємодії користувачів з об'єктами. Модель навчиться на цих даних, коригуючи свої параметри в напрямку градієнта функції, яка оцінює якість рекомендацій. Після навчання модель може використовуватися для генерування рекомендацій для інших користувачів.

Стохастичний градієнтний спуск є потужним інструментом, який може бути використаний для вирішення широкого спектру завдань. Однак, важливо розуміти його переваги та недоліки, щоб використовувати його ефективно.

Для визначення наскільки добре модель вирішує завдання використовуються функції втрати, такі як середньоквадратична помилка (MSE) для задач регресії. Середньоквадратична помилка (MSE) - це міра відстані між фактичними даними та прогнозованими даними. Вона визначається як середнє значення квадратів різниць між фактичними даними та прогнозованими даними.

Формула для розрахунку MSE така:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{1.9}$$

де:

- *yi* - фактичне значення *i*-го прикладу

- $y\hat{}i$ - прогнозоване значення i-го прикладу
- n - кількість прикладів

MSE є мірою дисперсії між фактичними даними та прогнозованими даними. Чим вище MSE, тим більша дисперсія, і тим гірше прогноз.

MSE широко використовується в машинному навчанні для оцінки якості прогнозів. Він також використовується в теорії статистичної оцінки для визначення оптимальних оцінок.

MSE має ряд переваг. Він є простим у розрахунку та інтерпретації. Крім того, він є стійким до шуму в даних.

Однак, MSE також має ряд недоліків. Він може бути чутливим до outliers, тобто до значних відхилень фактичних даних від прогнозованих даних. Крім того, MSE може не враховувати кореляцію між фактичними даними.

Перехресна ентропія (Cross-Entropy) [43] - це міра відстані між двома розподілами ймовірності. Вона використовується для оцінки якості класифікатора, який прогнозує ймовірність того, що даний приклад належить до певного класу.

Формула для розрахунку перехресної ентропії така:

$$H(p,q) = -\sum ip(i)\log(q(i)), \quad (1.10)$$

де: H(p,q) – перехресна ентропія між фактичним розподілом p і передбаченим розподілом q; i – індекс різних можливих подій або класів; p(i) – ймовірність події i у фактичному розподілі; q(i) – ймовірність події i у передбаченому розподілі.

Ця метрика часто використовується для оцінки якості класифікаційних моделей, таких як моделі машинного навчання, і допомагає виміряти, наскільки добре передбачені ймовірності відповідають фактичним даним. У задачах класифікації, менше значення перехресної ентропії вказує на кращу якість передбачення моделі. Розрахунок зворотнього розповсюдження помилки (Backpropagation) є важливим для навчання нейронних мереж і оновлення їх ваг після кожної ітерації. Глибинне навчання використовує цей математичний апарат для навчання нейронних мереж, що може бути використано для різних завдань, таких як розпізнавання образів, опрацювання природних мов, генерування рекомендацій, автономне навчання і багато інших.

Чим нижча перехресна ентропія, тим ближче розподіли ймовірності фактичних даних та прогнозованих даних. Таким чином, нижча перехресна ентропія означає, що класифікатор краще прогнозує ймовірність того, що даний приклад належить до певного класу. Перехресна ентропія широко використовується в машинному навчанні для оцінки якості класифікаторів. Вона також використовується в теорії інформації для визначення оптимальних кодів. Перехресна ентропія має ряд переваг. Вона є простою у розрахунку та інтерпретації. Крім того, вона є стійкою до шуму в даних. Однак, перехресна ентропія також має ряд недоліків. Вона може бути чутливою до outliers, тобто до значних відхилень фактичних даних від прогнозованих даних. Крім того, перехресна ентропія може не враховувати кореляцію між фактичними даними.

Ось кілька прикладів того, як використовується перехресна ентропія:

- Для оцінки якості класифікаторів у задачах класифікації. Наприклад, перехресну ентропію можна використовувати для оцінки якості класифікатора, який прогнозує ймовірність того, що даний зображення є зображенням людини або собаки.
- Для навчання класифікаторів у задачах класифікації. Наприклад, перехресну ентропію можна використовувати в якості функції втрат для навчання класифікатора, який прогнозує ймовірність того, що даний зображення є зображенням людини або собаки.
- Для оцінки якості кодів у задачах стиснення. Наприклад, перехресну ентропію можна використовувати для оцінки якості коду, який використовується для стиснення зображення.

Перехресна ентропія є потужним інструментом, який може використовуватися для вирішення широкого спектру завдань у машинному навчанні.

Гібридні системи (Hybrid Recommender Systems) [44] - це системи рекомендацій, які використовують два або більше різних методів фільтрування для генерування рекомендацій. Вони часто використовуються для того, щоб поєднати

переваги різних методів фільтрування та отримати більш точні та ефективні рекомендації.

Існує кілька різних способів використовувати гібридні системи для фільтрування. Один спосіб - використовувати два або більше методів фільтрування окремо, а потім поєднати результати цих методів. Наприклад, можна використовувати метод спільності для генерування списку потенційних рекомендацій, а потім використовувати метод подібності для відбору найкращих рекомендацій з цього списку.

Інший спосіб використовувати гібридні системи - використовувати два або більше методів фільтрування разом. Наприклад, можна використовувати метод спільності та метод подібності для генерування рекомендацій одночасно. Рекомендації, які генеруються кожним методом, будуть враховуватися при генеруванні остаточного списку рекомендацій.

Гібридні системи можуть бути ефективним способом генерувати точні та ефективні рекомендації. Вони можуть бути особливо корисними в ситуаціях, коли один метод фільтрування не може забезпечити достатню точність або ефективність.

Гібридні системи є потужним інструментом, який може використовуватися для вирішення широкого спектру завдань у галузі рекомендаційних систем.

Кожен з проаналізованих підходів має свої переваги та недоліки і в залежності від конкретного завдання та доступних даних обирається найзручніший. У рекомендаційній системі вважаємо за доцільне використати комбінацію цих підходів.

1.3 Постановка задачі дослідження

Формування команди для успішної реалізації ІТ проекту складна задача, для її успішного вирішення доцільно розробити рекомендаційну систему, функціональність якої має передбачати можливість підбору претендентів у команду. Для вирішення цього завдання доцільно:

1. Провести аналіз існуючих підходів до формування команд;
2. Дослідити сучасні підходи до розроблення рекомендаційних систем;

3. Проаналізувати ІТ середовище, яке доцільно створити у розумному місті;
4. Дослідити траєкторію залучення випускників шкіл до навчання на ІТ спеціальностях: мотиваційні чинники, обрання закладу вищої освіти для здобуття фаху, формування навчального контенту, що є передумовою формування ефективних команд для реалізації ІТ проєктів.
5. Формалізувати підходи до створення команд для використання цих формалізацій у рекомендаційній системі;
6. Обрати методи аналізу інформації та генерування рекомендацій;
7. Розробити інформаційну технологію підбору претендентів у команду;
8. Розробити архітектуру рекомендаційної системи.

провести аналіз ІТ середовища.

Висновки до 1 розділу

Проаналізовано переваги та обмеження функціонального, міжфункціонального, проектного, крос-функціонального підходів до формування команд.

Визначено та детально розглянуто чинники, необхідні для створення позитивної командної культури.

Проведене аналіз сучасних підходів до розроблення рекомендаційних, які розділено на два основні типи: фільтрація на основі схожості та фільтрація на основі прогнозування. Детально розглянуто методи, які використовуються в межах цих підходів, що дало можливість визначитись із підходом та методами, які використовуватимуться у розроблюваній системі.

Подано постановку задачі дослідження.

РОЗДІЛ 2
ТЕХНОЛОГІЇ АНАЛІЗУ ПОТРЕБ ІТ ГАЛУЗІ У ФАХІВЦЯХ З ВИЩОЮ ОСВІТОЮ

2.1 Аналізу ІТ ринку «розумного міста»

У сучасному інформаційному суспільстві провідні міста світу прагнуть досягти статусу “розумного”[45, 46], перетворення на центри розвитку вищої освіти та наукових досліджень, а також формування інтелектуального робочого потенціалу, зокрема для ІТ галузі[47, 48].

Рис. 2.1 – Фактори, що утворюють “Розумне місто”.

Ефективний розвиток ІТ ринку «розумного міста» передбачає взаємодію навчальних закладів, які готують фахівців для галузі, та ІТ фірм, ймовірно через структури, які отримали назву – ІТ кластер. Така взаємодія сприяє запровадження дуальної освіти, генерує потребу мотивації випускників шкіл до вступу на спеціальності ІТ галузі. Ідея концепту “розумне місто” реалізується завдяки загальносвітовій цивілізаційній тенденції до створення сучасного динамічного освітнього середовища. Поки що немає універсальних показників, що вичерпно визначали б “розумність міста”, хоча частково подані у стандартах (рис.2.2). Разом

з тим незаперечним є той факт, що формування розумного міста тісно пов’язане з прийняттям ефективних управлінських рішень, людськими ресурсами та розвитком університетів.

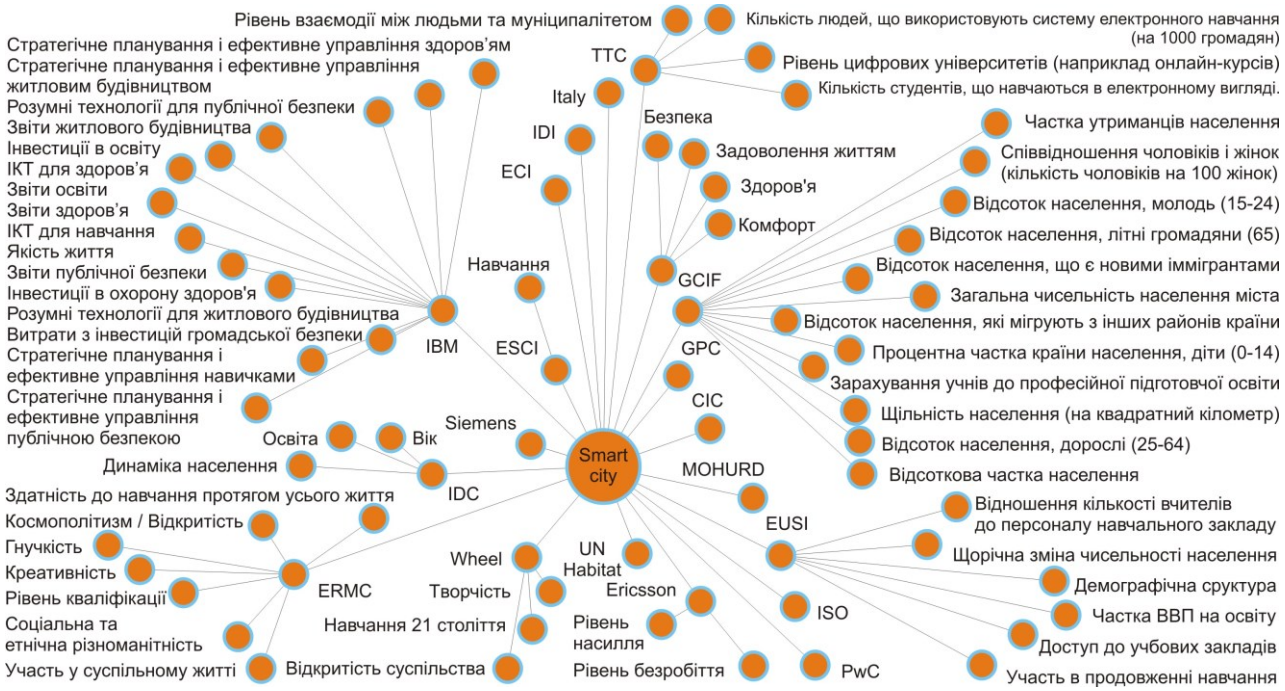


Рис. 2.2 – Стандарти «розумного міста» і та галузі охоплення.

Для проведення системного дослідження, яке передбачає вивчення ринку праці на ІТ ринку та діяльності всього комплексу вищих навчальних закладів в місті, розроблення стратегій та відповідних програм дій необхідно: зберігати і керувати даними обсягами десятки та сотні терабайт; опрацьовувати структуровані, та неструктуровані дані, аналізувати великі масиви статистичних даних; аналізувати різнотипні та різноформатні інформаційні ресурси з різнопланових джерел.

Сучасні умови функціонування ринку ІТ праці та освітніх послуг міста вимагають активного використання новітніх інформаційних та інтелектуальних технологій для професійної орієнтації молоді, вибору напряму підготовки та навчального закладу, які враховували б не лише можливості навчального закладу, а й суб’єктивні фактори пошукувача – мотивацію, схильність до певного виду

діяльності, рівень підготовки тощо. Один із способів, щоб стандартизувати значення показників є Z-перетворення [49, 50].

$$z_i = \frac{x_i - \bar{x}}{s} \tag{2.1}$$

За допомогою цього методу, всі значення індикаторів перетворюються в стандартизовані значення із середнім значенням 0 і стандартним відхиленням 1. Для отримання результатів на рівні факторів, характеристик і фінальних результатів кожного міста, необхідна агрегація на рівні індикатора. Для агрегації показників, розглядається рівень охоплення кожного показника.

Базову структуру критеріїв оцінювання «розумного» міста дослідники подають за допомогою матриці [51, 52]:

$$\begin{array}{cccc} & G_1 & \cdots & G_J \\ A_1 & \varphi_{11} & \cdots & \varphi_{1j} \\ \vdots & \vdots & & \vdots \\ A_i & \varphi_{i1} & \cdots & \varphi_{ij} \end{array} \tag{2.2}$$

де G_j вказує на характеристику розумного міста; $G_J = \{G_1, G_2, \cdots, G_J\}$ з перелік J, A_i є альтернативним варіантом $A_J = \{A_1, A_2, \cdots, A_J\}$; φ_{ij} вказує це результат застосування варіанту A_i для досягнення мети G_J. Зазвичай ваги $\{w_1, w_2, \cdots, w_J\}$ вводяться для представлення різних значень щодо можливостей проектів із перетворення міста.

Запропонований [53, 54] метод полягає у присвоєнні альтернативам $A_1, \cdots, A_m$, ваг на основі експертних оцінок.

Експерти висловлюють свою щодо показників оцінки, присвоюючи кожному критерію ваговий коефіцієнт в інтервалі [0, L]. Це дозволяє побудувати матрицю

Таблиця 2.1 – Матриця вагових коефіцієнтів критеріїв

	Критерії “розумного міста”																			
	J1				J2				J3				J4				J5			
1	1	1	2	2	2	2	3	3	1	1	1	1	2	2	2	2	1	1	2	2
2	3	3	3	3	2	2	3	3	2	2	2	2	2	2	3	3	1	1	2	2
3	3	3	3	3	2	2	3	3	2	2	2	2	2	2	3	3	1	1	2	2

4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	3	3	3	3	2	2	3	3	2	2	2	2	2	2	3	3	1	1	2	2
6	6	6	7	7	7	7	8	8	5	6	7	8	7	7	8	8	7	7	9	9
7	7	7	7	7	7	7	8	8	7	7	9	9	6	6	8	8	6	7	8	9
8	3	4	5	5	4	4	6	6	2	2	3	3	2	2	4	4	3	3	3	3
9	7	7	8	8	6	6	8	8	6	7	8	8	8	8	9	9	8	8	8	8
10	7	8	8	8	8	8	9	9	6	6	8	8	7	7	7	8	7	7	9	9
11	5	5	6	6	4	4	5	6	5	5	5	5	6	6	6	6	4	4	6	6
12	9	9	9	9	6	6	8	8	7	7	8	8	6	7	8	9	5	6	7	8
13	6	6	6	6	6	6	7	7	6	6	8	8	5	5	9	9	5	6	6	7
14	2	2	3	3	3	3	4	4	1	1	2	2	2	2	2	2	3	3	3	3
15	2	2	2	2	1	1	2	2	2	2	3	3	1	1	1	1	1	1	2	2
16	2	2	3	3	3	3	3	3	1	1	3	3	2	2	2	2	3	3	3	3
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	5	5	6	6	5	5	6	6	4	4	6	6	6	6	6	6	5	6	7	7

Необхідність ефективної взаємодії трьох елементів, таких як – розумні технології, розумні мешканці та розумна співпраця вимагає задіювати не тільки складні інтелектуальні інформаційні технології, але й ініціативи IT інфраструктур. Саме при такому підході місто може розвиватися успішніше [55, 56], а відповідно його мешканці отримують можливість формування комфортних умов проживання. Ключовим фактором розвитку систем «розумне місто» є соціум, з його потребами та перспективами. Цілі розвитку такого соціуму можна подати наступним алгоритмом (рис.2.3).

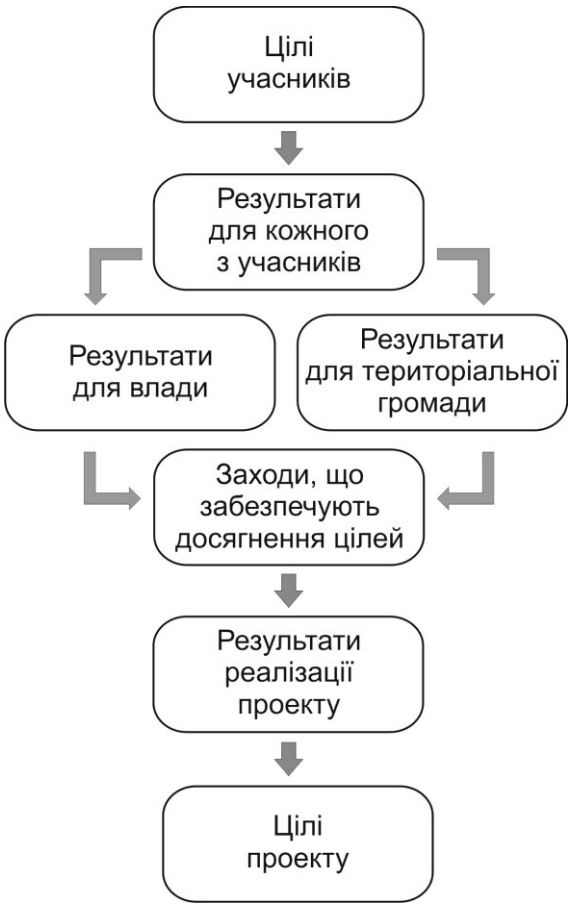


Рис. 2.3. Формування цілей проекту

Аналіз IT-ринку "розумного міста" вимагає дослідження різних аспектів, таких як технологічні інновації, тренди, ключові гравці, можливості та виклики. Ключовими гравцями IT ринку "розумного міста" є провідні IT компанії. Це можуть бути великі IT-компанії, провайдери мереж та зв'язку, постачальники додаткових послуг, девелопери програмного забезпечення тощо. Аналіз IT ринку дозволяє ідентифікувати ці виклики та розглядати можливості для розв'язання проблем та покращення процесів. Великі IT-компанії, такі як IBM, Cisco, Microsoft, Siemens, Oracle, SAP, займаються розробкою та наданням рішень для розумних міст. Вони пропонують платформи, системи управління, аналітичні рішення та послуги для оптимізації різних аспектів міста, включаючи транспорт, енергетику, управління будівлями та інфраструктурою.

Таким чином можемо представити модель IT галузі розумного міста, яка може включати різні аспекти, такі як системи збору інформації про працівників,

використання технологій для ІТ розробок та інше. Наведено приклад базової моделі у вигляді системи рівнянь:

Система збору інформації про працівників:

$I_{employee}=f(data\_sources\_employees)$

Використання технологій для ІТ розробок:

$I_{IT\_development}=g(technology\_usage)$

Інтеграція інформації та технологій для розумного міста:

$I_{smart\_city\_IT}=h(I_{employee}, I_{IT\_development})$

У цих рівняннях:

$I_{employee}$ - інформація про працівників.

$I_{IT\_development}$ - інформація про використання технологій для ІТ розробок.

$I_{TIsmart\_city\_IT}$ - інформація про ІТ галузь розумного міста.

Функції f, g, h представляють процеси збору, опрацювання та інтеграції інформації. Це базовий приклад, і конкретні рівняння будуть залежати від конкретних деталей та характеристик розумного міста і ІТ галузі.

Для стабільного поповнення ІТ галузі трудовими ресурсами необхідне вивчення чинників, які сприятимуть його розвитку, серед яких вивчення нахилів абітурієнтів та мотиваційних чинників до вступу у заклади вищої освіти на спеціальності ІТ галузі.

2.2 Визначення потенціалу випускників середньої школи

Для забезпечення команди якісними фахівцями необхідно налагоджувати ефективну систему підготовки ІТ фахівців. Для системного підходу до цієї процедури необхідно проводити аналіз нахилів абітурієнтів, які мають намір вступати у заклади вищої освіти на спеціальності ІТ галузі.

Та перш ніж визначати нахили абітурієнтів доцільно зорієнтуватися щодо складу випускників середніх шкіл тобто людським ресурсом міста [57].

Рис. 2.3 – Хмара слів термінів пов’язаних з людьми та людськими ресурсами

На найбільшому українському порталі відкритих даних http://data.gov.ua налічується 30 606 наборів даних, з них по освіті 2813 і по своїй суті це є не набори даних, а різноманітні документи в PDF, JPG форматах. Одним з найкращих ресурсів відкритих даних в українському сегменті інтернету в напрямку освіти є сайт українського центру оцінювання якості освіти https:// testportal.com.ua/, де вільно можна завантажити інформацію про дані ЗНО. Також для наповнення бази даних використовувалась інформаційна система управління освітою ІСУО https://isuo.org/. Але найбільше інформації про навчальні заклади середньої освіти в відкритому доступі розміщені на ресурсі Відкрита школа https://open-school.uspishnemisto.com.ua/.

Програмний ресурс має простий та зрозумілий інтерфейс. Наразі заповнюється база навчальних закладів для м. Тернопіль.

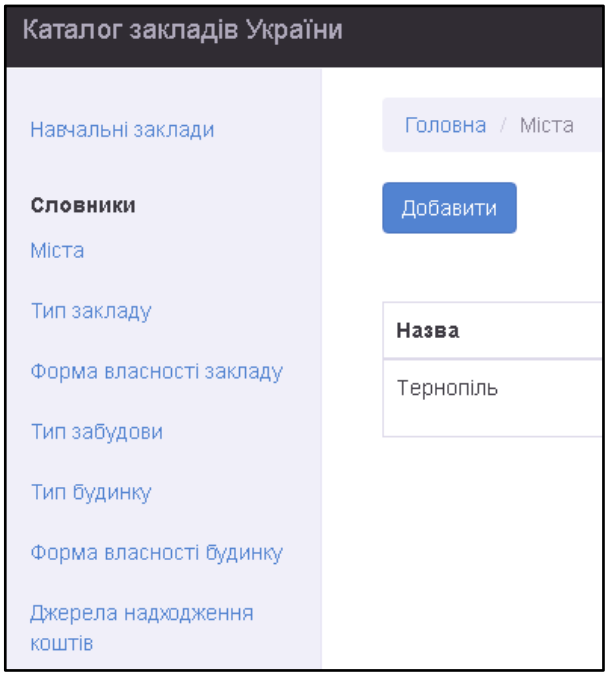


Рис. 2.4 – Скріншот з ресурсу «Відкрита школа»

Кожен навчальний заклад містить такий набір даних:

– заклад
– адреса
– ліцензія
– атестат
– земельна ділянка
– статистика
– режим роботи
– навчальний рік
– додатково

В меню додатково містяться:

– результати ЗНО
– мережі
– приміщення
– іноземні мови
– комп'ютерне обладнання
– мережеве обладнання

– фінансове забезпечення
– навчальні кабінети
– розклад дзвінків

Наповнення бази ресурсу здійснюється двома способами: вручну та за допомогою парсингу. Парсинг – це синтаксичний аналіз сайту, який здійснюється спеціальним скриптом чи іншим програмним засобом, для того щоб зберегти і відобразити зібрану інформацію.

Система призначена для організації баз даних про навчальні заклади міста та їх аналіз. Відразу ж після вдалого входу на веб-ресурс бачимо наступний інтерфейс. З лівого боку в головному меню сайту є такі пункти меню: навчальні заклади, міста, типи закладів, форма власності, ти забудови, тип будинку, форма власності будинку, джерела надходження коштів.

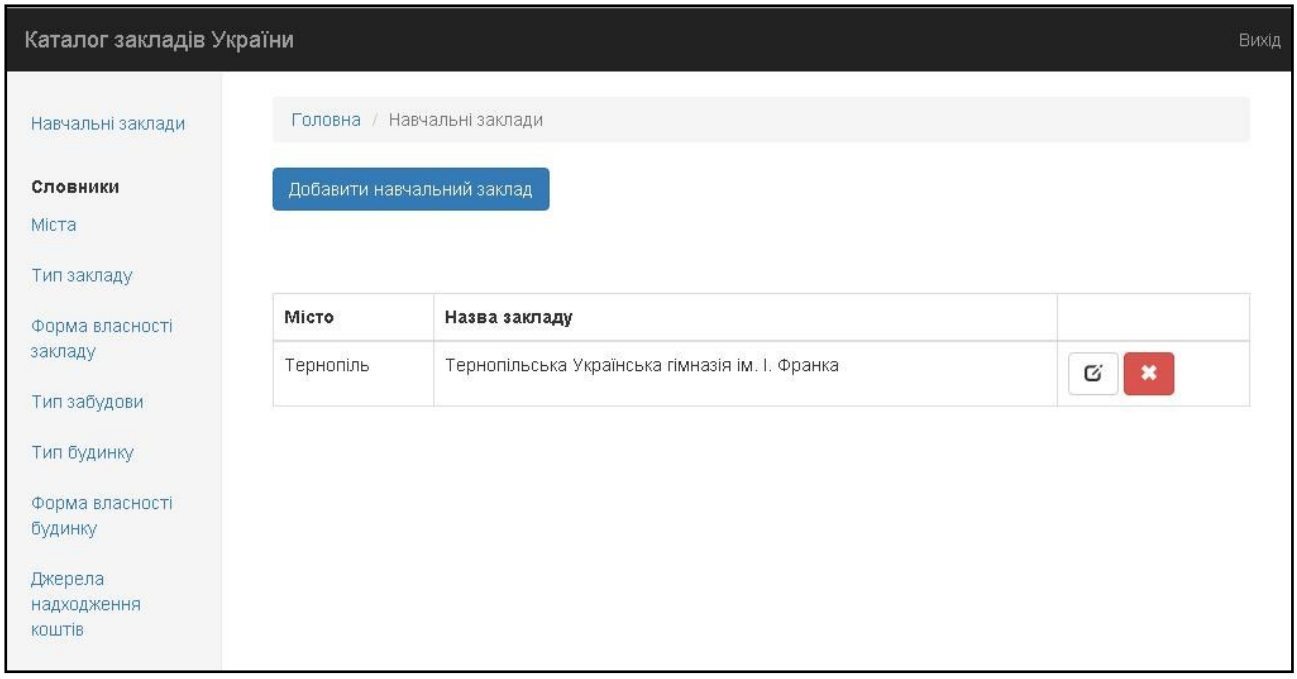


Рис. 2.5 – Скріншот з ресурсу «Відкрита школа». Вибір закладу.

Зручний та простий інтерфейс дає можливість легко добавляти та редагувати безліч параметрів, якими можна охарактеризувати навчальний заклад.

До успішного навчання учня чи студента в навчальному закладі можна віднести безліч факторів, які все ж таки прямо чи опосередковано впливають на якість навчання [58]. Це і кількість учнів в класі, доступність комп’ютерної техніки та технічних засобів, наповненість матеріальної бази та багато іншого. Не останнім чинником в такому підході є результати ЗНО, які показують рівень підготовки

учня, а якщо взяти в розрізі декількох років, то це може свідчити про професійність та майстерність вчителів.

При додаванні навчального закладу до бази будемо мати наступні пункти:

- тип закладу;
- форма власності;
- назва закладу;
- номер;
- засновник;
- мова навчання;
- територіальна підпорядкованість;
- дата заснування.

При редагуванні навчального закладу кількість пунктів меню збільшується і стають доступними:

- заклад;
- адреса;
- ліцензія;
- атестат;
- земельна ділянка;
- мережі;
- режим роботи;
- навчальний рік;
- додатково, яке містить набір додаткових даних (приміщення, комп’ютерне обладнання, мережеве обладнання, фінансове забезпечення, навчальні кабінети, розклад дзвінків).

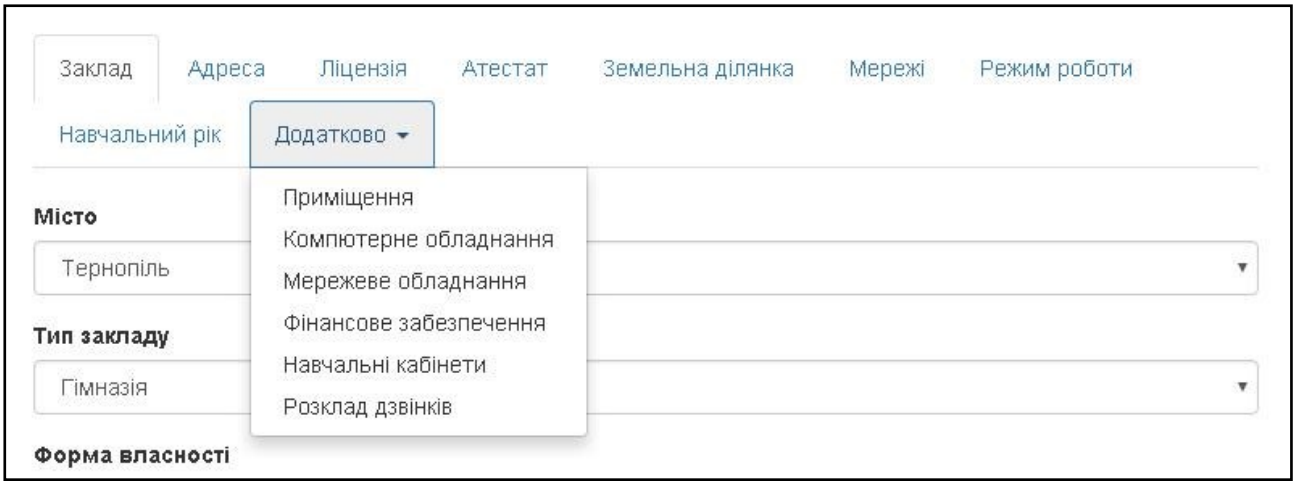


Рис. 2.6 – Скріншот з ресурсу «Відкрита школа». Додаткові можливості

Особливої уваги заслуговує пункт мережа, за змістом якого можна судити про доступність Інтернету для кожного учня чи вчителя навчального закладу. До пункту мережі входить такий набір даних:

- спосіб підключення до мережі Інтернет;
- середня швидкість доступу до мережі Інтернет;
- режим доступу до комп'ютерів ЗВО до мережі Інтернет;
- кількість годин роботи школи в мережі Інтернет за рік;
- кількість комп'ютерів підключених до локальної мережі;
- наявність підключення до мережі Інтернет;
- наявність підключення комп'ютерів у спеціалізованих комп'ютерних класах до локальної мережі;
- наявність Інтернет сторінки або сайту школи;
- адреса Інтернет-сторінки або сайту школи.

Зрозуміло, що доступ до мережі це дуже важливо, але не менш важливим є наступне: скільки точок доступу до Інтернету має навчальний заклад, чи є доступ до глобальної мережі в місцях, які не пов'язані безпосередньо з вивченням інформатики (класи математики, фізики літератури та інші), чи це є відкритий доступ. Саме учні є кінцевими користувачами Інтернету в школі і від цього є пряма залежність їх успішності. Немає потреби чекати закінчення уроку, щоб ознайомитись з потрібною інформацією.

Опрацювання результатів проходження ЗНО випускниками шкіл м. Тернополя [59]. Дані щодо кількості випускників шкіл та проходження зовнішнього незалежного оцінювання (ЗНО) можуть бути різного формату, з різних джерел, з офіційних звітів, опитування і т.д. Достовірна інформація про результати ЗНО надходить з офіційної сторінки ресурсу Український центр оцінювання якості освіти [60]. В вільному доступі є архів з результатами зовнішнього незалежного оцінювання за останні чотири роки. У таблиці 2 подано порівняльний аналіз результатів ЗНО з профільних для спеціальностей ІТ галузі предметів.

Таблиця 2.2 – Результати ЗНО у Тернопільській області у 2016-2019 рр. з математики, української та англійської мов

Регіон		**Кількість зареєстрованих**		**Бал за шкалою 100-200***		
Рік	**Предмет**	**усього**	**обрано як ДПА**	**min**	**avg**	**max**
2016	**математика**	2941	1376	100.0	137.5	199.0
2017	**математика**	2241	1423	100.0	141.2	200.0
2018	**математика**	2318	1285	100.0	138.8	196.0
2019	**математика**	3154	2225	100.0	140.9	200.0
2016	**англ. мова**	1933	141	146.2	199.0	200.0
2017	**англ. мова**	1775	990	100.0	147.6	199.0
2018	**англ. мова**	1839	939	100.0	145.8	197.0
2019	**англ. мова**	2004	1051	100.0	146.6	200.0
2016	**укр. мова**	6692	5014	100.0	149.0	200.0
2017	**укр. мова**	5757	4555	100.0	148.4	199.0

2018	**укр. мова**	10216	9656	100.0	141.2	200.0
2019	**укр. мова**	10216	9662	100.0	141.2	200.0

Виставивши необхідні фільтри отримаємо дані, що ЗНО з української мови та літератури здавало 1135 учнів з 35 навчальних закладів Тернополя, які дають повну середню освіту – 11 класів, незалежно від спрямування чи форми власності (рис. 2.7) (додаток Б, таблиця 2.3).

Рис. 2.7 – Розподіл за школами випускників м. Тернополя

З діаграми бачимо, що в місті явно видно три навчальні заклади, які за кількістю учнів на 20-25 перевищують решту. Це можна пояснити популярністю серед батьків, якістю навчання, а також досить високими результатами ЗНО, що безперечно призводить до отримання бажаного результату – вступу до обраного навчального закладу на бюджет.

З іншого боку бачимо навчальні заклади з невеликою кількістю учнів – 7-14, до них належать: 8, 25, 32, 35.

Використовуючи дані результатів ЗНО за 2019 рік візьмемо до прикладу по дві школи з максимальною та мінімальною кількістю 11-класників. В даному випадку показником стане середній бал з трьох предметів, який розділимо на такі діапазони: 200-190, 190-180, 180-170-170-160, 160-150, 150-140, 140-0. Бачимо, що

порівнюючи наповнюваність і кількість балів між ближніми школами результати дуже сильно відрізняються.

Розглянемо верхні діаграми (рис. 2.8), кількість учнів, що здали іспити зі ЗНО менше ніж на 140 балів – різниця – 14 осіб, від 190 до 180 балів – різниця 12 осіб. Аналогічна ситуація з нижніми діаграмами в процентному відношенні

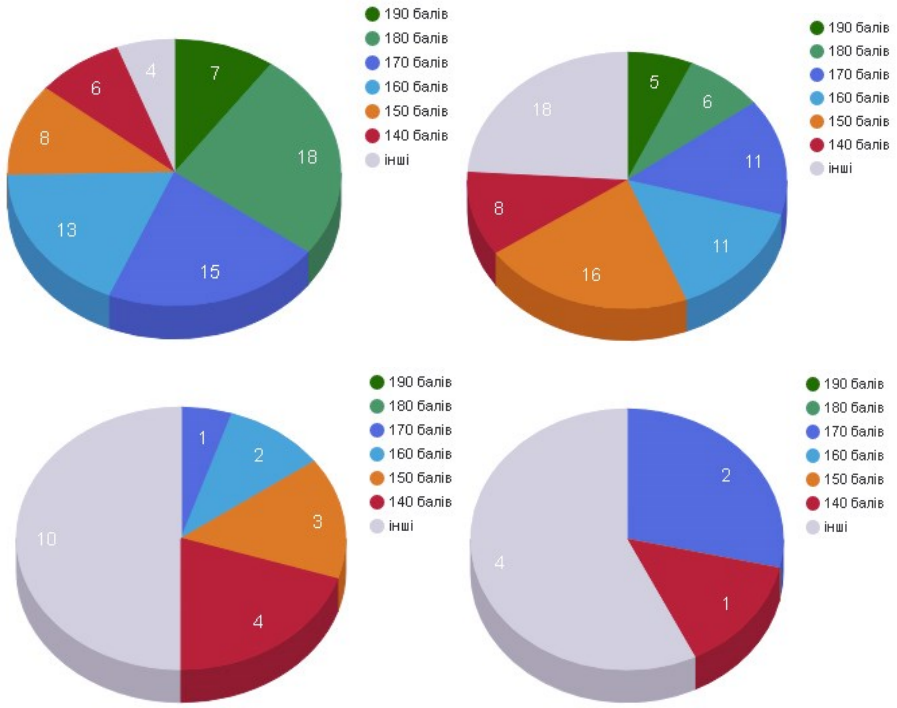


Рис. 2.8 – Діаграма розподілу випускників за кількістю балів

Використовуючи відкриті дані з сайту Відкритої школи [61], ми можемо оперувати ще більшою кількістю інформації. Для аналізу використовуємо відкриті дані про навчальний заклад, інформацію про учнів за 2016 -2019 навчальні роки, приміщення, фінансування, результати ЗНО, вступ до закладу вищої освіти України, педагогічний колектив школи, матеріальна база, кількість робочих місць за ПК, інтерактивні комплекси та багато інших даних.

Аналіз даних діаграми свідчить, що майже всі випускники 11 середніх навчальних закладів стали студентами ЗВО, подібна ситуація у 28 навчальних закладах – з незначною різницею. Це дає підстави стверджувати, що в даних навчальних закладах освітні процеси ведуться на досить високому рівні.

Якщо детальніше проаналізувати відомості про трійку шкіл-лідерів за кількістю учнів в школах, то бачимо, що різниця між кількістю випускників та

кількістю тих, що вступили до ЗВО складає від 5 до 9 осіб, подібна ситуація притаманна навчальним закладам – 2 (8), 4 (6), 5(10), 16 (9) та деяким іншим. Вивчивши результати ЗНО випускників даних навчальних закладах в звіті Центру оцінювання освіти та на сайті «Відкритої школи» можна стверджувати, що тих хто не склав іспитів практично немає. Така ситуація свідчить, що учні даних навчальних закладів з великою ймовірністю могли обирати для навчання широкий спектр ЗВО.

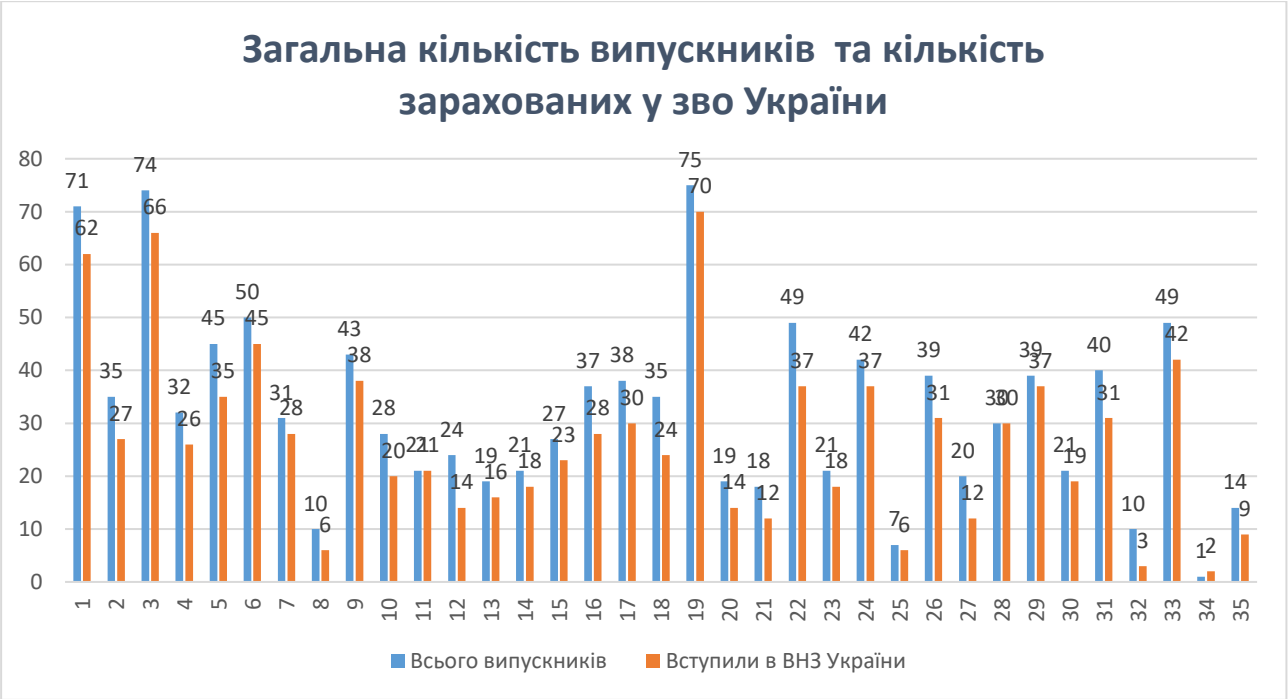


Рис. 2.9 – Загальна кількість випускників та кількість зарахованих до закладів вищої освіти України

В той же час, взявши до прикладу, результати ЗНО навчального закладу 27, де різниця між тими хто вступив на бюджет на та тими хто поступив на контракт склала 11 осіб. Бачимо, що тільки один з 12 випускників отримав 170 балів з трьох предметів. Такий результат може свідчити про низьку вмотивованість до продовження навчання, підбір учнів, незацікавленість батьків, складне матеріальне становище сімей, недостатній рівень викладання. Сім'ї восьми з двадцяти випускників, а це майже 40 відсотків, були змушені відмовитись від оплати здобуття вищої освіти донькою чи сином. Частина 11-класників, які мають бажання навчатись далі, можуть вступити до закладів професійно-технічної освіти.

За допомогою функціональних можливостей програмно-алгоритмічного комплексу сформовано діаграму, яка відображає результати обрання спеціальностей ІТ галузі випускниками шкіл м. Тернополя.

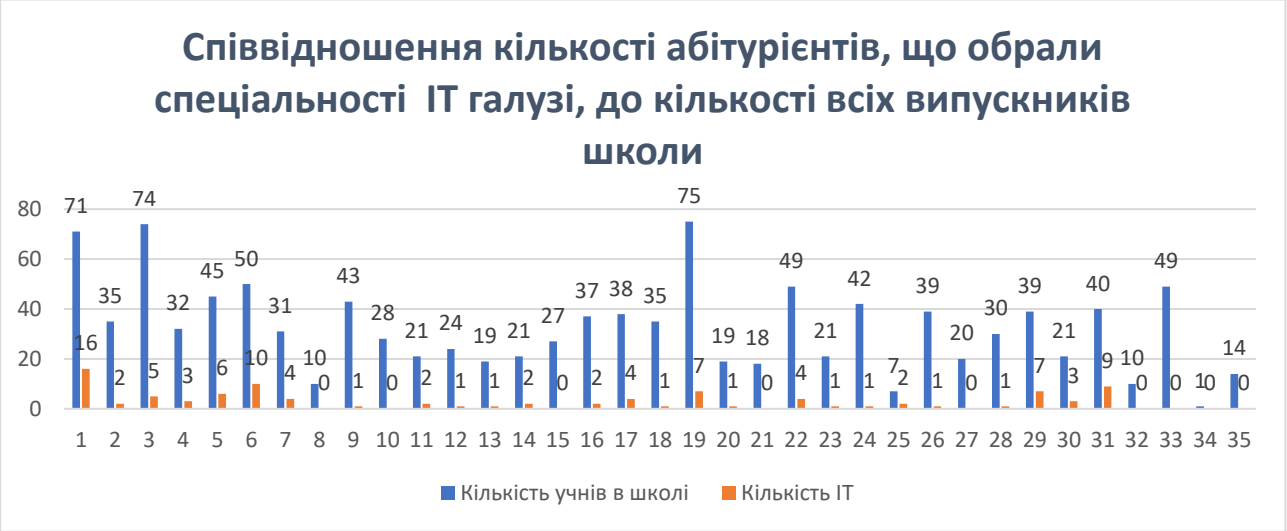


Рис. 2.10 – Кількість абітурієнтів, що вступили на спеціальності ІТ галузі та їх відсоткове значення по школах.

В Тернополі є 5 навчальних закладів, які за кількістю майбутніх ІТ спеціалістів в 2-3 рази перевищують результати інших навчальних закладів. Це вказує на те, що профіль школи відіграє важливу роль, проте, до прикладу навчальний заклад 6 має історично-філологічне спрямування, що не завадило його учням обрати технічні спеціальності.

Аналіз даних по школах, свідчить:

- 430 випускників шкіл зараховано на бюджетну форму навчання;
- 507 випускників шкіл зараховано на контрактну форму навчання;
- загальна кількість випускників шкіл м. Тернополя, що зараховано до ЗВО України – 937.

Як бачимо на діаграмі (рис. 2.11) з 937 учнів, які вступили на навчання в ЗВО України 97 осіб складають ті, що обрали такі спеціальності ІТ галузі:

- (121) інженерію програмного забезпечення;
- (122) комп'ютерні науки;
- (123) комп’ютерна інженерія;

- (124) системний аналіз;
- (125) кібербезпека;
- (126) інформаційні системи та технології.

Рис. 2.11. Кількість абітурієнтів, що вступили на спеціальності ІТ галузі та їх відсоткове значення по школах.

2.3 Методи комплексного аналізу нахилів абітурієнтів до ІТ профілю

Комплексний аналіз нахилів абітурієнтів (КА) для ІТ профілю може включати різноманітні методи та підходи для оцінки їхніх здібностей, інтересів та потенціалу в галузі інформаційних технологій. Подамо кортежем комплекс методів, які можуть бути використані в такому аналізі:

$$КА=\{ТЗ, ОП, МЗ, АП, ІС, АМ\}, \quad (2.3)$$

Тестування здібностей (ТЗ). Використання спеціальних тестів на логічне мислення, математичні здібності, аналітичний розум тощо, може допомогти виявити потенційні здібності абітурієнтів у галузі ІТ. Тестування на здібності є одним із методів оцінки нахилів абітурієнтів до конкретних реалій або областей. В контексті ІТ профілю, такі тести можуть виявити особливі здібності та потенціал абітурієнтів у сфері інформаційних технологій. Ось деякі аспекти, які можуть бути включені в тестування на здібності:

$$ТЗ=\{ЛМ, МЗ, АН, ПУ, ТІ, АС, КН\}, \quad (2.4)$$

де ЛМ- логічне мислення. Тести на логічне мислення можуть містити завдання, які вимагають аналізу паттернів, розв'язання логічних головоломок та встановлення зв'язків між об'єктами.

МЗ - математичні здібності. Тести на математичні здібності можуть включати завдання з розв'язування математичних задач, обчислень та використання алгоритмів.

АН - аналітичні навички. Завдання, спрямовані на розв'язання аналітичних завдань, можуть допомогти виявити абітурієнтів з здатністю розбиратися в складних ситуаціях та знаходити рішення.

ПУ - просторова уява. Тести на просторову уяву можуть оцінювати здатність абітурієнтів сприймати та працювати з тривимірними об'єктами та даними.

ТІ - творчість та інноваційність. Завдання, які спонукають абітурієнтів думати творчо та пропонувати нові рішення, можуть виявити їхню творчу природу.

АС - аналіз ситуацій. Тести, які ставлять перед абітурієнтами реальні або уявні ситуації та запитують про їхні реакції та рішення, можуть допомогти визначити їхню здатність аналізувати та вирішувати проблеми.

КН - комунікативні навички. Деякі тести можуть оцінювати комунікативні навички абітурієнтів, які можуть бути важливими для роботи в колективі.

Тестування здібностей може допомогти визначити сильні сторони та потенціал абітурієнтів у галузі ІТ, а також спрямувати їх на відповідний шлях розвитку та навчання.

Оцінка вмінь з програмування (ОП). Проведення тестування та вирішення задач, пов'язаних з програмуванням, може допомогти виявити абітурієнтів, які мають певний рівень навичок з програмувати та вміють розв'язувати технічні завдання. Оцінка навичок з програмування є важливим елементом аналізу нахилів абітурієнтів для вступу на спеціальності ІТ галузі. Вона дозволяє виявити їхні вміння, здібності та рівень практичних знань у галузі програмування. Одним із способів оцінки навичок із програмування є розв’язання тестових завдань, які вимагають написання програмного коду для вирішення конкретних задач. Такі завдання можуть тестувати рівень розуміння синтаксису, алгоритмічних

здібностей та способу мислення. Це може включати розроблення програмного коду, застосунку або веб-сайту з використанням різних технологій. Завдання з аналізу існуючого програмного коду та його розбір може допомогти оцінити здатність абітурієнта розуміти та впроваджувати вже існуючий код. Використання завдань, пов'язаних з розробкою алгоритмів, може допомогти виявити абітурієнтів з сильним алгоритмічним мисленням. Оцінюється вміння абітурієнта аналізувати принципи структури та стилю програмного коду, що може допомогти визначити їхній професійний рівень. Абітурієнти можуть представити портфоліо своїх IT проектів, яке відображає їхні здібності, досвід та креативність у програмуванні. Оцінка рівня навичок з програмування може допомогти ідентифікувати абітурієнтів з сильним потенціалом для подальшого навчання та розвитку у галузі IT.

Оцінка математичних здібностей (МЗ). Враховуючи, що знання математики є важливим для фахівців IT галузі, тестування математичних здібностей може допомогти виявити абітурієнтів з значним потенціалом у цій області. Оцінці математичних здібностей сприяють проведення тестування на основі завдань, які вимагають розв'язування математичних задач. Такі завдання можуть включати алгебру, геометрію, диференціальні рівняння та інші питання. Оцінка здатності абітурієнтів працювати з числами, розв'язувати задачі криптографії, теорії чисел та інші математичні завдання. Використання математичних головоломок та викликів може допомогти оцінити здатність абітурієнтів до креативного мислення та розв'язування нетривіальних математичних завдань. Проведення досліджень на базі математичних методів може допомогти визначити здатність абітурієнтів до аналітичного мислення та самостійного дослідження. Оцінювання навичок використання програм для математичного моделювання, аналізу даних та обчислень. Математичні здібності також включають розуміння алгоритмів, логічних висновків та вміння використовувати їх у вирішенні завдань. Оцінювання вмінь абітурієнтів використовують математичні методи для розв'язання реальних проблем, може вказати на їхні здібності та прикладний потенціал. Оцінка математичних здібностей допомагає визначити, наскільки абітурієнти можуть

успішно опанувати складні технічні та математичні аспекти спеціальності ІТ галузі, що важливо для їхнього подальшого успіху у ній.

Аналіз проектів (АП). Цей підхід передбачає представлення абітурієнтами своїх робіт або проектів. Це дозволяє оцінити їхній практичний досвід та креативність. Аналіз проектів є ефективним способом оцінки нахилів та потенціалу абітурієнтів у галузі інформаційних технологій. Реальні проекти можуть виявити їхні практичні навички, креативність та здатність до співпраці в команді. Аналіз того, наскільки великі та складні проекти, які абітурієнти реалізували, може вказати на їхні можливості у вирішенні технічних завдань. Врахування того, які технології та інструменти використовували абітурієнти, може показати їхні знання та досвід у сфері ІТ. Оцінка того, наскільки оригінальні та інноваційні були проекти, може вказати на креативний підхід абітурієнтів. Аналіз того, які результати були досягнуті у проектах, може показати ефективність та вплив їхньої роботи. Врахування, чи виконували абітурієнти проекти в командах, як вони співпрацювали та вирішували конфлікти, може вказати на їхні комунікативні навички. Оцінка дизайну та користувацького інтерфейсу відображає здатність абітурієнтів до створення зручних та привабливих продуктів. Вивчення, які проблеми були виявлені та вирішені під час реалізації проекту, може вказати на їхню аналітичну та здатність вирішувати проблеми. Абітурієнти можуть представити портфоліо своїх робіт, яке містить програмні проекти, творчі завдання тощо. Це дасть можливість оцінити їхній досвід та навички. Портфоліо робіт є важливим інструментом для оцінки нахилів, навичок та творчого потенціалу абітурієнтів у галузі інформаційних технологій. Воно дозволяє представити конкретні проекти, завдання або продукти, які абітурієнт створив або розробив. До портфоліо можуть бути включені низка реалізованих проєктів. Представлення власного програмного коду, який абітурієнт написав для різних проектів чи завдань. Це може бути як повний проект, так і окремі фрагменти коду. Опис та представлення проектів, які абітурієнт розробив. Це можуть бути програми, додатки, веб-сайти, ігри або інші цифрові продукти. Показ реалізованих алгоритмів, рішень задач або завдань з програмування. Представлення дизайну та

користувацького інтерфейсу проектів. Це може включати макети, ілюстрації, дизайн веб-сайтів та інше. Представлення технічної документації до проектів, включаючи опис функціональності, архітектури, використані технології та інше. Презентація опису кожного проекту чи завдання, його цілей, використаних методів та досягнених результатів. Демонстрація технічних навичок, які були застосовані при реалізації проектів, включаючи мови програмування, фреймворки, інструменти та інше. Портфоліо робіт дозволяє абітурієнтам показати свої практичні навички, творчість та здатність до співпраці в команді, що є важливим для визначення їхньої придатності до програми ІТ профілю. Аналіз проектів може допомогти отримати повнішу картину здібностей та знань абітурієнтів у сфері ІТ, а також виявити тих, хто може виявитися успішними та цікавими учасниками спеціалізованої програми.

Інтерв'ю та співбесіди (ІС). Проведення інтерв'ю або співбесід з абітурієнтами, під час яких можна визначити їхні зацікавленості, мотивацію та ставлення до спеціальностей ІТ галузі. Інтерв'ю та співбесіди є значущими засобами оцінки нахилів та можливостей абітурієнтів у галузі інформаційних технологій. Цей підхід дозволяє отримати більш глибоке розуміння їхніх навичок, мотивації та способів думання. Питання, пов'язані з тим, чому абітурієнти обрали ІТ профіль та які їхні професійні мети, можуть допомогти з'ясувати їхню мотивацію та зацікавленість. Обговорення попереднього досвіду у сфері ІТ, програмування, проектів та інших відповідних аспектів може допомогти оцінити їхні навички та підготовку. Інтерв'ю може включати питання, спрямовані на виявлення їхніх здібностей у сфері програмування, алгоритмів, математики та технічних знань. Аналіз реальних або уявних проектів, в яких вони брали участь, може показати їхні здібності до практичної роботи. Співбесіда дозволяє оцінити їхні комунікативні навички, вміння пояснювати складні концепції та виражати свої думки. Запитання, які ставлять перед абітурієнтами аналітичні завдання або сценарії, допомагають виявити їхню здатність до критичного мислення та аналізу. Обговорення того, як абітурієнти підходять до вирішення технічних проблем, може вказати на їхній стиль роботи та рівень спеціалізації. Вивчення того, як абітурієнти бачать свій

майбутній внесок у галузь ІТ, може показати їхні амбіції та довгострокові цілі. Представлення достатнього високого рівня розуміння та володіння матеріалом з інформатики, програмування, математики та інших відповідних предметів. Пояснення мотивації, яка веде абітурієнта до обраної галузі, та професійних цілей. Інтерв'ю та співбесіди дають можливість з'ясувати глибокі аспекти особистості, досвіду та амбіцій абітурієнтів, які можуть бути важливими для прийняття рішення щодо їхнього включення в програму ІТ профілю.

Оцінка аналітичного мислення (АМ). Використання тестів або завдань, що вимагають аналізу, логічного розуміння та вміння приймати рішення, може допомогти визначити абітурієнтів з потенціалом в галузі ІТ. Оцінка аналітичного мислення важлива для виявлення здатності абітурієнтів до розуміння та аналізу складних ситуацій, розв'язання проблем та здійснення логічних висновків. В контексті ІТ профілю, аналітичне мислення є ключовим для вирішення технічних завдань та розробки програмного коду. Надання абітурієнтам реальних або уявних ситуацій та питань, які вимагають аналізу та висновків. Це можуть бути завдання, що вимагають знаходження рішень, прогнозування та обґрунтування варіантів. Використання завдань, де абітурієнти повинні розглянути проблему або завдання, виявити можливі причини та шляхи розв'язання. Використання головоломок, логічних завдань та завдань, які вимагають послідовного мислення та логічних висновків. Завдання, де абітурієнти мають проаналізувати та зробити висновки з великих обсягів даних, можуть вказати на їхню здатність виділяти важливу інформацію та робити аналітичні висновки. Завдання, пов'язані з розв'язанням технічних проблем або розробкою алгоритмів, можуть вимагати глибокого аналізу та структурованого підходу. Використання математичних моделей для аналізу складних процесів та здійснення обчислень на основі даних. Аналіз технічного дизайну або архітектури програмних систем для виявлення потенційних проблем, оптимізацій та поліпшень. Оцінка аналітичного мислення допомагає визначити здатність абітурієнтів до розуміння та розв'язання складних технічних завдань, що є важливим аспектом у сфері інформаційних технологій.

Ця перевірка допомагає визначити загальну готовність абітурієнтів до роботи в сфері ІТ, їхнє розуміння ключових понять та принципів, які важливі для будь-якого технічного спеціаліста.

Тестування професійних знань допомагає визначити рівень готовності абітурієнтів до навчання на спеціальностях ІТ галузі, а також визначити їхню підготовку та потенціал у відповідних галузях ІТ. Комплексний аналіз об'єднує різні методи та оцінки. Інформаційні комплекси, які використовуються сьогодні в консультативних системах є не достатньо ефективними. Зокрема, практично відсутня можливість в одній інформаційній точці проаналізувати інформацію про людину як об’єкта профорієнтаційної та освітньої роботи та отримати вичерпні інформаційно-аналітичні дані регіонального ринку праці та освітніх послуг. Інформація в основному надається без належної достовірності і структуризації.

Модель може бути побудована з використанням різних методів машинного навчання, таких як дерева рішень, метод опорних векторів або нейронні мережі, залежно від обсягу даних та складності задачі. Така модель може допомогти абітурієнтам знайти ідеальну спеціальність в галузі ІТ відповідно до їхніх нахилів і здібностей. Модель процесу аналізу даних використаємо для подання процесу визначення професійних нахилів та здібностей особи.

На основі опрацювання результатів профорієнтаційного тестування розроблено метод для визначення нахилів до навчання на спеціальностях ІТ галузі, який складається з наступних кроків (рис. 2.12) [62]:

Крок1. Збір даних. Збір результатів тестування абітурієнтів, включаючи відповіді на питання або завдання, які оцінюють їхні професійні знання.

Крок 2. Попереднє опрацювання даних. Попереднє опрацювання даних дозволяє видалити помилки та аномальні значення.

Крок 3. Розподіл даних на групи. Визначення різних спеціальностей ІТ галузі, на які абітурієнти можуть подавати заяви.

Крок 4. Побудова моделі. Розроблення моделі, яка оцінює нахили абітурієнтів до конкретних спеціальностей ІТ. Модель може використовувати алгоритми машинного навчання, такі як класифікація, щоб передбачити, які

спеціальності найкраще підходять кожному абітурієнту на основі їхніх відповідей на тестові питання.

Крок 5. Навчання моделі. Використання навчальних даних для навчання моделі та налаштування її параметрів.

Крок 6. Валідація моделі. Перевірка точності та ефективності моделі на тестових даних.

Крок 7. Прогнозування нахилів. Використання моделі для передбачення нахилів кожного абітурієнта до конкретних спеціальностей ІТ галузі.

Крок 8. Виведення результатів. Представлення результатів аналізу у зручній формі, наприклад, графіках або таблицях.

Крок 9. Підведення підсумків і формування рекомендацій. Підведення підсумків та надання рекомендацій абітурієнтам щодо спеціальностей ІТ, які найкраще відповідають їхнім нахилам і знанням.

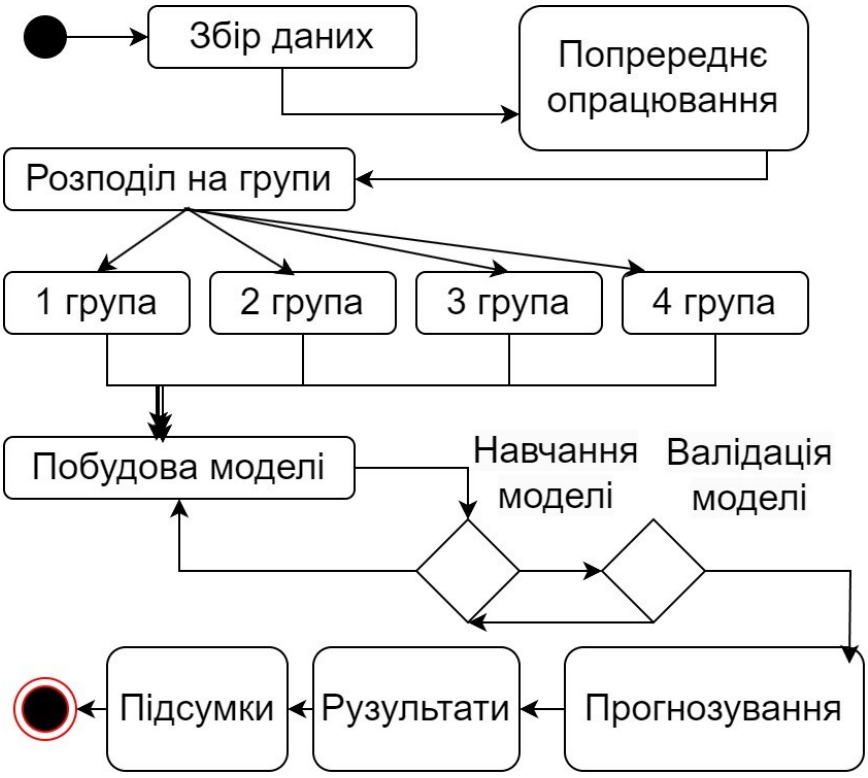


Рис. 2.12. Алгоритм процесу визначення професійних здібностей особистості

Розглянемо приклад побудови моделі з використанням дерева рішень для визначення нахилів абітурієнтів до спеціальностей ІТ. Для цього прикладу я

використовую бібліотеку scikit-learn для Python, яка надає інструменти для машинного навчання (*додаток Б, лістинг 2.1*). Точність моделі: 100.00%

Цей код побудує модель дерева рішень, навчить її на тренувальних даних та оцінить її точність на тестових даних. Точність визначає, наскільки добре модель передбачає нахил абітурієнтів до спеціальностей ІТ на основі їх відповідей на тестові питання. Окрім того, важливо враховувати, що використання дуже малої вибірки не сприяє ефективному навчанню моделі, а точність може варіюватися від запуску до запуску через випадковий розподіл даних для тренування і тестування. Збільшення розміру даних сприятиме поліпшенню стабільності оцінки точності. Це лише один з прикладів, і реальні дані та задачі можуть вимагати більш складних моделей та більшого обсягу даних для точних передбачень.

Розглянемо приклад побудови моделі з використанням методу опорних векторів (SVM) для визначення нахилів абітурієнтів до спеціальностей ІТ. Знову ж таки, для цього прикладу використовується бібліотека scikit-learn для Python, (*додаток Б, лістинг 2.2*).

У цьому прикладі ми використали модель SVM з ядром linear. Модель навчена на тренувальних даних і оцінена на тестових даних для визначення точності передбачень. Метод опорних векторів (SVM) особливо корисний у випадках, коли класи (спеціальності) мають складні границі розділення.

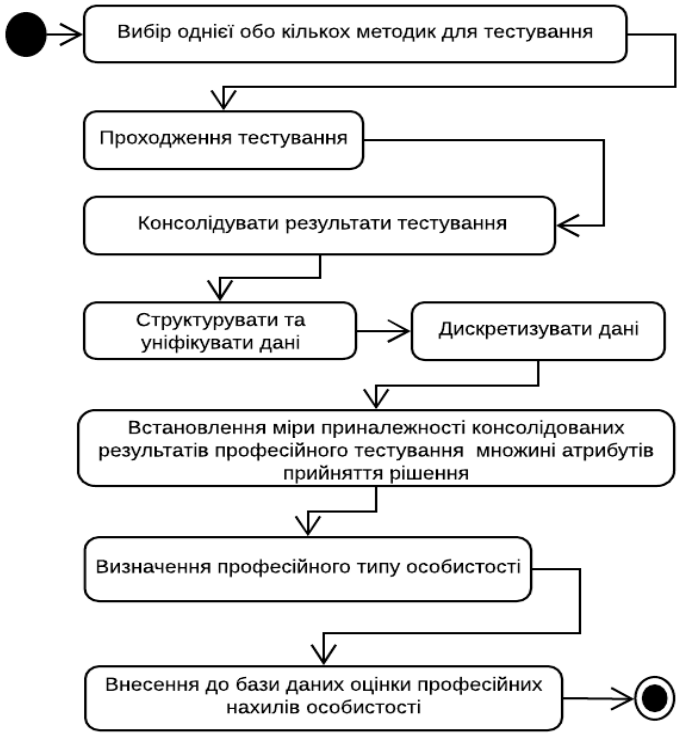


Рис. 2.13. «Діаграма діяльності» процесу визначення професійних здібностей особистості

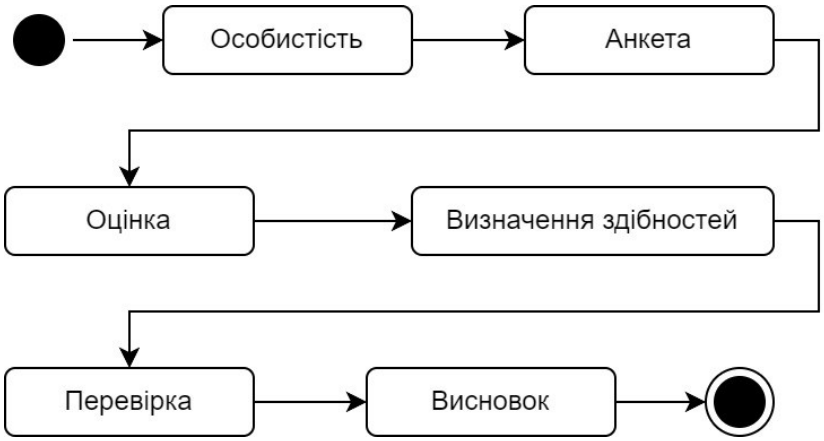


Рис. 2.14. Алгоритм визначення професійних здібностей особистості

1. Початок.
2. Визначення особистості..
3. Проведення анкетування.
4. Оцінка відповідей.
5. Визначення професійних здібностей.

6. Перевірка результатів.
7. Внесення отриманих результатів у базу даних.
8. Висновок.

За допомогою цієї діаграми діяльності, можна візуалізувати процес визначення професійних здібностей особистості та показати послідовність дій, які в ньому беруть участь.

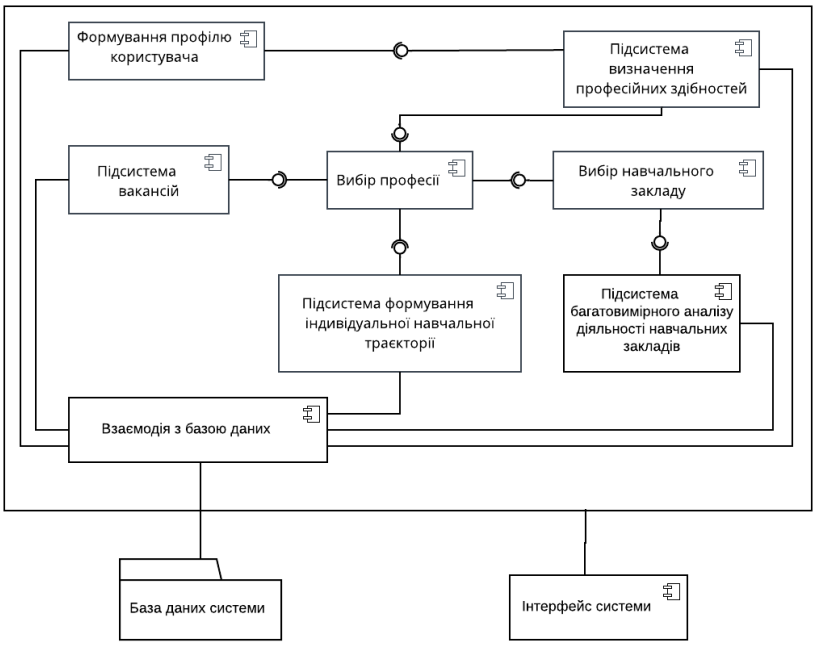


Рис. 2.15 – Структурна модель інформаційної технології персоналізованого супроводу вибору професій

2.4 Способи формування мотиваційної платформи майбутнього IT фахівця

Формування мотиваційної платформи майбутнього IT фахівця є важливим завданням, що допомагає створити позитивне ставлення до навчання та роботи в сфері IT. Доцільно пояснювати абітурієнтам, як важлива та впливова може бути робота в сфері IT. Повідомити їм про те, як технології впливають на сучасний світ і відкривають безмежні можливості. Розуміння значущості - це перший і один з найважливіших кроків у формуванні мотиваційної платформи для майбутніх IT фахівців. Якщо абітурієнти зрозуміють, чому робота в сфері IT є важливою та цікавою, вони будуть більш зацікавлені у поглибленому навчанні та подальшій

кар'єрі. Загалом, показавши абітурієнтам, як важлива та захоплива може бути робота в сфері ІТ, доцільно створити позитивну мотиваційну платформу для їхнього майбутнього.

Необхідно налаштувати позитивне ставленню родини абітурієнта та його вчителів до вибору кар'єри в ІТ. Це може збільшити мотивацію та підтримку абітурієнтів. Підтримка від родини та вчителів є важливим фактором у формуванні мотиваційної платформи для майбутніх ІТ фахівців. Ця підтримка може збільшити самовідчуття, сприяти позитивному ставленню та стимулювати розвиток в обраній галузі. Родина має бути забезпечена інформацією про переваги та можливості, які надає робота в ІТ. Доцільно залучити фахівців, які можуть надати інформацію про кар'єрні можливості в ІТ та відповісти на питання родини та вчителів та забезпечити підтримку у виборі навчального закладу.

Формування мотиваційної платформи передбачає залучення різних методів та підходів, які допоможуть абітурієнтам відчувати зацікавленість та ентузіазм щодо навчання та роботи в сфері ІТ.

У багатьох дослідженнях аргументовано доведено, що вибір професії впливає на успішність діяльності при обійманні професії, можливість проявити свій потенціал. Разом з тим, за інформацією Державної служби зайнятості, 51 % української молоді працює не за фахом [63]. Як свідчать дослідження Міжнародного кадрового порталу HeadHunter (hh.ua) [64], проведені у 2016 року, половини українців працювали не за тією спеціальністю, яку отримали під час навчання. Кожен десятий випускник обіймав посаду певний час після закінчення навчального закладу, але все ж змінив професію, а кожен третій випускник відразу після завершення навчання зміним фах.

Купар Д.М. вважає причиною такої статистики є часто необдуманий вибір майбутньої спеціальності та непристосованість вищої освіти до ринку праці [65]. 65% українців, які працюють за фахом, бажають змінити професію. 32% українців не обіймали посади за фахом, а 15% – певний час працювали за фахом, однак розчарувалися. Як свідчить статистика, половина мешканців України не задоволені здобутою професією, в Німеччині цей відсоток складає лише 10%. 21%

респондентів за фахом не працюють, оскільки їм подобається здобута професія, 43% відмовились через низьку заробітну плату [66].

Результати досліджень зарубіжних науковців засвідчують, що правильний вибір професії здійснений під час навчання в школі в 2-2,5 разів зменшує кількість випадків розчарування здобутою професією, й на 10-15% збільшує продуктивність праці випускника за обраним фахом. За даними статистики, поданої на сайті Кабінету міністрів України, загальні витрати на бюджетне навчання у закладах вищої освіти лише у 2021 р. склали 20,95 млрд грн. [67]. Враховуючи вище наведені дані, виходить, що близько 50 % цих коштів використовується не цільово, оскільки здобуті знання випускниками не використовуються через зміну фаху.

Моделювання процесів розвитку соціально-комунікаційного середовища сучасного міста є важливим інструментом формування цілісної інноваційної освітньої системи міста, яка надає широкий спектр інформаційних, телекомунікаційних та технологічних послуг, що сприяють підвищенню ефективності процесів отримання нових та закріплення раніше набутих знань; набуття необхідних для міста професій; удосконалення процесів обміну інформацією; просторового наближення та соціально-психологічної адаптації інформаційно-пізнавальних освітніх матеріалів до кінцевого користувача (рис. 2.16).

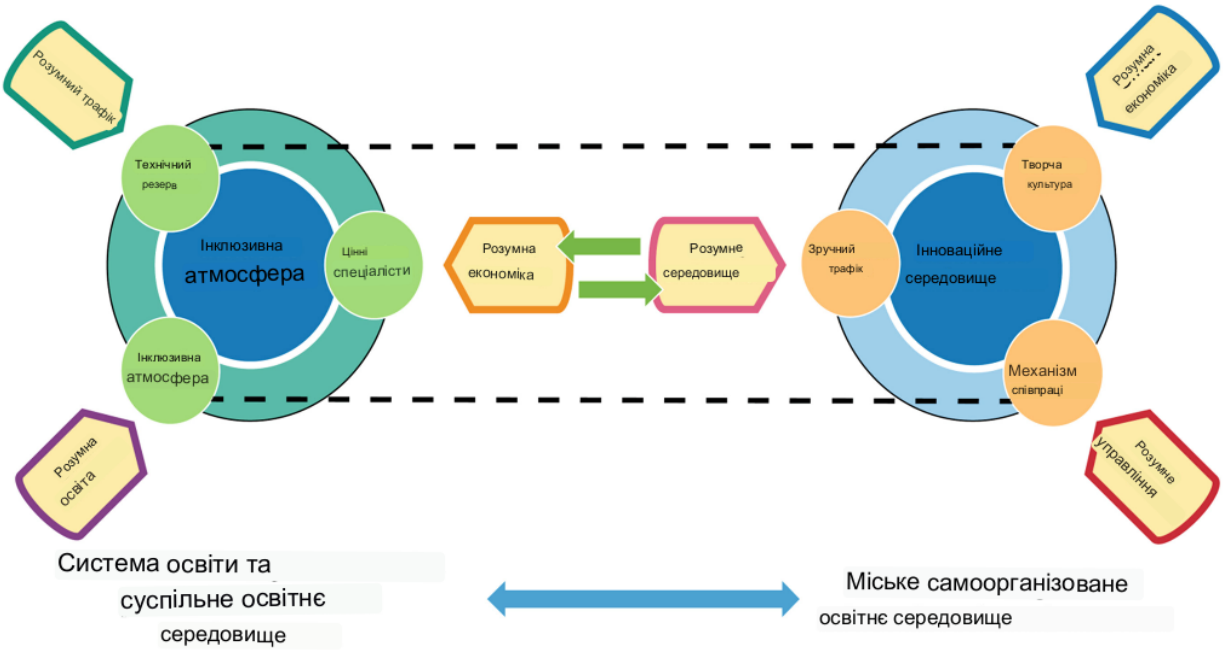


Рис. 2.16 – Освітнє середовище розумного міста

Процеси формування у абітурієнтів мотивації щодо обрання ним фаху, що відповідатиме його нахилам та здібностям, відбувається з врахуванням впливу декількох незалежних чинників - сім’ї, процедури зовнішнього незалежного оцінювання, школи, перспектив подальшого працевлаштування [68].

Середовище, в якому відбувається навчання людини, на відміну від наших очікувань, досить складне. Традиційне інформаційне суспільство породжує великі потоки інформації. Крім того, ця інформація постійно змінюється і часто втрачає свою достовірність. Така ситуація породжує необхідність виділення надійної інформації для прийняття рішень. Через використання недостовірної інформації певні рішення можуть бути помилковими, а рішення, прийняте на основі достовірної інформації, навпаки, може бути успішним. Звичайно, така ситуація призводить до того, що окрема особистість, стикаючись з інформацією, втрачає орієнтацію. Тому в «розумному місті» має бути сформоване соціально-комунікаційне середовище, засноване на сучасних технологіях опрацювання інформації, щоб навіть в умовах обмеженого досвіду та невизначеності прийматися ефективні рішення. У житті кожної людини настає момент, коли потрібно визначити свій подальший шлях. У цьому дослідженні ми хочемо дослідити, що саме найбільше впливає на рішення про навчання та які фактори впливають на цей процес.

У цій роботі проаналізовано лише основні чинники, серед яких: сім'я, ЗНО, школа та інші. При цьому враховуються умови невизначеності, оскільки більшість факторів мають довгостроковий вплив. Використання методології когнітивного моделювання, що сприятиме аналізу сили та спрямованості впливу, факторів приведення об’єкта в цільовий стан.

Основою таких когнітивних моделей зазвичай є класична когнітивна карта. Когнітивна карта поточної ситуації – це орієнтований зважений граф, вершини якого відповідають факторам, що є найбільш впливовими в даній ситуації [69]. Дуги визначають між факторами, взаємний вплив одного фактора на інший [70].

Розробка когнітивної моделі складної системи найчастіше починається з побудови когнітивної карти - знаково орієнтованого графа, отримується через структурування знань експертом з даної тематики на основі теоретичного понять, статистичних даних, застосування різних експертних методів. Більш складні когнітивні моделі - це векторний параметричний граф, та модифікований функціональний граф.

У когнітивній методології, що розробляється, знаходять застосування когнітивні моделі в різних формах, в залежності від їх призначення. Моделі розробляються на основі структуризації знань експертів, використання теоретичних знань, використання статистичних даних.

$$Фп = G, X,F,\theta , , \qquad (2.5)$$

де Фп - параметричний векторний функціональний граф, G V, E - когнітивна карта - знаковий орієнтований граф, в якому $V = \{v_i\}$, i = 1,2, ..., k - безліч вершин («концепти », «Об'єкти», «сутності» предметної області), $E = \{e_{ij}\}$ - безліч дуг, що з'єднують вершини v_i і v_j; $X = \{x_i\}$ - безліч параметрів вершин, $F = f \{v_i, v_j, e_{ij}\}$ - функція (або функціонал $f \{v_i, v_j, e_{ij}\}$, або коефіцієнт f_{ij}) зв'язку між вершинами, , θ - простір параметрів вершин [71].

Під час формування когнітивних карт певні труднощі, що виникають під час виборів, їх ранжування, для побудови ієрархічних залежностей, а також виявлення масштабного фактора впливу факторів (ваги для дугового графа). Передумовою для вибору базисних факторів став етап вербального соціально-комунікаційного моделювання середовища «Розумне місто» та застосування методу експертних оцінок для визначення ваг факторів впливу. Шкала вагових коефіцієнтів для оцінки впливів була обрана в сумі від 0 до 10. Зрозуміло, для кожної досліджуваної комплексної системи слід вибрати притаманний тільки йому набір факторів, які найбільш істотно впливають на її поведінку і розвиток.

Проаналізуємо, що ж впливає на абітурієнта, та взаємовплив різних факторів. Розглянемо фактор сім’я та взаємодію з абітурієнтом, обидва об’єкти мають однаковий вплив. Можна пояснити так, що абітурієнт разом з сім’єю радиться в який навчальний заклад йому вступати, чи може в разі чого сім’я його підтримати,

чи це буде вибір спрямований обирати ту ж професію, що й близькі родичі (батько, дідусь). Якщо розглядати дану ситуацію з інших позицій, побачимо, що майбутній абітурієнт має вже практично повністю сформований характер і свій погляд на речі, які вибирає в житті. Він має намір втупити до технічного вузу, бо бачить себе в подальшому програмістом (тим більше над цим працює ще зі школи, і досвід в нього вже є).

При побудові графу ми вибрали рівні оцінки, вагою в 5 балів. Якщо уважно подивитись на елементи, які ми досліджуємо, то побачимо так звані петлі. Тобто всі елементи крім ЗНО мають більш довготривалий вплив, який може змінюватися, зовнішнє незалежне тестування онлайн по-перше триває відносно короткий термін, по-друге його можна виміряти, тобто оцінки будуть мати для абітурієнта практично визначну роль, тому подали третьому елементу вагу в 10 балів. Зрозуміло, що на вступ впливають не тільки оцінки ЗНО, але й середній бал атестату випускника, престижність навчального закладу, факультету, конкурс, умови вступу та інші фактори [72].

Проаналізувавши граф, бачимо, що сім’я, школа та доступ до інтернет мають по три зв’язки кожен, а ЗНО, працевлаштування, доступ до інтернету та місце проживання мають найбільшу вагу.

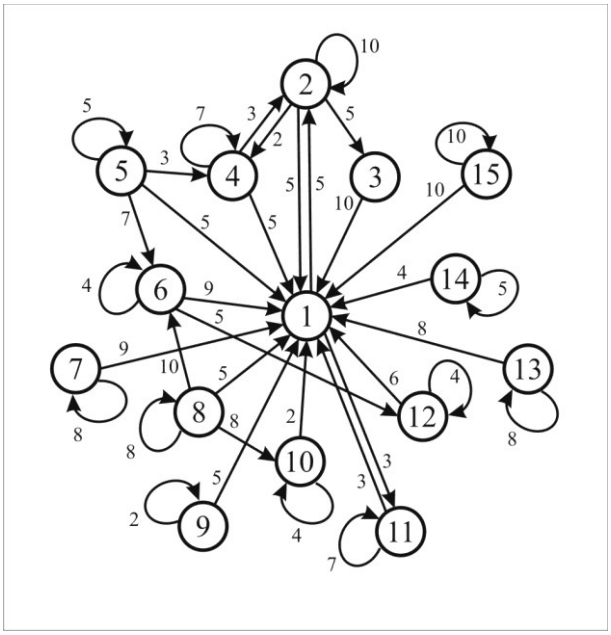


Рис. 2.17 Граф впливу на абітурієнта різних факторів.

Елементи: 1 - Абітурієнт, 2 - Сім’я, 3 - ЗНО, 4 - Школа, 5 - Достаток, 6 - Доступ інтернет, 7 - Самоосвіта, 8 - Місце проживання, 9 - Гурток, 10 - Здоров’я, 11 - Друзі, 12 - Соціальні мережі, 13 - Коледж, 14 - Вулиця, 15 - Працевлаштування

Оскільки немає механізму визначення реальних факторів взаємодії, така взаємодія описується неспотвореною. У міру використання результатів оцінки експертами буде використана формула [73]

$$\Delta Y_i(k) = Y_i(k) - Y_i(k-1) \quad (2.6)$$

$$\Delta Y_i(k+1) = \sum_{i=1}^{15} a_{ij} \cdot \Delta Y_i(k) \text{ - для першого графу.} \quad (2.7)$$

$$\Delta Y_i(k+1) = \sum_{i=1}^{19} a_{ij} \cdot \Delta Y_i(k) \text{ - для другого графу.} \quad (2.8)$$

Таблиця 2.4 – Матриця суміжності графу впливу на абітурієнта.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
V1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
V2	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
V3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
V4	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
V5	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
V6	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0
V7	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V8	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0
V9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V10	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
V11	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V12	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
V13	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V14	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V15	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

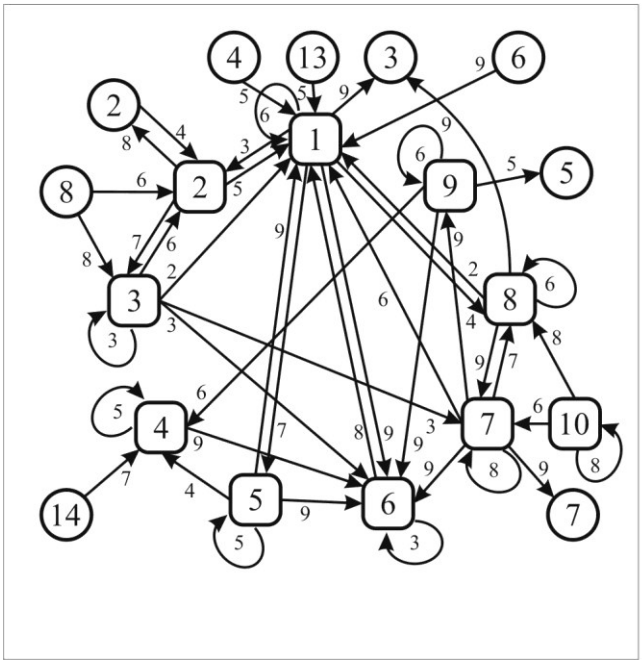
Таблиця 2.5 – Матриця вагових коефіцієнтів суміжності графу впливу на абітурієнта.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
V1	0	5	0	0	0	0	0	0	0	0	3	0	0	0	0
V2	5	10	5	2	0	0	0	0	0	0	0	0	0	0	0
V3	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V4	5	3	0	7	0	0	0	0	0	0	0	0	0	0	0

V5	5	0	0	3	5	7	0	0	0	0	0	0	0	0	0
V6	9	0	0	0	0	4	0	0	0	0	0	5	0	0	0
V7	9	0	0	0	0	0	8	0	0	0	0	0	0	0	0
V8	5	0	0	0	0	10	0	8	0	8	0	0	0	0	0
V9	5	0	0	0	0	0	0	0	2	0	0	0	0	0	0
V10	2	0	0	0	0	0	0	0	0	4	0	0	0	0	0
V11	3	0	0	0	0	0	0	0	0	0	7	0	0	0	0
V12	6	0	0	0	0	0	0	0	0	0	0	4	0	0	0
V13	8	0	0	0	0	0	0	0	0	0	0	0	8	0	0
V14	4	0	0	0	0	0	0	0	0	0	0	0	0	5	0
V15	10	0	0	0	0	0	0	0	0	0	0	0	0	0	10

В другому графі абітурієнту надано 10 основних якостей, замінивши самого абітурієнта на: знання, культуру, патріотизм, вміння працювати в команді, креативне мислення, досягнення поставленої мети, силу волі, здоров'я, працелюбність, заняття спортом. Елементи 10 якостей тут ми позначили прямокутниками. Таким чином створено двошаровий граф, додавши елементи першого, для того, щоб побачити як вони впливають на досягнення мети.

На другий шар ми додали основні, найвпливовіші елементи першого графу: 2 - сім'я, 3 – ЗНО, 4 – школа, 5 – достаток, 6 - доступ до інтернет, 7 – самоосвіта, 8 - місце проживання, 13 – коледж, 14 - друзі і вулиця. Для того щоб не сплутати елементи другого графу з новими 10 характеристиками ми позначили їх колами. Проаналізувавши другий граф бачимо наскільки вибір став ще складнішим. Головними якостями стали: знання – 10 впливів, досягнення поставленої мети – 7 впливів. На знання мають найбільший вплив не тільки елементи 2 графу, але й першого, тобто вони виходять ключовим елементом. При цьому вони є не короткотривалими в часі.



1. Знання
2. Культура
3. Патріотизм
4. Вміння працювати в команді
5. Креативне мислення
6. Досягнення поставленої мети
7. Сила волі
8. Здоров'я
9. Працелюбність
10. Заняття спортом

2. Сім'я
3. ЗНО
4. Школа
5. Достаток
6. Доступ інтернет
7. Самоосвіта
8. Місце проживання
13. Коледж
14. Друзі і вулиця

Рис. 2.18 – Граф впливу елементів першого графу на абітурієнта, як об’єкта, якого можна охарактеризувати такими якостями як: 1 – знання, 2 – культура, 3 – патріотизм, 4 - вміння працювати в команді, 5 - креативне мислення, 6 - досягнення поставленої мети, 7 - сила волі, 8 - здоров'я, 9 – працелюбність, 10 - заняття спортом.

Позначимо якості абітурієнта, яких визначаємо 10, буквою П – прямокутник, а зовнішні впливи з першого графу буквою К – коло.

Таблиця 2.6 – Матриця суміжності другого графу впливу на абітурієнта, як особи, яку характеризують сукупність характеристик.

	П1	П2	П3	П4	П5	П6	П7	П8	П9	П10	К2	К3	К4	К5	К6	К7	К8	К13	К14
П1	0	1	1	0	1	1	1	1	0	0	0	1	1	0	1	0	0	1	0
П2	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
П3	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
П4	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1
П5	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
П6	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
П7	1	0	1	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0	0
П8	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
П9	0	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0

П10	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
К2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К5	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
К6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
К8	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К13	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К14	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Таблиця 2.7. Матриця вагових коефіцієнтів суміжності графу впливу на абітурієнта, як особи, яку характеризують сукупність характеристик.

	П1	П2	П3	П4	П5	П6	П7	П8	П9	П10	К2	К3	К4	К5	К6	К7	К8	К13	К14
П1	6	3	0	0	7	9	0	4	0	0	0	9	0	0	0	0	0	0	0
П2	5	0	7	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0
П3	2	6	3	0	0	3	3	0	0	0	0	0	0	0	0	0	0	0	0
П4	0	0	0	5	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0
П5	9	0	0	4	5	9	0	0	0	0	0	0	0	0	0	0	0	0	0
П6	8	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
П7	6	0	0	0	0	9	8	7	9	0	0	0	0	0	0	9	0	0	0
П8	2	0	0	0	0	9	9	6	0	0	0	9	0	0	0	0	0	0	0
П9	0	0	0	6	0	0	6	0	6	0	0	0	0	5	0	0	0	0	0
П10	0	0	0	0	0	0	0	8	0	8	0	0	0	0	0	0	0	0	0
К2	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К8	0	6	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К13	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
К14	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

На другому рівні знаходяться істотні, найвпливовіші елементи першого графіка: 2 – сім’я, 3 – ЗНО, 4 – школа, 5 – достаток, 6 – доступ до Інтернету, 7 – самоосвіта, 8 – місце проживання, 13 – коледж. , 14 – друзі та вулиця, позначені

кружками. Аналіз другого графіка показує, що відбір став ще складнішим. Основні якості: знання – 10 впливів, успішність – 7 впливів.

На вершину «знань» впливають не тільки елементи другого графа, але й першого, оскільки вони стають ключовим елементом. Слід враховувати, що вони є не короткочасними, а скоріше тривалими елементами часу.

Елемент «патріотизм» включений в систему, оскільки ситуація в останніх років в Україні засвідчила нагальну потребу налагодження патріотичного виховання. Ні одна вершина графу не має стільки дуг, як ця. Вплив відбувається на інші вершини: «сила волі», «досягнення поставленої мети», «знання» та «культура»

В другому графі є елементи, які діють на найважливіші елементи опосередковано, не напряму. Наприклад вершина «вміння працювати в команді» для вчорашнього школяра, впливає на вершину «досягнення поставленої мети», що в свою чергу дає потрібні «знання».

Маючи 15 у першому та 19 пунктів у другому графіку, ми бачимо, що є багато способів вплинути на абітурієнта. Тим не менш, ми розуміємо, що можемо впливати на нього, коригуючи кілька факторів одночасно. З дослідження з'ясувалося, що вплив може здійснюватися навіть через проміжні фактори, які на перший погляд здаються неважливими, але насправді відіграють значну роль. Цей ефект здатний внести зміни, на які абітурієнт майже не розраховувати.

Але завжди слід дотримуватися золотої середини, щоб не нашкодити, адже витрачаючи час на, здавалося б, дуже важливі справи, ми можемо негативно вплинути на здоров’я, сім’ю чи щось важливе. Когнітивні карти дозволяють нам візуально сприймати кількість сценаріїв, не оновлюючи весь обсяг інформації[74].

Ми проаналізували сукупність факторів, що впливають на абітурієнта, який стоїть перед вибором майбутньої спеціальності, яку буде здобувати після закінчення школи [75]. Аналіз фактору "Сім'я" чітко ілюструє наявність його значної взаємодії з іншими факторами. Абітурієнт радиться з родичами, в який навчальний заклад йому вступати, яку професію обрати і наскільки він може розраховувати на підтримку сім'ї. При цьому важливу роль відіграє наявний сукупний сімейний досвід, який суттєво впливає на детермінацію рішення про

вибір тієї ж професії, представниками якої є близькі родичі. Крім того, існує ситуація, коли майбутній абітурієнт має упереджене переконання та усталений погляд на вибір майбутньої професії. Це, зокрема, може стосуватися вступу до технічного університету, і після його закінчення абітурієнт тісно пов'язує себе з IT-індустрією [22-25]. Експериментально цьому фактору присвоєно дві однакові оцінки по 5 балів.

Сформований таким чином графік може мати петлі (рис. 2.18). У нашому випадку всі елементи, окрім фактору ЗНО, мають довгостроковий вплив, рівень якого може змінюватися з часом. Специфіка фактору зовнішнього незалежного оцінювання полягає в тому, що, по-перше, час, протягом якого він суттєво впливає, є відносно коротким, по-друге, його можна реально виміряти, і отримані при цьому оцінки відіграватимуть помітну роль. Вплив цього фактору експерти визначили як високий, тому він отримав вагу в 10 балів. Зрозуміло, що на вступ впливають не лише результати зовнішнього незалежного оцінювання, але й середній бал атестата, престиж навчального закладу, факультету, конкурсна ситуація, умови вступу та низка інших факторів.

Вершини графіка відображають короткострокові та довгострокові цілі, які впливають на процес прийняття рішення. Скажімо, фактор U15 "Працевлаштування" абітурієнти вважають важливим, оскільки загальною метою вибору спеціальності абітурієнтом є отримання в майбутньому престижної високооплачуваної роботи, тому вплив цього фактору оцінюється в 10 балів, як такий, що має значний вплив на прийняття рішення.

Фактор U8 "Місце проживання" тісно пов'язаний з факторами "Доступ до Інтернету", "Здоров'я" і також має довгостроковий вплив за умови, що абітурієнт тривалий час проживає в містах. Заявники, які проживають у сільській місцевості, зазвичай не мають повноцінного доступу до Інтернету, що значно зменшує вплив цього фактору.

Вершини "Сім'я", "Школа" та "Доступ до Інтернету" мають по три зв'язки, а вершини "Зовнішнє незалежне оцінювання", "Працевлаштування", "Доступ до Інтернету" та "Місце проживання" є найбільш важливими.

Фактори: 1 - Абітурієнт, 2 - Сім'я, 3 - Зовнішнє незалежне оцінювання, 4 - Школа, 5 - Достаток, 6 - Доступ до Інтернету, 7 - Самоосвіта, 8 - Місце проживання, 9 - Гурток, 10 - Здоров'я, 11 - Друзі, 12 - Соціальні мережі, 13 - Коледж, 14 - Вулиця, 15 - Працевлаштування.

2.5 Обрання навчального закладу

Отримавши мотиваційні посили щодо обрання фаху, абітурієнт обирає заклад вищої освіти. Вибір відбувається шляхом рейтингування. Для обчислення рейтингової оцінки діяльності вищого навчального закладу використовуються різні методики та інструментарії підрахунку балів. Частіше за все реалізується принцип співвимірності окремих показників співставленням з показниками того вищого навчального закладу, для якого відповідний окремий показник має максимальне значення у вибірці

Сформуємо метод обчислення рейтингової оцінки діяльності вищого навчального закладу:

Рис. 2.19 – Процес рейтингування ВНЗ

Етап 1. Збір та аналіз даних.

Крок 1. Визначення індикаторів. Спершу визначаються індикатори, які будуть використовуватись для оцінки вищого навчального закладу. Це може включати такі дані, як кількість студентів, відсоток викладачів з науковим ступенем, бюджет, кількість опублікованих досліджень тощо.

Крок 2. Збір даних. Збираються дані, які потрібні для обчислення цих індикаторів. Це може включати аналіз сайтів ЗВО як доступні джерела даних та статистики.

Крок 3. Стандартизація даних. Привести дані до однієї одиниці виміру та структурувати таким чином, щоб їх можна було порівнювати між собою.

Крок 4. Визначення ваг індикаторів. Визначення ваги кожного індикатора відповідно до його важливості для оцінки вищого навчального закладу. Вага може бути виражена у відсотках або інших одиницях.

Етап 2. Розрахунок рейтингової оцінки.

Крок 1. Підсумок ваг індикаторів. Розраховується загальний рейтинг для кожного вищого навчального закладу, сумуючи значення індикаторів, помножені на їх вагу.

Крок 2. Ранжування. Відсортовуються вищі навчальні заклади за отриманими рейтингами у спадному чи зростаючому порядку.

Крок 3. Оцінка та відгуки: Відображаються рейтинги та відповідні відгуки для кожного вищого навчального закладу, щоб надати користувачам інформацію та контекст для їхніх рішень.

Крок 4. Оновлення та регулярний аналіз. Постійно оновлюються дані та рейтинги, оскільки вони можуть змінюватися з часом. Регулярно аналізуються дані, щоб переконатися, що оцінка відповідає поточному стану справ.

Крок 5. Інформування користувачів. Результати рейтингового оцінювання надаються користувачам.

Це загальний підхід до створення рейтингового оцінювання діяльності вищого навчального закладу. Точні індикатори та їхні ваги будуть залежати від конкретного випадку та цілей оцінки.

Тобто, для кожного показника визначається максимальне значення, після чого показники x_{ij} нормалізуються, зокрема, шляхом їх ділення на максимальне кількісне значення j-го показника у вибірці:

$$\bar{x}_{ij} = \frac{x_{ij}}{\max x_{ij}} 100, \quad j = 1,...,n \quad , \quad (2.9)$$

де $\bar{x}_{ij}$ – нормалізовані окремі показники.

Розглянемо приклад оцінювання рейтингу вищого навчального закладу за допомогою вказаних показників. Використаємо наступні дані для кількох університетів:

Університет А:

Індекс якості науково-педагогічного персоналу (індекс1): 85

Індекс якості контингенту студентів (індекс2): 75

Індекс доступу (індекс3): 90

Індекс ресурсного забезпечення навчального процесу (індекс4): 80

Індекс міжнародної активності (індекс5): 70

Університет В:

Індекс якості науково-педагогічного персоналу (індекс1): 90

Індекс якості контингенту студентів (індекс2): 80

Індекс доступу (індекс3): 85

Індекс ресурсного забезпечення навчального процесу (індекс4): 70

Індекс міжнародної активності (індекс5): 75

Університет С:

Індекс якості науково-педагогічного персоналу (індекс1): 80

Індекс якості контингенту студентів (індекс2): 70

Індекс доступу (індекс3): 75

Індекс ресурсного забезпечення навчального процесу (індекс4): 90

Індекс міжнародної активності (індекс5): 85

Розглянемо два можливих підходи для побудови рейтингу.

Зважена сума індексів. У цьому підході кожен індекс має певний ваговий коефіцієнт, який визначається важливістю цього показника. Наприклад, доцільно вважати, що індекс якості науково-педагогічного персоналу і індекс ресурсного забезпечення навчального процесу є найважливішими. Вагові коефіцієнти для інших індексів можуть бути меншими. Потім рейтинг розраховується за допомогою наступної формули:

Рейтинг = (ваговий коефіцієнт$_1$ * індекс$_1$ + ваговий коефіцієнт$_2$ * індекс$_2$ + ... + ваговий коефіцієнт$_n$ * індекс$_n$), (2.10)

де n - кількість індексів, а вагові коефіцієнти визначаються з використанням інформації з сайтів ЗВО.

У підході з використанням середніх геометричних індексів рейтинг визначається як середнє геометричне значення всіх індексів. Формула виглядає наступним чином:

$$\text{Рейтинг} = \sqrt{\text{індекс}_1 * \text{індекс}_2 * \ldots * \text{індек}_n} \quad (2.11)$$

Цей підхід враховує, як всі індекси взаємодіють між собою, і рейтинг буде вищим, якщо всі показники мають високі значення.

Обираючи один з цих підходів можна розрахувати рейтинг кожного університету на основі їхніх індексів та вагових коефіцієнтів і визначити, який з них має найвищий рейтинг в порівнянні з іншими університетами.

Зважене сумування індексів допоможе розрахувати рейтинг для кожного університету. Використаємо вагові коефіцієнти для індексів і розрахуємо рейтинги для вказаних університетів.

Нехай вагові коефіцієнти для індексів будуть такими:

Ваговий коефіцієнт для індексу якості науково-педагогічного персоналу (індекс1): 0.3

Ваговий коефіцієнт для індексу якості контингенту студентів (індекс2): 0.2

Ваговий коефіцієнт для індексу доступу (індекс3): 0.15

Ваговий коефіцієнт для індексу ресурсного забезпечення навчального процесу (індекс4): 0.2

Ваговий коефіцієнт для індексу міжнародної активності (індекс5): 0.15

Тепер ми можемо розрахувати рейтинг для кожного університету:

Університет А: Рейтинг = (0.3 * 85) + (0.2 * 75) + (0.15 * 90) + (0.2 * 80) + (0.15 * 70) = 25.5 + 15 + 13.5 + 16 + 10.5 = 80.5 (2.12)

Університет В: Рейтинг = (0.3 * 90) + (0.2 * 80) + (0.15 * 85) + (0.2 * 70) + (0.15 * 75) = 27 + 16 + 12.75 + 14 + 11.25 = 81 (2.13)

Університет С: Рейтинг = (0.3 * 80) + (0.2 * 70) + (0.15 * 75) + (0.2 * 90) + (0.15 * 85) = 24 + 14 + 11.25 + 18 + 12.75 = 80 (2.14)

За даними розрахунками, Університет В має найвищий рейтинг серед цих трьох університетів і оцінюється на 81 бал.

Цей приклад ілюструє зважене сумування індексів для розрахунку рейтингу вищого навчального закладу на основі заданих вагових коефіцієнтів. Вагові коефіцієнти визначаються залежно від важливості кожного індексу у вас.

Для цього доцільно сформувати багаторівневу ієрархічну структуру критеріїв, що містить на верхньому рівні інтегрований показник рейтингового оцінювання якості освітніх послуг, на наступних – часткові критерії.

Послуговуючись методом рейтингового оцінювання діяльності навчальних закладів можемо рекомендувати абітурієнтам певний навчальний заклад для здобуття освіти.

2.6. Адаптація навчального контенту під вимоги IT ринку

Кожна людина генерує унікальні особисті дані, аналізуючи їх, можна оцінити чи правильно люди використовують свій природній потенціал в різних життєвих ситуаціях, зокрема при обранні навчального закладу для здобуття освіти після закінчення середньої школи. Стрімко зростає потреба в освічених, висококваліфікованих людях, здатних до проектування, розроблення, розгортання інфраструктур, платформ і застосунків в “розумних містах”. Повинні бути запропоновані і створені такі умови для навчання та здобуття практичного досвіду, щоб формувалися кваліфіковані трудові ресурси для IT галузей цих міст.

Використання алгоритмів машинного навчання для виявлення залежностей, прогнозування попиту на IT освіту, використання індивідуальних навчальних траєкторій сприяють створенню персоналізованих систем навчання [76].

Для цього доцільно використовувати алгоритми адаптації навчального матеріалу до індивідуальних потреб кожного студента, що своєю чергою сприятиме формуванню у них унікальних компетентностей, які використовуватимуться для входження в команди реалізації IT проектів.

Процес адаптації навчального матеріалу до індивідуальних потреб студента та потреб IT галузі у фахівцях з певними компетентностями може ґрунтуватися на

алгоритмах машинного навчання для побудови персоналізованого шляху навчання. Подамо загальний опис концепцій, які можна використовувати для формування контенту, що задовольнятиме зазначеним умовам:

Студенти, аналізуючи вакансії IT ринку визначають компетенності, які необхідно здобути, щоб потрапити в команду IT проєкту. Позначимо це як інтереси студента - I_s. Використовуємо алгоритми аналізу дій студента, його переглядів матеріалів, оцінок і т.д., щоб оцінити I_s. Для кожної концепції контенту позначимо важливість як I_c. Алгоритм може враховувати статистику успішності студентів з врахуванням наповненості контенту, популярності тем і т.д. Проводимо оцінку відповідності між інтересами студентів та наповненням контенту навчальних курсів. Для кожної пари (I_s,I_c) можемо обчислити бали відповідності або подібності. Позначимо бали як $M(I_s,I_c)$. Можемо розрахувати бали для створення персоналізованого списку рекомендацій для студента. Результати розрахунків можуть бути оновлені з часом на основі нових даних про студента та потреби IT галузі у фахівцях з певними компетентностями.

Оцінка інтересів студента (User Interest) може бути проведена методами машинного навчання, зокрема з використанням алгоритму колаборативного фільтрування для оцінки інтересів студента, яке передбачає збір даних про дії студента, такі як перегляди сайтів IT компаній. Далі створюється матриця, де рядки представляють користувачів, а стовпці - предмети (наприклад, теми чи концепції навчального матеріалу). Можемо визначити схожість між інтересами користувачів або предметами на основі їхньої взаємодії. Для розрахунку схожості між інтересами користувачами та предметами на основі їхньої взаємодії використаємо алгоритм колаборативного фільтрування методом найближчих сусідів (k-NN) на прикладі гіпотетичних даних. Наприклад, маємо матрицю взаємодії користувачів із предметами:

Item1 Item2 Item3 Item4

User1 5 3 - 4

User2 - 4 4 3

User3 4 - 5 2

User4 2 4 3 5

У цьому прикладі "5" представляє взаємодію інтересів користувача із предметом, "- "означає, що немає взаємодії. Проведемо розрахунок схожості за допомогою k-NN, використавши косинусну схожість між векторами оцінок користувачів. Косинусна схожість між двома векторами A та B може бути обчислена за допомогою наступної формули:

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} AiBi}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2.15)$$

де A·B представляє скалярний добуток векторів A та B.

‖A‖ та ‖B‖ представляють їхні евклідові норми (довжини).

Для кожної пари інтересів користувачів та концепції контенту обчислюємо косинусну схожість User1 та Content 2:

Косинусна схожість = (54 + 34) / (sqrt(5^2 + 3^2) * sqrt(4^2 + 4^2)) ≈ 0.93

Повторюємо для всіх пар користувачів та контенту. Визначаємо k-сусідів. Вибираємо k (наприклад, k = 2) найбільш схожого контенту для кожного користувача.

Результат:

Similarity(User1, Content 1) ≈ 0.93

Similarity(User1, Content 2) ≈ 0.79

Similarity(User1, Content 3) ≈ 0.71

Це є загальним підходом для розрахунку схожості між інтересами користувачів або предметами. Результати можна використовувати для прогнозування наповнення навчальних предметів у відповідності до інтересів користувачів. Вибір конкретних методів буде залежати від конкретних завдань та характеристик проекту.

Таким чином, формування навчального контенту з врахуванням компетентностей, затребуваних на ІТ ринку має далекоглядні цілі, спрямовані на підготовку фахівців, яких айчари ІТ фірм будуть відбирати до команд для реалізації проєктів.

Висновки до 2 розділу

У другому розділі запропоновано модель ІТ галузі розумного міста. Проаналізовано результати проходження ЗНО випускниками шкіл Тернополя, як потенціал для формування команд ІТ проєктів в майбутньому.

Проведено порівняльний аналіз результатів ЗНО випускниками шкіл Тернополя з профільних для спеціальностей ІТ галузі предметів та рейтингування шкіл Тернополя за кількістю успішних випускників.

Розроблено методи комплексного аналізу нахилів абітурієнтів до ІТ профілю.

Запропоновано підхід до моделювання мотиваційних чинників для абітурієнтів з використанням когнітивних карт.

Розроблено методи обчислення рейтингової оцінки діяльності вищого навчального закладу та проведення розрахунку відповідності навчального контенту та опису вимог до вакантних посад в ІТ галузі з використанням косинусної схожості.

РОЗДІЛ 3
ФОРМУВАННЯ СТРУКТУР КОМАНД ВИКОНАВЦІВ ІТ ПРОЕКТУ

3.1 Цілевизначальний підхід до формування структури команди ІТ проекту

Формування структури команди виконавців ІТ проекту є важливим етапом, який визначає успіх проекту. Для формування самої команди виділяють чотири основні підходи: цілевизначальний, рольовий, проблемно-орієнтований, міжособистісний. Розглянемо кожен з них детальніше.

Цілевизначальний підхід – грунтується на цілях ІТ проєкту, які можуть бути сформульовані як стратегічна чи оперативна ціль, що дає змогу гнучко орієнтуватися при виборі групової мети. Цілевизначальний підхід до формування команди орієнтований на досягнення конкретних цілей або завдань. У цьому підході команда створюється з метою досягнення поставленої мети, а всі рішення щодо складу команди, ролей та відповідальностей, а також стратегії співпраці, спрямовані на виконання цієї мети. Подамо цілевизначальний підхід у вигляді кортежу

$$ЦВП=\{C, V, R, G, E, O\} \quad (3.1)$$

де С- чітка ціль, оскільки основна ідея цілевизначального підходу полягає в тому, що команда формується для досягнення конкретної цілі чи завдання. Ця ціль повинна бути чітко визначеною та орієнтованою на конкретний результат.

V - обгрунтований склад команди, у якому мають бути представники з різними компетентностями та навичками, які необхідні для досягнення цілі. Кожен член команди повинен вносити свій внесок в спільну мету.

R - розподіл ролей та відповідальності, які визначаються на основі потреб проекту чи його завдань. Члени команди можуть призначатися на різні ролі в залежності від їхніх компетентностей та здібностей.

G – гнучкість щодо формування команд, яка може бути сформована на певний період часу для вирішення конкретного завдання, після чого може

розформовуватися. Гнучкість і адаптивність важливі для забезпечення ефективності роботи та швидкого реагування на зміни.

Е- зосередженість усіх зусиль команди на досягнення поставленої цілі. Це допомагає зосередитися на важливих завданнях та уникнути відволікань від другорядних.

О - постійне оцінювання результатів командою, регулярне оцінювання досягнутого прогресу та отриманих результатів, співвідносячи їх з поставленою цілю. На основі цього можуть вноситися зміни в стратегію чи дії команди.

Побудуємо модель взаємозв'язків між цими елементами кортежу.

Модель взаємозв'язку обґрунтованості складу команди та визначення ефективності кожного члена команди подамо наступним чином:

$$E_i = f(V), \quad (3.2)$$

де E_i, обгрунтованість складу команди, а *i* - індекс члена команди.

V - об'єктивність відбору членів команди.

Чітка ціль С грунтується на формулюванні SMART-цілей, які характеризуються S - конкретністю, M - вимірюваністю, A - досяжністю, R - реалістичністю, T - часовими обмеженнями.

$$C = g(S,M,A,R,T). \quad (3.3)$$

Розподіл ролей та відповідальності потребує визначення обсягу робіт для кожної ролі подамо наступним чином:

$$W_i = h(R_i), \quad (3.4)$$

де W_i, обсяг робіт, а *i* - індекс ролі.

R_i - розподіл відповідальності, а *i* - індекс ролі.

Гнучкість щодо формування команд визначає кількість можливих комбінацій формування команди:

$$G = C_n^k, \quad (3.5)$$

де C_n^k - кількість можливих комбінацій формування команди, n- кількість членів команди, k - кількість місць у команді.

Зосередженість усіх зусиль команди на досягнення поставленої цілі подамо наступною формулою;

$$E_{ком} = \sum_{i=1}^{n} E_i \,, \qquad (3.6)$$

де $E_{ком}$ - сумарна ефективність команди, E_i -зусилля кожного *i* члена команди.

Постійне оцінювання результатів:

$$O_{рез} = \frac{dR}{dt}, \qquad (3.7)$$

де R(t), похідна за часом R, тобто зміна результатів з часом, а t - час.
Ця формула вказує на те, як змінюються результати (R) відносно часу (t), що може вказати на тенденції та ефективність діяльності команди.

Ці вирази відображають основні аспекти цілевизначального підходу та можуть бути доповнені або змінені залежно від конкретного контексту та вимог певного проекту.

Цілевизначальний підхід допомагає максимально зосередити зусилля команди на досягненні конкретної мети та забезпечити високу ефективність в вирішенні конкретних завдань проєкту. Цілевизначальний підхід до формування команди є важливою складовою успіху для реалізації проєкту. Цей підхід передбачає створення команди, спрямованої на досягнення конкретних цілей та завдань. Сформулюємо метод для формування цілевизначальної команди:

Крок 1.Визначення мети і завдання. Чітко визначаються мета та завдання, які команда має виконати. Це обумовлює навички, знання та досвід потрібні для досягнення цілей.

Крок 2. Відбір учасників. Відбір членів команди відбувається на основі їхніх навичок, досвіду та здатностей, які сприяють досягненню цілей. Розглядається також комунікативна здатність та спроможність працювати у команді.

Крок 3. Розподіл завдань. Ретельно розподіляються завдання серед членів команди на основі їхніх сильних сторін та здатностей. Кожен член команди має мати чіткі обов'язки та відповідальність.

Крок 4. Спрямованість дій. Важливо постійно спрямовувати дії команди на досягнення цілей та виконання завдань. Це допомагає утримувати мотивацію та фокус на досягненні мети.

Крок 5. Співпраця та комунікація. Забезпечується ефективна комунікацію між членами команди. Налагоджується співпрацю та взаємодія, щоб досягти кращих результатів.

Крок 6. Мотивація та підтримка. Підтримується ініціатива членів команди, надаючи їм необхідну мотивацію і сприяння в усіх етапах роботи над проектом.

Крок 7. Постійний аналіз та корекція. Постійно оцінюється робота команди, і при необхідності, вносяться зміни у склад команди або стратегію роботи.

Крок 8. Розвиток навичок. Забезпечується можливість для навчання та розвитку членів команди. Це сприяє підвищенню ефективності та конкурентоспроможності команди в майбутньому.

Загалом, успішна цілевизначальна команда має бути добре організованою, мотивованою і спроможною працювати разом для досягнення конкретних результатів. Врахування цих кроків сприяє формуванню ефективної команди, яка досягне своїх цілей.

Концептуальну модель цілевизначального підходу до формування команди може виглядати наступним чином:

$$МЦП=OF+DV+C+UF+MO, \quad (3.8)$$

Де МЦП - модель цілевизначального підходу

OF - цільова функція, яка виражає ціль, яку команда має досягнути. Вона може бути виражена числовою метрикою, яку необхідно максимізувати або мінімізувати. Наприклад, це може бути прибуток, продуктивність, задоволеність клієнтів тощо.

DV - змінні рішення, це параметри, які можна змінювати для досягнення цілі. Наприклад, це може бути склад команди, розподіл завдань, навички членів команди тощо.

C - обмеження, що визначають умови, які повинні бути виконані під час формування команди. Наприклад, обмеження можуть включати бюджет, наявність ресурсів, навички, досвід тощо.

UF - функція корисності, яка використовується для визначення важливості різних параметрів та вибору найкращого рішення. Функція корисності оцінює, наскільки задоволеність обраного рішення відповідає меті.

МО - методи оптимізації, які використовуються для досягнення мети, серед яких лінійне програмування, генетичні алгоритми, динамічне програмування тощо.

Розглянемо детальніше характеристику кожної складової. Цільова функція в моделі цілевизначального підходу до формування команди визначає, яким чином вимірюється досягнення мети або цілі. Форма цільової функції залежить від конкретного завдання та метрик, які використовуються для оцінки успішності команди. Зазвичай цільова функція є числовою функцією, і її форма може бути дуже різною. Залежно від конкретного завдання та метрик, які використовуються для оцінки успішності команди, формула цільової функції може мати різну структуру і її можна подати так:

Розрахунок змінних рішень в цілевизначальному підході до формування команди може бути складним завданням і зазвичай вимагає використання математичних методів та алгоритмів оптимізації для досягнення оптимальних результатів. Зважаючи на складність обчислення цільової функції та обмежень у контексті формування команди, а також на об'єм роботи, який потрібно виконати для розрахунку оптимальних команд, розглянемо приклад спрощеного завдання оптимізації, де цільова функція та обмеження не будуть включати складних параметрів.

Припустимо, є команда з 5 членів, і необхідно визначити оптимальний розподіл ролей для максимізації загальної продуктивності команди. Кожен член команди може обрати одну з двох ролей: "Розробник" (Developer) або "Тестувальник" (Tester).

Цільова функція для максимізації загальної продуктивності може бути вираженою через ваги ролей, які впливають на продуктивність команди. Наприклад, якщо ви визначили ваги наступним чином:

Вага "Розробник" = 3

Вага "Тестувальник" = 2

Тоді цільова функція може бути виражена як:

Цільова функція = (3 * Кількість розробників) + (2 * Кількість тестувальників), (3.9)

де "Кількість розробників" та "Кількість тестувальників" - це змінні рішення, які можна оптимізувати.

Обмеження можуть включати обмеження на загальну кількість членів команди (наприклад, 5), обмеження на кількість розробників та тестувальників (наприклад, кожен тип не може перевищувати 3 члени команди). За допомогою алгоритмів оптимізації, таких як лінійне програмування та генетичних алгоритмів, можна знайти оптимальний розподіл ролей, який максимізує цільову функцію і відповідає обмеженням. Розрахунок такого завдання оптимізації з використанням коду може виглядати наступним чином: *(додаток В, лістинг 3.1)*

Цей код використовує бібліотеку SciPy для лінійного програмування та знаходить оптимальний розподіл ролей, який максимізує цільову функцію (продуктивність) при врахуванні обмежень (кількість членів команди та обмеження на кількість розробників і тестувальників).

Результат оптимізації: Кількість розробників: 1.0; Кількість тестувальників: 1.0; Значення цільової функції (максимізована продуктивність): 5.0.

Зважаючи на складність обмежень у контексті формування команди, які можуть включати багато параметрів, наведемо спрощений приклад обмеження. Припустимо, формується команда для проекту розробки програмного забезпечення. Існують кількісні обмеження на кількість членів команди та кількість ролей. Наприклад, створюється команда з 10 членів, і є обмеження на кількість ролей, де:

Кількість розробників не може перевищувати 7 членів команди.

Кількість тестувальників не може перевищувати 5 членів команди.

Ці обмеження можуть бути виражені наступним чином:

Кількість розробників (D) повинна бути менше або рівно 7: $D \leq 7$

Кількість тестувальників (T) повинна бути менше або рівно 5: $T \leq 5$

Загальна кількість членів команди (N) повинна дорівнювати 10: $N = 10$

Такі обмеження враховують кількість членів команди, кількість розробників і тестувальників, що відповідають вимогам. Розв'язання такої задачі може бути здійснено за допомогою методів оптимізації, які б дозволили вибрати розподіл ролей, що враховує ці обмеження. Для визначення функції корисності розглянемо сценарій формування команди для проекту інноваційного розроблення нового продукту. У цьому сценарії ми маємо важливі параметри, які визначають важливість кожного члена команди та їхніх вмінь: технічний досвід, інноваційність, здатність до співпраці.

Функція корисності кожного члена команди може бути виражена у вигляді лінійної комбінації цих параметрів:

Функція корисності = α Технічний досвід + β Інноваційність + (3.10)

+ γ Здатність до співпраці,

де α, β, і γ - ваги, які визначають важливість кожного параметра. Якщо технічний досвід є найважливішим, то α може бути найбільшим позитивним числом, інші ваги можуть бути меншими значеннями або навіть нулями, якщо вони менш важливі. За допомогою такої функції корисності можна визначити вартість кожного члена команди на основі їхніх характеристик, і потім вибрати команду з максимальною загальною вартістю як оптимальний варіант для проекту. Розрахунок функції корисності можемо подати через код *(додаток В, лістинг 3.2).* У розглянутому прикладі оптимальний член команди: 2. Код розраховує вартість для кожного члена команди на основі їхніх характеристик, а потім вибирає члена команди з максимальною вартістю як оптимальний варіант для проекту.

Розглянемо приклад використання генетичного алгоритму для формування команди. Наприклад є команда з 20 потенційних членів, і необхідно вибрати оптимальну команду з 5 членів на основі їхньої ефективності (компетентностей). Проаналізуємо можливість максимізувати загальну якість роботи команди, обираючи 5 членів із загальної кількості 20 осіб. Генетичні алгоритми використовуються для оптимізації завдань шляхом емуляції еволюційних процесів. У нашому випадку, де потрібно максимізувати загальну якість роботи команди, можна сформулювати генетичний алгоритм наступним чином:

1. **Популяція (Population).** Початкова група індивідів, що представляють різні команди. Кожен індивід може бути представлений як послідовність ролей, де кожна роль відповідає конкретній особі в команді.
2. **Функція пристосованості (Fitness Function).** Оцінює якість кожної команди на основі певних метрик. Ця функція визначає, наскільки ефективно працює команда.
3. **Хромосоми (Chromosomes).** Представлені послідовністю ролей в команді.
4. **Селекція (Selection).** Вибір індивідів для наступного покоління на основі їхньої пристосованості. Індивіди з вищою якістю мають більше шансів потрапити в наступне покоління.
5. **Кросовер (Crossover).** Створення нових індивідів (команд) шляхом обміну частинами їхніх хромосом. Це може відбуватися на рівні ролей.
6. **Мутація (Mutation).** Можливість внесення випадкових змін в індивідів для розноманітності і уникнення застрягання в локальних максимумах.
7. **Запуск алгоритму (Algorithm Loop).** Повторення селекції, кросовера і мутації протягом декількох поколінь з метою збільшення загальної якості команд.
8. **Критерій зупинки (Stopping Criterion).** Умова зупинки алгоритму, наприклад, фіксована кількість поколінь або досягнення певного рівня пристосованості.

Формула генетичного алгоритму узагальнюється до цих етапів і може виглядати так:

1. **Ініціалізація популяції:**

   Population={Individual1,Individual2,…,Individual}Population={Individual1,Individual2,…,Individualn}

2. **Цикл генетичного алгоритму:**

- **Оцінка пристосованості:** Fitness(Individual)Fitness(Individuali)
- **Селекція:** Selection(Population)Selection(Population)

- **Кросовер і мутація:** Crossover(Selected Individuals) Crossover (Selected Individuals) Mutation(Offspring)Mutation(Offspring)

Наведемо приклад коду на мові Python, який демонструє цей процес: *(додаток В, лістинг 3.3).* Результат розрахунків:

Найкраща команда: [19, 20, 17, 18, 19]

Найкраща якість роботи команди: 93

У цьому прикладі генетичний алгоритм використовується для вибору найкращої команди з 20 можливих членів на основі якості їхньої роботи.

Отже, модель цілевизначального підходу включає в себе визначення цільової функції, змінних рішення, обмежень і метод оптимізації. Ця модель допомагає визначити оптимальний склад команди та спосіб її організації для досягнення поставленої мети.

3.2 Концептуальне подання інших структур проектних команд

Структура проєктної команди залежить від розміру проєкту, методу управління та особливості мети. Виокремлюють кілька видів структур проєктних команд, кожен з яких має свої плюси і мінуси:

Ізоморфна структура – команда рівних, легко контрольована, роботи ведуться паралельно, учасники команди здатні обмінюватися досвідом і ресурсами. Слабкими сторонами цієї структури є те, що така структура не може бути застосована (складно застосовна) в сильно інтегрованих проєктах. Ізоморфна структура команди проекту відображає концепцію ізоморфізму, де команда структурно відповідає структурі проекту або задачі. Тобто, структура команди відображає структуру та організацію проекту. Цей підхід допомагає забезпечити гармонійну взаємодію між різними частинами проекту та забезпечити більш ефективне виконання завдань.

Основні аспекти ізоморфної структури команди проекту: відповідність структурі проекту, спеціалізованість, синхронізація та координація, забезпечення цільової орієнтації, ефективне вирішення завдань, спрощення комунікації, адаптивність. Загалом, ізоморфна структура команди проекту допомагає

забезпечити збалансоване та ефективне виконання завдань у відповідності до структури та організації проекту. Модель ізоморфної структури команди подамо за допомогою теорії графів, де вузли представляють ролі або учасників команди, а ребра - зв'язки або взаємодії між ними. Для зручності будемо використовувати матрицю суміжності. Нехай G - граф, де V - множина вузлів (ролей чи учасників), а E - множина ребер (зв'язків чи взаємодій). Матриця суміжності A задається так, що A_{ij}=1, якщо існує зв'язок між вузлами i та j, і A_{ij}=0, якщо немає. Таким чином, якщо n - кількість вузлів у графі, то матриця суміжності A розмірності n×n буде мати вигляд:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \quad (3.11)$$

де a_{ij} - елемент матриці, який вказує на наявність (1) або відсутність (0) зв'язку між вузлами *i* та *j*.

Ця матриця визначає структуру команди та взаємодії між її членами.

Експертна структура команди– характеризується великою самостійністю команди, що розвантажує менеджера проєкту. Експертна структура команди - це тип командної організації, де кожен член команди є фахівцем у своїй області та має великий рівень компетенцій. Така команда складається з висококваліфікованих спеціалістів, кожний з яких відповідає за власну область експертизи. Експертна структура команди широко використовується в проектах, де важливий внесок з боку спеціалізованих фахівців. Основні особливості експертної структури команди: висока компетентність, спеціалізація, розподіл обов'язків, самостійність, взаємодія, підвищення якості, лідерство у своїй сфері, достовірність, виклики комунікації. Математичну модель експертної структури команди можна виразити, використовуючи набір параметрів та їх взаємодію.

Концептуальна модель може виглядати так:

Метрика команди=C·S·RO·I·Coop·Q·L·T·CC, (3.12)

де C - висока компетентність, S – спеціалізація, RO - розділення обов'язків, I – самостійність, Coop – взаємодія, Q - підвищення якості, L - лідерство у своїй сфері, T – достовірність, CC - виклики комунікації

Ця модель відображає, як кожен з параметрів впливає на загальну метрику команди. Значення кожного параметра може варіюватися від 0 до 1, де 1 вказує на високий рівень цього параметра, а 0 - на низький. Ця модель слугує загальним фреймворком, і його можна адаптувати відповідно до конкретних вимог формування команди. Експертна структура команди відмінно підходить для проектів, де важливі конкретні навички та спеціалізовані знання, а також коли головною метою є досягнення високої якості результатів завдяки залученню висококваліфікованих експертів. Недоліком є нечіткий розподіл відповідальності між членами команди, нерівномірність навантаження.

Колегіальна структура – характеризується сильною інтенсивністю інформаційного обміну між рольовими групами, згуртованістю, об'єднанням знань і зусиль. Колегіальна структура команди є підхідом, при якому всі члени команди мають рівні права та можливості в прийнятті рішень та управлінні проектом або завданням. В цьому підході наголошується на спільній природі роботи, співпраці та колективному прийнятті рішень. Колегіальна структура відрізняється від інших тим, що рішення приймаються колективно, а не за допомогою одного лідера або керівника. Основні особливості колегіальної структури команди: рівність членів команди, колективне прийняття рішень, співпраця та обмін ідеями, взаємодопомога, різноманітність поглядів, забезпечення лідерства, рівний доступ до інформації, розвиток навичок, відповідальність.

Концептуальна модель колегіальної структури команди може бути виражена через параметри, що визначають рівень різноманітності, співпраці, взаємодопомоги та інші елементи.

Концептуальна модель може бути представлена як добуток цих параметрів:

$$\text{Метрика команди} = E \cdot CD \cdot Coop \cdot Help \cdot D \cdot Lead \cdot IA \cdot Dev \cdot Resp, \quad (3.13)$$

де E - рівність членів команди, CD - колективне прийняття рішень, Coop - співпраця та обмін ідеями, Help – взаємодопомога, D - різноманітність поглядів, Lead - забезпечення лідерства, IA - рівний доступ до інформації, Dev - розвиток навичок, Resp – відповідальність. Ця модель відображає, як кожен з параметрів впливає на загальну метрику команди. Знову ж таки, значення кожного параметра

може варіюватися від 0 до 1, де 1 вказує на високий рівень цього параметра, а 0 - на низький. Ця модель є загальним представленням і може бути адаптована відповідно до конкретних вимог і контексту вашого дослідження.

Колегіальна структура команди може бути важливим інструментом для створення позитивної командної динаміки, забезпечення включеності та досягнення спільних цілей шляхом спільного прийняття рішень. З слабких сторін такої структури можна виділити відсутність лідерства, перевантаження інформацією членів команди. Недоліком є сильна залежність від керівника проєкту, висока ймовірність виникнення конфліктів та інформаційна перевантаженість.

У сфері IT проектів існує декілька типів команд, які можуть використовуватися для ефективного виконання різних завдань та проектів. Ось декілька видів команд IT проектів:

Розробницька команда зосереджена на розробці програмного забезпечення або інших технічних рішень. Вона включає програмістів, інженерів, тестерів та інших фахівців, які працюють разом для створення програм або продуктів. Розробницька команда є ключовим елементом в IT проектах, оскільки вона займається розробкою програмного забезпечення або інших технічних рішень. Ця команда об'єднує фахівців з різних областей для створення продукту, який відповідає вимогам та потребам замовника. Математична модель розробницької команди може включати різні елементи, враховуючи їхні ролі та взаємодії. Одна з можливих концептуальних моделей для розробницької команди може виглядати так:

$$\text{Розробницька команда} = P + T + A + D + BR + FR + QA + SA + SM, \quad (3.14)$$

де P - кількість програмістів, T - кількість тестерів, A - кількість архітекторів, D - кількість дизайнерів і UX-фахівців, BR - кількість бекенд-розробників, FR - кількість фронтенд-розробників, QA - кількість QA-фахівців, SA - кількість системних адміністраторів, SM - кількість Scrum-мастерів

Ця формула сумує кількість членів кожної групи, враховуючи їх прямопропорційну залежність. Важливо зауважити, що ця модель не враховує

взаємодії між різними ролями та інші аспекти. Модель може бути уточнена та розширена відповідно до конкретних вимог та характеристик проекту. Це лише деякі з ключових ролей в розробницькій команді. Склад та ролі можуть варіюватися в залежності від конкретних потреб проекту, важливо, щоб розробницька команда мала здатність співпрацювати, використовувати ефективні методи розробки та досягати спільних цілей.

Проектна команда формується для реалізації конкретного проекту. Вона включає представників різних фахових напрямків - розробки, тестування, дизайну, менеджменту тощо - які спільно працюють над досягненням цілей проекту. Проектна команда є групою фахівців, які об'єднуються для реалізації конкретного проекту. Ця команда включає різних спеціалістів з різних областей, які спільно працюють над досягненням цілей та завдань проекту. Основною метою проектної команди є виконання проекту в рамках встановленого обсягу, часу та бюджету. Математична модель проектної команди може бути подана як сума кількостей кожної групи:

$$\text{Проектна команда} = PM + BA + TA + QA + D + Dev + CM + IM + RM, \quad (3.15)$$

де PM - кількість проектних менеджерів, BA - кількість бізнес-аналітиків, TA - кількість технічних архітекторів, QA - кількість QA-інженерів, D - кількість дизайнерів, Dev - кількість розробників, CM - кількість комунікаційних менеджерів, IM - кількість спеціалістів з впровадження, RM - кількість ресурсних менеджерів.

Ця формула представляє собою суму кількостей членів кожної групи, подаючи прямопропорційну залежність від цих параметрів. Аналогічно до попередньої моделі, вона може бути уточнена та розширена в залежності від конкретних потреб та особливостей проекту. Ці ролі можуть варіюватися в залежності від типу та складності проекту. Проектна команда працює разом для забезпечення успішного виконання проекту та досягнення його цілей.

Аналітична команда займається дослідженням даних, розробкою аналітичних моделей та висновків на основі даних. Вона може включати аналітиків даних, науковців, статистиків та інших фахівців. Аналітична команда є групою

фахівців, які спеціалізуються на аналізі даних та вивченні інформації з метою отримання цінних висновків, рекомендацій та інсайтів. Ця команда грає важливу роль в процесі прийняття рішень, розробці стратегій та вдосконаленні бізнес-процесів.

Концептуальна модель аналітичної команди може бути подана як сума кількостей кожного елемента. Математична модель може виглядати так:

Аналітична команда=DA+BA+DA+WA+SMA+FA+MA+DA+RA,(3.16)

де DA - кількість дата-аналітиків, BA - кількість бізнес-аналітиків, DA - кількість аналітиків даних, WA - кількість веб-аналітиків, SMA - кількість аналітиків соціальних медіа, FA - кількість фінансових аналітиків, MA - кількість маркетингових аналітиків, DA - кількість дослідників даних, RA - кількість спеціалістів з аналізу ризиків.

Ця формула представляє собою суму кількостей членів кожної групи, об’єднаних прямопропорційною залежністю. Вона може бути уточнена або розширена в залежності від конкретних потреб та особливостей проекту.

Концептуальна модель *інтеграційної команди* може бути подана як сума кількостей кожного елемента.

Математична модель може виглядати так:

Інтеграційна команда=IM+TA+TI+SPI+CM+PM, (3.17)

де IM - кількість інтеграційних менеджерів, TA - кількість технічних архітекторів, TI - кількість тестувальників інтеграції, SPI - кількість спеціалістів з інтеграційними процесами, CM - кількість спеціалістів з управління конфігурацією, CM - кількість комунікаційних менеджерів, PM - кількість проектних менеджерів.

Ця формула також представляє собою суму кількостей членів кожної групи, об’єднаних прямопропорційною залежністю. Вона може бути уточнена або розширена в залежності від конкретних потреб та особливостей проекту. Інтеграційна команда грає важливу роль в забезпеченні взаємодії між різними компонентами чи системами, сприяючи досягненню цілей проекту або реалізації спільної стратегії.

Модель *інноваційної команди* може бути подана як сума кількостей кожного елемента.

Концептуальна модель може виглядати так:

Інноваційна команда=IM+IL+RT+ID+VI+MA+PM+IP, (3.18)

де M - кількість інноваційних менеджерів, IL - кількість ідеологічних лідерів, RT - кількість дослідників нових технологій, ID - кількість дизайнерів інновацій, VI - кількість спеціалістів з валідації інновацій, MA - кількість маркетингових аналітиків, PM - кількість проектних менеджерів, IP - кількість спеціалістів з патентознавства та захисту інтелектуальної власності.

Ця формула також представляє собою просту суму кількостей членів кожної групи, , об’єднаних прямопропорційною залежністю. Вона може бути уточнена або розширена в залежності від конкретних потреб та особливостей проекту. Інноваційна команда сприяє розвитку організації, забезпечуючи впровадження нових ідей та технологій, що сприяє підвищенню її конкурентоспроможності та здатності до адаптації до змін.

3.3 Рольовий підхід – заснований на переговорах, дискусіях членів команди, на яких приймається зважене рішення щодо ролей

Рольовий підхід до формування команди передбачає розподіл учасників команди для виконання певних ролей, кожна з яких має визначені функції, обов'язки та відповідальності. Цей підхід дозволяє забезпечити різноманітність компетентностей та навичок у команді, а також підвищити ефективність виконання завдань завдяки спеціалізації кожної ролі. Основні риси рольового підходу: формування комплекту ролей, розподіл обов'язків, вузька спеціалізація, взаємодія ролей, різноманітність навичок, наявність лідера, фокусування на конкретному аспекті роботи, розвиток навичок. Рольовий підхід дозволяє ефективно використовувати різнорідність навичок та компетентностей у команді, забезпечувати спеціалізацію та досягнення спільної мети.

Концертуальну модель рольового підходу до формування команди можна виразити через використання додаткових параметрів або змінних, які

представляють ролі, які члени команди повинні виконувати. Ось загальний приклад, як це може бути виражено:

Змінні рішення. Введемо змінні, які представляють вибір ролей для кожного члена команди. Наприклад, якщо у вас є команда з N членів, ви можете мати N змінних рішення:

$$DV=x_1, x_2, ..., x_N, \quad (3.19)$$

де DV - змінні рішення;

x_i - вибір ролі для члена команди під номером і. Роль може бути представлена числовим індексом або ідентифікатором ролі.

Для розрахунків за сценарієм, де вводяться змінні рішення для вибору ролей для кожного члена команди, можемо використовувати лінійне програмування. Розглянемо приклад з командою з 6 членів та 3 ролями: "Розробник," "Тестувальник," і "Дизайнер." Кожен член команди повинен вибрати одну з цих ролей.

Ми використовуватимемо бібліотеку PuLP для розв'язання цієї задачі лінійного програмування. Приклад коду для розрахукнів у *додатку В, лістинг 3.4.*

Статус розв'язку: 1

Член 1 вибрав роль: Тестувальник

Член 2 вибрав роль: Тестувальник

Член 3 вибрав роль: Тестувальник

Член 4 вибрав роль: Тестувальник

Член 5 вибрав роль: Тестувальник

Член 6 вибрав роль: Тестувальник

Таким чином, визначили задачу лінійного програмування для вибору ролей для кожного члена команди з обов'язковим обранням однієї ролі для кожного члена. Задача максимізує функцію корисності на основі виборів ролей, і ми отримуємо оптимальний розподіл ролей в команді.

Обмеження. Введемо обмеження, які визначають, які ролі доступні для вибору для кожного члена команди та які обмеження встановлюються на кількість

членів команди для кожної ролі. Наприклад, для ролі "Розробник" може бути обмеження на кількість розробників:

$$\Sigma x_i \leq C_{(\text{Кількість розробників})}, \quad (3.20)$$

де Σx_i - сума всіх обраних ролей, а $C_{(\text{Кількість розробників})}$ - обмеження на кількість розробників у команді.

Для моделювання сценарію з обмеженнями на кількість членів команди для кожної ролі, використаємо лінійне програмування. Розглянемо ситуацію, де команда має обмеження на кількість розробників. Припустимо маємо команду з 6 членів, і кожен член команди може вибрати одну з трьох ролей: "Розробник", "Тестувальник" або "Дизайнер". Обмеження для ролі "Розробник" полягає в тому, що кількість розробників повинна бути не більше 3.

Лінійне програмування допоможе нам максимізувати функцію корисності (наприклад, загальний вплив команди), дотримуючись обмежень на кількість розробників. Код на Python, який використовує бібліотеку PuLP для вирішення цієї задачі подано у додатку *(додаток В, лістинг 3.5).* У цьому прикладі ми використовуємо бібліотеку PuLP для визначення оптимального розподілу ролей в команді з метою максимізації функції.

Статус розв'язку: 1

Член 1 вибрав роль: Тестувальник

Член 1 вибрав роль: Дизайнер

Член 2 вибрав роль: Тестувальник

Член 2 вибрав роль: Дизайнер

Член 3 вибрав роль: Тестувальник

Член 3 вибрав роль: Дизайнер

Член 4 вибрав роль: Розробник

Член 4 вибрав роль: Тестувальник

Член 4 вибрав роль: Дизайнер

Член 5 вибрав роль: Розробник

Член 5 вибрав роль: Тестувальник

Член 5 вибрав роль: Дизайнер

Член 6 вибрав роль: Розробник

Член 6 вибрав роль: Тестувальник

Член 6 вибрав роль: Дизайнер

Цільова функція в цьому випадку може виражати досягнення мети з урахуванням ролей та їхнього внеску у результат команди. Наприклад, для максимізації продуктивності команди, цільова функція може виглядати як:

$$OF = \Sigma (x_i * \text{Внесок ролі у продуктивність}), \quad (3.21)$$

де OF- цільова функція;

x_i - вибір ролі, а "Внесок ролі у продуктивність" - числовий показник, що відображає, наскільки вибір певної ролі впливає на продуктивність команди.

Для розрахунків за цим сценарієм з цільовою функцією, яка враховує вибір ролей та їхній внесок у результат команди, розглянемо приклад з командою з 6 членів, де кожен член може вибрати одну з трьох ролей: "Розробник," "Тестувальник," або "Дизайнер." Ми також враховуватимемо числовий показник "Внесок ролі у продуктивність" для кожної ролі.

Приклад коду з використанням бібліотеки PuLP для вирішення цієї задачі лінійного програмування у додатку *(додаток В, лістинг 3.6)*.

Статус розв'язку: 1

Член 1 вибрав роль: Розробник, Внесок у продуктивність: 0.8

Член 2 вибрав роль: Розробник, Внесок у продуктивність: 0.8

Член 3 вибрав роль: Розробник, Внесок у продуктивність: 0.8

Член 4 вибрав роль: Розробник, Внесок у продуктивність: 0.8

Член 5 вибрав роль: Розробник, Внесок у продуктивність: 0.8

Член 6 вибрав роль: Розробник, Внесок у продуктивність: 0.8

У цьому прикладі ми розглянули задачу лінійного програмування для вибору ролей для кожного члена команди з метою максимізації продуктивності команди. Цільова функція враховує вибір ролей та їхній внесок у продуктивність. Ми отримуємо оптимальний розподіл ролей в команді з урахуванням цього критерію.

Отже, через використання змінних рішення, обмежень та цільової функції можна математично виразити рольовий підхід до формування команди, де вибір

ролей і їх вплив на результат можуть бути оптимізовані для досягнення бажаної мети проекту.

Розглянемо приклад проведення розрахунків рольового підходу до формування команди для проекту розробки програмного продукту з метою - максимізувати продуктивність команди, враховуючи ролі та їхній внесок в результат проекту. Припустимо, у нас є команда розробників, тестувальників і менеджерів, і необхідно визначити оптимальний розподіл ролей між ними для максимізації продуктивності, знаючи, які внески в прирості продуктивності вносять різні ролі:

Розробники мають коефіцієнт продуктивності 1.2.

Тестувальники мають коефіцієнт продуктивності 1.1.

Менеджери мають коефіцієнт продуктивності 1.05.

Враховуємо такі обмеження:

Команда повинна складатися з 10 осіб (N = 10).

Максимум 7 осіб можуть бути розробниками (Rozr = 7).

Максимум 3 особи можуть бути тестувальниками (Test = 3).

Якщо необхідно максимізувати загальну продуктивність команди, враховуємо суму продуктивність кожної ролі:

OF = Σ (xi * Внесок ролі у продуктивність)

де OF – цільова функція;

x_i - кількість членів команди, які вибирають певну роль, і "Внесок ролі у продуктивність" - коефіцієнт продуктивності цієї ролі.

Виконаємо розрахунки. Почнемо з визначення оптимального розподілу ролей:

Оптимальний розподіл ролей може бути приблизно таким:

Розробники (Rozr): 7 осіб

Тестувальники (Test): 3 особи

Менеджери (Managers): 0 осіб (оскільки вони мають менший коефіцієнт продуктивності)

Тепер обчислимо цільову функцію, використовуючи цей розподіл:

$$P = (7 * 1.2) + (3 * 1.1) + (0 * 1.05) = 8.4 + 3.3 + 0 = 11.7 \quad (3.22)$$

Отже, цільова функція для такого розподілу ролей становить 11.7, що представляє загальну очікувану продуктивність команди. Це приклад обчислення ефективності команди за рольовим підходом, де оптимальний розподіл ролей визначається для досягнення максимальної продуктивності команди.

Використаємо нейронні мережі для прогнозування оптимального складу та розподілу ролей у команді для нових проєктів. Нейронні мережі використовуються для задач прогнозування та класифікації, включаючи оптимальний склад та розподіл ролей у команді. Однак точна формула нейронної мережі може значно варіюватися в залежності від конкретної задачі та архітектури мережі. Основною ідеєю є те, що нейронна мережа отримує вхідні дані (наприклад, характеристики команди, навички учасників, інші фактори) і виводить прогнозовані значення (наприклад, оптимальний розподіл ролей). Важливо підібрати архітектуру мережі, кількість шарів, кількість нейронів у кожному шарі та інші параметри для конкретного завдання.

Спрощена формула може виглядати приблизно так:

Вхідний шар:

Кожен нейрон в цьому шарі представляє один вхідний ознаку (наприклад, характеристику учасника команди).

Приховані шари:

Містять нейрони, які вивчають складні взаємозв'язки між вхідними ознаками.

Застосовують функції активації для вирішення нелінійних проблем.

Вихідний шар:

Кожен нейрон у цьому шарі представляє вихідну змінну (наприклад, ймовірність кожної ролі).

Формула може бути узагальнена як:

$$y=f(W_2 \cdot f(W_1 \cdot X+b_1)+b_2) \quad (3.23)$$

де X - вектор вхідних ознак (властивостей команди), W_1 та b_1 - ваги та зсуви для першого шару, f - функція активації, W_2 та b_2 - ваги та зсуви для вихідного шару, y - вихідні значення (наприклад, ймовірності ролей).

Це базовий приклад, і для конкретних задач можуть використовуватися різні архітектури та функції активації. Оптимальну формулу слід визначити експериментальним шляхом, шляхом налаштування параметрів мережі та вибору відповідної архітектури. У додатку подано приклад простої нейронної мережі з використанням бібліотеки Python, яка може використовуватися для цього типу задач *(додаток В, лістинг 3.7).* У цьому прикладі використовується штучна нейронна мережа з одним прихованим шаром. Модель навчається на даних з параметрами проєктів та розподілом ролей. Після тренування вона може використовуватися для прогнозування оптимального розподілу ролей для нового проєкту. Для розв'язання цієї задачі можна використовувати метод оптимізації. Основна ідея полягає в тому, щоб створити модель, яка оцінює внесок кожної ролі в продуктивність команди та рекомендує оптимальний розподіл ролей для максимізації продуктивності. Однією з можливих моделей для задачі оптимізації є лінійна регресія. Лінійна регресія широко використовується для прогнозування значень залежної змінної на основі лінійної комбінації незалежних змінних.

Припустимо, що у нас є команда розробників, тестувальників і менеджерів, і ми хочемо оптимізувати їхні ролі для максимізації продуктивності. Ми можемо використовувати лінійну регресію для прогнозування продуктивності команди на основі характеристик кожної ролі. Приклад коду для використання лінійної регресії мовою Python та з бібліотекою scikit-learn подано у додатку *(дивись додаток В, лістинг 3.8).* У цьому прикладі ми використовуємо лінійну регресію для побудови моделі, яка прогнозує продуктивність команди на основі характеристик кожної ролі. Потім ми оцінюємо точність моделі за допомогою коефіцієнта детермінації (R-squared). В реальних умовах важливо враховувати різноманітність факторів та наявність достатньої кількості даних для навчання моделі.

3.4 Проблемно-орієнтований підхід

Проблемно-орієнтований підхід – ґрунтується на досягненні бажаної мети через вирішення виявленої проблеми або низки проблем. Проблемно-орієнтований підхід до формування команди грунтується на вирішенні конкретної проблеми або

завдання, що вимагає творчого підходу та співпраці всіх членів команди. Основною ідеєю цього підходу є об'єднання фахівців із різних областей для пошуку інноваційних та творчих рішень для складних проблем. Основні складові проблемно-орієнтованого підходу: постановка задачі, Творчий підхід, різноманітність навичок, колективна співпраця, інтердисциплінарність, постійна оцінка результатів, розвиток критичного мислення, стимулювання інновацій. Проблемно-орієнтований підхід сприяє вирішенню складних та невизначених проблем, коли необхідно спільно працювати над знаходженням творчих та оригінальних рішень.

Розрахунки можна провести лише для складової, яка сприяє команді постійно аналізувати та оцінювати прогрес, коригувати свої дії та стратегію, використовуючи ітераційний процес. Розглянемо приклад використання ітераційного аналізу для розв'язання проблеми командою.

Припустимо, є команда розробників, яка працює над створенням програмного продукту. Необхідно максимізувати продуктивність команди та якість продукту. Команда працює над проектом із багатьма задачами та функціональністю. Продемонструємо приклад коду на Python, який ілюструє цей процес *(додаток В, лістинг 3.9)*

Iteration 1: Team Productivity: 0.8, Quality: 0.9,

Iteration 2: Team Productivity: 0.8, Quality: 0.9

Iteration 3: Team Productivity: 0.8, Quality: 0.9

Iteration 4: Team Productivity: 0.8, Quality: 0.9

Iteration 5: Team Productivity: 0.8, Quality: 0.9

Final Team Productivity: 0.8, Final Quality: 0.9

У цьому прикладі команда працює над проектом упродовж 5 ітерацій, моніторить свій прогрес, аналізує результати та, якщо необхідно, коригує стратегію для поліпшення продуктивності та якості. Скажімо при наявності списку кандидатів та вимог до ролей, можна створити матрицю, в якій вказувати, наскільки кожен кандидат підходить для кожної ролі на основі оцінки

відповідності. Для створення матриці, що відображає відповідність кандидатів до ролей на основі оцінок, спершу потрібно визначити, які оцінки відповідності можуть бути використані. Наприклад, оцінки можуть бути в діапазоні від 0 до 1, де 0 вказує на жодну відповідність, а 1 - на повну відповідність. Визначимо деякі умовні оцінки для кожного кандидата для трьох ролей: "Розробник", "Тестувальник" і "Аналітик". Приклад матриці оцінок відповідності для кожного кандидата до ролей *(додаток В, лістинг 3.10):*

Матриця оцінок відповідності:

[[0.9 0.6 0.7]

[0.6 0.8 0.6]

[0.7 0.7 0.9]

[0.8 0.5 0.8]]

У цьому прикладі ми створили матрицю, де кожний рядок відповідає кандидатові, а кожний стовпчик відповідає ролі. Значення в кожній комірці матриці показують оцінку відповідності кандидата до ролі. Наприклад, "0.9" в першому рядку та першому стовпчику означає, що перший кандидат має високу відповідність до ролі "Розробник". Ця матриця може бути використана для прийняття рішень щодо призначення ролей кандидатам на основі їхньої відповідності. Цей підхід більше фокусується на кваліативному аналізі та врахуванні специфічних потреб проекту чи завдання. Зважаючи на те, що кваліативний аналіз базується на якісних аспектах та оцінках, а не на кількісних даних, можемо розглянути приклад використання кваліативного аналізу для формування команди проєкту із впровадження нової технології. Компанія вирішила впровадити нову технологію в своїй діяльності і потребує формування команди для успішної реалізації цього проекту.

Алгоритм кваліативного аналізу містить наступні кроки:

Крок 1. Аналіз вимог та цілей проекту. Важливо спершу зрозуміти обсяг та цілі проекту. Наприклад, впровадження нової технології може потребувати експертів у різних областях, включаючи інженерів, розробників, аналітиків, менеджерів проектів тощо.

Крок 2. Визначення ключових ролей і функцій: Визначення ключових ролей і функцій, які потрібно включити до команди. Наприклад, для проекту впровадження технології може бути важливою роллю "технічний лідер," "проектний менеджер," "експерт з користувачами" і т. д.

Крок 3. Оцінка навичок і досвіду кандидатів: Вибір кандидатів для кожної ролі здійснюється на основі їхнього досвіду, навичок, знань та відповідності вимогам ролі.

Крок 4. Оцінка комунікаційних навичок і командного досвіду: Оцінка, наскільки кандидати можуть ефективно співпрацювати в команді та володіють комунікаційними навичками, що є важливими для успіху проекту.

Крок 5. Оцінка взаємодії в команді: Аналіз взаємодії між кандидатами та їхніх здатностей співпрацювати в команді, а також їхньої взаємодії зі сторонніми структурами в організації. Оцінка відповідності цілям проекту та корпоративній культурі: Переконання, що кандидати дійсно відповідають цілям проекту та вписуються в корпоративну культуру компанії.

Крок 6. Формування команди: На основі аналізу кваліфікацій і аспектів вищезазначених кроків формується команда, яка відповідає потребам проекту та має високий потенціал для успішності.

Цей процес ґрунтується на якісних аналітичних підходах та досвіді. Основна мета - вибрати команду, яка може спільно досягти цілей проекту, враховуючи якісні параметри та вимоги, які не завжди можна виразити числами.

3.5 Міжособистісний підхід – сфокусований на поліпшенні міжособистісних відносин усередині команди

Міжособистісний підхід до формування команди фокусується на взаємодії та відносинах між учасниками команди. Основні риси міжособистісного підходу: розвиток командних відносин, довіра та взаєморозуміння, ефективне спілкування, розуміння різниць, емоційний інтелект, розвиток лідерства, кооперація та підтримка, гнучкість та адаптивність, підвищення мотивації, Міжособистісний підхід до формування команди зазвичай розглядає взаємодію та відносини між

членами команди як ключовий фактор успіху командного проекту. Цей підхід більше зорієнтований на аспекти характеру, особистості, комунікації та міжособистісних відносин. Мета полягає в створенні гармонійної та взаємодопоміжної команди, щоб досягти спільних цілей.

Для представлення міжособистісного підходу до формування команди математичною моделлю, було б важко використовувати точні числові обчислення, оскільки цей підхід більше базується на якісних та суб'єктивних аспектах. Пропонуємо використовувати якісні критерії та методи для оцінки взаємодії та відносин між членами команди через анкетування та опитування.

3.6 Формування команд виконавців на базі компетентнісного підходу

Інформаційні технології можуть бути використані для аналізу та формування команд виконавців на базі компетентнісного підходу. Використання інформаційних технологій для збору, аналізу даних про компетентності та навички потенційних виконавців. Це може включати створення профілів співробітників, оцінку їхнього досвіду, освіти та сертифікацій, а також використання інструментів оцінки компетентностей. Це дозволить легко знаходити та аналізувати інформацію про виконавців, їхні навички та потенційні можливості. Використання аналітики даних та машинного навчання для аналізу та виявлення залежностей між компетентностями виконавців та успіхом проектів. Це допоможе зрозуміти, які компетентності є ключовими для успішної команди та підвищення їх ефективності. Використання інформаційних технологій для аналізу даних та формування команд виконавців на базі компетентнісного підходу допомагає ефективно оцінювати та підбирати виконавців з потрібними навичками та компетентностями для досягнення успіху проектів.

Компетентнісний аналіз є процесом оцінки навичок, знань, досвіду та інших ключових компетентностей учасників команди. Для проведення компетентнісного аналізу можна використовувати декілька підходів. Визначаються компетентності, які є важливими для успішного виконання завдань в IT проекті. Це можуть бути технічні навички, міжособисті комунікаційні вміння, лідерські якості, знання

специфіки галузі тощо. Пошук інформації про потенційних учасників команди, яка стосується їхнього досвіду, освіти, сертифікації, участі у попередніх проектах, досягнень тощо. Доцільно також зібрати та проаналізувати відгуки колег, керівників або клієнтів про їхні вміння та виконавські якості. На основі зібраної інформації проводиться оцінка компетентностей кожного учасника команди. Для цього використовується шкала оцінювання, матриці порівнянь або інші методи, що відображають рівень компетентностей кожної особи. Проводиться аналіз результатів оцінки для виявлення прогалин в компетентностях команди. Для цього визначається, які навички або знання відсутні або потребують покращення. Компетентнісний аналіз допомагає забезпечити, що у команді є необхідні навички та компетентності для виконання завдань проекту. Це дозволяє краще розподіляти ролі та відповідальності, підвищує ефективність роботи та забезпечує успішне виконання ІТ проекту. Вважається, що проект успішний, якщо керівник і команда дотримувалися встановлених термінів його реалізації та бюджету, а якість створеного програмного продукту відповідає вимогам. Менеджер проекту несе відповідальність за реалізацію конкретних цілей проекту в рамках обмежень проекту. У PMBoK Project Manager вони визначаються як особа, яка керує командою та відповідає за досягнення цілей проекту [77]. При цьому чітко розмежовані ролі керівника проекту та функціонального менеджера. Перший керівник відповідає за нагляд за окремим підрозділом, а другий — за забезпечення ефективності господарських операцій. Керівник проекту працює з менеджером портфоліо або програмою, щоб переконатися, що команда працює над досягненням цілей проекту та відповідає розкладу програми. Керівник проекту також тісно співпрацює з членами команди, які виконують інші ролі [78].

Формування команди проекту – це процес підтвердження можливості залучення людських ресурсів, необхідних для досягнення мети проекту, для роботи в команді. Але реалізацію великих проектів віддають перевагу більші команди або можна сформувати кілька команд, які мають більше інструментів і ресурсів. Для визначення оптимальної кількості учасників команди доцільно розподілити ролі з

урахуванням професійної компетентності кожного учасника, розробити правила взаємодії та алгоритми прийняття рішень.

Безперечно, успіх колективу багато в чому залежить від кожного його учасника. Для успішної реалізації проекту необхідна ефективна команда однодумців, концептуальна модель якої визначається низкою факторів, які можна представити у вигляді кортежу елементів з $n > 2$ елементів:

$$K=(B1+B2+ B3+ B4+\dots Bn-1 +Bn), \quad (3.24)$$

де для всіх k, $1 \leq k \leq n$, маємо $Bk \in Kk$.

При цьому можемо вважати, що першими чотирма елементами кортежу є:

B1 – ефективна комунікація серед учасників команди, при якій правильно і з мінімальними втратами передається інформація.

B2 - відповідальність, яка покладається на кожного учасника за призначену зону, а вся команда несе солідарну відповідальність за результат.

B3 – атмосфера довіри, співпраці, взаємодопомоги, яка панує у команді, яка генерує у працівників бажання ефективно виконувати роботу і співпрацювати з колегами.

B4 – мотивація учасників команди до успішної реалізації проекту, усвідомлення цінностей, цілей як спільного досягнення.

B5 - синергетичний ефект від спільної роботи команди якісно перевершує ефект від роботи окремих її членів.

Як показує досвід проектної роботи в IT-сфері, раннє залучення учасників команди до розподілу ролей на етапі планування підвищує мотивацію членів команди. У цьому випадку компетенції та кількість членів проектної команди можуть змінюватися, оскільки проект реалізується в невеликих проектах, використовуючи гнучку методологію, відповідальність за управління проектом може бути розподілена між усіма членами команди [79].

Слід зазначити, що PMBoK визначає, що набір процесів управління людськими ресурсами проекту містить низку процесів, серед яких важливу роль відіграє планування людських ресурсів. На цьому етапі визначається перелік вимог до претендентів, коло їх обов'язків і окреслення зв'язків з іншими проектами, що

лягає в основу плану управління персоналом. Безпосереднє формування команди проекту включає не тільки процедуру відбору її персоналу, а й підтримку процедур підвищення рівня навичок і вмінь кожного члена команди, налагодження взаємодії між членами команди. Саме ці фактори підвищують ефективність реалізації проекту (рис. 3.1).

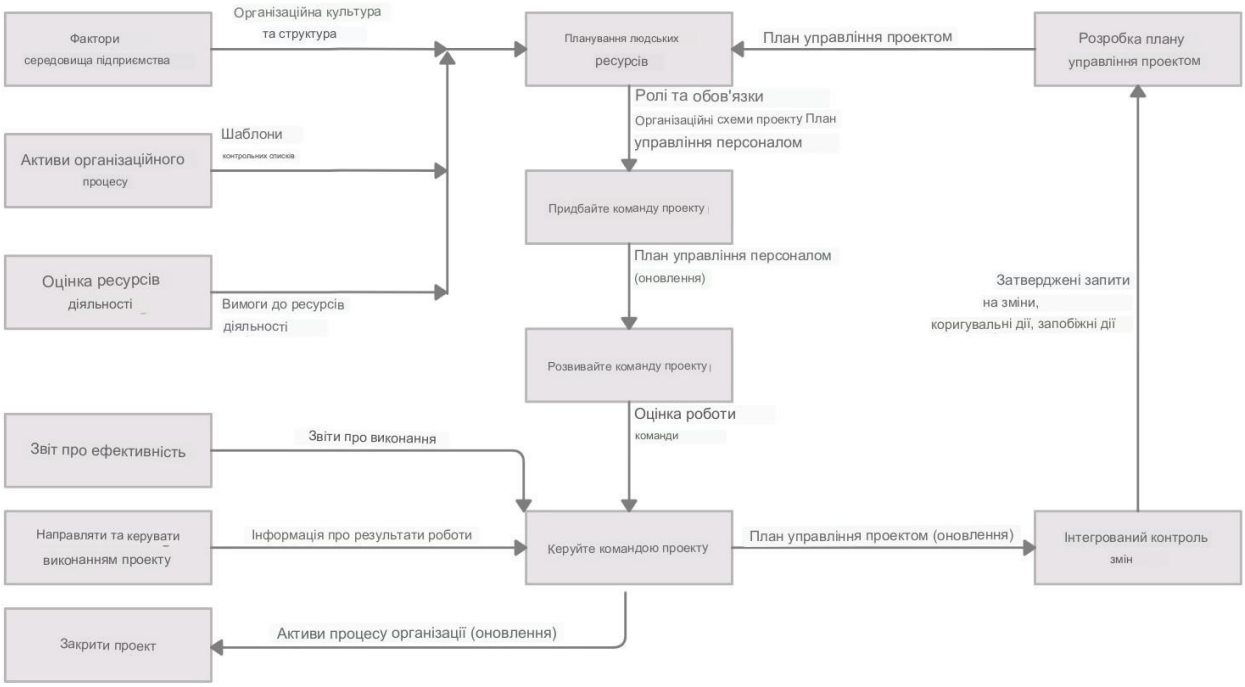


Рис. 3.1 Блок-схема процесу управління людськими ресурсами проекту

Планування людських ресурсів включає ряд процесів: розподіл ролей між членами команди проекту; визначення обов'язків; встановлення продуктивних робочих стосунків; створення плану управління персоналом. Розподіл ролей може відбуватися між окремими особами або групами людей, які перебувають у зовнішньому чи внутрішньому середовищі проекту. Планування управління персоналом може визначати спосіб і час формування команди проекту, критерії оцінки їх компетенцій, необхідність навчання претендентів, винагороди за якісне завчасне впровадження процесів.

При формуванні команди відповідно до рекомендацій PMBoK враховується низка факторів (рис. 3.2), таких як:

• Організаційні;

• Технічні;

• Міжособистісне спілкування;

• Логістика;

• Політичний.

Організаційні.

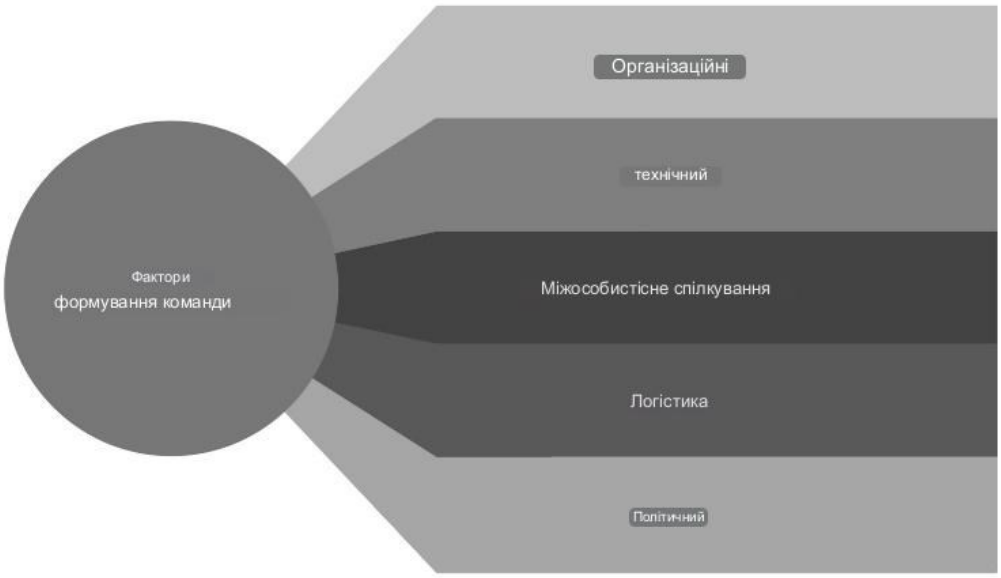


Рис. 3.2. Фактори формування команди згідно з рекомендаціями PMBoKних відносин.

3.7 Поведінкова модель команди як сукупний рух імітованої зграї

Утворення ефективної команди виконавців ІТ проекту є багатоаспектним процесом, який потребує уваги до деталей та гнучкості. Успіх команди залежить від керівництва, спільної відповідальності та здатності пристосовуватися до змін у процесі реалізації проекту. Злагоджену роботу команди можна порівняти з поведінкою зграї птахів. Зімітована зграя — це вдосконалення системи частинок, в якій імітовані птахи є частинками [80]. Сукупний рух імітованої зграї створюється розподіленою поведінковою моделлю, яка функціонує за тим самим принципом, за яким діють природні зграї; птахи самі обирають свій шлях. Кожен змодельований птах реалізується як незалежний актор [81], який орієнтується відповідно до свого локального сприйняття динамічного середовища, законів імітованої фізики, які керують її рухом, і набору поведінок, запрограмованих у неї «аніматором».

Сукупний рух імітованої зграї є результатом щільної взаємодії відносно простих поведінок окремих імітованих птахів [82].

Формування команди ІТ проекту можемо розглядати як аналог зграї. Змодельована зграя, тісно пов’язана із системами частинок [83], які використовуються для представлення динамічних «нечітких об’єктів» неправильної та складної форми [84]. Системи частинок — це сукупність великої кількості окремих частинок, кожна з яких має свою власну поведінку. Частинками є претенденти у команду. Частинки створюються, старіють і відмирають, своєю чергою претендентів відбирають, їх компетентності старіють і послуг таких фахівців команди не потребують. Впродовж свого життя частинки вирізняються певною поведінкою, яка може змінити власний стан частинки, який складається з кольору, розташування та швидкості. В основі моделі птахоподібних об’єктів лежить узагальнення систем частинок [85].

Щоб побудувати змодельовану зграю, варто почати з моделі птахоподібного об’єкта, яка підтримує геометричний політ, додано поведінку, яка відповідає протилежним силам уникнення зіткнення та бажанням приєднатися до зграї. Аналогічним є підхід до формування команди для ІТ проекту. Кожна з поведінкових компонент, пов’язаних зі зграюванням, дають окремі пропозиції щодо того, яким шляхом спрямувати об’єкт зграї. Вони виражаються як запити на прискорення. Кожна поведінка має декілька параметрів, які контролюють її функцію; один — це «сила», дробове значення від нуля до одиниці, яке може ще більше послабити запит прискорення. Аналогія поведінки зграї та формування команди за допомогою ройових алгоритмів може бути цікавою та корисною. Ройові алгоритми базуються на спостереженні природної поведінки зграї або інших колективів у природі для розв'язання завдань оптимізації та координації. Ця аналогія може бути корисною для розуміння, як команди формуються, співпрацюють та оптимізують свою продуктивність.

Взаємодія членів команди за допомогою ройового алгоритму може бути змодельована за допомогою ройового алгоритму оптимізації. Ройові алгоритми згенеровані природною поведінкою зграї птахів, де кожен член зграї взаємодіє з

іншими, орієнтуючись на досягнення спільних цілей. Ось метод моделювання взаємодії членів команди за допомогою ройового алгоритму:

Крок 1. Ініціалізація. Створення початкової популяції агентів команди, де кожен агент може мати певні параметри або властивості, що представляють його характеристики або ролі в команді.

Крок 2. Оцінка. Оцінка продуктивності кожного агента в команді на основі його властивостей або параметрів. Це може включати обчислення вкладу кожного агента в досягнення цілей команди.

Крок 3. Моделювання руху. Розробка правил, за якими агенти взаємодіють один з одним і приймають рішення. Це може включати такі аспекти, як взаємодія з ближніми сусідами, вплив комунікаційних зв'язків тощо.

Крок 4. Оновлення популяції. Агенти виконують дії згідно з модельними правилами, можуть змінювати свої властивості або параметри відповідно до результатів взаємодії.

Крок 5. Оцінка. Після кожної ітерації команди оцінюються їхня продуктивність та результати досягнення цілей.

Крок 6. Повторення. Цей процес повторюється протягом кількох ітерацій для досягнення оптимальних результатів.

Приклад коду Python, який демонструє моделювання взаємодії команди за допомогою ройового алгоритму подано у додатку *(додаток В, лістинг 3.11).* У цьому прикладі ми створили команду агентів імпровізовано, визначили їхню продуктивність та моделювали взаємодію між ними на основі ближніх сусідів. Продуктивність кожного агента оновлюється після кожної ітерації, і у кінці ми оцінюємо продуктивність команди в цілому.

Ось деякі аналогії між поведінкою зграї та формуванням команди: лідерство, взаємодія і співпраця, координація і синхронізація, пошук рішень. В зграї та команді може бути лідер, який надає напрямок та координує дії інших членів. Лідер може бути обраний на підставі своїх навичок, досвіду або інших якостей, які забезпечують ефективне керівництво. Для розрахунків за цим сценарієм, де в команді є лідер, який обирається на підставі своїх якостей, можна використовувати

лінійне програмування. У цьому прикладі ми розглянемо ситуацію з командою з 6 членів, де кожен член може вибрати одну з трьох ролей: "Розробник", "Тестувальник" або "Дизайнер". Лідер обирається на основі свого впливу, який вимірюється числовим показником. Приклад коду з використанням бібліотеки PuLP для вирішення цієї задачі лінійного програмування подано у додатку *(додаток В, лістинг 3.12).*

Статус розв'язку: 1

Член 1 вибрав роль: Розробник. Член 1 обраний лідером

Член 2 вибрав роль: Розробник. Член 2 обраний лідером

Член 3 вибрав роль: Розробник. Член 3 обраний лідером

Член 4 вибрав роль: Розробник. Член 4 обраний лідером

Член 5 вибрав роль: Розробник. Член 5 обраний лідером

Член 6 вибрав роль: Розробник. Член 6 обраний лідером

У цьому прикладі визначено задачу лінійного програмування для вибору ролей для кожного члена команди та обрання лідера з метою максимізації продуктивності команди. Цільова функція враховує вибір ролей та внесок кожної ролі, а також вибір лідера. В зграї та команді важливо вміти взаємодіяти та співпрацювати. Члени команди повинні взаємодіяти, щоб досягти спільних цілей, аналогічно до того, як члени зграї співпрацюють для максимальної ефективності. Приклад розрахунків за сценарієм обрання лідера у команді з використанням лінійного програмування. В даному випадку, ми визначимо цільову функцію для максимізації продуктивності команди з урахуванням обраного лідера та вибору ролей для кожного члена команди *(додаток В, лістинг 3.13).*

Статус розв'язку: 1

Член 1 вибрав роль: Розробник. Член 1 обраний лідером

Член 2 вибрав роль: Розробник. Член 2 обраний лідером

Член 3 вибрав роль: Розробник. Член 3 обраний лідером

Член 4 вибрав роль: Розробник. Член 4 обраний лідером

Член 5 вибрав роль: Розробник. Член 5 обраний лідером

Член 6 вибрав роль: Розробник. Член 6 обраний лідером

Загальна продуктивність команди: 5.699999999999999

У цьому прикладі ми визначили задачу лінійного програмування для вибору ролей для кожного члена команди та обрання лідера з метою максимізації продуктивності команди. Результати включають вибір ролей, обраного лідера та загальну продуктивність команди з урахуванням цих виборів.

Зграя птахів або риб повинна координувати свої рухи для уникнення зіткнень та досягнення колективних цілей. Так само і команда повинна синхронізувати свої зусилля для досягнення успіху. Приклад розрахунків за сценарієм обрання лідера у команді за допомогою лінійного програмування. У цьому прикладі ми розглянемо команду з 5 членів, де кожен член може вибрати одну з двох ролей: "Розробник" або "Тестувальник". Лідер обирається на основі свого впливу на команду, а також на основі обраних ролей у команді *(додаток В, лістинг 3.14)*

Статус розв'язку: 1

Член 1 вибрав роль: Розробник

Член 2 вибрав роль: Розробник

Член 3 вибрав роль: Розробник

Член 4 вибрав роль: Розробник

Член 5 вибрав роль: Розробник

Один із членів команди обраний лідером

Загальна продуктивність команди: 4.8999999999999995

Лінійне програмування (ЛП) може бути використано для оптимізації вибору ролей для кожного члена команди та визначення лідера з метою максимізації продуктивності команди. Давайте визначимо деякі базові елементи, які можна врахувати в цьому контексті:

Ролі команди (variables):

x_i - змінна, яка вказує, яку роль і-й учасник команди виконує. Може приймати значення 0 або 1, де 1 вказує на те, що учасник виконує дану роль, а 0 - ні.

Визначення ролей:

Наприклад, якщо у нас є ролі $R_1, R_2, \ldots, R_n$, то змінні $x_1, x_2, \ldots, x_n$ визначають, яку роль кожен учасник виконує.

Лідерство (variable):

y - змінна, яка вказує, хто з учасників є лідером (якщо *y*=1) і хто не є (якщо *y*=0).

Функція максимізації продуктивності (objective function):

$Z=c_1x_1+c_2x_2+\ldots+c_nx_n,$ (3.25)

де c_i - це показник продуктивності для кожної ролі R_i.

Обмеження (constraints):

$\sum_{i=1}^{n} x_i=1$- кожен учасник може виконувати лише одну роль.

y=1 - існує тільки один лідер.

$\in\{0,1\}x_i,y\in\{0,1\}$ - змінні можуть приймати лише значення 0 або 1.

Таким чином, задачу ЛП можна формалізувати наступним чином:

Максимізувати: $Z=c_1 x_1 + c_2 x_2+ \ldots+ c_n x_n$

Обмежено: $\sum_{i=1}^{n} x_i = 1$

y=1

$x_i,y\in\{0,1\}$

Ця формула представляє базовий фреймворк для задачі вибору ролей для членів команди та визначення лідера з метою максимізації продуктивності команди. Параметри c_i можна адаптувати залежно від конкретного контексту та важливості кожної ролі для досягнення мети.

У цьому прикладі ми використовуємо лінійне програмування для вибору ролей для кожного члена команди та вибору лідера з метою максимізації продуктивності команди. Результати включають вибір ролей, обраного лідера та загальну продуктивність команди з урахуванням цих виборів.

В природі зграї шукають оптимальні шляхи для досягнення певних цілей, наприклад, пошук харчів або уникнення небезпеки. У команді також часто потрібно шукати рішень для вирішення завдань та проблем. Сценарій пошуку рішень в команді або зграї може бути математично представлений за допомогою різних моделей і методів, залежно від конкретної ситуації та поставленої мети. Ось приклади математичних підходів для моделювання сценарію пошуку рішень:

Команду можна розглядати як систему агентів, кожен з яких має свої цілі та обмеження. Застосування агентних моделей дозволяє моделювати взаємодію агентів під час пошуку рішень та розв'язання проблем. Якщо потрібно знайти оптимальний спосіб розподілу ресурсів або вирішити проблему маршрутизації, то моделі мережевого потоку можуть бути корисними для представлення та розв'язання завдань пошуку рішень в команді. Сценарій пошуку рішень за допомогою моделі мережевого потоку може бути використаний для вирішення задач розподілу ресурсів або маршрутизації в команді. Ось приклад розрахунків за цим сценарієм, де команда повинна прийняти рішення щодо розподілу завдань між її членами для досягнення оптимального результату *(додаток В, лістинг 3.15)*. У цьому прикладі використовується модель мережевого потоку для розподілу завдань між членами команди з метою мінімізації загального часу виконання завдань. Встановлюються пропускні здатності на ребрах графу, які представляють, скільки завдань може виконати кожен член команди, і розв'язуємо задачу мінімізації за допомогою методу linear_sum_assignment. Якщо задача пошуку рішень складна та має багато можливих варіантів, генетичні алгоритми та еволюційні підходи можуть використовуватися для знаходження оптимальних рішень під час еволюції команди. Генетичні алгоритми та еволюційні підходи можуть бути використані для пошуку оптимальних рішень в команді. Ось приклад розрахунків за цим сценарієм, де команда використовує генетичний алгоритм для оптимізації розподілу ролей *(додаток В, лістинг 3.16)*.

Найкращий розподіл ролей: [4, 4, 4, 4, 4, 4, 4, 4, 4, 4]

Значення метрики для найкращого розподілу: 40

У цьому прикладі використано генетичний алгоритм для пошуку оптимального розподілу ролей в команді. Починаючи з випадкової початкової популяції, алгоритм еволюціонує протягом декількох поколінь, обираючи найкращих особин, схрещуючи їх і застосовуючи мутації. Результатом є найкращий розподіл ролей, що максимізує визначену метрику (у цьому випадку суму ролей). Еволюційні підходи також можуть бути використані для пошуку оптимальних рішень в команді, але вони можуть використовувати різні методи

оптимізації та еволюційні стратегії. Ось приклад розрахунків за цим сценарієм, де команда використовує еволюційний підхід для оптимізації розподілу ролей, *(дивись додаток В, лістинг 3.17)*.

Найкращий розподіл ролей: [4, 4, 3, 4, 4, 3, 2, 3, 2, 3]

Значення метрики для найкращого розподілу: 32

У цьому прикладі ми використовуємо еволюційні підходи для пошуку оптимального розподілу ролей в команді. Еволюційні підходи можуть використовувати генетичні алгоритми, але також можуть включати інші методи оптимізації та пошуку рішень. Ось приклад розрахунків за цим сценарієм, де команда використовує еволюційний підхід для оптимізації розподілу ролей *(додаток В, лістинг 3.18)*.

Найкращий розподіл ролей: [4, 4, 4, 4, 4, 4, 4, 4, 4, 4]

Значення метрики для найкращого розподілу: 40

У цьому прикладі використано еволюційний підхід для пошуку оптимального розподілу ролей в команді. Процес еволюції включає оцінку, відбір, схрещування та генерацію нових особин у популяції з метою максимізації визначеної метрики. В результаті отримується найкращий розподіл ролей, що відповідає цільовій функції.

Моделі на основі правил можуть бути використані для моделювання пошуку рішень в команді на основі конкретних правил, що керують поведінкою членів команди. У цьому прикладі ми розглянемо сценарій, де команда вирішує, як розподілити ресурси між своїми членами на основі правил розподілу, *(дивись додаток В, лістинг 3.20)*.

Член команди 1: [5 4 4 2 0 4 4 3 0 0]

Член команди 2: [2 1 1 3 2 4 3 4 4 2]

Член команди 3: [1 0 4 4 0 0 0 2 0 3]

Член команди 4: [0 2 0 0 2 0 0 0 1 1]

Член команди 5: [0 0 1 0 0 0 0 0 1 0]

Залишок нерозподілених ресурсів: [0 0 0 0 2 0 0 0 4 0]

У цьому прикладі використано матрицю правил розподілу ресурсів, де кожний рядок представляє правила для одного з членів команди, а кожний стовпчик відповідає одному ресурсу. За допомогою цих правил та доступних ресурсів ми розподіляємо ресурси між членами команди. Результати розподілу виводяться на екран, а також вказується залишок нерозподілених ресурсів.

Команди та зграї можуть змінювати свою поведінку, якщо умови змінюються. Вони адаптуються до нових вимог та середовища для забезпечення ефективності та виживання. Ця аналогія може бути корисною для розуміння та оптимізації роботи команди, особливо в ситуаціях, де важлива координація, взаємодія та співпраця між членами команди для досягнення спільних цілей. Ройові алгоритми можуть також надати інспірацію для розробки інтелектуальних систем, які моделюють природну поведінку зграї для розв'язання різних завдань, таких як оптимізація маршрутів або роботи у розподіленому середовищі. Поведінку зграї та формування команди можна зобразити математичними моделями, такими як агентні моделі, графи, системи рівнянь, автомати та інші. Математичні моделі дозволяють абстрагувати та вивчати ключові аспекти взаємодії, координації та поведінки в колективі. Ось деякі приклади математичних моделей для цих сценаріїв:

Для моделювання поведінки зграї можна використовувати агентні моделі. Кожен агент (птах або риба) може мати певні правила для руху, інтеракції з іншими агентами та прийняття рішень. Це може бути представлено системою диференціальних рівнянь для руху кожного агента та правилами для реагування на зміни у середовищі.

Для формування та діяльності команди можна використовувати агентні моделі, де кожен член команди виступає як агент зі своїми власними властивостями та правилами.

Задача формування та діяльності команди, використовуючи агентні моделі, може бути сформульована наступним чином:

Агенти:

$A_1, A_2, \ldots, An$ - агенти, що представляють учасників команди.

Властивості агентів:

P_i - набір властивостей (навички, досвід, спеціалізація тощо) для кожного агента A_i.

Обов'язки:

$R_1,R_2,\ldots,R_m$ - обов'язки членів команди або різні ролі, які можуть бути виконані.

Взаємодія:

C_{ij} - рівень співпраці між агентами A_i та A_j. Агенти можуть взаємодіяти між собою, обираючи ролі та обговорюючи стратегії для досягнення спільних цілей.

Функція вартості:

$F(A_1,A_2,\ldots,A_n)$ - функція вартості або ефективності команди, яка залежить від властивостей агентів і рівня їх взаємодії. Цільова функція може визначати продуктивність команди, враховуючи внесок кожного агента та відповідність обраної ролі його навичкам.

Ця задача може бути сформульована для максимізації функції вартості при врахуванні обмежень на властивості агентів. З урахуванням цих елементів, агентна модель може бути представлена формальною системою правил та відносин, де кожен агент має визначений внесок у командну діяльність.

Взаємодія між агентами може бути модельована за допомогою графів, де ребра представляють комунікаційні зв'язки, а вузли - агентів. Для розрахунків за сценарієм, де кожен член команди виступає як агент зі своїми властивостями та правилами взаємодії, ми можемо використовувати агентну модель та графовий підхід. В даному прикладі, ми розглянемо формування команди з двома типами агентів: лідерами та послідовниками, які спільно приймають рішення за допомогою графової структури.

Почнемо з опису агентів та їх властивостей:

Лідери (Leaders). Кожен лідер має свої персональні характеристики, такі як рішучість (p), вплив (інші агенти, на які він впливає), інші властивості тощо. Вони приймають рішення на основі своїх властивостей та взаємодіють з іншими агентами.

Послідовники (Followers). Кожен послідовник також має свої характеристики та вплив на інших агентів. Вони віддаватимуть перевагу лідерам, які мають великий вплив. Взаємодія між агентами може бути описана наступними правилами:

Лідери та послідовники формують комунікаційний граф. Вузли графу відповідають агентам, а ребра - комунікаційним зв'язкам. Лідери намагаються збільшити свій вплив та рішучість через взаємодію з іншими агентами. Послідовники намагаються визначити, які лідери мають найбільший вплив, і приймати рішення на основі цього впливу.

Розглянемо приклад розрахунків, де ми будемо моделювати динаміку взаємодії між лідерами та послідовниками у команді за допомогою графової агентної моделі, *(додаток В, лістинг 3.22)*.

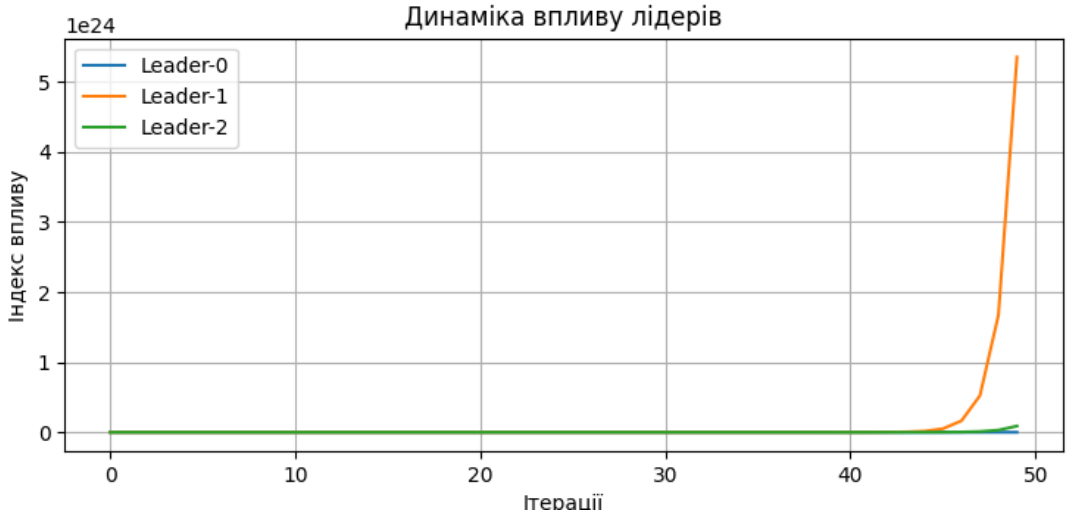


Рис. 3.3 – Динаміка впливу лідерів

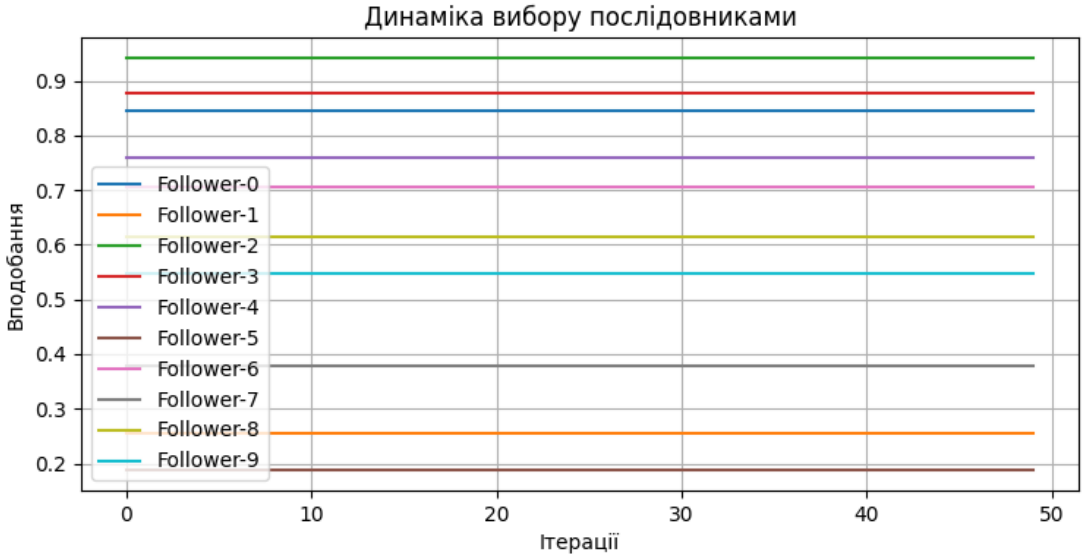


Рис. 3.4 – Динаміка впливу послідовників

Цей код продовжує попередню конфігурацію команди, додає агентів лідерів і послідовників до графу, визначає комунікаційні зв'язки між ними та моделює динаміку взаємодії та впливу в команді протягом декількох ітерацій. Візуалізація показує динаміку впливу лідерів та вибір послідовниками на основі їхнього впливу. Можна побачити, як вплив лідерів змінюється з часом і як це впливає на вибір послідовниками щодо того, якого лідера обрати. Для розрахунків за сценарієм використання агентних моделей для формування та діяльності команди, де кожен член команди виступає як агент зі своїми властивостями та правилами, ми можемо розглянути приклад на основі агентної моделі з використанням графів.

Припустимо, що ми маємо команду з двох видів агентів: "Лідери" і "Послідовники", які взаємодіють між собою за допомогою комунікаційних зв'язків, представлених графом. Кожен агент має свої властивості, такі як "рішучість" і "підпорядкованість", і вони впливають на спільне прийняття рішень командою.

Подамо агентну модель імпровізованої наради, де команда має прийняти спільне рішення щодо сценарію виконання певного процесу *(додаток В, лістинг 3.23)*.

У цьому прикладі агенти команди представлені в графі, і вони взаємодіють між собою з урахуванням їхньої "рішучості" та "підпорядкованості". Кожен агент приймає рішення на основі впливу від своїх сусідів, і результати відображаються

на графі. Цей приклад ілюструє, як агенти можуть взаємодіяти в команді, приймати спільні рішення та взаємодіяти між собою за допомогою агентних моделей і графової структури.

Можна використовувати системи диференціальних рівнянь для опису динаміки та взаємодії між членами зграї чи команди. Наприклад, модель взаємодії членів команди під час прийняття спільних рішень. Зважаючи на сценарій взаємодії членів команди під час прийняття спільних рішень, ми можемо використовувати систему диференціальних рівнянь для моделювання динаміки цього процесу. Розглянемо приклад, де дві групи агентів (лідери та послідовники) намагаються досягти консенсусу щодо вибору оптимального рішення.

Модель може бути представлена наступними рівняннями:

Для групи лідерів (L): $dL/dt = \alpha\_L (L\_target - L)$, (3.26)

де dL/dt - зміна кількості лідерів з часом, α_L - коефіцієнт впливу, L_target - бажана кількість лідерів.

Для групи послідовників (F): $dF/dt = \alpha\_F (F\_target - F)$, (3.27)

де dF/dt - зміна кількості послідовників з часом, α_F - коефіцієнт впливу, F_target - бажана кількість послідовників.

Клітинні автомати та автомати з правилами можуть бути використані для моделювання мікрорівня інтеракцій в зграї чи команді. Ці моделі дозволяють визначити, як зміни в стані одного агента впливають на стан інших.

Використання теорії графів для представлення структури взаємодії між членами команди або агентами в зграї допомагає аналізувати взаємодії, виявляти ключових акторів і спрямовувати комунікацію. Нижче подані приклади розрахунків з використанням теорії графів для аналізу команди.

Припустимо, маємо команду з 6 агентів, і ми хочемо побудувати граф взаємодії на основі їхніх зв'язків. Для цього спочатку створимо матрицю суміжності, яка відображає, які агенти спілкуються між собою, *(дивись додаток В, лістинг 3.24)*.

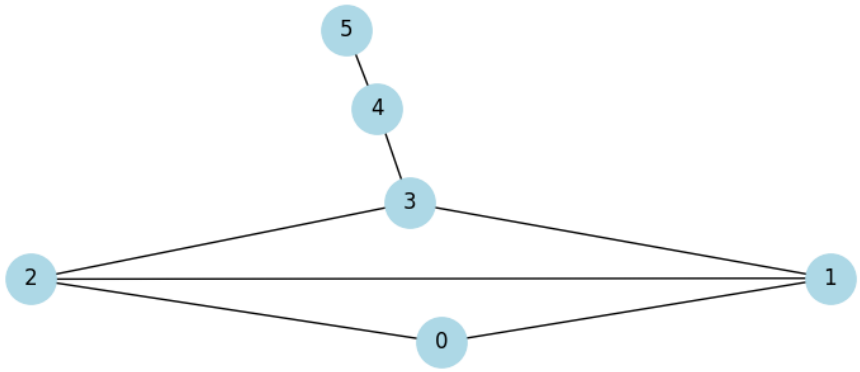


Рис. 3.5 – Граф взаємодії команди на основі матриці суміжності

У цьому прикладі ми створили граф взаємодії команди на основі матриці суміжності. Тепер можна проводити аналіз цього графу для визначення центральних членів команди, груп і взаємодій. Якщо необхідно здійснити аналіз графу, наприклад, знайти найкоротший шлях між двома агентами, центральність агентів тощо, вам може знадобитися використовувати бібліотеку NetworkX для Python. Ці математичні моделі можуть допомогти досліджувати та розуміти, як взаємодія та координація впливають на поведінку групи агентів та членів команди. Вони також можуть бути корисними для оптимізації та прогнозування результатів, а також для вивчення впливу різних факторів на динаміку колективної діяльності. Наведемо приклад використання агентних моделей в контексті формування команди з математичним поданням:

Розглянемо приклад проведення розрахунків з використанням диференціальних рівнянь для опису динаміки процесу формування команди. В цьому прикладі ми будемо моделювати, як команда формується в часі та як її склад може змінюватися в залежності від внутрішньої динаміки.

Припустимо, що ми маємо групу агентів, які мають певні властивості або навички, і ми хочемо вивчити, як ця команда формується з часом. Ми можемо використовувати диференціальні рівняння для опису динаміки зміни складу команди.

Розглянемо таку систему диференціальних рівнянь:

Для агента 1: $dA_1(t)/dt = -k_1 * A_1(t)$, (3.28)

Для агента 2: $dA_2(t)/dt = -k_2 * A_2(t)$, (3.29)

Для агента 3: $dA_3(t)/dt = -k_3 * A_3(t)$, (3.30)

де $A_1(t)$, $A_2(t)$ і $A_3(t)$ - кількість кожного агента в команді в часі, k_1, k_2 і k_3 - коефіцієнти, що впливають на швидкість зменшення кількості агентів. У цьому прикладі ми припускаємо, що кількість агентів зменшується з часом.

Код для розв'язання цієї системи диференціальних рівнянь методом Ейлера може виглядати так (в мові Python), *(дивись додаток В, лістинг 3.25)*.

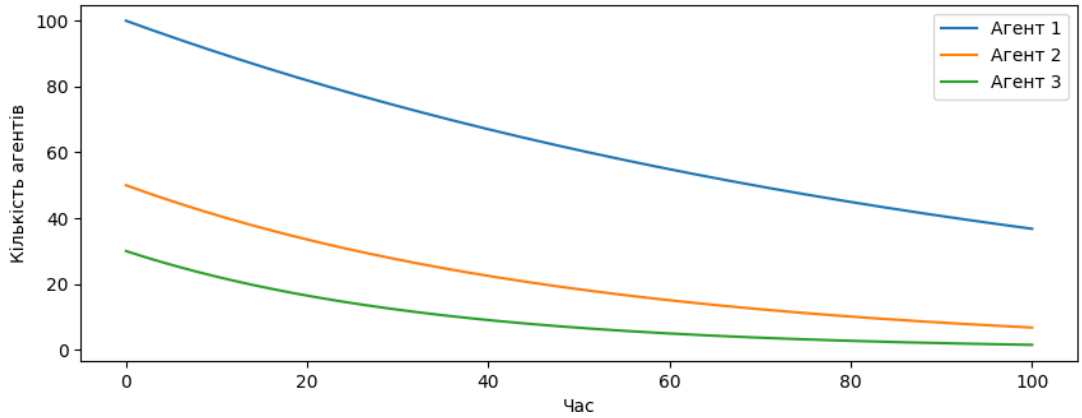


Рис. 3.6 – Зміна команди з часом, відповідно до заданих коефіцієнтів

Цей код моделює, як кількість агентів у команді зменшується з часом відповідно до системи диференціальних рівнянь. В результаті отримується графік, який показує, як команда складається та змінюється з часом відповідно до заданих коефіцієнтів.

Для опису динаміки взаємодії агентів за допомогою диференціальних рівнянь розглянемо простий приклад моделі, де агенти взаємодіють у процесі спільної роботи та навчання. Припустимо, що ми маємо два агенти, і ми хочемо вивчити, як змінюються їхні навички з часом.

Припустимо, що навички агентів представлені як функції часу. Ми можемо описати динаміку цих навичок за допомогою диференціальних рівнянь. Давайте позначимо навички першого агента через $S_1(t)$ і навички другого агента через $S_2(t)$.

Тоді диференціальні рівняння для динаміки навичок можуть виглядати наступним чином:

Для першого агента: $dS_1(t)/dt = \alpha_1 * (S_2(t) - S_1(t))$, (3.31)

Для другого агента: $dS_2(t)/dt = \alpha_2 * (S_1(t) - S_2(t))$, (3.32)

де α_1 і α_2 - коефіцієнти, що визначають швидкість навчання агентів. Ці рівняння вказують, що навички кожного агента змінюються з часом відповідно до різниці між їхніми навичками та залежать від коефіцієнтів навчання.

Тепер ми можемо використовувати ці рівняння для моделювання динаміки навичок агентів в часі та вивчення, як взаємодія між ними впливає на навички. За допомогою числових методів розв'язання диференціальних рівнянь, таких як метод Ейлера чи метод Рунге-Кутта, можна провести розрахунки та вивчити, як змінюються навички агентів у процесі їхньої взаємодії.

Зважаючи на рівняння динаміки навичок агентів, які були представлені раніше, можемо розглянути приклад розрахунків за допомогою методу Ейлера. Ми розглянемо випадок з двома агентами, де їхні навички змінюються відповідно до рівнянь:

Для першого агента: $dS_1(t)/dt = \alpha_1 * (S_2(t) - S_1(t))$, (3.33)

Для другого агента: $dS_2(t)/dt = \alpha_2 * (S_1(t) - S_2(t))$. (3.34)

Для спрощення припустимо, що $\alpha_1 = \alpha_2 = 0.1$. Також, початкові значення навичок $S_1(0)$ і $S_2(0)$ будуть відомі.

Використаємо метод Ейлера для чисельного розв'язання цих рівнянь протягом певного часового інтервалу. Ми можемо виразити зміни навичок як:

$\Delta S_1 = \alpha_1 (S_2 - S_1) \Delta t$, $\Delta S_2 = \alpha_2 (S_1 - S_2) \Delta t$, (3.35)

де Δt - крок часу.

Почнемо з початкових значень $S_1(0) = 0.5$ і $S_2(0) = 0.3$, і виконаємо кілька кроків методу Ейлера. Нехай крок часу буде $\Delta t = 0.1$.

Крок 1: $\Delta S_1 = 0.1 (0.3 - 0.5) 0.1 = -0.002$, $\Delta S_2 = 0.1 (0.5 - 0.3) 0.1 = 0.002$. $S_1(0.1) = S_1(0) + \Delta S_1 = 0.5 - 0.002 = 0.498$, $S_2(0.1) = S_2(0) + \Delta S_2 = 0.3 + 0.002 = 0.302$.

Крок 2: $\Delta S_1 = 0.1 (0.302 - 0.498) 0.1 = -0.0196$, $\Delta S_2 = 0.1 (0.498 - 0.302) 0.1 = 0.0196$. $S_1(0.2) = S_1(0.1) + \Delta S_1 = 0.498 - 0.0196 = 0.4784$, $S_2(0.2) = S_2(0.1) + \Delta S_2 = 0.302 + 0.0196 = 0.3216$.

Таким чином, можемо продовжити розв'язувати рівняння методом Ейлера на багато інших кроків для вивчення, як змінюються навички агентів з часом під впливом їхньої взаємодії.

Розглянемо приклад розрахунків з використанням методу Ейлера для розв'язання системи диференціальних рівнянь, що описують динаміку навичок двох агентів, які взаємодіють. Використовуючи метод Ейлера, ми можемо наблизити значення навичок агентів на послідовних часових кроках.

Система диференціальних рівнянь виглядає наступним чином:

Для першого агента: $dS_1(t)/dt = \alpha_1 * (S_2(t) - S_1(t))$, (3.36)

Для другого агента: $dS_2(t)/dt = \alpha_2 * (S_1(t) - S_2(t))$, (3.37)

де α_1 і α_2 - коефіцієнти навчання агентів.

Розглянемо приклад, де $\alpha_1 = 0.2$ і $\alpha_2 = 0.1$. Початкові значення навичок агентів будуть $S_1(0) = 1$ і $S_2(0) = 2$. Ми будемо обчислювати значення навичок на протязі певного часу, використовуючи метод Ейлера з кроком часу Δt. Код для розв'язання цієї системи за допомогою методу Ейлера подано у додатку *(дивись додаток В, лістинг 3.26)*.

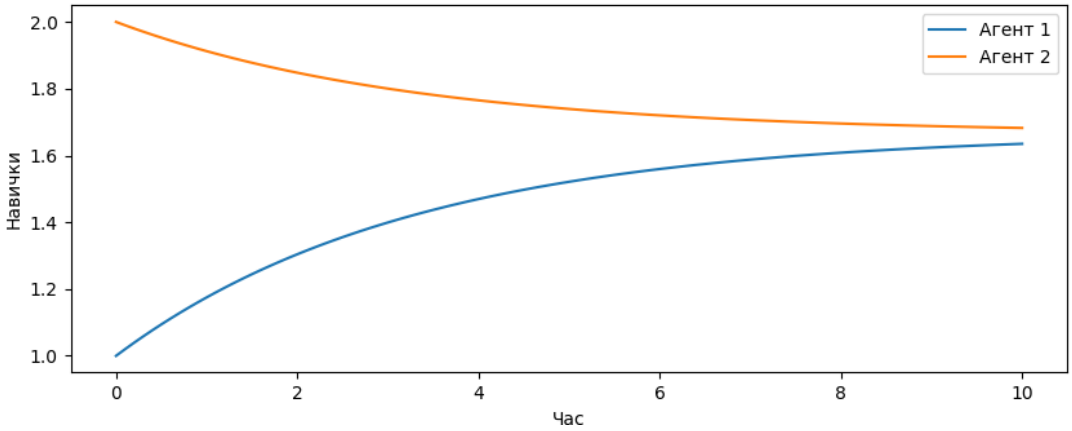


Рис. 3.7 – Графік динаміки навичок протягом певного часу.

Цей код обчислює та візуалізує динаміку навичок двох агентів протягом певного часу, використовуючи метод Ейлера з заданим кроком часу. В результаті отримується графік зміни навичок агентів з часом.

Диференціальні рівняння можуть використовуватися для оптимізації процесу формування команди у випадку, коли необхідно знайти команди, які максимізують певний об'єктив (наприклад, мінімізують час виконання завдання). Диференціальні рівняння можуть служити основою для оптимізаційних алгоритмів, які покращують обрані команди з плином часу. Для вирішення завдання оптимізації формування команди, яка максимізує певний об'єкт (наприклад, мінімізує час виконання завдання), можемо використовувати диференціальні рівняння разом із методами оптимізації. Розглянемо приклад, де ми маємо групу агентів і нам потрібно знайти оптимальний склад команди для мінімізації загального часу виконання завдання. Припустимо, що для кожного агента i у нас є параметр T_i, який представляє час, який цей агент потребує для виконання завдання. Наша мета - вибрати команду, яка мінімізує загальний час виконання завдання. Можемо визначити об'єктивну функцію, яка визначає загальний час виконання завдання для обраної команди:

Загальний час T_total = Σ(T_i) для всіх агентів у команді.

Для оптимізації цієї функції ми можемо використовувати диференціальні рівняння, які будуть змінювати склад команди з часом з метою покращення результатів. Можемо визначити рівняння для кожного агента, які керуватимуть його приєднанням до команди:

Для агента i: $dN_i(t)/dt = \alpha * \partial T_{total}/\partial N_i$, (3.38)

де $dN_i(t)/dt$ - швидкість зміни кількості агентів i в команді з часом, α - коефіцієнт, що визначає швидкість оптимізації, $\partial T_{total}/\partial N_i$ - часткова похідна об'єктивної функції відносно кількості агентів i в команді.

Це рівняння може бути розв'язане методом Ейлера чи іншими методами чисельної оптимізації для пошуку оптимального складу команди з мінімальним часом виконання завдання. Наведемо приклад розв'язання цієї системи диференціальних рівнянь за допомогою методу Ейлера у Python, *(додаток В, лістинг 3.27).*

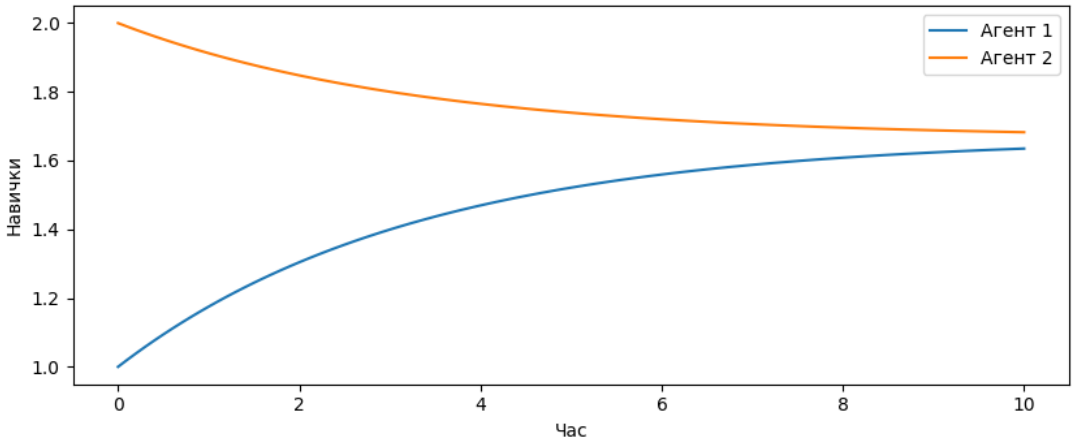


Рис. 3.8 – Графік відображення кількості агентів у команді та загальний час виконання завдання

У цьому прикладі оптимізуємо склад команди, щоб мінімізувати загальний час виконання завдання. Результати моделювання показують, як змінюється кількість агентів у команді та загальний час виконання завдання з часом при використанні методу Ейлера.

Найцікавіший рух імітованої зграї відбувається в результаті взаємодії з іншими об’єктами навколишнього середовища. Ізольована поведінка зграї має тенденцію досягати стійкого стану. Проте перешкоди з оточуючого середовища та спроби птахоподібного об’єкта орієнтуватися на рух навколо них посилюють складність поведінки зграї.

Висновки до 3 розділу

Запропоновано концептуальні моделі цілевизначального та рольового підходів до формування структури команди ІТ проекту,

Розроблено концептуальні моделі ізоморфної, експертної, колегіальної, розробницької, проектної, аналітичної, інтеграційної, інноваційної структур команди. Модель ізоморфної структури команди за допомогою теорії графів, де вузли представляють ролі або учасників команди, а ребра - зв'язки або взаємодії між ними. Модель рольового підходу до формування команди побудовано з використанням лінійного програмування.

Запропоновано підхід до розрахунку функції корисності кожного члена команди для пошуку оптимального розподілу ролей в команді.

Запропоновано метод для формування команди з використанням генетичного алгоритму, а оптимального складу та розподілу ролей у команді з використанням нейронних мереж, що навчаються на даних з параметрами проєктів та генетичного алгоритму. Розроблено метод максимізації продуктивності команди з використанням ітераційного та квалітативного аналізу.

Розроблена поведінкова модель команди як сукупний рух імітованої зграї, використовуючи метод моделювання взаємодії членів команди за допомогою ройового алгоритму. Розроблено метод для формування команди за аналогією зграї з використанням агентних моделей, де кожен член команди виступає як агент зі своїми власними властивостями та правилами, моделюючи взаємодію між агентами за допомогою графової агентної моделі, де ребра представляють комунікаційні зв'язки, а вузли - агентів.

РОЗДІЛ 4
ПРОТОТИП РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ФОРМУВАННЯ КОМАНД ДЛЯ ІТ ПРОЕКТІВ

4.1 Портфельне управління людськими ресурсами команд

Портфельне управління організаційними ресурсами команд — це стратегічний підхід до керування різнорідними ресурсами, що використовуються для досягнення цілей команди або організації. Цей підхід дозволяє краще забезпечити використання ресурсів та забезпечити оптимальну балансу між різними проектами та завданнями.

Портфель ресурсів включає в себе різні види ресурсів, такі як фінансові, людські, технічні та інші. Важливо визначити, які саме ресурси є доступними для використання командою чи організацією. Визначення портфеля ресурсів - це процес ідентифікації, оцінки і керування різнорідними ресурсами, які доступні для використання в рамках організації або команди. Ми розглядатимемо лише аспект портфельного управління людськими ресурсами, які використовуються для досягнення стратегічних цілей та завдань.

Портфельне управління організаційними ресурсами команд має багато переваг, які сприяють ефективному використанню ресурсів та досягненню стратегічних цілей організації. Ось деякі з них:

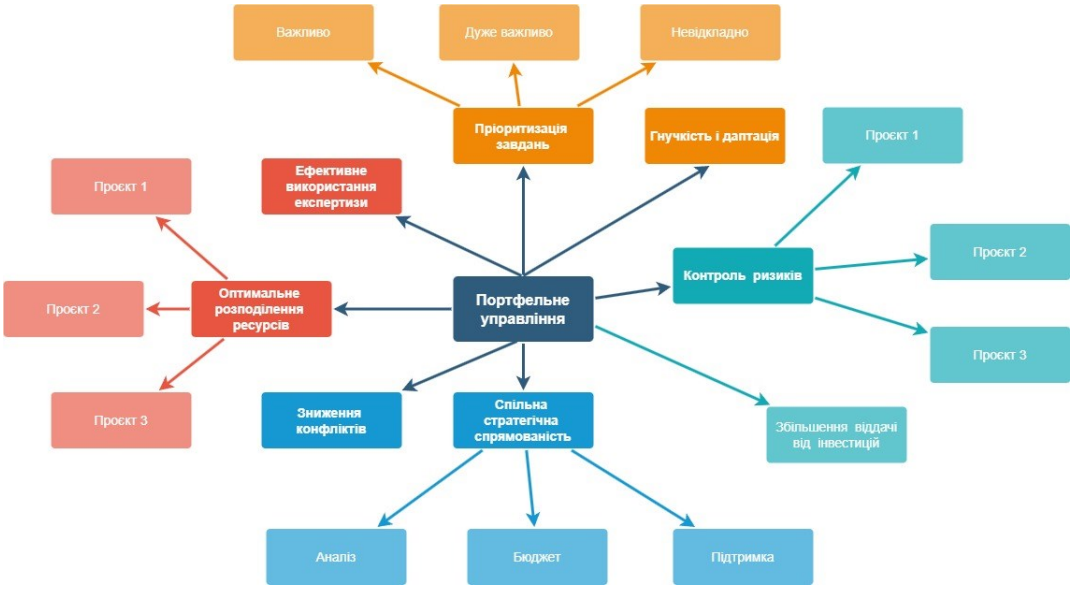


Рис. 4.1 – Портфельне управління як інструмент успішного виконання проєктів.

1. Оптимальне розподілення ресурсів. Портфельне управління дозволяє забезпечити раціональне розподілення обмежених ресурсів (бюджет, людські ресурси, обладнання) між різними проектами та ініціативами. Це допомагає уникнути перевантажень та забезпечити оптимальний виконання завдань.

2. Пріоритизація завдань. Портфельне управління допомагає визначити найважливіші та найефективніші проекти та завдання, які мають найбільший внесок у досягнення стратегічних цілей. Це дозволяє виконувати пріоритетні завдання першими.

3. Спільна стратегічна спрямованість. Портфельне управління допомагає забезпечити спільну спрямованість всієї команди або організації на досягнення загальних стратегічних цілей. Всі проекти та завдання відображають спільну стратегію.

4. Контроль ризиків. Методи портфельного управління допомагають ідентифікувати та управляти ризиками на рівні портфелю та окремих проектів. Це сприяє зменшенню можливих негативних наслідків та збільшенню шансів на успішне завершення.

5. Збільшення віддачі від інвестицій. Портфельне управління допомагає визначити проекти та ініціативи, які мають найбільший потенціал внеску до вартості організації. Це допомагає забезпечити віддачу від інвестицій у розробку та реалізацію проектів.

6. Гнучкість та адаптація. Портфельне управління надає можливість адаптуватися до змін в оточенні, змінювати пріоритети та ресурси відповідно до нових умов, що дозволяє організації бути більш гнучкою.

7. Ефективне використання експертизи. Використання методів портфельного управління допомагає враховувати експертні оцінки та думки різних фахівців при вирішенні питань розподілу ресурсів та виконання проектів.

8. Зниження конфліктів. Структуроване портфельне управління дозволяє уникнути конфліктів через чітке визначення пріоритетів, розподіл ресурсів та обґрунтування рішень.

В цілому, портфельне управління допомагає організаціям досягати кращих результатів, забезпечуючи ефективне використання ресурсів, пріоритизацію завдань та вирішення стратегічних завдань. Такий підхід сприяє зниженню дублювання ресурсів та зусиль у різних проектах, ефективному розподіленню людських ресурсів для досягнення стратегічних цілей організації, забезпеченню більшої видимості та контролю над всіма аспектами діяльності.

Портфельне управління організаційними ресурсами команд є потужним інструментом для забезпечення успішного виконання проектів та досягнення стратегічних цілей. Воно допомагає організаціям оптимізувати використання своїх ресурсів, уникати перевантажень та забезпечувати ефективну координацію між різними ініціативами.

Управління людськими ресурсами в команді включає в себе ряд процесів та завдань, спрямованих на ефективне використання потенціалу членів команди, забезпечення команди необхідними ресурсами, залучення та розвиток співробітників. Процедуру управління людськими ресурсами в команді можна подати кортежем

$$hr\_procedure = \{ P, R, N, T, K, O, S, ZT, PK, M, V\}, \quad (4.1)$$

де:

P -Визначення потреб команди - це перший і важливий етап управління людськими ресурсами. Цей процес включає в себе аналіз ідентифікації потреб та ресурсів команди для досягнення її цілей. В результаті визначення потреб команди формується база для подальшого управління людськими ресурсами та розвитку команди з урахуванням конкретних вимог і обставин;

R - рекрутинг та відбір кандидатів - це важливий етап формування команди;

N - оцінка навичок та досвіду кандидатів - це ключовий етап у процесі рекрутингу. Оцінка навичок та досвіду дозволяє забезпечити, що обрані кандидати відповідають потребам команди та можуть успішно виконувати покладені на них завдання;

T - тренінг та розвиток команди можуть включати в себе різноманітні дії та ініціативи для покращення навичок, знань та ефективності роботи команди;

K - управління конфліктами - це складний та важливий процес для забезпечення гармонійної роботи команди та досягнення спільних цілей;

O - оцінка та звітування про результати роботи є важливою частиною управління командою;

S - стимулювання та мотивація гравців у команді - це ключовий аспект успішного управління;

ZT - збереження талантів є важливим аспектом управління людськими ресурсами в команді;

PK - планування кар'єри та розвитку в команді — це важливий процес, який дозволяє працівникам розуміти їхні кар'єрні цілі та шляхи до їхнього досягнення;

M - моніторинг задоволеності працівників - це важливий аспект управління командою, що передбачає постійний аналіз настроїв працівників;

V - розв'язання проблем та викликів в команді - це процес, який передбачає формування стратегій та визначення відповідальних осіб.

Проаналізуємо кожну складову кортежу детальніше, оскільки на портфельному підході до управління людськими ресурсами ґрунтується робота рекомендаційної системи. *Визначення потреб команди* містить декілька кроків, які можуть бути включені в цей етап:

Крок 1. Аналіз цілей команди. Ретельний огляд цілей і завдань команди. Це може включати визначення стратегічних і тактичних цілей, а також основних завдань, які потрібно виконати для їх досягнення.

Крок 2. Оцінка поточних ресурсів. Аналіз наявних ресурсів команди, таких як людські, фінансові, технічні і інші. Це включає в себе експертну оцінку навичок, досвіду, знань і можливостей членів команди.

Крок 3. Визначення невиправданих прогалин. Виявлення областей, де є невиправдані прогалини або потреба в додаткових ресурсах. Це може бути пов'язано з відсутністю певних навичок у команді або обмеженими фінансовими ресурсами.

Крок 4. Прогнозування майбутніх потреб. Розглядання перспектив розвитку команди та визначення майбутніх потреб у навичках, ресурсах та інших аспектах.

Крок 5. Консультації з членами команди. Важливо враховувати думку та відгуки членів команди щодо їхніх потреб та очікувань. Це може включати індивідуальні розмови, опитування або групові зустрічі.

Крок 6. Створення стратегії управління ресурсами. Розробка плану управління ресурсами на основі виявлених потреб. Це може включати план розвитку персоналу, фінансові стратегії та інші аспекти.

Рекрутинг та відбір кандидатів відбувається за наступним методом:

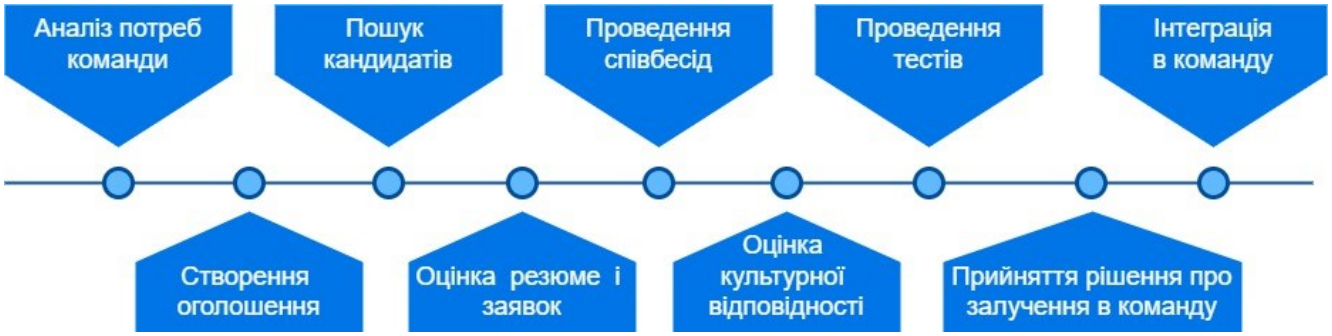


Рис. 4.2 – Процес рекрутингу

1. Аналіз потреб команди. Визначення конкретних потреб команди у нових членах. Це може включати в себе аналіз прогалин у навичках, необхідних для виконання завдань команди.

2. Створення оголошення. Розроблення оголошення, яке чітко визначає вимоги до кандидатів, описуючи роль, обов'язки, вимоги та очікування від потенційних членів команди.

3. Пошук кандидатів. Використання різних джерел для залучення потенційних кандидатів, таких як рекрутингові агентства, онлайн-платформи, соціальні мережі, внутрішні рекомендації та інші.

4. Оцінка резюме і заявок. Перегляд резюме та заявок від кандидатів для відбору тих, які відповідають вимогам та критеріям вакансії.

5. Проведення співбесід. Організація співбесід з відібраними кандидатами для оцінки їхніх навичок, досвіду, мотивації та взаємодії з командою.

6. Оцінка культурної відповідності. Врахування культурної відповідності кандидата до корпоративної культури та цінностей команди.

7. Проведення тестів. Використання тестів або практичних вправ для оцінки конкретних навичок та здатностей кандидата, які є важливими для роботи в команді.

8. Прийняття рішення про залучення в команду. Прийняття рішення щодо відбору та пропозиції роботи найкращим кандидатам.

9. Інтеграція в команду. Проведення процесу інтеграції для нового члена команди, включаючи ознайомлення із командною культурою, з колегами та інші аспекти.

Цей процес сприяє вибору кваліфікованих та мотивованих кандидатів, які відповідають потребам команди і сприяють досягненню її цілей.

Оцінка навичок та досвіду кандидатів відбувається за наступним алгоритмом:

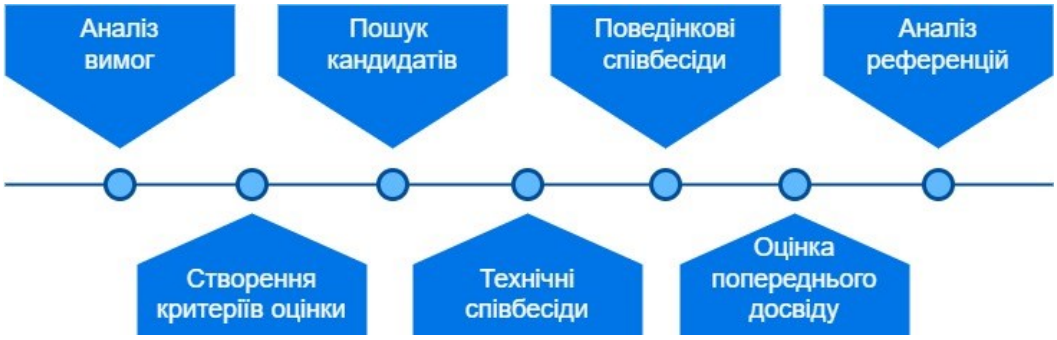


Рис. 4.3 – Оцінка навичок та досвіду кандидатів.

1. Аналіз вимог. Чітке визначення навичок та досвіду, які необхідні для успішного виконання робочих обов'язків в команді. Це може включати технічні навички, м'які навички, освіту та попередній досвід роботи.

2. Створення критеріїв оцінки. Розроблення конкретних критеріїв, за якими буде оцінюватися навички та досвід кандидатів. Це може бути здібність до розв'язання завдань, попередній досвід у схожих проектах, вміння працювати в команді тощо.

3. Збір інформації. Отримання інформації про навички та досвід кандидатів. Це може включати аналіз резюме, портфоліо, рекомендацій, а також відбіркові тести або завдання.

4. Технічні співбесіди. Проведення технічних співбесід або тестів для перевірки конкретних технічних навичок, необхідних для виконання роботи.

5. Поведінкові співбесіди. Проведення співбесід, що оцінюють м'які навички, такі як комунікація, співпраця, лідерство та рішення проблем.

6. Оцінка попереднього досвіду. Аналіз попередніх робочих досягнень кандидата, його ролі в попередніх проектах, успішність та внесок у попередні організації.

7. Аналіз референцій. Звертання до референцій від попередніх роботодавців чи колег для отримання додаткової інформації про досвід та професійні якості кандидата.

Тренінг та розвиток команди відбувається з врахуванням потреб команди, визначення областей, в яких необхідно покращення навичок або здобуття нових

Наступні елементи кортежу сприяють налагодженню роботи команди та характеризують м’які навички членів команди. Про них подамо лише коротку інформацію.

Управління конфліктами передбачає декілька кроків, серед яких *в*изначення причини конфлікту, визначення спільної мети або інтересів, які можуть об'єднати членів команди, визначення стратегії врегулювання конфліктів.

Оцінка роботи та звітування передбачає регулярну перевірку продуктивності команди на основі встановлених критеріїв, розроблення плану подальших дій для досягнення стратегічних цілей, спрямованих на поліпшення та розвиток.

Стимулювання та мотивація гравців у команді передбачає чітке розуміння членами команди, що внесок кожного члена команди впливає на загальний успіх, своєї важливості для успіху команди.

Наведені підходи можуть комбінуватися залежно від особливостей команди та її завдань. Ключовою є індивідуальна адаптація стратегій мотивації до потреб та особливостей кожного члена команди.

Збереження талантів охоплює стратегії та практики, спрямовані на утримання та розвиток найцінніших працівників, створення стимулюючого робочого середовища, надаються можливості для навчання та розвитку.

Планування кар'єри та розвитку команди передбачає визначення конкретних цілей, які працівник хоче досягти, створення можливостей для росту в межах команди.

Моніторинг задоволеності працівників допомагає визначити ключові проблеми або позитивних аспектів, які впливають на задоволеність працівників.

Розв'язання проблем та викликів в команді потребує ідентифікації проблеми, аналізу причин, формулювання стратегії розв'язання та запобігання проблемам.

Реалізація процедури управління людськими ресурсами проєкту потребує розроблення рекомендаційної системи з наступними вимогами до неї[86]:

На основі проведеного аналізу підходів до формування команд для IT проєктів можемо сформулювати *первинні вимоги* до рекомендаційної системи для підбору команди проєкту:

Збір інформації про відповідних фахівців. Система повинна знайти здібних та кваліфікованих фахівців для участі в проєкті відповідно до вимог і особливостей проєкту.

Аналіз навичок і досвіду. Система повинна оцінити навички, досвід і компетентності потенційних членів команди та рекомендувати осіб із відповідними навичками.

Врахування специфіки проєкту. Система повинна враховувати конкретні вимоги та специфіку проєкту, такі як технології, галузь, мови програмування тощо.

Врахування ролей. Система повинна давати можливість визначити ролі в команді, такі як розробник, тестувальник, дизайнер, проектний менеджер тощо.

Оцінка індивідуальних і командних навичок. Система може оцінювати як індивідуальні, так і командні навички потенційних членів команди.

Аналіз сумісності. Врахування сумісності між членами команди є важливим аспектом для успішного виконання проєкту.

Інтерфейс користувача. Система повинна мати інтуїтивний та зручний інтерфейс для користувачів, які здійснюють підбір команди.

Захист даних. Забезпечення безпеки та конфіденційності інформації про фахівців та проєкти є критичним аспектом.

Аналітика та звітність. Система повинна надавати аналітичні засоби для оцінки та відстеження ефективності підбору команди.

Скальованість. Система повинна бути здатною працювати як з невеликими, так і з великими командами та проєктами.

Інтеграція. Можливість інтеграції з іншими системами, такими як інструменти управління проєктами чи кадрові системи.

Підтримка. Наявність технічної підтримки та оновлень для системи.

Навчання користувачів. Надання навчальних матеріалів та підтримки для користувачів системи.

Другорядні вимоги до системи підбору команди проєкту можуть включати такі аспекти:

Мови та географічні вимоги. Здатність вибирати фахівців з певних мовами програмування або регіонів, якщо це важливо для проєкту.

Способи комунікації. Підтримка різних способів комунікації між членами команди, таких як чат, відеоконференції тощо.

Гнучкі фільтри. Можливість налаштовувати різноманітні фільтри для точного підбору фахівців.

Графічне відображення даних. Візуалізація інформації для зручності користувачів, наприклад, графіки та діаграми.

Можливість перегляду портфоліо. Відображення портфоліо робіт фахівців для оцінки їхньої попередньої роботи.

Підтримка мобільних пристроїв. Забезпечення роботи системи на різних типах мобільних пристроїв та планшетів.

Аналіз історії вибору. Зберігання історії попередніх виборів для аналізу та покращення рекомендацій.

Підтримка керування профілями. Можливість користувачів змінювати та оновлювати свої профілі.

Підтримка звітів. Звіти про ефективність та результати вибору команди для аналітики та покращення системи.

Підтримка оголошень та сповіщень. Можливість користувачів отримувати сповіщення та оголошення про можливі вибори чи зміни у команді.

Це другорядні вимоги можуть бути додатковими функціями, які покращують зручність користування системою і надають додатковий функціонал користувачам.

Для порівняння вимог до ролі в команді та компетентностей претендента можна використовувати наступні методи та інструменти:

Матриці порівняння. Створення матриці, де в одному стовпці перераховані вимоги до ролі, а в іншому — компетентності претендента. Визначення відповідності між ними за допомогою оцінок або символів.

SWOT-аналіз. Визначення сильних та слабких сторін претендента щодо вимог до ролі, а також можливостей та загроз. Порівняння отриманих даних з характеристиками ролі в команді.

Оціночні шкали. Створення шкал для оцінки рівня відповідності компетентностей претендента вимогам до ролі. Використання числових значень або категорій (наприклад, "повністю відповідає", "частково відповідає", "не відповідає").

Оцінювання за допомогою компетентнісних моделей допомагає визначити ключові навички та характеристики для конкретної ролі. Порівняння здібностей претендента з цими моделями.

Важливо враховувати специфіку ролі та бізнес-контексту для визначення тих аспектів, які є найбільш критичними. Комбінування різних методів допоможе створити повний та об'єктивний образ про відповідність.

Для порівняння вимог до ролі в команді та компетентностей претендента можна використовувати всі вище перераховані методи та інструменти.

Розглянемо приклад використання матриці порівняння для оцінки відповідності компетентностей претендента вимогам до ролі в команді.

Припустимо, у нас є роль "Проектний менеджер", і ми маємо вимоги до цієї ролі, такі як:

Лідерські навички

Організаційні здібності

Технічний досвід

Комунікативність

Тепер ми створимо матрицю порівняння, де рядки представляють вимоги, а стовпці — компетентності претендента. На основі цієї матриці ми можемо визначити, наскільки кожна компетентність претендента відповідає вимогам: *(додаток Г. Таблиця 4.1)*

У цьому прикладі оцінка від 1 до 5 вказує, наскільки претендент володіє певною компетентністю в контексті визначених вимог. Наприклад, якщо лідерські навички є ключовими для ролі "Проектний менеджер", то претендент у цьому прикладі має високий рівень лідерських навичок (оцінка 5).

Компетентнісні моделі використовуються для оцінювання та розвитку компетентностей працівників. Ось алгоритм оцінювання за допомогою компетентнісних моделей:

1.Визначення компетентностей. Спочатку потрібно визначити необхідні компетентності для конкретної ролі чи посади. Це може включати технічні навички, міжособистісні вміння, лідерські якості та інші аспекти.

2. Створення компетентнісної моделі. Розроблення чіткої компетентнісної моделі, яка визначає кожну компетентність та описує, як вона пов'язана з успішністю на роботі.

3.Оцінка компетентностей. Застосування інструментів для оцінки компетентностей працівника.

4. Збір обґрунтованих даних. Забезпечення обґрунтованих даних про рівень компетентностей. Це може включати регулярні оцінки, вивчення історії участі у попередніх проектах.

Такий підхід допомагає не лише оцінити компетентності працівників. Розглянемо приклад *створення компетентнісної моделі* для ролі "Менеджер проекту", спершу визначивши компетентності.

Лідерство. Здатність керувати та мотивувати команду, приймати стратегічні рішення.

Комунікації. Ефективне спілкування з різними стейкхолдерами, написання звітів та презентацій.

Організаційні навички. Здатність ефективно планувати та вирішувати завдання, управління ресурсами.

Технічні знання. Розуміння основних методологій управління проектами, використання інструментів для планування.

Далі переходимо до створення компетентісної моделі:

Лідерство.

Рівень 1. Здатність мотивувати команду.

Рівень 2. Здатність приймати рішення під час кризових ситуацій.

Комунікації.

Рівень 1. Ефективне спілкування внутрішньо в команді.

Рівень 2. Спроможність представляти проект перед клієнтами.

Організаційні навички.

Рівень 1. Здатність планувати щоденні завдання.

Рівень 2. Управління проектом з високим ризиком.

Технічні знання:

Рівень 1. Знання основних інструментів для управління проектами.

Рівень 2. Досвід використання конкретної методології (наприклад, Scrum або Kanban).

Оцінка компетентностей відбувається з використанням системи оцінювання (наприклад, шкала від 1 до 5) для оцінки працівників за кожною компетентністю. Цей приклад є загальним та може бути адаптований залежно від конкретних вимог та особливостей організації.

Проаналізуємо модель ефективної команди

$$UR=\{FK, OP, B, R, K\}, \quad (4.2)$$

FK - *формування команди*, яке передбачає вибір та розміщення правильних людей на правильних позиціях в команді. Важливо враховувати навички, досвід та характеристики кожного члена команди. Ця процедура є критичним завданням управління людськими ресурсами. Для цього використовуються різні методи та процедури. Ось деякі з них:

Оголошення вакансій та рекрутинг. Розміщення оголошень про вакансії та пошук кандидатів, які відповідають вимогам, через рекрутингові агентства або власний рекрутинговий відділ.

Інтерв'ю та співбесіди. Проведення інтерв'ю та співбесід з кандидатами для оцінки їхніх навичок, досвіду та підходу до роботи. Це може бути стандартні інтерв'ю або поведінкові співбесіди.

Оцінка компетентностей. Використання тестів або оцінки компетентностей для визначення, наскільки кандидати відповідають вимогам до ролі. Вибір методу залежить від конкретних потреб та особливостей організації, а також від ролі, на яку робиться вибір. Враховується, що правильний вибір та розміщення людей у команді сприяють досягненню кращих результатів та високої продуктивності.

Аналіз резюме та портфоліо: Перегляд резюме та портфоліо кандидатів для оцінки їхньої попередньої роботи та досягнень.

Референси. Перевірка рекомендацій від попередніх роботодавців або колег.

Оцінка культурного підходу. Визначення, наскільки кандидат вписується в корпоративну культуру та командний дух.

Оцінка м'яких навичок. Врахування м'яких навичок, таких як комунікаційні навички, креативність, лідерство тощо.

Внутрішні переміщення. Внутрішні переміщення працівників, коли вже працюючий співробітник переходить на іншу посаду в межах організації.

Рекомендації з розвитку умінь та навичок. Забезпечення навчання та навчальних можливостей для покращення професійних навичок членів команди. Це може включати навчання на певних освітніх рівнях закладів вищої освіти.

Управління людськими ресурсами в команді варіюється в залежності від розміру команди, її мети та галузі. Ефективне управління людськими ресурсами в команді допомагає досягати бажаних результатів і забезпечує високу продуктивність команди.

OP -оцінка потреб та пріоритетів, яка передбачає визначення поточних та перспективних потреби команди чи проектів, що входять до портфеля, встановлення пріоритетів щодо використання ресурсів для досягнення стратегічних цілей. Оцінка потреб та пріоритетів є важливим кроком в портфельному управлінні організаційними ресурсами команд. Цей процес допомагає визначити, які завдання, проекти чи ініціативи мають найвищий пріоритет і які ресурси слід виділити для їх виконання.

B - балансу, забезпечення якого між різними видами ресурсів та проектами, дозволяє уникнути перевантаження або нестатку ресурсів. Цей процес передбачає раціональне розподілення ресурсів між різними проектами, завданнями або ініціативами з метою досягнення оптимальних результатів та забезпечення ефективного використання ресурсів. Створення балансу у портфельному управлінні є важливим кроком для забезпечення ефективності та успішності реалізації проектів та завдань. Він дозволяє оптимально використовувати ресурси та досягати стратегічних цілей організації чи команди.

R - *визначення ролей і обов'язків* учасників процесу портфельного управління. Чітко визначається, хто має приймати рішення, хто має відповідати за виконання завдань, хто має надавати інформацію тощо.

K - *забезпечення відкритої комунікації* для своєчасного обміну інформацією між всіма учасниками з врахуванням різних каналів комунікації, таких як особиста зустріч, електронна пошта, відеоконференції тощо. Ефективна взаємодія та комунікація забезпечують згуртованість команди, зменшують ризики невірного розуміння та допомагають досягати спільних стратегічних цілей управління ресурсами.

4.2 Аналіз функціоналу рекомендаційної системи для формування команди

Для розроблення рекомендаційної системи обрана технологія з гібридним підходом, яка збирає дані про членів команди, такі як їх навички, досвід, освіта, історія командної взаємодії.

При відборі претендентів в команду, враховуючи специфіку проекту, розробляються критерії, які стають основою для аналізу компетенцій претендентів та їх ранжування. В основу процедури ранжування покладена експертна оцінка даних про претендента з присвоєнням певних вагових коефіцієнтів, які враховують вагу критерію та рівень знань і навичок, наявних у претендента (рис. 4.4).

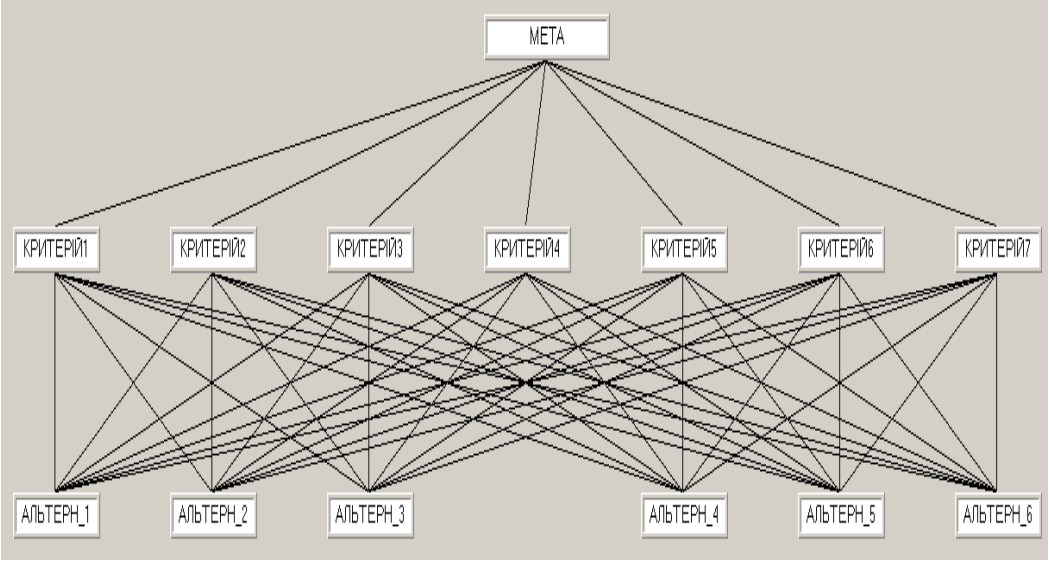


Рис. 4.4 – Загальна схема методу аналізу ієрархій, яка використана для аналізу переваг претендентів

Критерій 1 – працевлаштування. Враховується можливість участі заявника (п. 1, 2, ..., п) у роботі групи, тобто відсутність інших пропозицій чи проектів протягом періоду реалізації проекту.

Критерій 2 – досвід. Претендент у команду має необхідний досвід роботи для виконання певного переліку завдань.

Критерій 3 – це здатність. За цим критерієм оцінюється кваліфікація кандидата в команду.

Критерій 4 – знання. Враховується рівень знань про предметну область проекту, вимоги замовника до продукту проекту, середовище проекту.

Критерій 5 – навички. Формування командних навичок здобувача у використанні проектних інструментів, технологій реалізації проекту.

Критерій 6 – адаптивність. Це здатність кандидата в команду встановлювати стосунки та співпрацювати з іншими членами для створення згуртованої команди.

З використанням методу попарних порівнянь аналізуються переваги кожного з претендентів за певним критерієм. Приклад такого порівняння за критерієм кваліфікації наведено в табл. 4.2.

Таблиця 4.2 – Матриця попарних порівнянь для вибору претендента за критерієм “навички”

Альтер-нативи	**Претен-дент 1**	**Претен-дент 2**	**Претен-дент 3**	**Претен-дент 4**	**Претен-дент 5**	**Претен-дент 6**
Претен-дент 1	1,00	3,00	5,00	9,00	3,00	9,00
Претен-дент 2	0,33	1,00	3,00	7,00	3,00	7,00
Претен-дент 3	0,20	0,33	1,00	3,00	0,33	3,00
Претен-дент 4	0,11	0,14	0,33	1,00	0,33	1,00
Претен-дент 5	0,33	0,33	3,00	3,00	1,00	3,00
Претен-дент 6	0,11	0,14	0,33	1,00	0,33	1,00
Сума	**2,09**	**4,95**	**12,67**	**24,00**	**8,00**	**24,00**

Далі використовуючи алгоритм фільтрації за вмістом для аналізу характеристик кандидатів і порівняння їх з вимогами до ролей в команді.

Алгоритм фільтрації за вмістом ґрунтується на аналізі характеристики об'єкта (у цьому випадку, кандидата). Проводиться порівняння характеристик

кандидата з компетентностями, які мають бути притаманні претенденту для виконання певної ролі в команді та згенерувати рекомендації. Основні кроки методу фільтрації за вмістом у рекомендаційній системі:

1. Створення профілю ролі в команді. Для кожної ролі в команді створюється профіль, який включає в себе переліки компетентностей та вимог до претендента.
2. Створення профілю користувача. Користувач надає інформацію про свої вподобання, і ця інформація використовується для створення профілю користувача.
3. Проведення порівняння профілів. Система порівнює профіль ролі і профіль користувача, використовуючи метод обчислення схожості-косинусну схожість.

Цей метод особливо ефективний, оскільки у нас є чітко визначені характеристики об'єктів та достатньо інформації для побудови профілів. Розглянемо приклад використання косинусної схожості для порівняння профілю претендента і кваліфікаційних вимог до ролі в команді.

Припустимо, що у нас є два вектори, які представляють профіль претендента (A) і кваліфікаційні вимоги до ролі в команді (B). Кожен елемент цих векторів може відображати різні характеристики, такі як рівень освіти, робочий досвід, навички тощо.

Профіль претендента (A): [освіта, досвід, навички, комунікативність, творчість].

Кваліфікаційні вимоги (B): [освіта, досвід, навички, аналітичність, комунікативність]

Визначимо значення цих елементів. Наприклад, для спрощення можемо використовувати числові значення від 1 до 5, де 1 - найнижчий рівень, а 5 - найвищий рівень.

Профіль претендента (A): [4, 3, 5, 4, 4]

Кваліфікаційні вимоги (B): [5, 4, 5, 3, 4]

Тепер використаємо косинусну схожість для визначення міри схожості між цими векторами:

$$\text{Косинусна схожість } \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} AiBi}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \qquad (4.3)$$

де A • B - скалярний добуток векторів, ||A|| і ||B|| - їхні довжини (норми).

Скалярний добуток (A • B) = (4 * 5) + (3 * 4) + (5 * 5) + (4 * 3) + (4 * 4) = 20 + 12 + 25 + 12 + 16 = 85

Довжина вектора A (||A||) = √(4^2 + 3^2 + 5^2 + 4^2 + 4^2) = √(16 + 9 + 25 + 16 + 16) = √82

Довжина вектора B (||B||) = √(5^2 + 4^2 + 5^2 + 3^2 + 4^2) = √(25 + 16 + 25 + 9 + 16) = √91

Косинусна схожість = 85 / (√82 * √91) ≈ 0.926

Отже, косинусна схожість приблизно 0.926, що є високим показником схожості. Це може інтерпретуватися як те, що профіль претендента схожий на кваліфікаційні вимоги до ролі в команді.

Користувачі (кандидати), які мають високу схожість з профілем ролі в команді, рекомендуються для включення в команду. Система постійно оновлює рекомендації на основі нових даних про претендента та вимог до ролі в команді.

Проводячи фільтрацію на основі спільних дій (Collaborative Filtering) рекомендаційна система враховує взаємодію між членами команди в минулому та шукає подібних кандидатів на основі успішних командних взаємодій.

Розглянемо приклад реалізації фільтрації на основі спільних дій (Collaborative Filtering) для визначення подібних кандидатів на основі успішних командних взаємодій: *(додаток Г, лістинг 4.1).* Подібні кандидати для Кандидата 1: [2 3]

У цьому прикладі ми створюємо матрицю взаємодії між членами команди, де 1 вказує на те, що кандидати взаємодіють, а 0 - що не взаємодіють. Потім ми використовуємо косинусну схожість для визначення ступеня подібності між кандидатами. Функція find_similar_candidates визначає та повертає топ-N подібних кандидатів для заданого кандидата.

Цей підхід дозволяє знаходити кандидатів, які мають схожу історію взаємодії з командою, і тим самим можуть бути ефективними у новій команді. Використання моделей машинного навчання для прогнозування, які члени команди можуть найкраще взаємодіяти між собою для досягнення успіху.

Розглянемо приклад реалізації сценарію використання моделей машинного навчання для прогнозування ефективності взаємодії між членами команди: *(додаток Г, лістинг 4.2).* Точність моделі: 0.5. Прогноз взаємодії нового члена команди: 0

У цьому прикладі ми використовуємо модель Random Forest Classifier для класифікації успішної взаємодії членів команди. Після навчання моделі ми можемо використовувати її для прогнозування ефективності взаємодії нових членів команди на основі їхніх характеристик.

Рекомендаційна система використовує зважені коефіцієнти для регулювання ваги різних факторів, таких як навички, досвід, та історія взаємодії.

Розглянемо приклад реалізації сценарію з використанням зважених коефіцієнтів для регулювання ваги різних факторів, таких як навички, досвід та історія взаємодії:

Припустимо, що у нас є дані про членів команди, де кожен запис містить навички, досвід та історію взаємодії. Розглянемо приклад навички, досвід та історія взаємодії:

[5, 3, 8] -це вагові коефіцієнти навичок, досвіду, історія взаємодії для члена команди 1

[7, 5, 6] -це вагові коефіцієнти навичок, досвіду, історії взаємодії для члена команди 2

[4, 6, 7]-це вагові коефіцієнти навичок, досвіду, історія взаємодії для члена команди 3 і так далі

Провівши розрахунки отримуємо результат (успішність взаємодії, 1 – успішно, 0 – невдача) отримуємо результати для 10 команд ([1, 0, 1, 0, 1, 1, 0, 1, 1, 0]). Зважені коефіцієнти для навичок, досвіду та історії взаємодії

([0.4, 0.3, 0.3])

Проводимо розрахунки, стандартизуємо ознаки, створюємо та тренуємо модель (наприклад, Logistic Regression). Передбачаємо результати на тестових даних та оцінюємо точність моделі.

Отримуємо результат (успішність взаємодії, 1 - успішно, 0 - невдача). Застосовуємо зважені коефіцієнти до ознак. З використанням програми на Python, проведемо розрахунки. *(додаток Г, лістинг 4.3)*. Точність моделі: 0.0.

У цьому прикладі ми використовуємо зважені коефіцієнти для навичок, досвіду та історії взаємодії членів команди. Зважені коефіцієнти визначаються попередньо, і потім ми застосовуємо їх до відповідних ознак. Модель Logistic Regression використовує ці зважені ознаки для передбачення успішності взаємодії. Аналіз здатності претендента до командної роботи розраховується з використанням алгоритмів RandomForestClassifier та визначає як добре кандидат впишеться в команду.

Розглянемо приклад реалізації сценарію використання алгоритмів аналізу здатності до командної роботи для визначення, як добре кандидат впишеться в команду, проведемо розрахунки з використанням умовних даних про кандидатів та їх оцінки за атрибутами, які впливають на командну роботу. Оцінюємо точність моделі. *(дивись додаток Г, лістинг 4.4).* Точність моделі: 0.5

У цьому прикладі ми використовуємо алгоритм аналізу здатності до командної роботи, такий як Random Forest Classifier. Модель навчається на основі оцінок атрибутів кандидатів та їх впливу на командну роботу. Після навчання модель може передбачити вплив нових кандидатів на командну ефективність.

Система генерує персоналізовані рекомендації для кожного члена команди на основі аналізу його характеристик і взаємодії з іншими.

Для створення персоналізованих рекомендацій використовуємо підхід на основі алгоритмів машинного навчання. Проведемо розрахунки з використанням алгоритму рекомендацій для кожного члена команди із застосуванням навченої моделі для прогнозу взаємодії. Визначаємо точність моделі. *(додаток Г, лістинг 4.5)* Точність моделі: 1.0. Можливо, варто розглянути корекції для поліпшення взаємодії.

У цьому прикладі ми використовуємо класифікатор Random Forest Classifier для навчання моделі на основі характеристик членів команди та міток взаємодії (успішна чи неуспішна). Після навчання моделі ми можемо використовувати її для прогнозування взаємодії нового члена команди та генерації персоналізованих рекомендацій на основі прогнозу.

Система постійно моніторить ефективність роботи команди та оновлює рекомендації на основі нових даних та змін в умовах проекту.

Використання колаборативного підходу для генерування рекомендацій при підборі членів IT команди система аналізує профілі користувачів та групує їх на основі подібностей у навичках, досвіді та раніше виконаних проектах.

На основі схожості профілів система генерує рекомендації для створення оптимальної команди для конкретного проекту. Це включає в себе рекомендації щодо ролей, які потрібно включити, та підбору членів команди для забезпечення балансу у навичках. Система надає можливість користувачам налаштовувати ваги різних критеріїв для генерації рекомендацій, щоб враховувати унікальні вимоги конкретного проекту чи команди. Такий колаборативний підхід дозволяє створювати ефективні та збалансовані IT-команди на основі спільних навичок та досвіду її членів.

Запропонований гібридний підхід генерування рекомендацій дозволяє враховувати різноманітні фактори для ефективного формування команди та забезпечення її успішної взаємодії.

4.3 Аналіз функціоналу рекомендаційної системи

Для розроблення рекомендаційної системи було обрано об'єктно-орієнтовану мову програмування Java. Це надало розробленій рекомендаційній системи ряд переваг:

Використання об'єктно-орієнтованого підходу, що сприяє створенню легко розширюваної та зручної для обслуговування системи і дозволило реалізацію ансамблю алгоритмів та моделей.

Наявність широкого спектру бібліотек та фреймворків, спростило процедуру розроблення рекомендаційної системи, а використання бібліотеки Apache Mahout , яка містить інструменти машинного навчання забезпечило реалізацію рекомендаційних алгоритмів.

Java, маючи добру підтримку для роботи з реляційними базами даних, дозволила рекомендаційній системі коректно використовувати дані користувачів і об'єктів, які можна зручно зберігати та опрацьовувати у реляційних базах даних.

Платформонезалежність дозволяє виконання рекомендаційних процедур на різних платформах, що робить рекомендаційну систему універсальним інструментом, який може працювати на різних пристроях та серверах.

Послуговування мовою Java дозволимо використовувати різні інструменти для реалізації завдань машинного навчання, зокрема бібліотек, таких як Deeplearning4j або Weka для створення та навчання моделей рекомендацій.

Оскільки Java є потужним інструментом для розробки рекомендаційних систем, і вона часто використовується в індустрії для реалізації подібних проектів, саме тому вибір впав на неї.

Використання бібліотек Apache Mahout і Apache Spark MLlib для опрацювання даних і рекомендацій завдяки їхнім широким власним можливостям дозволяє: Генерувати рекомендації з використанням колаборативного та контентного підходів послуговуючись вбудованими алгоритмами рекомендацій.

Кластеризація та класифікація даних у рекомендаційній системі відбувається завдяки Mahout, яка містить алгоритми кластеризації і класифікації.

Масштабованість передбачає здатність до роботи з великими обсягами даних.

Apache Spark MLlib також має алгоритми рекомендацій, зокрема, Alternating Least Squares (ALS) для колаборативного фільтрування та пропонує широкий спектр алгоритмів машинного навчання для класифікації, регресії, кластеризації та інших завдань. Фреймворк Apache Spark забезпечує масштабованість та оптимізований для розподіленого обчислення, зокрема, може використовуватись для паралельного опрацювання даних на кластері.

Обидві бібліотеки можуть бути використані для створення систем рекомендацій в запропонованій рекомендаційній системі, що надає їй ряд переваг у порівнянні з іншими застосунками.

Загальний алгоритм рекомендаційної системи підбору команди IT проектів на основі гібридних алгоритмів фільтрації передбачає виконання ряду кроків:

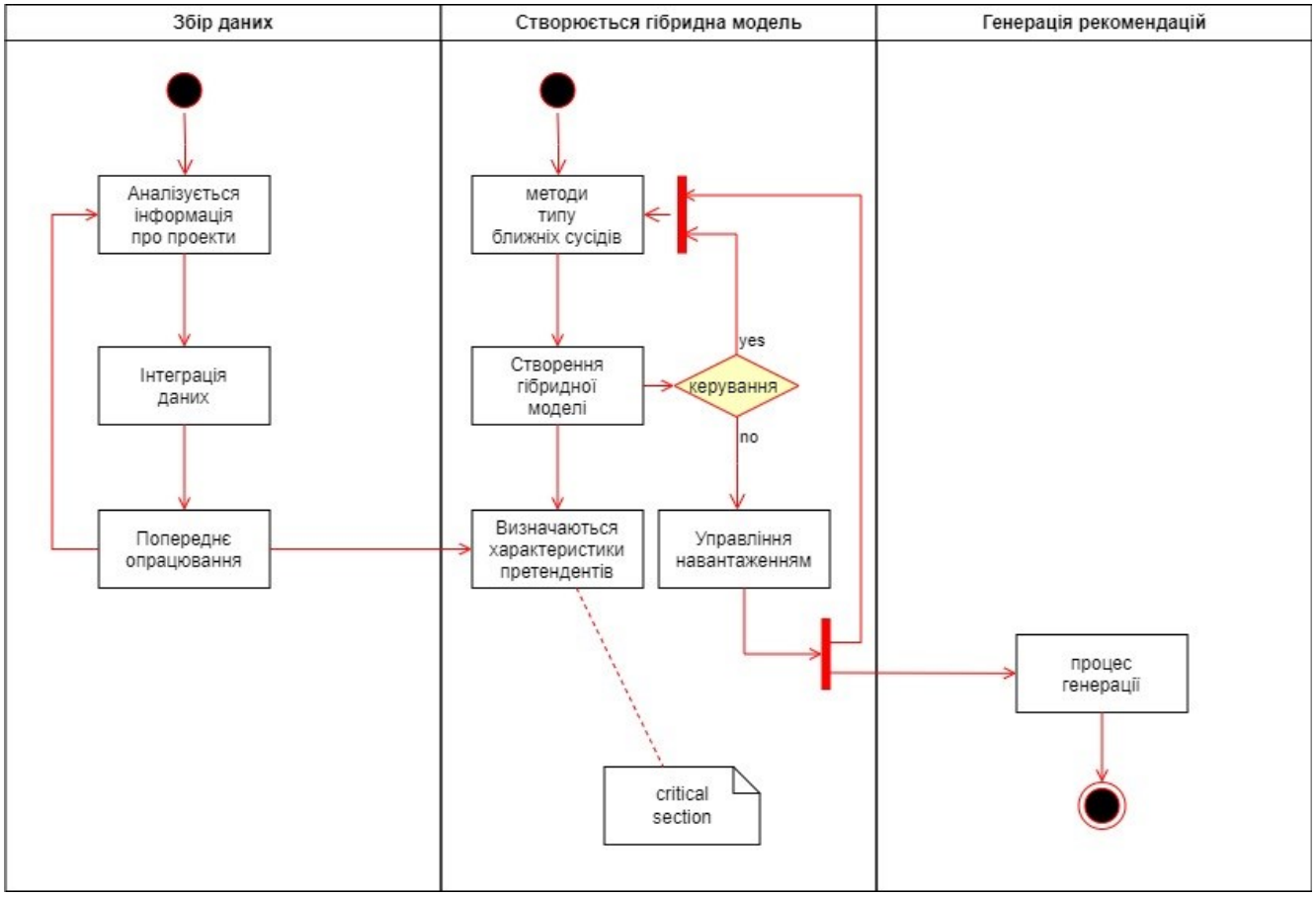


Рис. 4.5 – Алгоритм рекомендаційної системи підбору команди

1. Збір даних. Збираються дані про претендентів, їхні навички, а також вимоги до виконавців ролей розробників, тестувальників, менеджерів. Аналізується інформація про проекти, над якими працювали претенденти в минулому.
2. Інтеграція даних. Об'єднуються дані з різних джерел і структуруються для подальшого використання.
3. Проводиться попереднє опрацювання. Дані очищуються від надлишковості та доповнюються відсутніми значеннями. Визначається рівень важливості різних навичок та досвіду.

4. Визначаються характеристики претендентів і ролей. Визначається профіль кожного користувача, зокрема, його навички, досвід, області інтересів.
5. Створюється гібридна модель. Використовується комбінація алгоритмів фільтрації за вмістом, на основі спільних дій, колаборативний та контентний підходи для формування рекомендацій. Колаборативний підхід може використовувати методи типу "ближніх сусідів" для знаходження схожих користувачів або проектів. Контентний підхід може базуватися на аналізі ключових слів, які описують проекти та навички користувачів. Алгоритми фільтрації за вмістом використовує інформацію про характеристики претендента та кваліфікаційні вимоги до ролі в команді, генерування профілю користувача на основі його дій.
6. Проводиться аналіз текстового змісту об'єктів для визначення ключових слів, тегів або тематичних ознак, які описують контент.
7. Управління навантаженням. Ефективне керування обчислювальними ресурсами та оптимізація процесу рекомендацій для швидкого та ефективного використання.
8. Генерація рекомендацій. На основі гібридної моделі генеруються персоналізовані рекомендації для кожного користувача.

Цей алгоритм враховує індивідуальні характеристики користувачів та профілі ролей в команді та використовує гібридний підхід для покращення точності та релевантності рекомендацій.

У рекомендаційній системі розроблено 5 фільтрів:

country (UKRAINE, USA, EUROPE, ASIA, OTHER)

role (JAVA_DEV, PYTHON_DEV, JS_DEV, C_DEV, HTML_CSS_DEV, DEV_OPS, DESIGN, QA, SALES, PROJECT_MANAGER, BUSINESS_ANALYST)

experience (TRAINEE, JUNIOR, MIDDLE, SENIOR)

englishLevel (NO_ENGLISH, BEGINNER, PRE_INTERMEDIATE, INTERMEDIATE, UPPER_INTERMEDIATE, ADVANCED)

salary ExpectationFrom(Integer)

Реалізація фільтру «country» в контексті бази даних є частиною запиту. Це виглядає так:

SELECT * FROM users WHERE country = 'UKRAINE';

Фільтр role в базі даних може використовуватися для класифікації користувачів за їхніми ролями в IT-проектах. Кожен користувач може бути призначений на одну або декілька ролей з перелічених у фільтрі. Ось опис деяких можливих ролей:

JAVA_DEV - розробник, спеціалізується на мові програмування Java.

PYTHON_DEV - розробник, спеціалізується на мові програмування Python.

JS_DEV - розробник, спеціалізується на мові програмування JavaScript.

C_DEV - розробник, спеціалізується на мові програмування C/C++.

HTML_CSS_DEV - розробник, спеціалізується на розробці фронтенду з використанням HTML і CSS.

DEV_OPS - фахівець з операційної підтримки розробки (DevOps).

DESIGN - дизайнер, відповідає за створення дизайну інтерфейсу та користувацьких взаємодій.

QA - інженер з якості, відповідає за тестування і контроль якості продукту.

SALES - менеджер з продажу, відповідає за комерційну діяльність та залучення клієнтів.

PROJECT_MANAGER - керівник проекту, відповідає за організацію та контроль ходу проекту.

BUSINESS_ANALYST - бізнес-аналітик, відповідає за аналіз бізнес-вимог та взаємодію з клієнтом.

Цей фільтр допомагає класифікації користувачів за їхніми компетенціями та фаховістю в певних областях для подальшого використання в рекомендаційних алгоритмах.

Фільтр experience використовується для класифікації користувачів за їхнім рівнем досвіду в IT-галузі. Кожен користувач може мати один із чотирьох рівнів досвіду, визначених у фільтрі. Ось опис можливих рівнів досвіду:

TRAINEE - стажер або особа з мінімальним досвідом в IT.

JUNIOR - молодший розробник, зазвичай з невеликим досвідом роботи.

MIDDLE - середній рівень досвіду, розробник із середнім рівнем навичок та робочим досвідом.

SENIOR - високий рівень досвіду, досвідчений розробник або фахівець із значним стажем роботи в галузі.

Цей фільтр може бути використаний для визначення, наскільки кожен користувач досвідчений в своїй області, що може бути важливим критерієм для формування команд в IT-проектах. Розподіл користувачів за рівнем досвіду може також використовуватися в алгоритмах рекомендацій для забезпечення ефективної роботи команд.

Фільтр englishLevel в базі даних може використовуватися для класифікації користувачів чи команд за рівнем володіння англійською мовою. Кожен користувач чи учасник команди може бути визначений в одній із наступних категорій:

NO_ENGLISH: Відсутність знань англійської мови.

BEGINNER: Початковий рівень володіння англійською мовою.

PRE_INTERMEDIATE: Рівень до середнього (підготовка до середнього рівня).

INTERMEDIATE: Середній рівень володіння англійською мовою.

UPPER_INTERMEDIATE: Високий середній рівень володіння англійською мовою.

ADVANCED: Високий рівень володіння англійською мовою.

Цей фільтр може бути використаний для підбору команди або учасників проєктів з врахуванням рівня знань англійської мови, що може бути важливим у співпраці на міжнародних проєктах, комунікації з клієнтами та інших сценаріях.

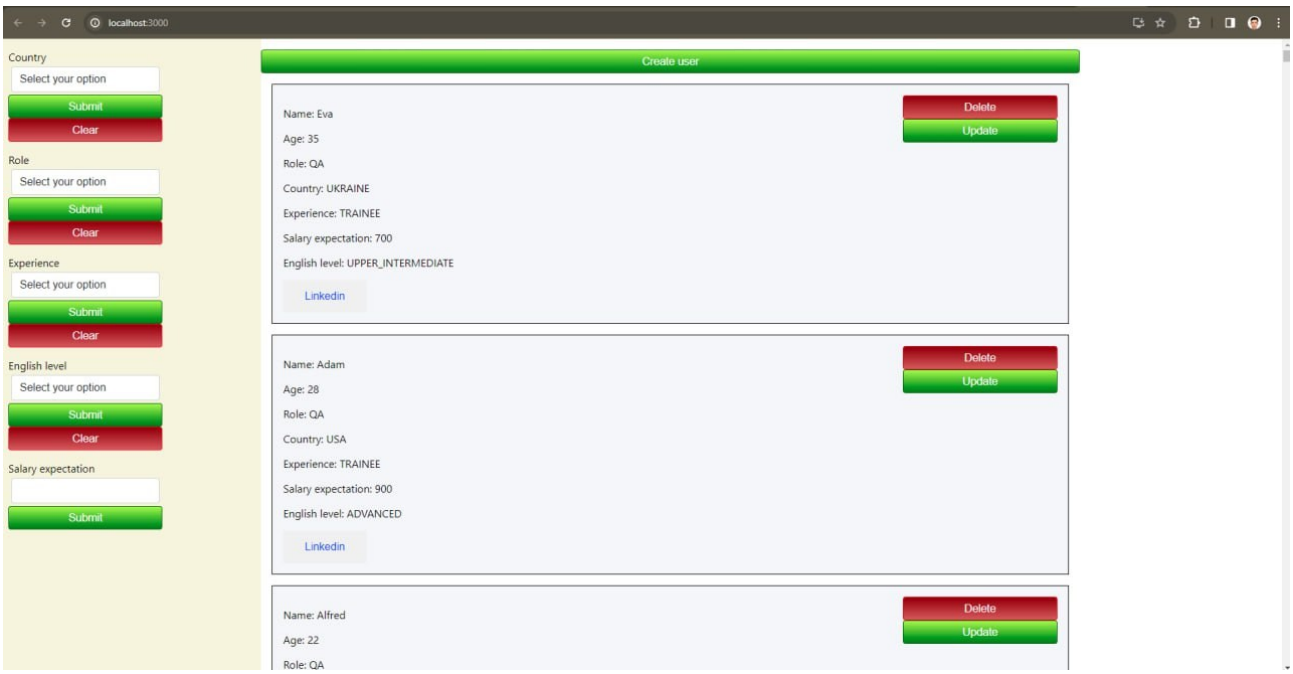


Рис. 4.6 – Створення профілю фахівця

Таким чином рекомендаційна система дозволяє виконувати функції «створити профіль працівника», «редагувати профіль працівника», «видалити профіль працівника», «переглянути профіль конкретного працівника», «переглянути профілі всіх працівників з можливістю фільтрування».

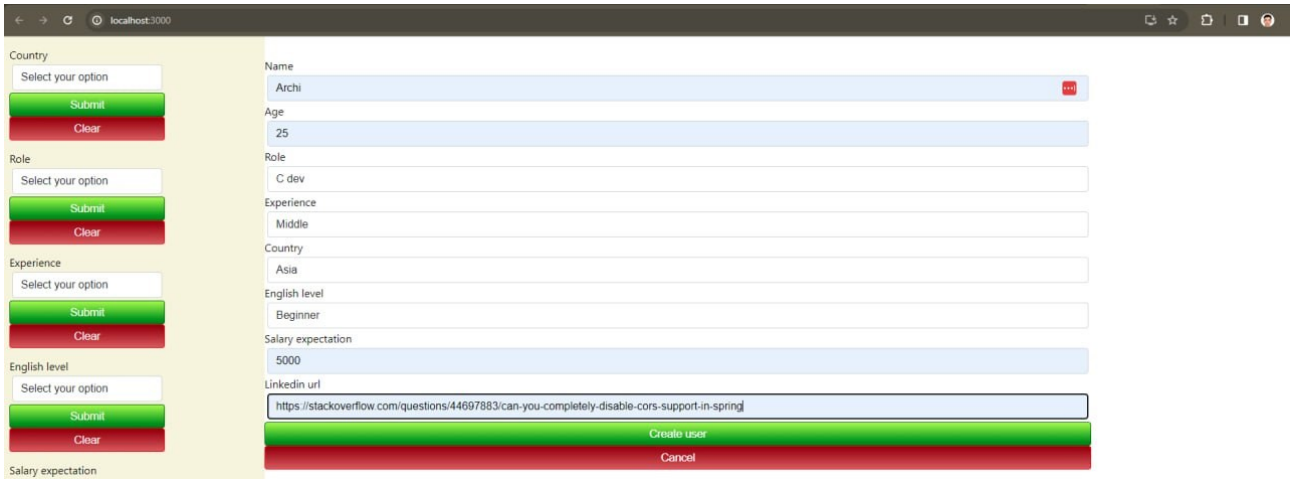


Рис. 4.7 – Автоматичне заповнення інформаційних полів

У рекомендаційній системі розроблено скрипт, який дозволяє автоматично заповнюватиме базу даних із профілем працівниками. Після стартування аплікації, тобто після запуску програми можемо переглянути профілі працівників. Поля в об'єкті працівник є обов'язковими для заповнення.

id": 6,
name": "Julia",
age": 101,
role": "QA",
country": "UKRAINE",
experience": "JUNIOR",
salaryExpectation": 500,
englishLevel": "BEGINNER",
linkToCVorLinkedIn": "link"

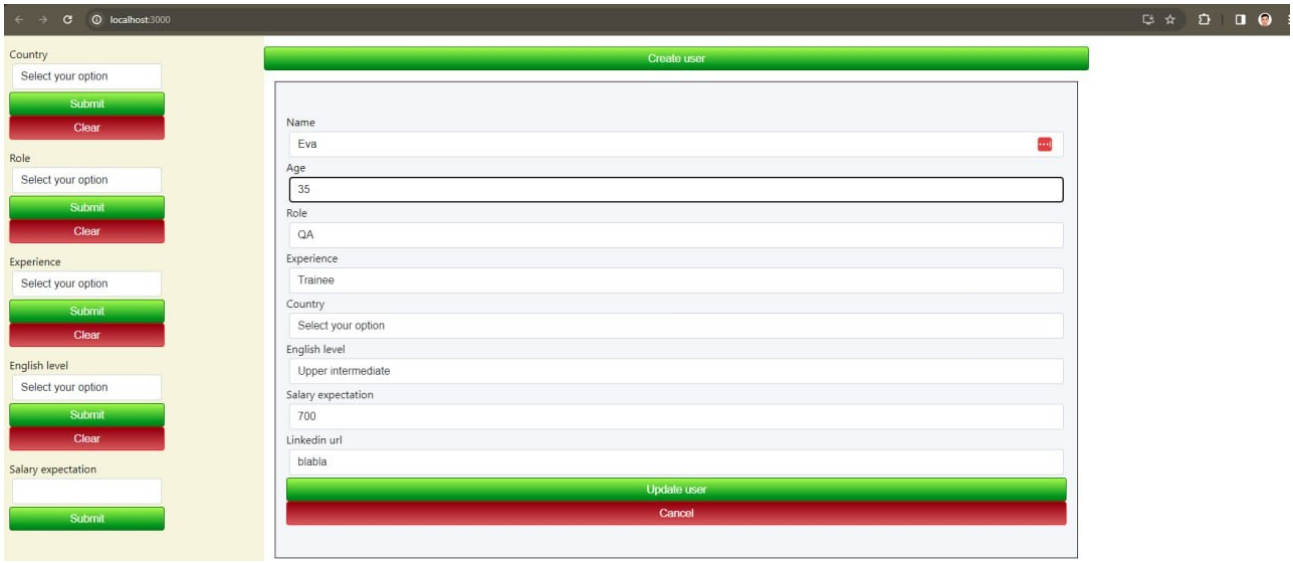


Рис. 4.8 – Коригування потрібної інформації

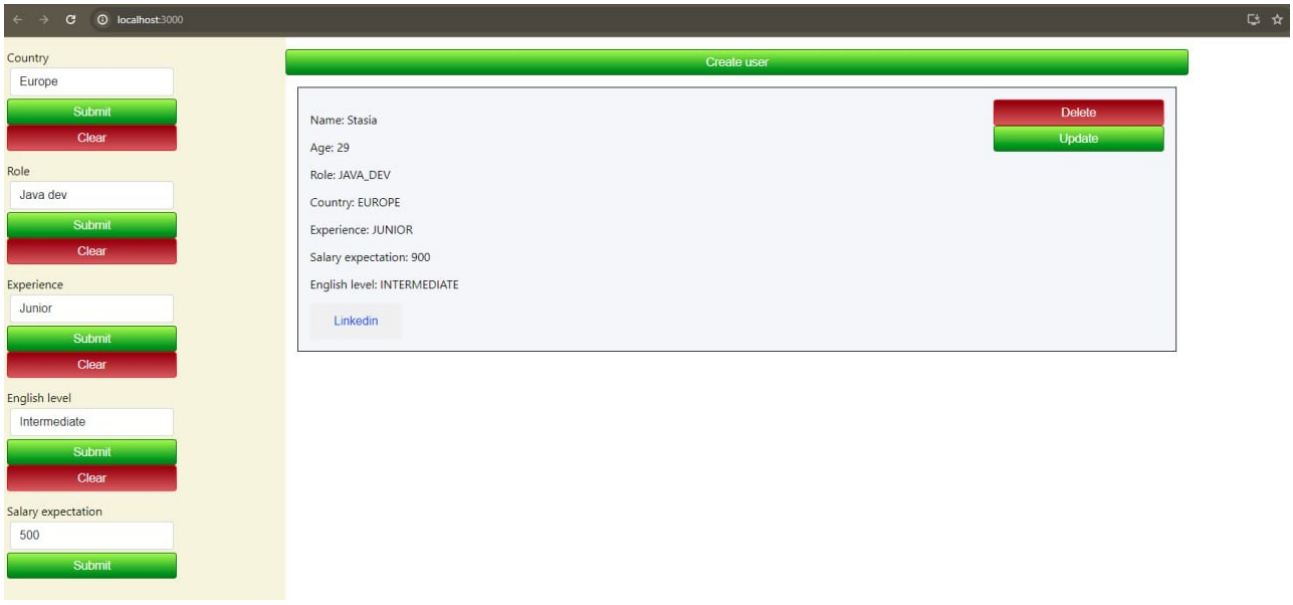


Рис. 4.9 – Перегляд відкоригованого профілю

Використання гібридного методу генерування рекомендацій, до складу якого входить метод спільної фільтрації потребує аналізу великих наборів даних.

У рекомендаційній системі по формуванню команди IT проекту для збагачення або вдосконалення процесу рекомендацій використаний ChangeLog, який фіксує всі зміни, внесені до системи протягом певного часу. Функції ChangeLog включають:

Фіксація змін. Відбувається реєстрація всіх змін, які вносяться в код програми, документацію, базу даних чи інші компоненти системи. Це може включати виправлення помилок, нові функції, модифікації інтерфейсу тощо.

Відстеження версій. ChangeLog фіксує інформацію про версії програмного продукту або проекту, зазначаючи, які конкретні зміни внесені в кожну версію.

Дата та час змін. Зафіксовуючи точний час і дату внесення змін, ChangeLog дозволяє відслідковувати, коли вони були впроваджені. Це корисно для відстеження історії розвитку продукту.

Опис змін. Кожна зміна зазвичай супроводжується коротким описом, який пояснює, що саме було змінено або виправлено. Це допомагає розробникам і айчарам розуміти призначення змін.

Визначення відповідального. У ChangeLog може бути вказано, хто відповідав за кожну конкретну зміну. Це сприяє відповідальності та може бути корисним при подальшій комунікації між учасниками проекту.

Взаємодія з командою. ChangeLog може використовуватися для комунікації з усією командою розробників, тестувальників та інших учасників проекту. Він допомагає всім залишатися в курсі останніх подій і змін.

Структурованість і організація. ChangeLog може бути структурованим документом зі зручним форматуванням, яке спрощує читання та розуміння змін.

Автоматизація. Процес збору даних для ChangeLog може бути автоматизованим, наприклад, за допомогою систем контролю версій.

Визначення компетенцій розробників. Інформація з ChangeLog може допомогти визначити, над якими конкретними завданнями працювали розробники.

Наприклад, якщо розробник часто вносить зміни в частину коду, пов'язану з веб-розробкою, це може свідчити про його високу компетенцію в цьому напрямку.

Оцінка рівня досвіду. ChangeLog може містити інформацію про те, скільки часу розробник витрачає на різні завдання та як часто він вносить зміни. Це може служити основою для оцінки рівня досвіду розробника.

Пошук експертів з певних областей. Інформація про внесені зміни може вказувати на галузі, в яких розробник може бути експертом. Рекомендаційна система може використовувати ці дані для пошуку експертів з певних технологій або напрямків.

Оптимізація командної роботи. Аналіз ChangeLog може допомогти визначити, які розробники ефективно співпрацюють між собою. Рекомендаційна система може враховувати ці взаємодії при формуванні команди для нового проєкту.

Прогнозування інтересів розробників. Зміни в коді можуть свідчити про інтереси розробника. Наприклад, якщо він регулярно вносить зміни в області штучного інтелекту, це може свідчити про його інтерес до цієї теми.

Інтеграція ChangeLog у рекомендаційну систему поліпшує якість рекомендацій і забезпечує створення більш адаптованих та ефективних команд для IT проектів.

Використання MySQL для зберігання даних про користувачів, їх навички та досвід у розробленій рекомендаційній системі формування команд для IT проектів надало їй кілька переваг:

Простота використання, оскільки MySQL відомий своєю простотою використання та налаштування. Це легко встановлюється і використовується, що полегшує розробку та обслуговування системи.

Доступність, так як MySQL є відкритим програмним забезпеченням і доступним для безкоштовного використання. Це робить його привабливим вибором для стартапів та проєктів з обмеженим бюджетом.

Ефективність. MySQL добре оптимізований для використання в невеликих та середніх проектах. Він може забезпечити ефективне зберігання та витяг даних, якщо обсяги не є надто великими.

Підтримка ACID. MySQL забезпечує ACID-властивості (Atomicity, Consistency, Isolation, Durability), що робить його надійним для зберігання важливих даних та транзакцій.

Широкі можливості інтеграції. MySQL може легко інтегруватися з багатьма іншими інструментами та мовами програмування, що полегшує розробку системи рекомендацій.

Спільнота та документація. MySQL має широку спільноту користувачів і обширну документацію, що полегшує вирішення проблем та отримання підтримки.

Для розроблення фронтенту рекомендаційної системи використано інструмент для дизайну та прототипування Figma, і його використання для розробки фронтенду рекомендаційної системи для формування команди IT проекту надало системі деякі переваги:

Figma дозволив створити прототип з інтерактивністю. Це важливо для розробки фронтенду рекомендаційної системи, оскільки можна створювати прототипи та взаємодіючі компоненти.

Figma був використаний для розробки дизайну на різних етапах проекту, починаючи від прототипування та завершуючи детальним дизайном. Це полегшило масштабованість роботи.

Figma дозволив швидко створювати та редагувати дизайн, що було корисним під час ітераційного розроблення фронтенду рекомендаційної системи.

Figma забезпечує можливість зберігання та документації дизайну, включаючи компоненти, кольори та стилі, що сприяло консистентності та ефективному управлінню дизайном.

Висновки до 4 розділу

Проаналізовано доцільність використання портфельного управління людськими ресурсами команд, визначено які аспекти такого управління можна реалізувати з використанням рекомендаційної системи.

Запропоновано використовувати метод аналізу ієрархій та експертного оцінювання для побудови ієрархії претендентів в команду.

Запропоновано концептуальну модель процедури управління людськими ресурсами в команді, яка покладена в основу розробленої рекомендаційної системи. Визначено особливості кожної складової моделі.

На основі проведеного аналізу підходів до формування команд для IT проєктів сформульовано первинні та вторинні вимоги до рекомендаційної системи для підбору команди проєкту.

Проаналізовано засоби побудови рекомендаційної системи, її функціонал та обґрунтовано переваги.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальне наукове завдання, яке полягало у розробленні методів і засобів реалізації процедур формування команд для успішного виконання ІТ проектів. При цьому отримано такі основні результати:

1. Розроблено комплекс формалізмів, який покладено в основу процесів створення концептуальних моделей цілевизначального та рольового підходів, ізоморфної, експертної, колегіальної, розробницької, проектної, аналітичної, інтеграційної, інноваційної структур команди, що дозволило спростити процедури побудови ефективних команд виконавців ІТ проектів, які формуються на платформі рекомендаційної системи;

2. Створено поведінкову модель команди проєкту як сукупний рух імітованої зграї, подаючи взаємодії членів команди за допомогою ройового алгоритму, що дозволило розробити метод формування команди з використанням агентного підходу, що забезпечило проведення серії симуляційних процесів для формування ефективних команд для проєктів;

3. Побудовано модель ізоморфної структури команди з використанням формалізмів теорії графів, що дозволило візуалізувати процеси побудови команд проєктів та формування оптимальної структури команди виконавців проєктів;

4. Розроблено метод формування команди з використанням генетичного алгоритму та оптимального розподілу ролей у команді з використанням нейронних мереж, що дозволило здійснювати пошук рішення щодо оптимального складу команди, базуючись на певній популяції претендентів;

5. Запропоновано архітектуру рекомендаційної системи відбору претендентів з заданою системою компетентностей для створення ефективної команди розробників;

6. Розроблено інформаційну технологію відбору претендентів до складу команди з певними компетентностями для успішного виконання ІТ проекту.

7. Побудовано рекомендаційну систему відбору претендентів в команду з певними компетентностями, що використовує гібридний метод генерування рекомендацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] PMBOK® Guide . Seventh Edition Member Pennsylvania, 2021.-370 s.

[2] Nota, L., Santilli, S., Soresi, S.: A life design based online career intervention for early adolescents: description and initial analysis. Career Dev. Q. 64(1), 4–19 (2016)

[3] Charles, E., Wang, Y.: Social influence in career choice: evidence from a randomized field experiment on entrepreneurial mentorship. Res. Policy 46(3), 636–650 (2017)

[4] Mann, I.: Hacking the Human: Social Engineering Techniques and Security Countermeasures. Routledge (2017) 8. Ceschi, A.: The career decision-making competence: a new construct for the career realm. Eur. J. Train. Dev. 41(1), 8–27 (2017)

[5] Умови, чинники і критерії успішної реалізації проекту. Критерії успішності проекту. Критерії успіхів і невдач в управлінні проектами. URL: https://prowines.ru/uk/business-ideas/usloviya-faktory-i-kriterii-uspeshnoi-realizaciiproekta-kriterii.html (дата звернення: 10.11.2022)

[6] Онишкевич О.В. Актуальність проектного підходу в управлінні підприємствами Економіка і суспільство 2016 Випуск 6 С.203-207

[7] Черчата А. О. Проектний менеджмент на підприємстві: застосування в контексті взаємодії з функціональним та процесним підходами Науковий вісник ІФНТУНГ. Серія: Економіка та управління в нафтовій і газовій промисловості. 2019. № 1 (19). С.172-179.

[8] Надія Миколаївна САПИЧ, Карина Олександрівна ХАМЛИКА, Проєктний менеджмент: теорія та практика застосування Шлях успіху і перспективи розвитку (до 26 річниці заснування Харківського національного університету внутрішніх справ). Харків, 2020. С.450-453

[9] Верба В.А. Гармонізація процесного і проектного підходів до управління розвитком компанії / В.А. Верба / / Управління проектами та розвиток виробництва: Зб.наук.пр. - Луганськ: вид-во СНУ ім. В.Даля, 2009. - № 3 (31). - С. 14-22. - Режим доступу: http://www.pmdp.org.ua/images/Journal/31/09vvaurk.pdf

[10] Бушуєва Н. С., Хрутьба В.О., Філатов А.С. Модель формування крос - функціональної команди для управління соціальними проектами в швидкозростаючій організації Управління проектами, системний аналіз і логістика. К.: НТУ, 2015. Вип. 15. С.25-35.

[11] Project Management Institute, A Guide to the Project Management Body of Knowledge – Fifth Edition, Project Management Institute Inc., 2013, Page 2.

[12] Лєбєдєва І. Ю. Методика формування команди проекту. URL: https://www.sworld.com.ua/simpoz8/103.pdf (дата звернення: 10.11.2022).

[13] Данченко О. Б., Занора В. О. Проектний менеджмент: управління ризиками та змінами в процесах прийняття управлінських рішень: монографія / О. Б. Данченко, В. О. Занора. – Черкаси: ПП Чабаненко Ю.А., 2019. – 278 с.

[14] Шерстюк О. І., Тесленко П. О. Аналіз компетенцій команди проекту при її взаємодії із зацікавленими сторонами. Управління проектами у розвитку суспільства: матеріали XVI міжнародної конференції. Київ: КНУБА, 2019. С. 248 – 249.

[15] Лєбєдєва І.Ю. Методика формування команди Режим доступу: https://www.sworld.com.ua › simpoz8

[16] Медведєва, О.М. Корпоративна культура та культурний контекст проекту розвитку організації. Частина 3. Модель представлення культурного контексту проекту в компонента корпоративної культури Управління проектами та розвиток виробництва: Зб. наук. праць. – Луганськ: Східноукр. нац. ун-т ім. В.Даля, 2009. - №1(29). – С.17-27.

[17] The Standard for Program Management Джорджії: Project Management Institute, 2017.-176s.

[18] Батенко Л. П. Цінність проекту з позицій різних зацікавлених сторін Ефективна економіка № 9, 2013

[19] Сеек, А. М., Тесленко, П. О., & Бєлова, О. І. (2019). Ціннісні-орієнтований підхід управління проектами як джерело виникнення ризиків. Вчені записки Університету «КРОК», ((3) 55), 128–133. https://doi.org/10.31732/2663-2209-2019-55-128-133

[20] Медведєва, О.М. Механізм управління взаємодією в проектах Управління розвитком складних систем. Зб. наук. праць. – К.: КНУБА, 2012. – Вип. 12. – С. 65-74

[21] Управление проектами: Основы профессиональных знаний и система оценки компетентности проектних менеджеров (National Competence Base Line, NCB UA Version 3.1) [Текст]/ С.Д. Бушуев, Н.С. Бушуева; изд. 2-е. - К.: ІРІДІУМ, 2010. – 208 с.

[22] Медведєва, О.М. Моделювання комунікації в проектах на основі теорії нечітких множин Управління проектами у розвитку суспільства: Управління програмами організаційного розвитку в конкурентному середовищі: тез. доп. III між. конф. 24-25 травня 2007 р. - К.: КНУБА, 2007. - С. 87-89

[23] Бушуев С. Д. Формальная модель ментального простору проєкту чи програми / С. Д. Бушуев, Е. В. Веренич, Д. А. Бушуев, Р. Ф. Ярошенко // Радіоелектроніка, інформатика, управління. – 2017. – № 1.– С. 153-160.

[24] О. І. Бєлова, Інноваційна активність персоналу та способи її стимулювання на підприємстві, Вчені записки Університету «КРОК»: № 3 (51) (2018)

[25] Куценко М.Н. Створення цінності проектів на основі системи управління знаннями / М.Н. Куценко // Управління розвитком складних систем. Зб. наук. праць. – К.: КНУБА, 2012. – Вип. 9. - С. 36-39

[26] Aubry M. Organizational design in public administration: typology of project management offices. Paper presented at Project Management Institute Research and Education Conference / M. Aubry, M. Brunet. – Phoenix, AZ. Newtown Square, PA: Project Management Institute, 2014. – Retrieved from: https://www.pmi.org/learning/library/organizational-design-public-administration-8941

[27] О. І. Бєлова, Ю. А. Поскрипко, Мотиваційний механізм управління проєктами: стратегічний аспект, Вчені записки Університету «КРОК»: № 2 (58) (2020)

[28] Бушуєв, С.Д. Динамічне лідерство в управлінні проектами [Текст]: Монографія / С.Д. Бушуєв, В.В. Морозов. – К.: Українська асоціація управління проектами, 1999. – 312 с.

[29] S.G. Fisher, T.A. Hunter & W.D.K. MacRosson (2001) A validation study of Belbin's team roles, European Journal of Work and Organizational Psychology, 10:2, 121-144, DOI: 10.1080/13594320143000591

[30] Newton John Organisational role analysis, London, 2013, 22 p.

[31] Ярошенко, Ф. Проактивне антикризове управління програмами розвитку за умов поширення фінансової кризи: Матеріали майстер-класу в Україні, січень 2012 [Текст]/ Ф.Ярошенко, Х.Танака, С. Бушуєв – К., 2012. – 130с.

[32] R. C K and K. C. Srikantaiah, "Similarity Based Collaborative Filtering Model for Movie Recommendation Systems," 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2021, pp. 1143-1147, doi: 10.1109/ICICCS51141.2021.9432354.

[33] Yi, S.-r.; Song, J. Particle Filter Based Monitoring and Prediction of Spatiotemporal Corrosion Using Successive Measurements of Structural Responses. Sensors 2018, 18, 3909. https://doi.org/10.3390/s18113909

[34] Dominik Kowald and Emanuel Laci Popularity Bias in Collaborative Filtering-Based Multimedia Recommender Systems International Workshop on Algorithmic Bias in Search and Recommendation,2022, DOI:10.48550/arXiv.2203.00376

[35] Chhavi Sharma, Punam Bedi, Sabu M. Thampi, and El-Sayed M. El-Alfy. 2017. CCFRS – Community based Collaborative Filtering Recommender System. J. Intell. Fuzzy Syst. 32, 4 (2017), 2987–2995. https://doi.org/10.3233/JIFS-169242

[36] Frans Prathama, Wenny Franciska Senjaya, Bernardo Nugroho Yahya, Jei-Zheng Wu,Personalized recommendation by matrix co-factorization with multiple implicit feedback on pairwise comparison, Computers & Industrial Engineering,Volume 152, 2021, 107033, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2020.107033.

[37] Li, J., Fu, A. & Zhang, L. An Overview of Scoring Functions Used for Protein–Ligand Interactions in Molecular Docking. Interdiscip Sci Comput Life Sci 11, 320–328 (2019). https://doi.org/10.1007/s12539-019-00327-w

[38] Prelec, Drazen. “The Probability Weighting Function.” Econometrica, vol. 66, no. 3, 1998, pp. 497–527. JSTOR, https://doi.org/10.2307/2998573. Accessed 6 Dec. 2023.

[39] Shangsong Li and Xuesong Li (2020) Collaborative Filtering Recommendation Algorithm Based on User Characteristics and User Interests: J. Phys.: Conf. Ser. 1616 012032

[40] Sarker, I.H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. SN COMPUT. SCI. 2, 420 (2021). https://doi.org/10.1007/s42979-021-00815-1

[41] Ouaknine A. (2018) «Review of Deep Learning Algorithms for Image Semantic Segmentation» Medium, Regime of access: https://medium.com/@arthur_ouaknine/review-of-deep-learning-algorithmsfor-image-semantic-segmentation-509a600f7b57/

[42] Bottou, L. (2012). Stochastic Gradient Descent Tricks. In: Montavon, G., Orr, G.B., Müller, KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_25

[43] de Boer, PT., Kroese, D.P., Mannor, S. et al. A Tutorial on the Cross-Entropy Method. Ann Oper Res 134, 19–67 (2005). https://doi.org/10.1007/s10479-005-5724-z

[44] Burke, R. Hybrid Recommender Systems: Survey and Experiments. User Model User-Adap Inter 12, 331–370 (2002). https://doi.org/10.1023/A:1021240730564

[45] Pankiv Y., Kunanets N., Artemenko O., Veretennikova N. and Nebesnyi R. Project of an Intelligent Recommender System for Parking Vehicles in Smart Cities. *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, LVIV, Ukraine, 2021, pp. 419-422, doi: 10.1109/CSIT52700.2021.9648687 (Index Scopus)

[46] Кунанець Н. Е., Небесний Р. М., Мацюк О. В. Особливості формування цілей соціальних та соціокомунікаційних складових у проектах "Розумних міст" *Вісник Національного університету "Львівська політехніка". Серія : Інформаційні системи та мережі*. 2016. Випю 854. С. 257-274. - Режим доступу: http://nbuv.gov.ua/UJRN/VNULPICM_2016_854_26 (Index Copernicus)

[47] Caragliu, Andrea; DEL BO, Chiara; Nijkamp, Peter. Smart cities in Europe.Journal of urban technology, 2011, 18.2: 65-82

[48] Лавриненко, Л. М. Інноваційний розвиток трудового потенціалу в сучасних умовах. Сталий розвиток економіки, 2014, 1: 18-25.

[49] Jäger, Andreas, et al. "Industry 4.0: challenges for the human factor in future production scenarios." (2015).

[50] Knox, Paul L., and Sallie A. Marston. Human geography: Places and regions in global context. Pearson, 2014.

[51] Khan, Zaheer, et al. Towards cloud based big data analytics for smart future cities. Journal of Cloud Computing, 2015, 4.1: 1-11.

[52] Табачишин, Д. Р., Ленько, В. С., Кунанець, Н. Е., Пасічник, В. В., & Щербина, Ю. М. (2017). Експертне оцінювання" розумності міста" із застосуванням нечіткої логіки. Штучний інтелект..

[53] Циганок, В. В. "Агрегація групових експертних оцінок, що отримані у різних шкалах." Реєстрація, зберігання і обробка даних (2011).

[54] Lazaroiu, George Cristian, and Mariacristina Roscia. "Definition methodology for the smart cities model." Energy 47.1 (2012): 326-332.

[55] Claude Trigano 2016 What is the "Smart Human City?"

[56] Бабенцова, Орина Сергіївна, et al. "СУЧАСНІ ТЕНДЕНЦІЇ РОЗВИТКУ МІСЬКИХ ПРОСТОРІВ." The 6th International scientific and practical conference “Multidisciplinary scientific notes. Theory, history and practice”(November 01–04, 2022) Edmonton, Canada. International Science Group. 2022. 712 p.. 2022.

[57] Кунанець Н. Небесний Р. Людський ресурс “розумного міста” та відкриті дані *Матеріали V науково-технічної конференції „Інформаційні моделі, системи та технології“*, 1-2 лютого 2018 року. Т. : ТНТУ, 2018. С. 41–42. (Секція 2. Інформаційні системи).

[58] Кунанець Н., Кунанець О., Небесний Р., Пасічник В. Освітня соціокомунікаційна складова у портфелях проектів «Розумних міст»: досвід Великобританії. *Proceedings of the tenth international scientific-practical*

conference «Internet-Education-Science» (IES-2016), Vinnytsia, 11-14 October, 2016. Vinnytsia : VNTU, 2016. С. 192-194.

[59] Kunanets Natalia, Pasichnyk Volodymyr, Nebesnyi Ruslan, Nazaruk Mariia Аналіз вибору IT спеціальностей учнями випускних класів на прикладі м. Тернополя *Вісник Національного університету" Львівська політехніка". Інформаційні системи та мережі* 2019. №6. С. 79 - 89 https://doi.org/10.23939/sisn2019.02.079 (Index Copernicus)

[60] Український центр оцінювання якості освіти. Статистичні дані основної сесії ЗНО. Отримано з https://zno.testportal.com.ua/opendata.

[61] https://open-school.uspishnemisto.com.ua

[62] Kunanets N. E., Nazaruk M. V., Nebesnyi R. M., and Pasichnyk V. V. Information technology of personalized choice of profession in smart cities, *ITLT*, vol. 65, no. 3, pp. 277–290, Jul. 2018. https://doi.org/10.33407/itlt.v65i3.2172 (Web of Science (ESCI), USA)

[63] Демографічна та соціальна статистика // Державна служба статистики України. URL: http://www.ukrstat.gov.ua/ (дата звернення: 15.11.2018)

[64] HeadHunter (hh.ua).-Режим доступу: https://www.otzyvua.net/headhunter-hhua

[65] Купар Д.М. Освіта як цінність у контексті міграційних процесів в Україні // Міжнародний науковий вісник. Випуск 2 (18). Ужгород, 2018.-С.38-46.

[66] Судакова Н., Мегединюк М. Чому диплом не допомагає знайти роботу? // STUDWAY. 20.05.2016. URL: https://studway.com.ua/diplom-nedopomagaie/ (дата звернення: 11.11.2018)

[67] Бюджет МОН на 2021 рік: майже 140 млрд грн для розвитку освіти і науки.-Режим доступу: https://mon.gov.ua/ua/news/byudzhet-mon-na-2021-rik-majzhe-140-mlrd-grn-dlya-rozvitku-osviti-i-nauki

[68] Дуда О.М., Кунанець Н.Е., Липак Г.І., Мацюк О.В., Небесний Р.М., Пасічник В.В. Консолідація інформаційних ресурсів соціокомунікаційного середовища в проектах “Розумне місто” *System Analysis and Information Technologies 18-th International Conference SAIT 2016* Kyiv, Ukraine, May 30 – June 2, 2016 p.214 ISBN 978-966-2748-83-3

[69] Лебідь О. Ю., Побудова когнітивної моделі для аналізу діяльності електронних магазинів. Електронне наукове фахове видання "Ефективна економіка". № 11, 2015

[70] Лебідь О. Ю. Деякі аспекти застосування когнітивного моделювання в державному управлінні. Електронне наукове фахове видання "Державне управління: удосконалення та розвиток" № 11, 2015

[71] Горелова Г. В. Когнитивный анализ, синтез, прогнозирование развития больших систем в интеллектуальных РИУС. / Г. В. Горелова, Э. В. Мельник, Я. С. Коровин // Штучний інтелект.-.-2010.-№3.-С.61-72.

[72] Pasichnyk V., Nazaruk M., Kunanets N., Veretennikova N. and Nebesnyi R. Information Analysis of Procedures for Choosing a Future Specialty Using Cognitive Cards. *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 2018, pp. 215-220, https://doi.org/10.1109/STC-CSIT.2018.8526626 (Index Scopus).

[73] Романенко В. Д. Обеспечение устойчивости импульсных процессов в когнитивных картах на основе моделей в пространстве состояний / В. Д. Романенко, Ю. Л. Милявский //System research & information technologies.-2014.- № 1.- С.26-42

[74] Кунанець Н., Пасічник В., Небесний Р. Інформаційні технології прогнозування розвитку освітнього середовища «розумного міста» *Proceedings of the tenth international scientific-practical conference «Internet-Education-Science» (IES-2016)*. Vinnytsia, 11-14 October, 2016. Vinnytsia : VNTU, 2016. С. 188-189.

[75] Matsyuk, O., Nazaruk, M., Turbal, Y., Veretennikova, N., Nebesnyi, R. Information Analysis of Procedures for Choosing a Future Specialty. *Advances in Intelligent Systems and Computing III. CSIT* 2018. vol 871. Springer, Cham. https://doi.org/10.1007/978-3-030-01069-0_26 (Index Scopus)

[76] Pasichnyk , V., Kunanets , N., Artemenko , O., Fedorka , P., & Nebesnyi , R. Using mobile crowd sensing for social distancing real-time navigation. *Управління розвитком складних систем.* 2021. №47. С. 57–62. https://doi.org/10.32347/2412-9933.2021.47.57-62 (Index Copernicus)

[77] S. Petersen, “The role of enterprise modeling in virtual enterprises”, Collaborative Networks and Their Breeding Environments: IFIP TC5 WG 5.5 Sixth IFIP Working Conference on VIRTUAL ENTERPRISES (26–28 September, 2005, Valencia, Spain), pp. 109–117.

[78] Nebesnyi R., Kunanets N., Vaskiv R. and Veretennikova N. Formation of an IT Project Team in the Context of PMBOK Requirements *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, LVIV, Ukraine, 2021, pp. 431-436, https://doi.org/10.1109/CSIT52700.2021.9648612. (Index Scopus)

[79] Nebesnyi R., Pasichnyk V., Kunanets N., Veretennikova N. and Kunanets O. Formation of IT Project Implementation Team. *2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT)*, Zbarazh, Ukraine, 2020, pp. 203-206, https://doi.org/10.1109/CSIT49958.2020.9322005. (Index Scopus)

[80] Калинич Ю., Білак Ю.Ю., Небесний Р., Федорка П. Аналіз процесів формування симуляцій з використанням графічного процесора *Вісник Національного університету" Львівська політехніка". Інформаційні системи та мережі*. 2022. №11. С.110-126 https://doi.org/10.23939/sisn2022.11.110 (Index Copernicus)

[81] Reynolds. C. W. (1982). Computer Animation with Scripts and Actors, Computer Graphics, 16 (3), (acm SIGGRAPH `82 Proceedings), 289-296.

[82] Reynolds, C.W. (1987). Flocks, herds and schools: A distributed behavioral model. SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. Association for Computing Machinery, 25–34. DOI:10.1145/37401.37406

[83] Gille, W. (2020). Particle and Particle Systems Characterization. Small-Angle Scattering (SAS) Applications. CRC Press.

[84] Reeves, W. (1983) Particle Systems-A Technique for Modeling a Class of Fuzzy Objects, ACM Transactions on Graphics, 2, 91-108

[85] Shaw, E. (1979) Fish in Schools, Natural History, 84 (8), 4046

[86] Мартинюк С. В., Небесний Р. М. Розробка функціонуючої структури програмного консолідованого ресурсу *Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“*, 16-17 листопада 2017 року. Т. : ТНТУ, 2017. Том 2. С. 112–113. (Комп’ютерно-інформаційні технології та системи зв’язку).

ДОДАТКИ

ДОДАТОК А

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ ТА ВІДОМОСТІ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Kunanets N. E., Nazaruk M. V., Nebesnyi R. M., and Pasichnyk V. V. Information technology of personalized choice of profession in smart cities, *ITLT*, vol. 65, no. 3, pp. 277–290, Jul. 2018. https://doi.org/10.33407/itlt.v65i3.2172 (Web of Science (ESCI), USA)
2. Kunanets Natalia, Pasichnyk Volodymyr, Nebesnyi Ruslan, Nazaruk Mariia Аналіз вибору IT спеціальностей учнями випускних класів на прикладі м. Тернополя *Вісник Національного університету" Львівська політехніка". Інформаційні системи та мережі* 2019. №6. С. 79 - 89 https://doi.org/10.23939/sisn2019.02.079 (Index Copernicus)
3. Pasichnyk , V., Kunanets , N., Artemenko , O., Fedorka , P., & Nebesnyi , R. Using mobile crowd sensing for social distancing real-time navigation. *Управління розвитком складних систем.* 2021. №47. С. 57–62. https://doi.org/10.32347/2412-9933.2021.47.57-62 (Index Copernicus)
4. Калинич Ю., Білак Ю.Ю., Небесний Р., Федорка П. Аналіз процесів формування симуляцій з використанням графічного процесора *Вісник Національного університету" Львівська політехніка". Інформаційні системи та мережі*. 2022. №11. С.110-126 https://doi.org/10.23939/sisn2022.11.110 (Index Copernicus)
5. Кунанець Н. Е., Небесний Р. М., Мацюк О. В. Особливості формування цілей соціальних та соціокомунікаційних складових у проектах "Розумних міст" *Вісник Національного університету "Львівська політехніка". Серія : Інформаційні системи та мережі*. 2016. Випю 854. С. 257-274. - Режим доступу: http://nbuv.gov.ua/UJRN/VNULPICM_2016_854_26 (Index Copernicus)

Наукові праці, які засвідчують апробацію матеріалів дисертації:

1. Nebesnyi R., Kunanets N., Vaskiv R. and Veretennikova N. Formation of an IT Project Team in the Context of PMBOK Requirements *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, LVIV, Ukraine, 2021, pp. 431-436, https://doi.org/10.1109/CSIT52700.2021.9648612. (Index Scopus)
2. Nebesnyi R., Pasichnyk V., Kunanets N., Veretennikova N. and Kunanets O. Formation of IT Project Implementation Team. *2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT)*, Zbarazh, Ukraine, 2020, pp. 203-206, https://doi.org/10.1109/CSIT49958.2020.9322005. (Index Scopus)
3. Matsyuk, O., Nazaruk, M., Turbal, Y., Veretennikova, N., Nebesnyi, R. Information Analysis of Procedures for Choosing a Future Specialty. *Advances in Intelligent Systems and Computing III. CSIT* 2018. vol 871. Springer, Cham. https://doi.org/10.1007/978-3-030-01069-0_26 (Index Scopus)
4. Pasichnyk V., Nazaruk M., Kunanets N., Veretennikova N. and Nebesnyi R. Information Analysis of Procedures for Choosing a Future Specialty Using Cognitive Cards. *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 2018, pp. 215-220, https://doi.org/10.1109/STC-CSIT.2018.8526626 (Index Scopus)
5. Pankiv Y., Kunanets N., Artemenko O., Veretennikova N. and Nebesnyi R. Project of an Intelligent Recommender System for Parking Vehicles in Smart Cities. *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, LVIV, Ukraine, 2021, pp. 419-422, doi: 10.1109/CSIT52700.2021.9648687 (Index Scopus)
6. Дуда О.М., Кунанець Н.Е., Липак Г.І., Мацюк О.В., Небесний Р.М., Пасічник В.В. Консолідація інформаційних ресурсів соціокомунікаційного середовища в проектах “Розумне місто” *System Analysis and Information*

Technologies 18-th International Conference SAIT 2016 Kyiv, Ukraine, May 30 – June 2, 2016 p.214 ISBN 978-966-2748-83-3

7. Кунанець Н. Небесний Р. Людський ресурс “розумного міста” та відкриті дані *Матеріали V науково-технічної конференції „Інформаційні моделі, системи та технології“*, 1-2 лютого 2018 року. Т. : ТНТУ, 2018. С. 41–42. (Секція 2. Інформаційні системи).
8. Кунанець Н.Е., Небесний Р.М., Мацюк О.В., Пасічник В.В. Соціокомунікаційна складова у портфелях проектів «Розумних міст». *Управління проектами: стан та перспективи : матеріали XII Міжнародної науково-практичної конференції*. Миколаїв : НУК, 13-16 вересня 2016, ст. 84-85 https://nuos.edu.ua/wp-content/uploads/2021/12/Konf_Upravlenie_Proekt.pdf
9. Кунанець Н., Кунанець О., Небесний Р., Пасічник В. Освітня соціокомунікаційна складова у портфелях проектів «Розумних міст»: досвід Великобританії. *Proceedings of the tenth international scientific-practical conference «Internet-Education-Science» (IES-2016)*, Vinnytsia, 11-14 October, 2016. Vinnytsia : VNTU, 2016. С. 192-194.
10. Кунанець Н., Пасічник В., Небесний Р. Інформаційні технології прогнозування розвитку освітнього середовища «розумного міста» *Proceedings of the tenth international scientific-practical conference «Internet-Education-Science» (IES-2016)*. Vinnytsia, 11-14 October, 2016. Vinnytsia : VNTU, 2016. С. 188-189.
11. Мартинюк С. В., Небесний Р. М. Розробка функціонуючої структури програмного консолідованого ресурсу *Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“*, 16-17 листопада 2017 року. Т. : ТНТУ, 2017. Том 2. С. 112–113. (Комп’ютерно-інформаційні технології та системи зв’язку).

Наукові праці, які додатково відображають наукові результати дисертації:

1. Кормило І. В., Небесний Р. М. Побудова інформаційної технології візуалізації інформаційних ресурсів *Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“*, 16-17 листопада 2017 року. Т. : ТНТУ, 2017. Том 2. С. 96–97. (Комп’ютерно-інформаційні технології та системи зв’язку).

Додаток Б
Додатки з 2 розділу дисертації

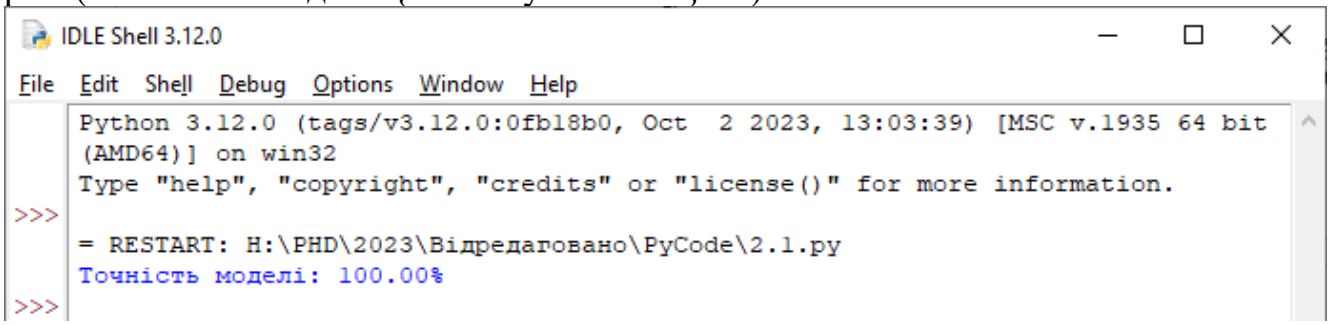
Таблиця 2.3 – Відповідність назвам навчальних закладів їх номерів в діаграмах.

01	Тернопільська Українська гімназія ім. І. Франка Тернопільської міської ради Тернопільської області.
02	Тернопільський навчально-виховний комплекс "Загальноосвітня школа І-ІІІ ступенів-правовий ліцей № 2".
03	Тернопільська спеціалізована школа І-ІІІ ступенів № 3 з поглибленим вивченням іноземних мов Тернопільської міської ради Тернопільської області.
04	Тернопільська загальноосвітня школа І-ІІІ ступенів № 4 Тернопільської міської ради Тернопільської області.
05	Тернопільська спеціалізована школа І-ІІІ ступенів № 5 з поглибленим вивченням іноземних мов Тернопільської міської ради Тернопільської області.
06	Тернопільський навчально-виховний комплекс "Школа-ліцей № 6 ім. Н. Яремчука".
07	Тернопільська спеціалізована школа І-ІІІ ступенів № 7 з поглибленим вивченням іноземних мов Тернопільської міської ради Тернопільської області.
08	Тернопільська загальноосвітня школа І-ІІІ ступенів № 8.
09	Тернопільський навчально-виховний комплекс "Загальноосвітня школа І-ІІІ ступенів-економічний ліцей № 9 імені Іванни Блажкевич".
10	Тернопільська загальноосвітня школа І-ІІІ ступенів № 10 Тернопільської міської ради Тернопільської області.
11	Тернопільська загальноосвітня школа І-ІІІ ступенів № 11 Тернопільської міської ради Тернопільської області.
12	Тернопільський навчально-виховний комплекс "Школа-колегіум Патріарха Йосифа Сліпого" Тернопільської міської ради Тернопільської області.
13	Тернопільська загальноосвітня школа І-ІІІ ступенів № 13 імені Андрія Юркевич Тернопільської міської ради Тернопільської області.
14	Тернопільська загальноосвітня школа І-ІІІ ступенів № 14 ім. Б. Лепкого тернопільської міської ради тернопільської області.
15	Тернопільський навчально-виховний комплекс "Загальноосвітня школа І-ІІІ ступенів-медичний ліцей № 15" Тернопільської міської ради Тернопільської області.

16	Тернопільська загальноосвітня школа І-ІІІ ступенів № 16 імені Володимира Левицького Тернопільської міської ради Тернопільської області.
17	Тернопільська спеціалізована школа І-ІІІ ступенів № 17 імені Володимира Вихруща з поглибленим вивченням іноземних мов Тернопільської міської Ради Тернопільської області.
18	Тернопільська загальноосвітня школа І-ІІІ ступенів № 18.
19	Тернопільська загальноосвітня школа І-ІІІ ступенів № 19 Тернопільської міської ради Тернопільської області.
20	Тернопільська загальноосвітня школа І-ІІІ ступенів № 20 Тернопільської міської ради.
21	Тернопільська загальноосвітня школа І-ІІІ ступенів № 21 Тернопільської міської ради Тернопільської області.
22	Тернопільська загальноосвітня школа І-ІІІ ступенів № 22 Тернопільської міської ради Тернопільської області.
23	Тернопільська загальноосвітня школа І-ІІІ ступенів № 23 Тернопільської області Тернопільської міської ради.
24	Тернопільська загальноосвітня школа І-ІІІ ступенів № 24 Тернопільської міської ради Тернопільської області.
25	Тернопільська загальноосвітня школа І-ІІІ ступенів № 25 Тернопільської міської ради Тернопільської області.
26	Тернопільська загальноосвітня школа І-ІІІ ступенів № 26 Тернопільської міської ради Тернопільської області.
27	Тернопільська загальноосвітня школа І-ІІІ ступенів № 27 імені Віктора Гурняка Тернопільської міської ради.
28	Тернопільська загальноосвітня школа І-ІІІ ступенів № 28 Тернопільської міської ради Тернопільської області.
29	Тернопільська спеціалізована школа І-ІІІ ступенів № 29 з поглибленим вивченням іноземних мов Тернопільської міської ради Тернопільської області.
30	Тернопільська класична гімназія Тернопільської міської ради Тернопільської області.
31	Тернопільський технічний ліцей Тернопільської міської ради Тернопільської області.
32	Тернопільська вечірня школа Тернопільської міської ради Тернопільської області.
33	Тернопільський педагогічний ліцей спортивного профілю Тернопільської міської ради Тернопільської області.
34	Заклад освіти І-ІІІ ступенів "Тернопільський обласний навчально-реабілітаційний центр".
35	Тернопільська обласна експериментальна комплексна школа мистецтв імені Ігоря Герети.

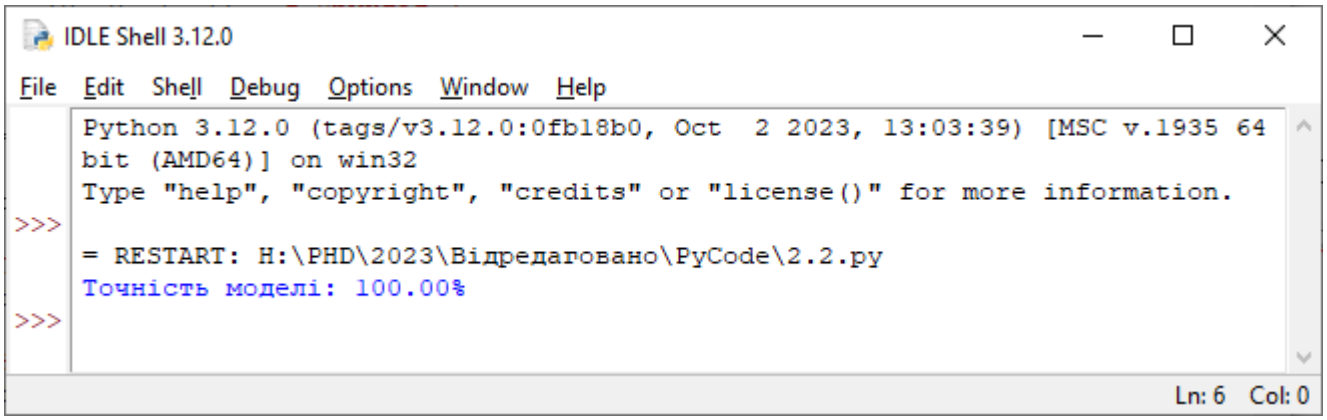
Лістинг 2.1 – лістинг коду з використанням бібліотеки scikit-learn для Python, яка надає інструменти для машинного навчання

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Приклад даних
# Представимо, що у нас є дані про відповіді абітурієнтів на тестові питання та їх вибірка спеціальності (0 - не зацікавлені в спеціальності IT, 1 - зацікавлені в IT).

# Приклади відповідей абітурієнтів (тестові питання)
X = [[0, 1, 1, 0],  # Приклад 1
     [1, 1, 0, 0],  # Приклад 2
     [0, 0, 1, 1],  # Приклад 3
     [1, 0, 1, 0],  # Приклад 4
     [0, 1, 0, 1]]  # Приклад 5

# Відповідний вибір спеціальності (0 - не зацікавлені в IT, 1 - зацікавлені в IT)
y = [0, 1, 0, 1, 0]

# Розділення даних на тренувальну та тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Побудова моделі дерева рішень
model = DecisionTreeClassifier()

# Навчання моделі на тренувальних даних
model.fit(X_train, y_train)

# Прогнозування результатів на тестових даних
y_pred = model.predict(X_test)

# Оцінка точності моделі
accuracy = accuracy_score(y_test, y_pred)
print(f"Точність моделі: {accuracy * 100:.2f}%")
```



Лістинг 2.2 – лістинг прикладу побудови моделі з використанням методу опорних векторів (SVM) для визначення нахилів абітурієнтів до спеціальностей IT

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Приклад даних (використаємо ті ж дані як у попередньому прикладі)
X = [[0, 1, 1, 0],  # Приклад 1
     [1, 1, 0, 0],  # Приклад 2
     [0, 0, 1, 1],  # Приклад 3
     [1, 0, 1, 0],  # Приклад 4
     [0, 1, 0, 1]]  # Приклад 5

y = [0, 1, 0, 1, 0]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Побудова моделі методу опорних векторів (SVM)
model = SVC(kernel='linear', C=1.0)

# Навчання моделі на тренувальних даних
model.fit(X_train, y_train)

# Прогнозування результатів на тестових даних
y_pred = model.predict(X_test)

# Оцінка точності моделі
accuracy = accuracy_score(y_test, y_pred)
print(f"Точність моделі: {accuracy * 100:.2f}%")
```

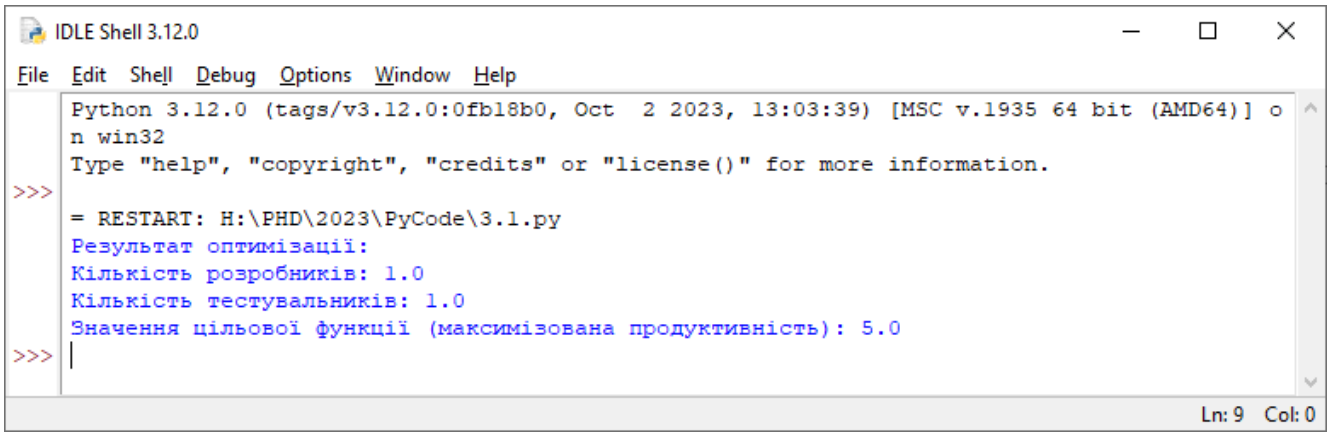


Додаток В

Додатки з 3-го розділу дисертації

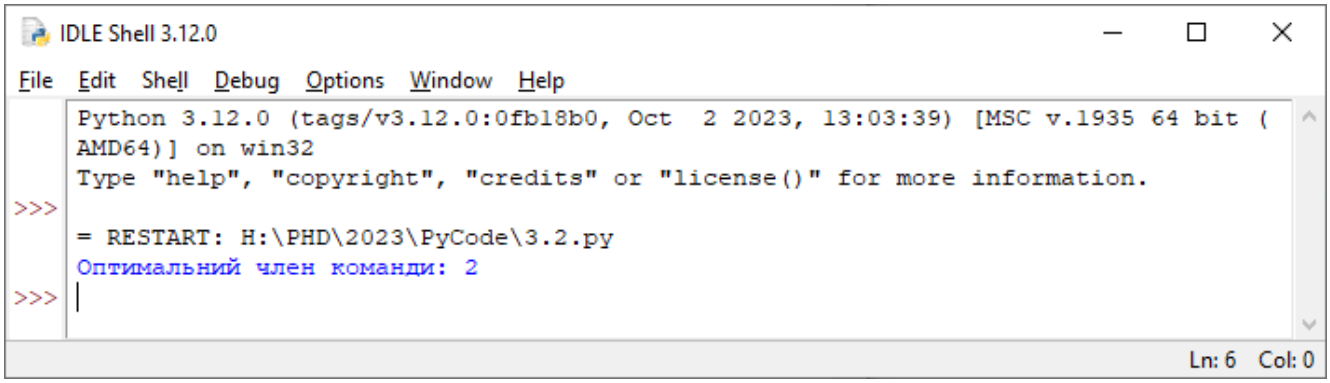
Лістинг 3.1 – лістинг лінійного програмування або генетичний алгоритм

```
from scipy.optimize import linprog
# Коефіцієнти цільової функції (ваги ролей)
c = [-3, -2]  # Вага "Розробник" = -3, Вага "Тестувальник" = -2
# Матриця обмежень (кількість членів команди)
A_eq = [[1, 1]]  # Сума розробників і тестувальників = 1
# Обмеження на кількість розробників і тестувальників
b_eq = [5]  # Загальна кількість членів команди
# Межі для змінних рішень (0 або 1, оскільки кожен член команди обирає одну з двох ролей)
bounds = [(0, 1), (0, 1)]
# Розв'язання оптимізаційної задачі
result = linprog(c, A_eq=A_eq, b_eq=b_eq, bounds=bounds, method='highs')
print("Результат оптимізації:")
print("Кількість розробників:", result.x[0])
print("Кількість тестувальників:", result.x[1])
print("Значення цільової функції (максимізована продуктивність):", -result.fun)
```



Лістинг 3.2 – лістинг розрахунок функції корисності

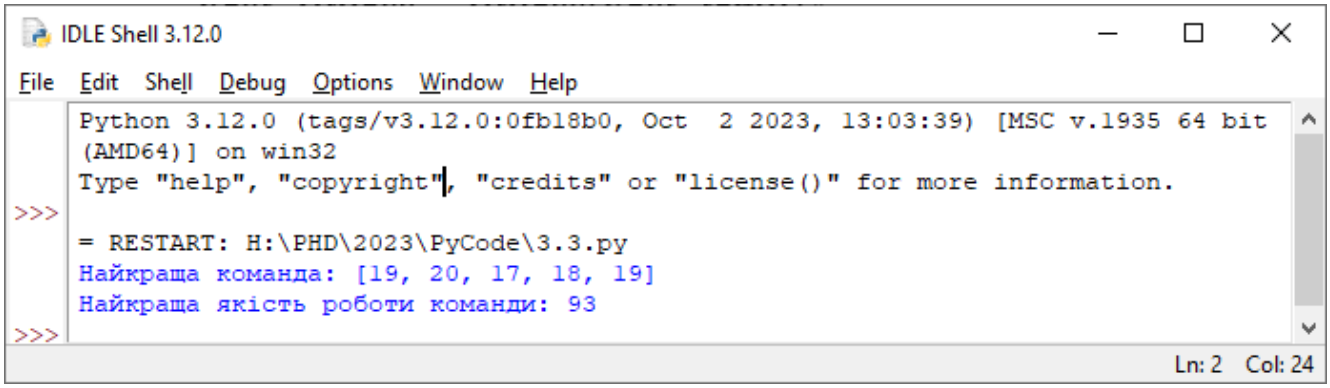
```
# Параметри команди
technical_expertise = [0.8, 0.7, 0.9, 0.6, 0.75]
innovativeness = [0.7, 0.9, 0.8, 0.6, 0.85]
collaboration_ability = [0.6, 0.7, 0.9, 0.8, 0.75]

# Ваги функції корисності
alpha = 2.0
beta = 1.5
gamma = 1.0

# Розрахунок функції корисності для кожного члена команди
utility_values = [alpha * te + beta * i + gamma * ca for te, i, ca in zip(technical_expertise, innovativeness, collaboration_ability)]

# Знайти члена команди з максимальною вартістю
optimal_member = utility_values.index(max(utility_values))

print("Оптимальний член команди:", optimal_member)
```



Лістинг 3.3 – лістинг, коли генетичний алгоритм використовується для вибору найкращої команди

```
import random
# Параметри задачі
population_size = 20  # Розмір популяції
team_size = 5  # Розмір команди
generations = 50  # Кількість поколінь

# Генерація початкової популяції (випадковий вибір)
population = [random.sample(range(1, 21), team_size) for _ in range(population_size)]

# Функція придатності (максимізуємо суму вибраних членів)
def fitness(team):
    return sum(team)

# Основний цикл генетичного алгоритму
for generation in range(generations):
    # Оцінка придатності кожної команди в популяції
    fitness_scores = [fitness(team) for team in population]

    # Вибір найкращих команд
    best_teams = [population[i] for i in sorted(range(len(fitness_scores)), key=lambda k: fitness_scores[k], reverse=True)[:population_size // 2]]

    # Схрещування і мутація
    new_population = []
    while len(new_population) < population_size:
        parent1, parent2 = random.sample(best_teams, 2)
        crossover_point = random.randint(1, team_size - 1)
        child = parent1[:crossover_point] + parent2[crossover_point:]
        mutation_prob = 0.1  # Ймовірність мутації
        if random.random() < mutation_prob:
            mutated_gene = random.choice(child)
            child[child.index(mutated_gene)] = random.choice([i for i in range(1, 21) if i not in child])
        new_population.append(child)

    population = new_population

# Знайти найкращу команду
best_team = max(population, key=fitness)
best_fitness = fitness(best_team)

print("Найкраща команда:", best_team)
```

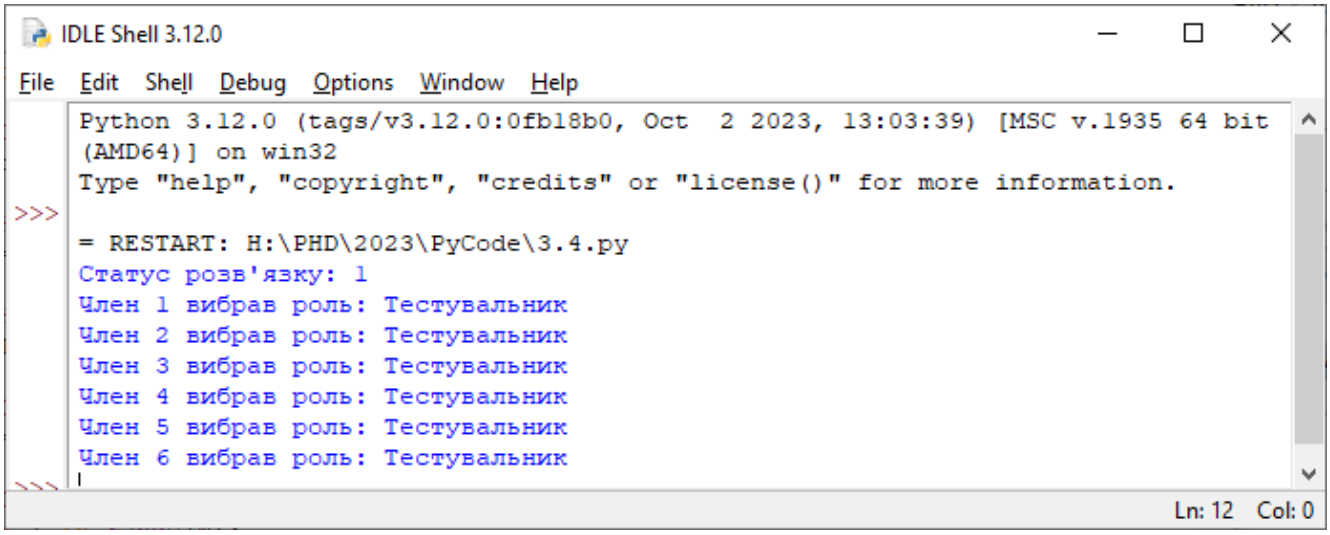
print("Найкраща якість роботи команди:", best_fitness)



Лістинг 3.4 – лістинг використання бібліотеки PuLP

```
from pulp import LpProblem, LpVariable, LpMaximize

# Створення задачі лінійного програмування
prob = LpProblem("TeamRolesOptimization", LpMaximize)

# Кількість членів команди
N = 6

# Кількість ролей
roles = ["Розробник", "Тестувальник", "Дизайнер"]

# Створення змінних рішення для вибору ролей для кожного члена команди
x = LpVariable.dicts("Role", (range(N), roles), cat="Binary")

# Додавання обмеження, що кожен член команди повинен вибрати роль
for i in range(N):
    prob += sum(x[i][role] for role in roles) == 1

# Функція корисності (наприклад, загальний вплив команди)
utility_function = sum((i + 1) * x[i][role] for i in range(N) for role in roles)

# Додавання функції до задачі
prob += utility_function

# Вирішення задачі
prob.solve()

# Вивід результатів
print("Статус розв'язку:", prob.status)
```

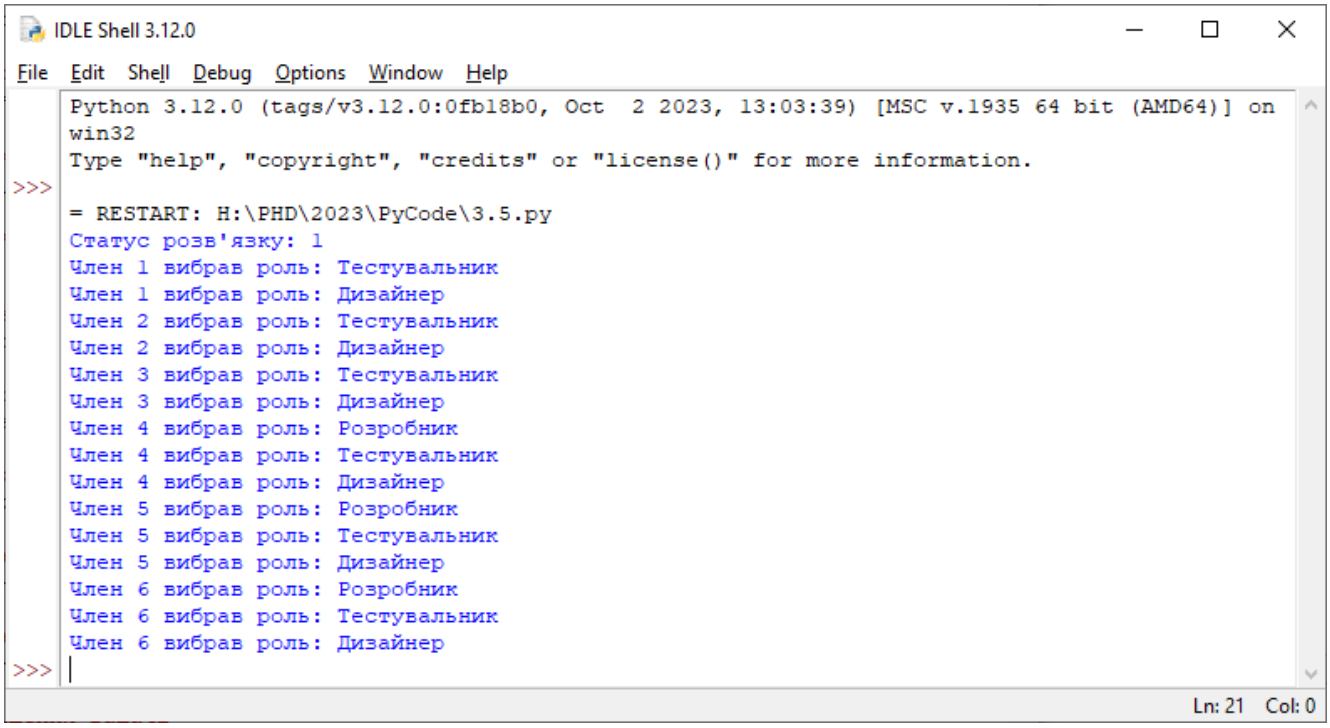
```
# Виведення виборів ролей для кожного члена команди
for i in range(N):
    for role in roles:
        if x[i][role].varValue == 1:
            print(f"Член {i + 1} вибрав роль: {role}")
```



Лістинг 3.5 – лістинг використання лінійного програмування, що допомагає максимізувати функцію корисності

```
from pulp import LpMaximize, LpProblem, LpVariable
# Створення задачі лінійного програмування
prob = LpProblem("TeamOptimization", LpMaximize)
# Кількість членів команди
total_members = 6
# Кількість ролей
roles = ["Розробник", "Тестувальник", "Дизайнер"]
# Обмеження на кількість розробників
developer_limit = 3
# Створення змінних вибору для кожного члена команди та кожної ролі
choices = LpVariable.dicts("Choice", (range(total_members), roles), cat="Binary")
# Функція корисності (наприклад, загальний вплив команди)
utility_function = sum(choices[i][role] * (i + 1) for i in range(total_members) for role in roles)
# Додавання функції до задачі
prob += utility_function
# Додавання обмеження на кількість розробників
prob += sum(choices[i]["Розробник"] for i in range(total_members)) <= developer_limit
# Вирішення задачі
prob.solve()
# Вивід результатів
print("Статус розв'язку:", prob.status)
# Виведення виборів кожного члена команди
for i in range(total_members):
    for role in roles:
        if choices[i][role].varValue == 1:
```

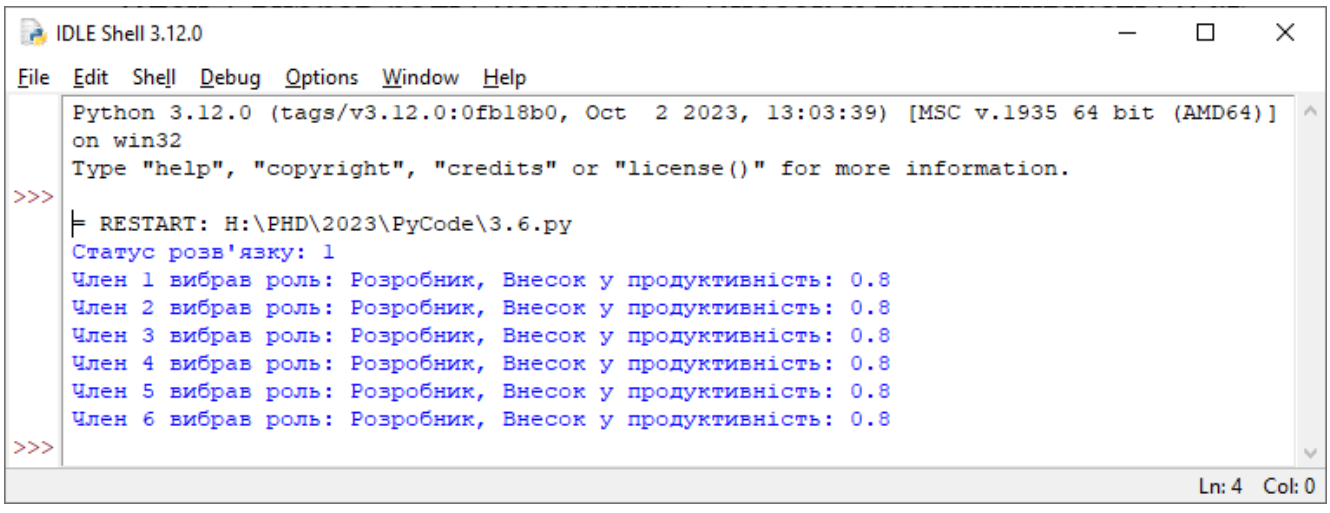
print(f"Член {i + 1} вибрав роль: {role}")



Лістинг 3.6 – лістинг

```
from pulp import LpProblem, LpVariable, LpMaximize
# Створення задачі лінійного програмування
prob = LpProblem("TeamProductivityOptimization", LpMaximize)
# Кількість членів команди
N = 6
# Кількість ролей
roles = ["Розробник", "Тестувальник", "Дизайнер"]
# Створення змінних рішення для вибору ролей для кожного члена команди
x = LpVariable.dicts("Role", (range(N), roles), cat="Binary")
# Внесок кожної ролі у продуктивність (припустимо, що це випадкові значення)
role_contributions = {"Розробник": 0.8, "Тестувальник": 0.7, "Дизайнер": 0.6}
# Цільова функція для максимізації продуктивності команди
```

```
    objective_function = sum(x[i][role] * role_contributions[role] for i in range(N) for role in roles)
    # Додавання функції до задачі
    prob += objective_function
    # Додавання обмеження, що кожен член команди повинен вибрати роль
    for i in range(N):
        prob += sum(x[i][role] for role in roles) == 1
    # Вирішення задачі
    prob.solve()
    # Вивід результатів
    print("Статус розв'язку:", prob.status)
    # Виведення виборів ролей та їх внеску у продуктивність
    for i in range(N):
        for role in roles:
            if x[i][role].varValue == 1:
                print(f"Член {i + 1} вибрав роль: {role}, Внесок у продуктивність: {role_contributions[role]}")
```

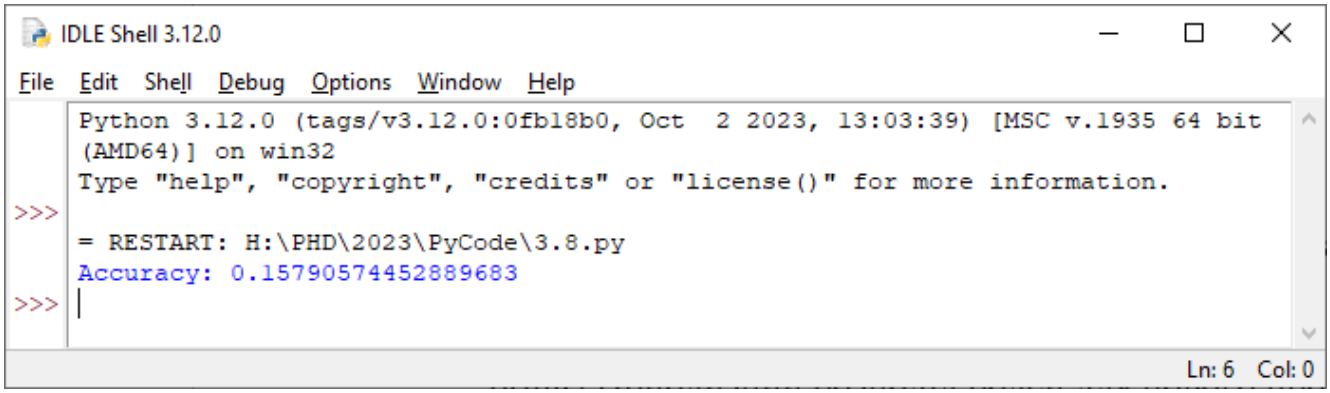


Лістинг 3.7 – лістинг прикладу простої нейронної мережі з використанням бібліотеки Python

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
# Приклад даних для навчання
# Параметри проєкту та роль у команді
data = np.array([
    [5, 3, 1, 0],  # Параметри проєкту 1 та розподіл ролей
    [3, 1, 4, 1],  # Параметри проєкту 2 та розподіл ролей
    [8, 4, 2, 1],  # Параметри проєкту 3 та розподіл ролей
    [1, 2, 3, 0],  # Параметри проєкту 4 та розподіл ролей
])
# Вхідні дані (параметри проєкту)
X = data[:, :3]
# Вихідні дані (розподіл ролей)
y = data[:, 3]
# Створення моделі нейронної мережі
model = Sequential()
model.add(Dense(10, input_dim=3, activation='relu'))
model.add(Dense(1, activation='linear'))
# Компіляція моделі
model.compile(loss='mean_squared_error', optimizer='adam')
# Тренування моделі
model.fit(X, y, epochs=1000, batch_size=1)
# Передбачення для нового проєкту
new_project = np.array([[6, 3, 2]])  # Параметри нового проєкту
prediction = model.predict(new_project)
print(f'Приблизний розподіл ролей для нового проєкту: {prediction[0][0]}')
```

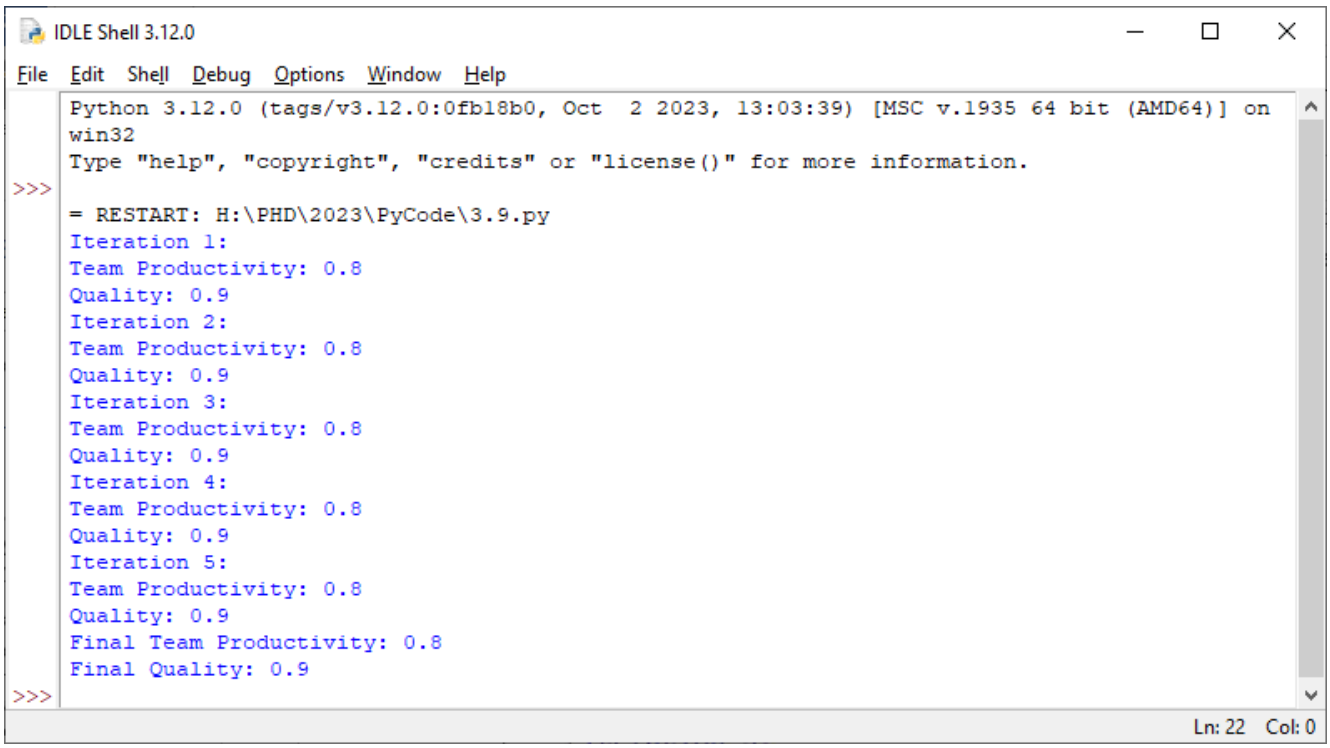
Лістинг 3.8 – лістинг приклад коду для використання лінійної регресії.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

# Згенеруйте випадкові дані для X і Y (припустимо, це ваші характеристики і мітки)
np.random.seed(0)
X = np.random.rand(100, 1)  # Припустимо, у вас є одна характеристика
Y = 2 * X.squeeze() + np.random.randn(100)  # Припустимо, що це ваша мітка

# Розділіть дані на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

# Ініціалізуйте модель лінійної регресії
model = LinearRegression()

# Навчання моделі
model.fit(X_train, y_train)

# Зробіть прогнози на тестовому наборі
y_pred = model.predict(X_test)

# Оцінка точності моделі
accuracy = metrics.r2_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```



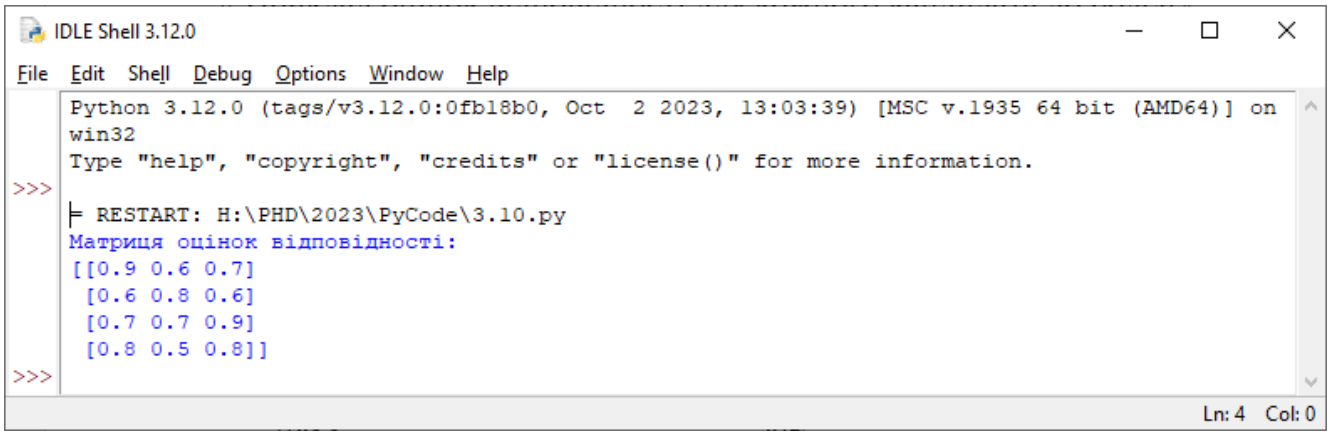
Лістинг 3.9 – лістинг, програмного коду, що демонструє роботу команди над проєктом з багатьма задачами.

```
# Визначення функції calculate_team_productivity
def calculate_team_productivity():
    # Заміни це на реальні обчислення або логіку для визначення продуктивності команди
    return 0.8  # Приклад значення, заміни його на власні обчислення

# Визначення функції calculate_quality
def calculate_quality():
    # Заміни це на реальні обчислення або логіку для визначення якості
    return 0.9  # Приклад значення, заміни його на власні обчислення

# Визначення функції adjust_strategy
def adjust_strategy():
    # Логіка для коригування стратегії
    print("Adjusting the strategy...")

# Ініціалізація команди та початкового плану
team_productivity = []
quality = []

for iteration in range(5):  # 5 ітерацій
    # Виконання завдань
    team_productivity.append(calculate_team_productivity())
    quality.append(calculate_quality())

    # Моніторинг та оцінка
    print(f"Iteration {iteration + 1}:")
    print(f"Team Productivity: {team_productivity[-1]}")
    print(f"Quality: {quality[-1]}")

    # Аналіз результатів
    if iteration > 0:
        if quality[-1] < quality[-2]:
            print("Quality is decreasing. Adjusting the strategy.")
            adjust_strategy()

# Фінальний результат
print("Final Team Productivity:", team_productivity[-1])
print("Final Quality:", quality[-1])
```


IDLE Shell 3.12.0
File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: H:\PHD\2023\PyCode\3.9.py
Iteration 1:
Team Productivity: 0.8
Quality: 0.9
Iteration 2:
Team Productivity: 0.8
Quality: 0.9
Iteration 3:
Team Productivity: 0.8
Quality: 0.9
Iteration 4:
Team Productivity: 0.8
Quality: 0.9
Iteration 5:
Team Productivity: 0.8
Quality: 0.9
Final Team Productivity: 0.8
Final Quality: 0.9
>>>
Ln: 22 Col: 0

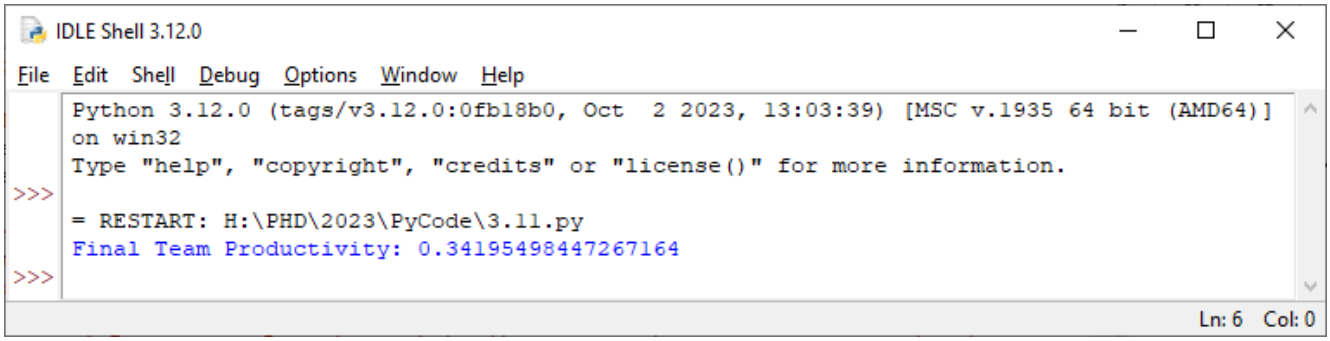
Лістинг 3.10 – лістинг коду приклад матриці оцінок відповідності для кожного кандидата до ролей

```
import numpy as np

# Приклад списку кандидатів
candidates = ["Кандидат1", "Кандидат2", "Кандидат3", "Кандидат4"]

# Приклад оцінок відповідності для кожного кандидата до ролей
role_ratings = {
    "Розробник": [0.9, 0.6, 0.7, 0.8],
    "Тестувальник": [0.6, 0.8, 0.7, 0.5],
    "Аналітик": [0.7, 0.6, 0.9, 0.8]
}

# Створення матриці оцінок відповідності
matrix = np.array([role_ratings[role] for role in role_ratings]).T

# Вивід матриці
print("Матриця оцінок відповідності:")
print(matrix)
```



Лістинг 3.11 – лістинг приклад коду Python, який демонструє моделювання взаємодії команди за допомогою ройового алгоритму

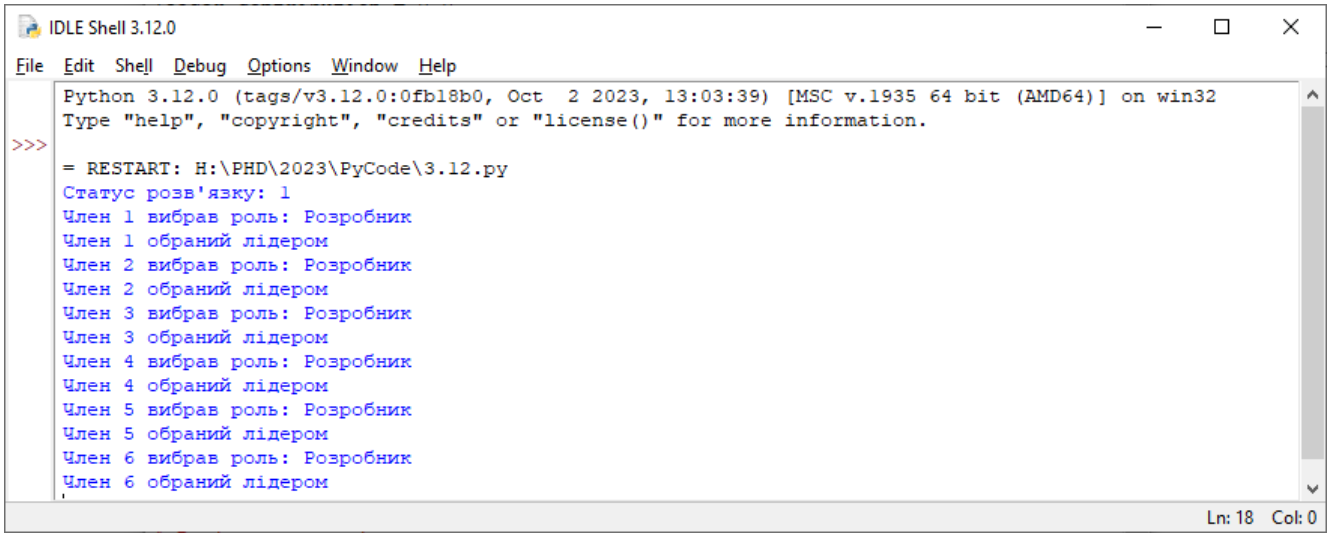
```
import random

# Ініціалізація команди (початкова популяція агентів)
team_size = 20
team = [{'agent_id': i, 'productivity': random.uniform(0, 1)} for i in range(team_size)]

# Моделювання взаємодії та руху команди (рой)
num_iterations = 100
for iteration in range(num_iterations):
    for agent in team:
        # Визначення ближніх сусідів (інших агентів, з якими взаємодіємо)
        neighbors = random.sample(team, 5)  # Взяли 5 випадкових сусідів

        # Моделювання взаємодії з сусідами та оновлення продуктивності
        total_productivity = agent['productivity'] + sum(neighbor['productivity'] for neighbor in neighbors)
        agent['productivity'] = total_productivity / 6  # Оновлюємо продуктивність

# Оцінка продуктивності команди
team_productivity = sum(agent['productivity'] for agent in team) / team_size

# Виведення результатів
print(f"Final Team Productivity: {team_productivity}")
```



Лістинг 3.12 – лістинг прикладу коду з використанням бібліотеки PuLP для вирішення цієї задачі лінійного програмування

```
from pulp import LpProblem, LpVariable, LpMaximize

# Створення задачі лінійного програмування
prob = LpProblem("TeamLeaderSelection", LpMaximize)

# Кількість членів команди
N = 6

# Кількість ролей
roles = ["Розробник", "Тестувальник", "Дизайнер"]

# Внесок кожної ролі у продуктивність (припустимо, що це випадкові значення)
role_contributions = {"Розробник": 0.8, "Тестувальник": 0.7, "Дизайнер": 0.6}

# Внесок лідера у продуктивність (випадковий показник)
leader_contribution = 0.9

# Створення змінних рішення для вибору ролей для кожного члена команди та вибору лідера
x = LpVariable.dicts("Role", (range(N), roles), cat="Binary")
leader = LpVariable("Leader", cat="Binary")

# Додавання обмеження, що кожен член команди повинен вибрати роль
for i in range(N):
    prob += sum(x[i][role] for role in roles) == 1

# Цільова функція для максимізації продуктивності команди з лідером
objective_function = (
    leader_contribution * leader
    + sum(x[i][role] * role_contributions[role] for i in range(N) for role in roles)
)

# Додавання цільової функції до задачі
prob += objective_function

# Вирішення задачі
prob.solve()

# Вивід результатів
print("Статус розв'язку:", prob.status)

# Виведення виборів ролей та лідера
for i in range(N):
```

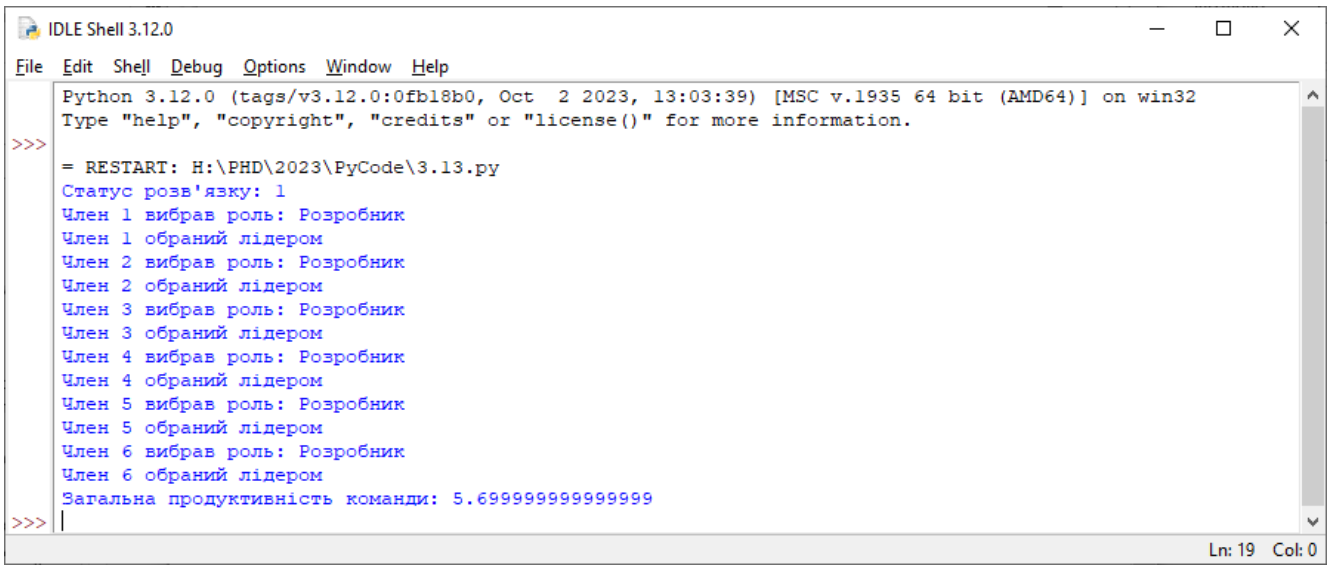
```
    for role in roles:
        if x[i][role].varValue == 1:
            print(f"Член {i + 1} вибрав роль: {role}")
    if leader.varValue == 1:
        print(f"Член {i + 1} обраний лідером")
```



Лістинг 3.13 – лістинг приклад розрахунків за сценарієм обрання лідера у команді з використанням лінійного програмування

```
from pulp import LpProblem, LpVariable, LpMaximize

# Створення задачі лінійного програмування
prob = LpProblem("TeamLeaderSelection", LpMaximize)

# Кількість членів команди
N = 6

# Кількість ролей
roles = ["Розробник", "Тестувальник", "Дизайнер"]

# Внесок кожної ролі у продуктивність (припустимо, що це випадкові значення)
role_contributions = {"Розробник": 0.8, "Тестувальник": 0.7, "Дизайнер": 0.6}

# Внесок лідера у продуктивність (випадковий показник)
leader_contribution = 0.9

# Створення змінних рішення для вибору ролей для кожного члена команди та вибору лідера
x = LpVariable.dicts("Role", (range(N), roles), cat="Binary")
```

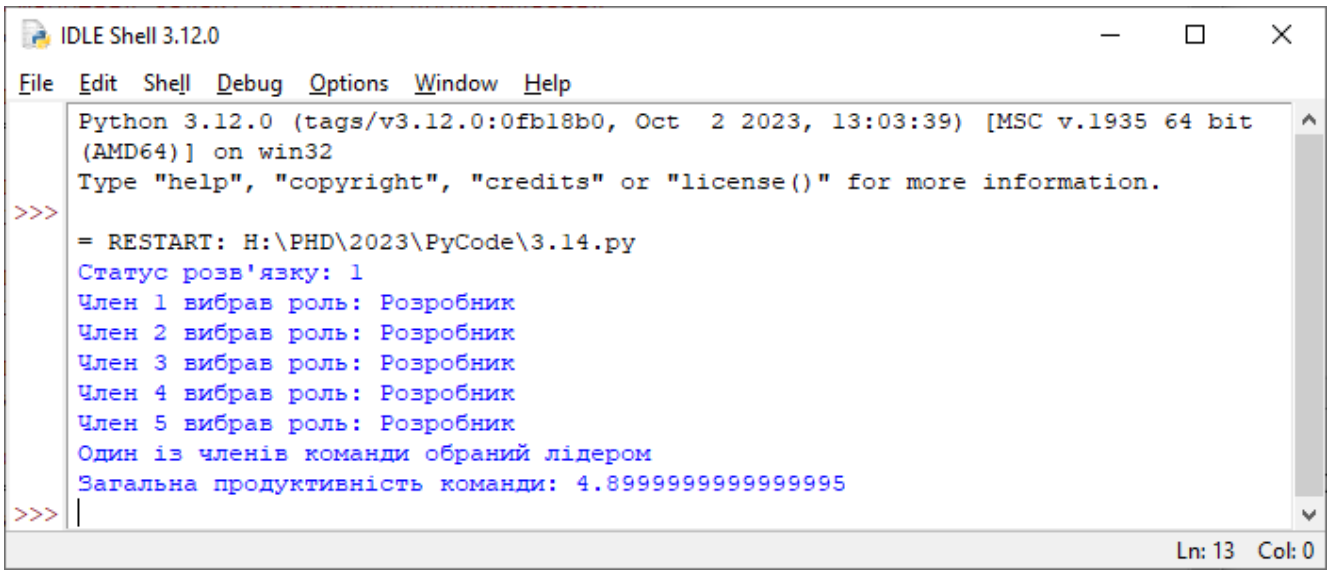
```
leader = LpVariable("Leader", cat="Binary")

# Додавання обмеження, що кожен член команди повинен вибрати роль
for i in range(N):
    prob += sum(x[i][role] for role in roles) == 1

# Цільова функція для максимізації продуктивності команди з лідером
objective_function = (
    leader_contribution * leader
    + sum(x[i][role] * role_contributions[role] for i in range(N) for role in roles)
)

# Додавання цільової функції до задачі
prob += objective_function

# Вирішення задачі
prob.solve()

# Вивід результатів
print("Статус розв'язку:", prob.status)

# Виведення виборів ролей та лідера
for i in range(N):
    for role in roles:
        if x[i][role].varValue == 1:
            print(f"Член {i + 1} вибрав роль: {role}")
    if leader.varValue == 1:
        print(f"Член {i + 1} обраний лідером")

# Виведення загальної продуктивності команди з урахуванням обраного лідера
total_productivity = leader_contribution * leader.varValue
for i in range(N):
    for role in roles:
        total_productivity += x[i][role].varValue * role_contributions[role]
print(f"Загальна продуктивність команди: {total_productivity}")
```



Лістинг 3.14 – лістинг прикладу розрахунків за сценарієм обрання лідера у команді за допомогою лінійного програмування

```
from pulp import LpProblem, LpVariable, LpMaximize

# Створення задачі лінійного програмування
prob = LpProblem("TeamLeaderSelection", LpMaximize)

# Кількість членів команди
N = 5

# Кількість ролей
roles = ["Розробник", "Тестувальник"]

# Внесок кожної ролі у продуктивність (припустимо, що це випадкові значення)
role_contributions = {"Розробник": 0.8, "Тестувальник": 0.7}

# Внесок лідера у продуктивність (випадковий показник)
leader_contribution = 0.9

# Створення змінних рішення для вибору ролей для кожного члена команди та вибору лідера
x = LpVariable.dicts("Role", (range(N), roles), cat="Binary")
leader = LpVariable("Leader", cat="Binary")

# Додавання обмеження, що кожен член команди повинен вибрати роль
for i in range(N):
    prob += sum(x[i][role] for role in roles) == 1
```

```
# Цільова функція для максимізації продуктивності команди з лідером
objective_function = (leader_contribution * leader
    + sum(x[i][role] * role_contributions[role]
          for i in range(N) for role in roles))

# Додавання цільової функції до задачі
prob += objective_function

# Додавання обмеження, що лідером може бути обраний тільки член команди, який обрав роль "Розробник"
for i in range(N):
    prob += leader <= x[i]["Розробник"]

# Вирішення задачі
prob.solve()

# Вивід результатів
print("Статус розв'язку:", prob.status)

# Виведення виборів ролей та лідера
for i in range(N):
    for role in roles:
        if x[i][role].varValue == 1:
            print(f"Член {i + 1} вибрав роль: {role}")
if leader.varValue == 1:
    print("Один із членів команди обраний лідером")

# Виведення загальної продуктивності команди з урахуванням обраного лідера
total_productivity = leader_contribution * leader.varValue
for i in range(N):
    for role in roles:
        total_productivity += x[i][role].varValue * role_contributions[role]
print(f"Загальна продуктивність команди: {total_productivity}")
```



Лістинг 3.15 – лістинг сценарію пошуку рішень за допомогою моделі мережевого потоку

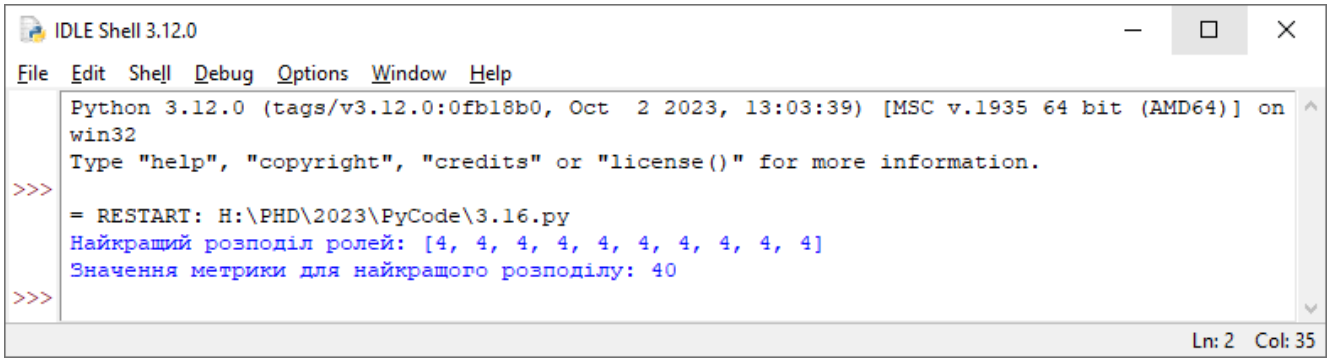
```
import networkx as nx
from scipy.optimize import linear_sum_assignment
# Створення графу для моделі мережевого потоку
G = nx.DiGraph()
# Додавання членів команди до графу
team_members = ['A', 'B', 'C', 'D']
G.add_nodes_from(team_members)

# Додавання вузлів-завдань, які потрібно розподілити
tasks = ['Task1', 'Task2', 'Task3']
G.add_nodes_from(tasks)

# Додавання ребер для визначення можливостей розподілу завдань
for member in team_members:
    for task in tasks:
        G.add_edge(member, task)

# Визначення пропускних здатностей ребер (наприклад, скільки завдань може виконати кожен член команди)
capacities = {
    ('A', 'Task1'): 1,
    ('A', 'Task2'): 1,
    ('B', 'Task2'): 1,
    ('C', 'Task1'): 1,
    ('C', 'Task2'): 1,
    ('D', 'Task3'): 1,
}
```

```
# Встановлення пропускних здатностей на ребрах графу
nx.set_edge_attributes(G, capacities, 'capacity')

# Розв'язання задачі розподілу завдань як задачі мінімізації
# Мета - знайти оптимальний розподіл завдань між членами команди
# За допомогою методу linear_sum_assignment знаходимо оптимальний розподіл
_, assignment = linear_sum_assignment(-nx.max_flow_min_cost(G, capacity='capacity'))

# Вивід результатів
for i, task in enumerate(tasks):
    member = team_members[assignment[i]]
    print(f"Завдання '{task}' було призначено члену команди '{member}'")
```

Лістинг 3.16 – лістинг генетичні алгоритми та еволюційні підходи

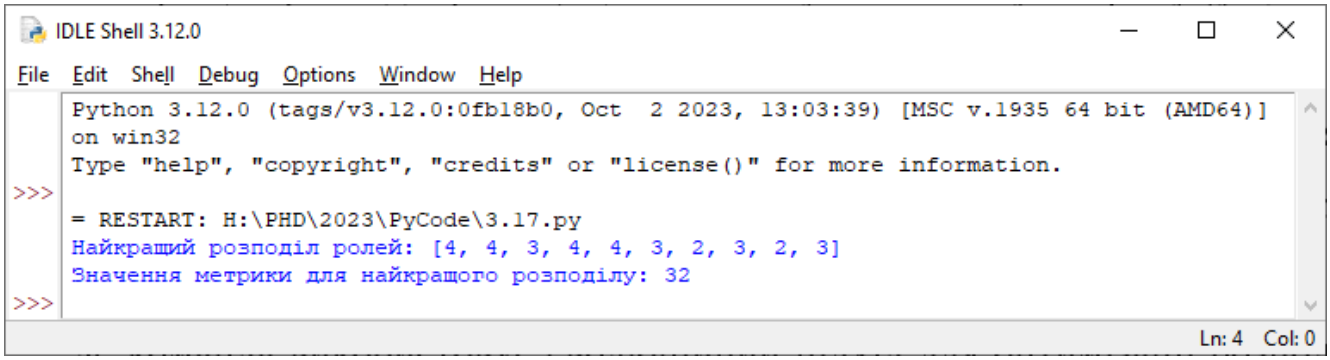
```
import random
import numpy as np

# Кількість членів команди
num_members = 10

# Кількість ролей
num_roles = 5

# Метрика, яку потрібно максимізувати (приклад)
def fitness_function(assignment):
    return sum(assignment)

# Функція для генерації випадкового розподілу ролей для початкової популяції
def generate_random_assignment(num_members, num_roles):
    return [random.randint(0, num_roles - 1) for _ in range(num_members)]

# Генетичний алгоритм
population_size = 100
generations = 50
mutation_rate = 0.1

# Ініціалізація початкової популяції
population = [generate_random_assignment(num_members, num_roles) for _ in range(population_size)]

for generation in range(generations):
    # Оцінка кожної особини в популяції
```

```
    fitness_scores = [fitness_function(assignment) for assignment in population]

    # Відбір найкращих особин
    elite_indices = np.argsort(fitness_scores)[-population_size // 2:]
    elite_population = [population[i] for i in elite_indices]

    # Створення нової популяції
    new_population = []

    # Відбір найкращих особин із елітної популяції
    new_population.extend(elite_population)

    # Генерація нових особин за допомогою схрещування
    while len(new_population) < population_size:
        parent1, parent2 = random.choices(elite_population, k=2)
        crossover_point = random.randint(1, num_members - 1)
        child = parent1[:crossover_point] + parent2[crossover_point:]
        new_population.append(child)

    # Мутація
    for i in range(len(new_population)):
        if random.random() < mutation_rate:
            mutation_point = random.randint(0, num_members - 1)
            new_population[i][mutation_point] = random.randint(0, num_roles - 1)

    population = new_population

# Знаходимо найкращий розподіл ролей
best_assignment = max(population, key=fitness_function)
print("Найкращий розподіл ролей:", best_assignment)
print("Значення метрики для найкращого розподілу:",
fitness_function(best_assignment))
```



Лістинг 3.17 – лістинг прикладу еволюційного підходи також можуть бути використані для пошуку оптимальних рішень в команді

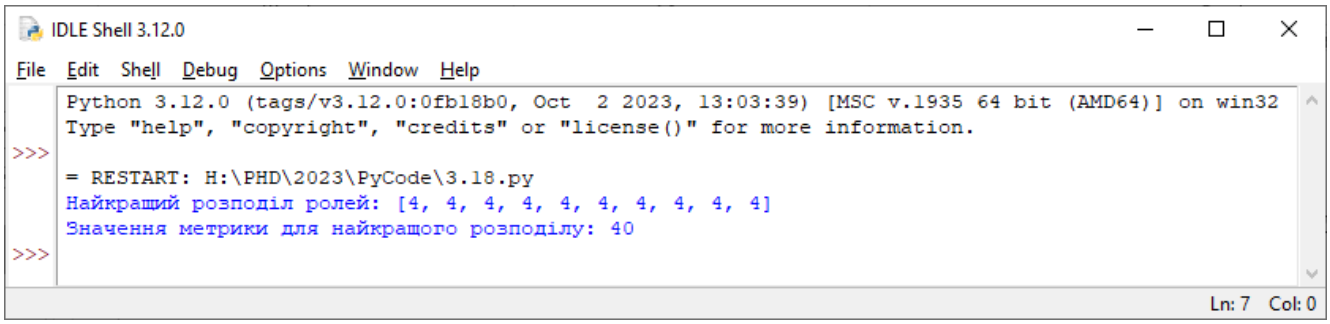
```
import random

# Кількість членів команди
num_members = 10

# Кількість ролей
num_roles = 5

# Метрика, яку потрібно максимізувати (приклад)
def fitness_function(assignment):
    return sum(assignment)

# Ініціалізація початкового розподілу ролей
current_assignment = [random.randint(0, num_roles - 1) for _ in range(num_members)]

# Ініціалізація параметрів еволюційного підходу
num_generations = 50
mutation_rate = 0.1

for generation in range(num_generations):
    # Оцінка поточного розподілу ролей
    current_fitness = fitness_function(current_assignment)

    # Створення нового розподілу ролей шляхом мутації
    new_assignment = current_assignment.copy()
    for i in range(num_members):
        if random.random() < mutation_rate:
            new_assignment[i] = random.randint(0, num_roles - 1)

    # Оцінка нового розподілу ролей
    new_fitness = fitness_function(new_assignment)

    # Порівняння нового розподілу ролей з поточним та прийняття рішення
    if new_fitness >= current_fitness:
        current_assignment = new_assignment

# Знаходимо найкращий розподіл ролей
best_assignment = current_assignment
best_fitness = fitness_function(best_assignment)

print("Найкращий розподіл ролей:", best_assignment)
print("Значення метрики для найкращого розподілу:", best_fitness)
```



Лістинг 3.18 – лістинг прикладу розрахунків за цим сценарієм, де команда використовує еволюційний підхід для оптимізації розподілу ролей

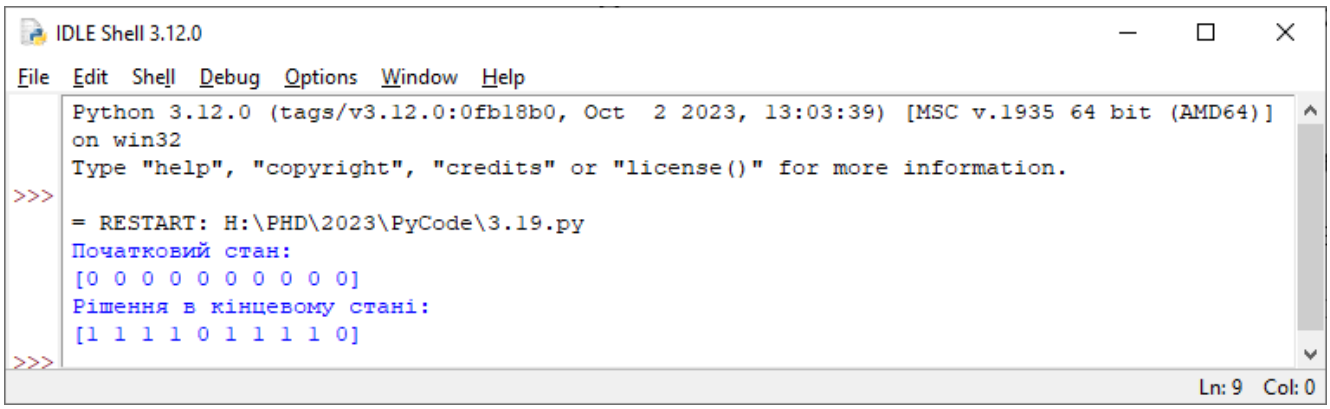
```
import random
import numpy as np

# Кількість членів команди
num_members = 10

# Кількість ролей
num_roles = 5

# Метрика, яку потрібно максимізувати (приклад)
def fitness_function(assignment):
    return sum(assignment)

# Функція для генерації випадкового розподілу ролей для початкової популяції
def generate_random_assignment(num_members, num_roles):
    return [random.randint(0, num_roles - 1) for _ in range(num_members)]

# Еволюційний підхід
population_size = 100
generations = 50

# Ініціалізація початкової популяції
population = [generate_random_assignment(num_members, num_roles) for _ in range(population_size)]

for generation in range(generations):
    # Оцінка кожної особини в популяції
    fitness_scores = [fitness_function(assignment) for assignment in population]

    # Відбір найкращих особин
    elite_indices = np.argsort(fitness_scores)[-population_size // 2:]
    elite_population = [population[i] for i in elite_indices]
```

```
    # Створення нової популяції
    new_population = []

    # Відбір найкращих особин із елітної популяції
    new_population.extend(elite_population)

    # Генерація нових особин за допомогою схрещування
    while len(new_population) < population_size:
        parent1, parent2 = random.choices(elite_population, k=2)
        child = [random.choice(pair) for pair in zip(parent1, parent2)]
        new_population.append(child)

    population = new_population

# Знаходимо найкращий розподіл ролей
best_assignment = max(population, key=fitness_function)
print("Найкращий розподіл ролей:", best_assignment)
print("Значення метрики для найкращого розподілу:",
fitness_function(best_assignment))
```



Лістинг 3.19 – лістинг прикладу як клітинні автомати та моделі на основі правил можуть бути використані для моделювання пошуку рішень в команді

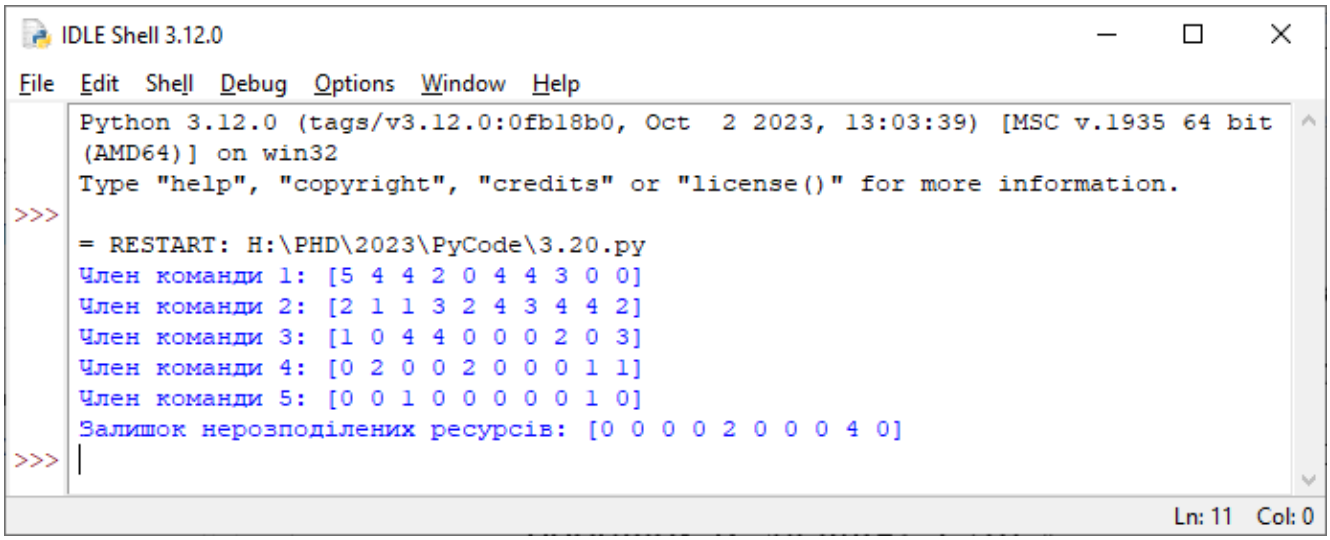
```
import numpy as np

# Розмірність клітинного автомата (решітки)
n = 10  # Розмірність по осі X
m = 10  # Розмірність по осі Y

# Створення початкового стану клітинного автомата (решітки)
cellular_automaton = np.zeros((n, m), dtype=int)

# Визначення правил для прийняття рішень (приклад)
decision_rule = np.random.randint(2, size=(n, m))  # Випадково визначені правила (0 або 1)

# Ініціалізація початкового стану
cellular_automaton[0, :] = np.random.randint(2, size=n)  # Випадково ініціалізований початковий рядок

# Моделювання роботи клітинного автомата
for t in range(1, m):
    for x in range(n):
        left = cellular_automaton[x - 1, t - 1] if x > 0 else 0
        center = cellular_automaton[x, t - 1]
        right = cellular_automaton[(x + 1) % n, t - 1]  # Обрізаємо для замкненого розміру решітки
        neighborhood = [left, center, right]

        # Використання правил для прийняття рішення
        cellular_automaton[x, t] = decision_rule[x, t] if neighborhood in [[0, 0, 0], [1, 1, 1]] else 1

# Результати прийнятих рішень в кінцевому стані клітинного автомата
final_decision = cellular_automaton[:, -1]

# Вивід результатів
print("Початковий стан:")
print(cellular_automaton[:, 0])
print("Рішення в кінцевому стані:")
print(final_decision)
```



Лістинг 3.20 – лістинг прикладу коли розглянемо сценарій, де команда вирішує, як розподілити ресурси між своїми членами на основі правил розподілу.

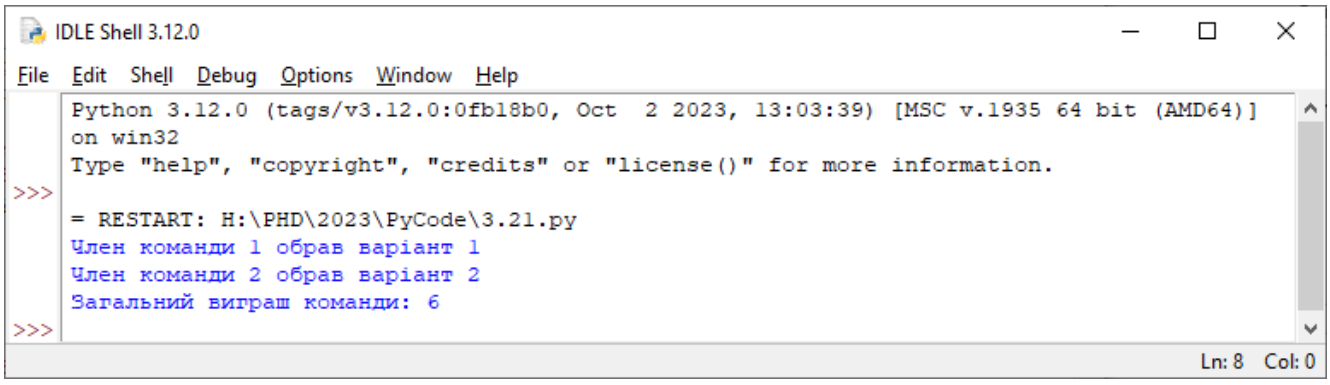
```
import numpy as np

# Кількість членів команди
num_members = 5

# Кількість ресурсів для розподілу
num_resources = 10

# Матриця правил розподілу ресурсів (приклад)
# В цьому прикладі, правила випадково визначають, скільки ресурсів отримає кожен член команди
distribution_rules = np.random.randint(0, 6, size=(num_members, num_resources))

# Ресурси, доступні для розподілу
available_resources = np.array([8, 7, 10, 9, 6, 8, 7, 9, 10, 6])

# Розподіл ресурсів на основі правил
resource_allocation = np.zeros((num_members, num_resources), dtype=int)

for member in range(num_members):
    for resource in range(num_resources):
        if available_resources[resource] > 0:
            allocation = min(distribution_rules[member, resource], available_resources[resource])
            resource_allocation[member, resource] = allocation
            available_resources[resource] -= allocation

# Вивід результатів
for member in range(num_members):
    print(f"Член команди {member + 1}: {resource_allocation[member]}")
```

```
# Залишок нерозподілених ресурсів
print(f"Залишок нерозподілених ресурсів: {available_resources}")
```



Лістинг 3.21 – лістинг розрахунків за сценарієм пошуку рішень у команді за допомогою теорії ігор.

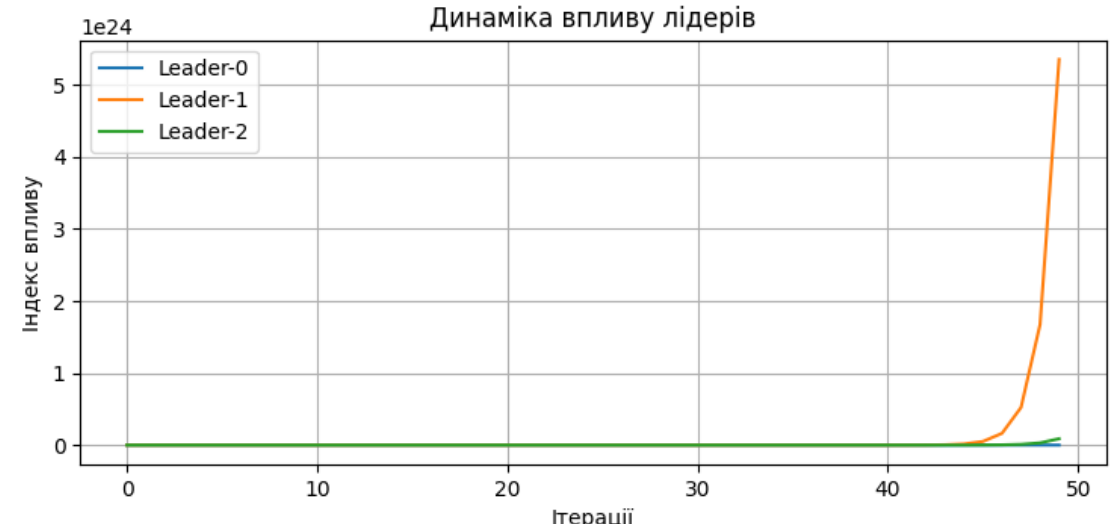
```
import numpy as np
from scipy.optimize import linear_sum_assignment

# Матриця виграшів (рядки - члени команди, стовпці - варіанти дій)
# За замовчуванням, вибір першого варіанта виглядає більш привабливим для кожного члена команди.
payoff_matrix = np.array([
    [4, 1],  # Виграші для члена команди 1 при виборі варіанту 1 та 2 відповідно
    [3, 2],  # Виграші для члена команди 2 при виборі варіанту 1 та 2 відповідно
    [2, 3],  # Виграші для члена команди 3 при виборі варіанту 1 та 2 відповідно
])

# Використовуємо метод лінійного програмування для знаходження оптимального рішення
row_ind, col_ind = linear_sum_assignment(-payoff_matrix)

# Отримані оптимальні вибори для кожного члена команди
optimal_choices = col_ind

# Вивід результатів
for i, choice in enumerate(optimal_choices):
    print(f"Член команди {i + 1} обрав варіант {choice + 1}")

# Виграш команди (сума виграшів обраних варіантів)
team_payoff = np.sum(payoff_matrix[np.arange(len(optimal_choices)), optimal_choices])
print(f"Загальний виграш команди: {team_payoff}")
```

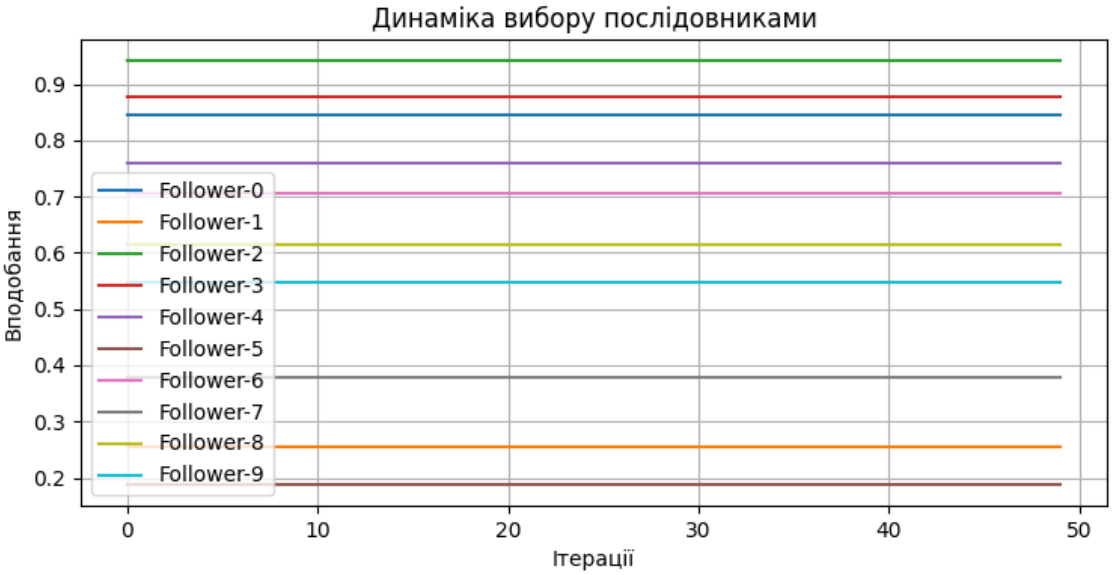


Лістинг 3.22 – лістинг прикладу коду розрахунків, де ми будемо моделювати динаміку взаємодії між лідерами та послідовниками у команді за допомогою графової агентної моделі

```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

# Початкова конфігурація команди
G = nx.Graph()
num_leaders = 3
num_followers = 10

# Додавання лідерів до графу
for i in range(num_leaders):
    G.add_node(f'Leader-{i}', agent_type='leader', influence=np.random.uniform(0.5, 1.0))

# Додавання послідовників до графу
for i in range(num_followers):
    G.add_node(f'Follower-{i}', agent_type='follower', preference=np.random.rand())

# Визначення комунікаційних зв'язків (ребер) між агентами
for leader in G.nodes:
    for follower in G.nodes:
        if leader != follower:
            if G.nodes[leader]['agent_type'] == 'leader' and G.nodes[follower]['agent_type'] == 'follower':
                G.add_edge(leader, follower, communication=np.random.uniform(0.1, 0.5))

# Моделювання динаміки взаємодії та впливу в команді
num_iterations = 50
leader_influence_values = {leader: [] for leader in G.nodes if G.nodes[leader]['agent_type'] == 'leader'}
follower_preference_values = {follower: [] for follower in G.nodes if G.nodes[follower]['agent_type'] == 'follower'}

for iteration in range(num_iterations):
    for leader in leader_influence_values:
        total_influence = G.nodes[leader]['influence']
        for follower in G.neighbors(leader):
            communication_strength = G[leader][follower]['communication']
            if leader_influence_values[leader]:
                total_influence += communication_strength * leader_influence_values[leader][-1]
```

```
        leader_influence_values[leader].append(total_influence)

    for follower in follower_preference_values:
        best_leader = None
        max_influence = -1
        for leader in G.nodes:
            if G.nodes[leader]['agent_type'] == 'leader':
                leader_influence = leader_influence_values[leader][-1]
                if leader_influence > max_influence:
                    best_leader = leader
                    max_influence = leader_influence

        follower_preference = G.nodes[follower]['preference']
        influence_difference = max_influence - leader_influence_values[best_leader][-1]
        follower_preference_values[follower].append(follower_preference +
influence_difference)

# Візуалізація динаміки впливу та вибору послідовників
plt.figure(figsize=(12, 6))

for leader, values in leader_influence_values.items():
    plt.plot(range(num_iterations), values, label=leader)

plt.xlabel('Ітерації')
plt.ylabel('Індекс впливу')
plt.title('Динаміка впливу лідерів')
plt.legend()
plt.grid()

plt.figure(figsize=(12, 6))

for follower, values in follower_preference_values.items():
    plt.plot(range(num_iterations), values, label=follower)

plt.xlabel('Ітерації')
plt.ylabel('Вподобання')
plt.title('Динаміка вибору послідовниками')
plt.legend()
plt.grid()

plt.show()
```


Динаміка впливу лідерів
1e24
Leader-0
Leader-1
Leader-2
Індекс впливу
0
1
2
3
4
5
0
10
20
30
40
50
Ітерації


Динаміка вибору послідовниками
Follower-0
Follower-1
Follower-2
Follower-3
Follower-4
Follower-5
Follower-6
Follower-7
Follower-8
Follower-9
Вподобання
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50
Ітерації

Лістинг 3.23 – лістинг прикладу агентної моделі команди імпровізованої наради

```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

# Створення графу для представлення комунікаційних зв'язків
G = nx.Graph()
G.add_node("Лідер_1", рішучість=0.8, підслуховуваність=0.2)
G.add_node("Лідер_2", рішучість=0.7, підслуховуваність=0.3)
G.add_node("Послідовник_1", рішучість=0.3, підслуховуваність=0.7)
G.add_node("Послідовник_2", рішучість=0.4, підслуховуваність=0.6)

G.add_edge("Лідер_1", "Лідер_2", вплив=0.5)
G.add_edge("Лідер_1", "Послідовник_1", вплив=0.3)
G.add_edge("Лідер_2", "Послідовник_1", вплив=0.2)
G.add_edge("Послідовник_1", "Послідовник_2", вплив=0.4)

# Визначення правил прийняття рішень
def прийняти_рішення(агент, сусіди):
    рішучість_агента = агент["рішучість"]
    вплив_сусіда = [сусід["вплив"] for сусід in сусіди]

    загальний_вплив = np.dot(вплив_сусіда, рішучість_агента)
    if загальний_вплив >= 0.5:
        return "Позитивне рішення"
    else:
        return "Негативне рішення"

# Прийняття рішень агентами
результати = {}
for агент in G.nodes:
    сусіди = list(G.neighbors(агент))
    рішення = прийняти_рішення(G.nodes[агент], [G.nodes[сусід] for сусід in сусіди])
    результати[агент] = рішення

# Візуалізація графу та прийнятих рішень
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_color="lightblue", node_size=800)
labels = {агент: результати[агент] for агент in G.nodes}
nx.draw_networkx_labels(G, pos, labels=labels, font_size=12)
plt.show()
```

Лістинг 3.24 – лістинг побудови графу взаємодії на основі зв'язків

```
import networkx as nx
import matplotlib.pyplot as plt

# Матриця суміжності для взаємодії між агентами
interaction_matrix = [
    [0, 1, 1, 0, 0, 0],
    [1, 0, 1, 1, 0, 0],
    [1, 1, 0, 1, 0, 0],
    [0, 1, 1, 0, 1, 0],
    [0, 0, 0, 1, 0, 1],
    [0, 0, 0, 0, 1, 0]
]

# Отримання ребер з матриці суміжності
edges = []
for i in range(len(interaction_matrix)):
    for j in range(i + 1, len(interaction_matrix[i])):
        if interaction_matrix[i][j] == 1:
            edges.append((i, j))

# Створення графу на основі ребер
G = nx.Graph(edges)

# Візуалізація графу
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size=800)
plt.title("Граф взаємодії команди")
plt.show()
```

Лістинг 3.25 – лістинг прикладу розв’язанням системи диференціальних рівнянь методом Ейлера може виглядати так

```
import numpy as np
import matplotlib.pyplot as plt

# Початкові значення кількості агентів
A1 = 100  # Початкова кількість агентів 1
A2 = 50   # Початкова кількість агентів 2
A3 = 30   # Початкова кількість агентів 3

# Коефіцієнти
k1 = 0.01
```

```
k2 = 0.02
k3 = 0.03

# Величина кроку часу та загальний час моделювання
delta_t = 0.1
total_time = 100

# Масиви для збереження значень кількості агентів та часу
time_values = []
A1_values = []
A2_values = []
A3_values = []

# Розв'язання системи рівнянь методом Ейлера
current_time = 0
while current_time <= total_time:
    time_values.append(current_time)
    A1_values.append(A1)
    A2_values.append(A2)
    A3_values.append(A3)

    dA1_dt = -k1 * A1
    dA2_dt = -k2 * A2
    dA3_dt = -k3 * A3

    A1 += dA1_dt * delta_t
    A2 += dA2_dt * delta_t
    A3 += dA3_dt * delta_t

    current_time += delta_t

# Візуалізація результатів
plt.plot(time_values, A1_values, label='Агент 1')
plt.plot(time_values, A2_values, label='Агент 2')
plt.plot(time_values, A3_values, label='Агент 3')
plt.xlabel('Час')
plt.ylabel('Кількість агентів')
plt.legend()
plt.show()
```

Лістинг 3.26 – лістинг розв'язання цієї системи за допомогою методу Ейлера може виглядати наступним чином.

```
# Початкові значення
S1 = 1.0
S2 = 2.0
alpha1 = 0.2
alpha2 = 0.1
delta_t = 0.1  # Величина кроку часу
total_time = 10.0  # Загальний час моделювання

# Масиви для збереження значень навичок в часі
S1_values = []
S2_values = []
time_values = []

# Розв'язання системи рівнянь методом Ейлера
current_time = 0.0
while current_time <= total_time:
    S1_values.append(S1)
    S2_values.append(S2)
    time_values.append(current_time)

    dS1_dt = alpha1 * (S2 - S1)
    dS2_dt = alpha2 * (S1 - S2)

    S1 += dS1_dt * delta_t
    S2 += dS2_dt * delta_t

    current_time += delta_t

# Виведення результатів
import matplotlib.pyplot as plt

plt.plot(time_values, S1_values, label='Агент 1')
plt.plot(time_values, S2_values, label='Агент 2')
plt.xlabel('Час')
plt.ylabel('Навички')
plt.legend()
plt.show()
```

Лістинг 3.27 – лістинг прикладу розв'язання системи диференціальних рівнянь за допомогою методу Ейлера у Python

```
nd() plt.tight_layout() plt.show()
import numpy as np
import matplotlib.pyplot as plt

# Початкові значення кількості агентів
N1 = 10  # Початкова кількість агентів 1
N2 = 10  # Початкова кількість агентів 2

# Коефіцієнт оптимізації
alpha = 0.01

# Величина кроку часу та загальний час моделювання
delta_t = 0.1
total_time = 100

# Масиви для збереження значень кількості агентів та часу
time_values = []
N1_values = []
N2_values = []
T_total_values = []

# Розв'язання системи рівнянь методом Ейлера
current_time = 0
while current_time <= total_time:
    time_values.append(current_time)
    N1_values.append(N1)
    N2_values.append(N2)

    # Обчислення об'єктивної функції (загальний час)
    T_total = N1 * T1 + N2 * T2
    T_total_values.append(T_total)

    # Обчислення часткових похідних об'єктивної функції
    dT1_dN1 = T1
    dT2_dN2 = T2

    # Оновлення кількості агентів
    dN1_dt = alpha * dT_total_dN1
    dN2_dt = alpha * dT_total_dN2

    N1 += dN1_dt * delta_t
    N2 += dN2_dt * delta_t
```

```
    current_time += delta_t

# Візуалізація результатів
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(time_values, N1_values, label='Кількість агентів 1')
plt.plot(time_values, N2_values, label='Кількість агентів 2')
plt.xlabel('Час')
plt.ylabel('Кількість агентів')
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(time_values, T_total_values, label='Загальний час')
plt.xlabel('Час')
plt.ylabel('Загальний час')
plt.legend()

plt.tight_layout()
plt.show()
```

Лістинг 3.28 – лістинг

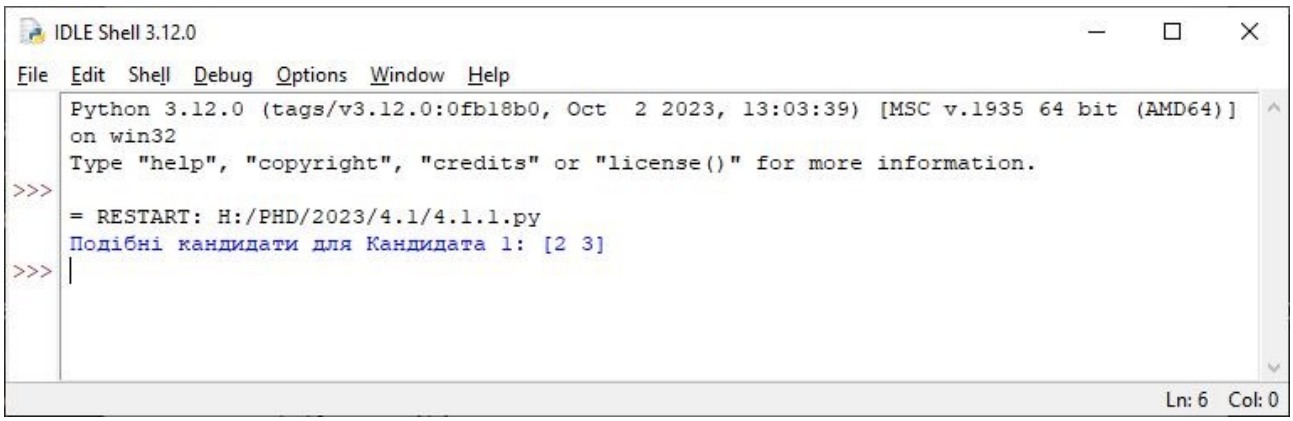
```
import numpy as np
import matplotlib.pyplot as plt

# Початкові значення
L = 5  # Початкова кількість лідерів
F = 10  # Початкова кількість послідовників
alpha_L = 0.1
alpha_F = 0.2
L_target = 8  # Бажана кількість лідерів
F_target = 15  # Бажана кількість послідовників
delta_t = 0.1
total_time = 100

# Масиви для збереження значень
time_values = []
L_values = []
F_values = []

# Розв'язання системи рівнянь методом Ейлера
current_time = 0
while current_time <= total_time:
    time_values.append(current_time)
    L_values.append(L)
    F_values.append(F)

    # Обчислення зміни кількості лідерів та послідовників
    dL_dt = alpha_L * (L_target - L)
    dF_dt = alpha_F * (F_target - F)

    L += dL_dt * delta_t
    F += dF_dt * delta_t

    current_time += delta_t

# Візуалізація результатів
plt.figure(figsize=(12, 6))
plt.plot(time_values, L_values, label='Лідери')
plt.plot(time_values, F_values, label='Послідовники')
plt.xlabel('Час')
plt.ylabel('Кількість агентів')
plt.legend()
plt.show()
```

Додаток Г

Додатки до 4-го розділу дисертації

Лістинг 4.1 – лістинг реалізації фільтрації на основі спільних дій (Collaborative Filtering) для визначення подібних кандидатів на основі успішних командних взаємодій

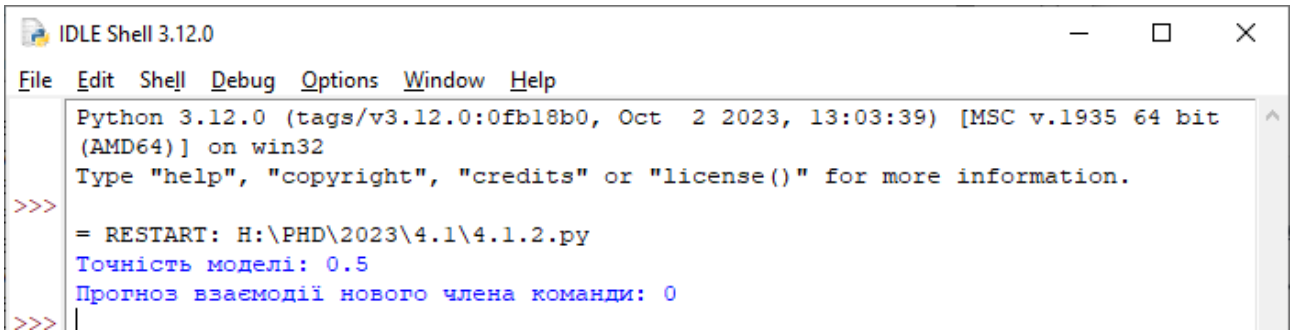
```
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
# Припустимо, що у нас є дані про взаємодію між членами команди (0 - не взаємодія, 1 - взаємодія)
team_interaction_matrix = np.array([
    [1, 1, 0, 1, 0],  # Кандидат 1
    [0, 1, 1, 0, 1],  # Кандидат 2
    [1, 0, 0, 1, 1],  # Кандидат 3
    [0, 1, 0, 0, 1],  # Кандидат 4
    # і так далі
])
# Визначимо подібність між кандидатами на основі косинусної схожості
similarity_matrix = cosine_similarity(team_interaction_matrix)
# Функція для знаходження подібних кандидатів для заданого кандидата
def find_similar_candidates(candidate_index, n=2):
    candidate_similarity = similarity_matrix[candidate_index]
    similar_candidates = np.argsort(candidate_similarity)[::-1][1:n+1]  # Вибираємо топ-N подібних кандидатів
    return similar_candidates
# Приклад використання: знаходження двох подібних кандидатів для Кандидата 1
similar_candidates_for_candidate1 = find_similar_candidates(0, n=2)
print(f'Подібні кандидати для Кандидата 1: {similar_candidates_for_candidate1}")
```



Лістинг 4.2 – лістинг прикладу реалізації сценарію використання моделей машинного навчання для прогнозування ефективності взаємодії між членами команди

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# Припустимо, що у нас є дані про роботу команди, де кожен запис містить характеристики членів команди та їх результат
# Ось спрощений приклад:
# Ознаки (характеристики членів команди)
features = [
    [3, 25, 5],  # Параметри для члена команди 1
    [5, 30, 7],  # Параметри для члена команди 2
    [4, 28, 6],  # Параметри для члена команди 3
    # і так далі
]
# Результат (успішність взаємодії, 1 - успішно, 0 - невдача)
results = [1, 0, 1, 0, 1, 1, 0, 1, 1, 0]  # Приклад результатів для 10 команд
# Розділімо дані на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, results, test_size=0.2, random_state=42)
```

```
# Створимо модель класифікації (наприклад, RandomForestClassifier)
model = RandomForestClassifier()
# Тренуємо модель на навчальних даних
model.fit(X_train, y_train)
# Передбачаємо результати на тестових даних
predictions = model.predict(X_test)
# Оцінюємо точність моделі
accuracy = accuracy_score(y_test, predictions)
print(f"Точність моделі: {accuracy}")
# Тепер, можемо використовувати натреновану модель для прогнозування взаємодії нових членів команди
new_member = [[4, 29, 6]]  # Параметри нового члена команди
prediction_new_member = model.predict(new_member)
print(f"Прогноз взаємодії нового члена команди: {prediction_new_member[0]}")
```

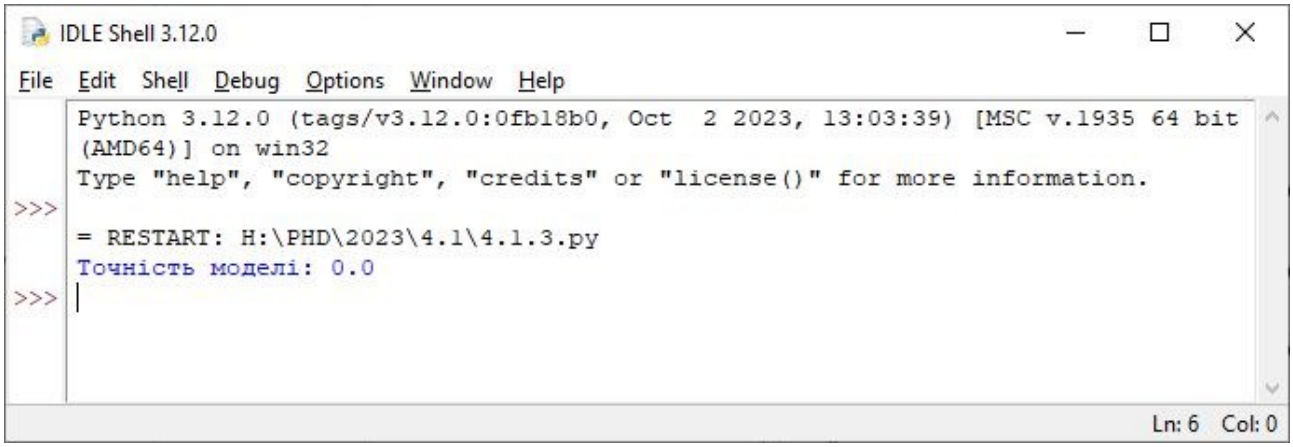


Лістинг 4.3 – лістинг розрахунків, стандартизуємо ознаки, створюємо та тренуємо модель (наприклад, Logistic Regression). Передбачаємо результати на тестових даних та оцінюємо точність моделі

```
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
# Припустимо, що у нас є дані про членів команди, де кожен запис містить навички, досвід та історію взаємодії
# Ось спрощений приклад:
# Навички, досвід та історія взаємодії (приклад)
features = np.array([
    [5, 3, 8],  # Навички, досвід, історія взаємодії для члена команди 1
    [7, 5, 6],  # Навички, досвід, історія взаємодії для члена команди 2
    [4, 6, 7],  # Навички, досвід, історія взаємодії для члена команди 3
    # і так далі
])
# Результат (успішність взаємодії, 1 - успішно, 0 - невдача)
results = np.array([1, 0, 1, 0, 1, 1, 0, 1, 1, 0])  # Приклад результатів для 10 команд
# Зважені коефіцієнти для навичок, досвіду та історії взаємодії
weights = np.array([0.4, 0.3, 0.3])
# Застосовуємо зважені коефіцієнти до ознак
weighted_features = features * weights
# Розділімо дані на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(weighted_features, results, test_size=0.2, random_state=42)
# Стандартизуємо ознаки
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Створимо та тренуємо модель (наприклад, Logistic Regression)
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
# Передбачаємо результати на тестових даних
predictions = model.predict(X_test_scaled)
# Оцінюємо точність моделі
accuracy = accuracy_score(y_test, predictions)
```

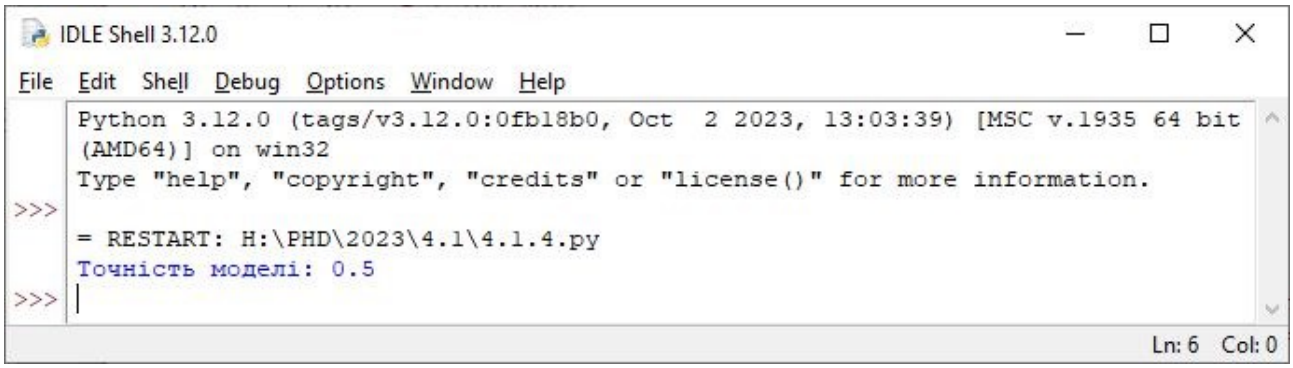
print(f"Точність моделі: {accuracy}")



Лістинг 4.4 – лістинг прикладу реалізації сценарію використання алгоритмів аналізу здатності до командної роботи для визначення, як добре кандидат впишеться в команду, проведемо розрахунки з використанням умовних даних про кандидатів та їх оцінки за атрибутами, які впливають на командну роботу. Оцінюємо точність моделі

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Припустимо, що у нас є дані про кандидатів та їх оцінки за атрибутами, які впливають на командну роботу
# Ось спрощений приклад:
# Оцінки атрибутів (приклад)
attributes = np.array([
    [5, 8, 7, 9],  # Атрибути кандидата 1
    [6, 7, 8, 6],  # Атрибути кандидата 2
    [8, 6, 9, 7],  # Атрибути кандидата 3
    # і так далі
])
# Результат (вплив на командну роботу, 1 - позитивний вплив, 0 - негативний вплив)
```

```
impact_on_teamwork = np.array([1, 0, 1, 0, 1, 1, 0, 1, 1, 0])  # Приклад результатів для 10 кандидатів
# Розділімо дані на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(attributes, impact_on_teamwork, test_size=0.2, random_state=42)
# Створимо та тренуємо модель (наприклад, Random Forest Classifier)
model = RandomForestClassifier()
model.fit(X_train, y_train)
# Передбачаємо результати на тестових даних
predictions = model.predict(X_test)
# Оцінюємо точність моделі
accuracy = accuracy_score(y_test, predictions)
print(f"Точність моделі: {accuracy}")
```

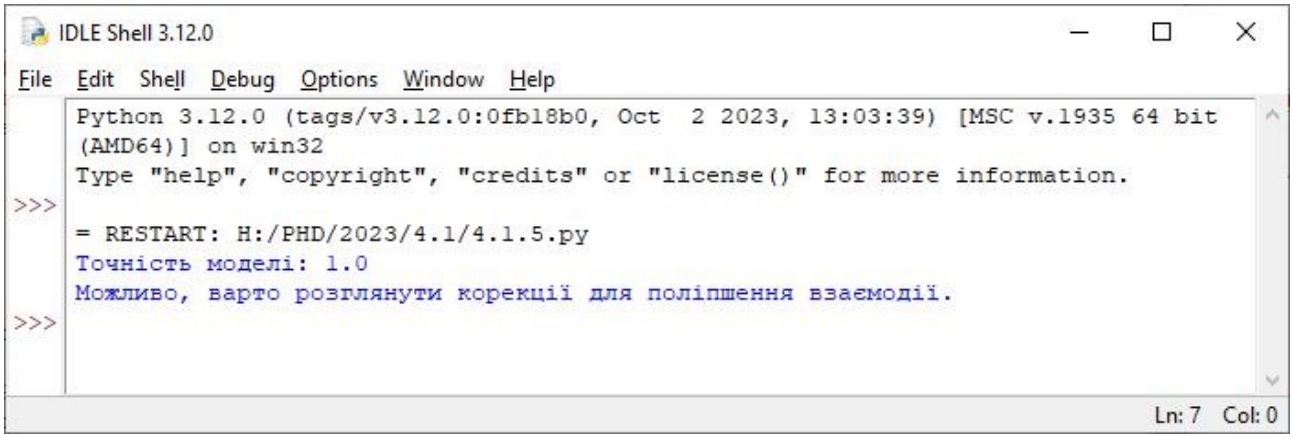


Лістинг 4.5 – лістинг розрахунків з використанням алгоритму рекомендацій для кожного члена команди із застосуванням навченої моделі для прогнозу взаємодії. Визначаємо точність моделі.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Припустимо, що у нас є дані про характеристики членів команди та їх взаємодію
# features - характеристики (наприклад, навички, досвід)
```

```
# interaction - взаємодія (1 - успішна взаємодія, 0 - не успішна)
features = [
    [3, 5, 2],  # Член команди 1
    [4, 6, 3],  # Член команди 2
    [2, 3, 1],  # Член команди 3
    # і так далі
]
interaction = [1, 0, 1]  # Приклад міток успішної взаємодії (може бути більше даних)
# Розділити дані на навчальний і тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, interaction, test_size=0.2, random_state=42)
# Навчання моделі
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
# Прогнози для тестового набору
predictions = model.predict(X_test)
# Оцінка точності
accuracy = accuracy_score(y_test, predictions)
print(f"Точність моделі: {accuracy}")
# Функція для генерації персоналізованих рекомендацій
def generate_personalized_recommendations(member_features):
    # Застосування навченої моделі для прогнозу взаємодії
    prediction = model.predict([member_features])[0]
    if prediction == 1:
        return "Рекомендуємо продовжувати успішну взаємодію!"
    else:
        return "Можливо, варто розглянути корекції для поліпшення взаємодії."
# Приклад використання: генерація рекомендацій для нового члена команди
new_member_features = [4, 5, 2]
recommendations = generate_personalized_recommendations(new_member_features)
```

print(recommendations)



Таблиця 4.1 - Матриця порівняння, де рядки представляють вимоги, а стовпці — компетентності претендента

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Лідерські навички	Комунікація	1	2	2	3	1	5	5	5	3	1	3	1	1	4	3	1	5	2	5	4
	Мотивація	3	1	1	1	3	3	2	5	3	2	1	3	5	5	3	2	1	3	1	4
	Планування та стратегія	3	5	5	5	2	1	1	2	1	1	1	5	5	2	2	2	3	2	3	3
	Рішучість	4	2	4	5	4	1	1	1	5	2	2	1	4	2	3	1	2	3	3	3
	Співпраця	1	2	3	4	3	5	1	1	4	4	5	3	5	3	4	4	3	5	1	2
Організаційні здібності	Планування процесів	1	3	4	5	5	3	5	5	4	5	1	4	4	1	3	3	3	3	1	4
	Ресурсоуправління	2	5	5	1	3	1	2	5	1	1	5	2	3	2	1	2	3	2	5	1
	Керування проектами	2	5	1	2	1	3	4	3	1	4	4	4	3	5	1	5	2	3	5	3
	Моніторинг та оцінка	5	2	3	1	3	3	2	4	5	4	4	4	2	2	4	3	2	2	3	4
	Тайм-менеджмент	4	5	4	2	3	2	5	3	4	2	3	2	3	5	3	4	1	4	5	3
Технічний досвід	Розуміння технологій	4	1	4	1	1	4	5	4	3	5	5	5	5	3	4	3	1	4	1	4
	Аналітичні навички	2	3	2	3	3	3	1	2	2	1	2	2	1	5	5	3	4	4	2	5
	Керування проектами в галузі IT	1	1	5	1	1	5	5	1	3	4	4	3	5	5	4	5	4	2	5	1
	Лідерство в технічних командах	4	2	5	5	5	1	5	5	5	1	5	1	5	5	5	3	2	2	1	4
	Кібербезпека та захист даних	2	3	3	2	2	2	4	5	1	2	4	2	5	3	4	4	1	5	3	2
Комунікативність	Технічна комунікація	2	2	4	2	2	2	4	5	1	1	3	5	2	1	5	4	2	5	3	2
	Вміння слухати	1	4	2	5	1	3	3	4	5	2	5	1	4	5	1	3	3	4	3	2
	Документація	5	4	5	5	3	4	4	3	3	5	1	3	1	4	4	1	1	5	3	1
	Керування конфліктами	2	3	2	5	2	5	2	1	1	5	2	1	3	4	5	3	1	5	5	1
	Ефективне ведення зборів	4	5	1	3	4	1	4	1	3	1	2	2	4	3	5	1	3	4	2	3

1 - Здатність розробляти плани проекту, визначати завдання та керувати ресурсами для досягнення мети.

2 - Навички розробки та впровадження графіків робіт, враховуючи терміни та обсяги робіт.

3 - Здатність ефективно управляти фінансовими ресурсами проекту та визначати бюджет.

4 - Знання та управління ризиками проекту, виявлення та зменшення можливих проблем.

5 - Здатність мотивувати та спілкуватися з різними членами команди, стимулювати ефективну комунікацію.

6 - Здатність вести команду, надихати та створювати сприятливий робочий клімат.

7 - Навички вирішення конфліктів у команді та взаємодії зі стейкхолдерами.

8 - Взаємодія з різними сторонами, врахування їхніх інтересів та потреб.

9 - Ефективне розподілення та використання ресурсів для максимального вигоди проекту.

10 - Розуміння основних технологій та їх вплив на проект.

11 - Досвід роботи з Agile-підходами та впровадження Scrum для покращення продуктивності.

12 - Навички впровадження стратегій тестування для забезпечення якості продукту.

13 - Здатність відстежувати та оцінювати прогрес виконання робіт у проекті.

14 - Здатність писати докладну та зрозумілу документацію проекту.

15 - Здатність ефективно презентувати результати проекту перед стейкхолдерами.

16 - Здатність побудови позитивних взаємин із членами команди та стейкхолдерами.

17 - Здатність розуміти та враховувати вимоги клієнтів та користувачів.

18 - Освіченість у теорії та практиці управління проектами.

19 - Навички виявлення та вирішення технічних та організаційних проблем.

20 - Здатність ефективно управляти своїм часом та завданнями, вибудовуючи продуктивну робочу рутину.