Вісник ЛДУБЖД
Bulletin of Lviv State University of Life Safety
https://journal.ldubgd.edu.ua/index.php/Visnuk
ISSN 2078-4643 (print), ISSN 2708-1389 (online)

UDC 004.942:658.5
DOI: 10.32447/20784643.29.2024.04

Ю. С. Назар, О. В. Придатко
Львівський державний університет безпеки життєдіяльності, м. Львів, Україна
ORCID: https://orcid.org/0000-0003-0151-8285 – Ю. С. Назар
https://orcid.org/0000-0002-0719-9118 – О. В. Придатко
kordunovayulia@gmail.com

МОДЕЛЮВАННЯ ПРОЦЕСУ ОБХОДУ МЕРЕЖЕВОГО ГРАФА ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ КОРОТКОСТРОКОВОГО ПЛАНУВАННЯ ІТ-ПРОЄКТІВ

Вступ. З практики застосування мережевих графів відома їх ефективність в управлінні проєктами для довгострокового планування та оцінки стану виконання проєкту на поточному етапі. Мережевий граф дає змогу розрахувати ключові показники для знаходження критичного шляху виконання робіт, а також резерви часу для кожної роботи окремо. Цей метод добре зарекомендував себе в управлінні масштабними проєктами, проте вивчення ефективності його застосування для короткострокового планування розробки програмного забезпечення у динамічних умовах (умовах постійних змін до змісту, людських та часових ресурсів) досліджено не достатньо. Крім того складність використання стандартних підходів проєктного менеджменту до управління процесом розроблення спеціалізованого програмного забезпечення у динамічному оточенні (умовах постійних змін) спонукає до вирішення актуальної задачі щодо дослідження нових, науково-обґрунтованих методів та підходів для оцінки тривалості розроблення спеціалізованого програмного забезпечення, які будуть адаптованими під специфіку роботи у динамічних умовах із значною часткою невизначеності та ресурсних обмежень (людських та часових).

Мета. Моделювання процесів обходу мережевого графа для розрахунку ключових параметрів мережевої моделі, що дасть змогу швидко та якісно оцінити тривалість проєктних робіт в умовах динамічності ресурсів та змісту робіт.

Методи досліджень. Для досягнення поставленої мети в роботі використано методи імітаційного моделювання дискретних систем засобами мережі Петрі. Зважаючи на те, що об'єктом моделювання є процес обходу мережевого графа, для опису його ключових властивостей використано понятійний апарат теорії графів.

Основні результати дослідження. Отримано уніфіковані моделі процесів обходу мережевих графів різної складності та конфігурації, що дає змогу вирішувати задачі короткострокового планування проєктів з розроблення безпеко-орієнтованих сервісів у динамічних умовах шляхом розрахунку раннього та пізнього термінів виконання подій та з одержанням на їх основі характеристики про тривалість проєктних робіт. Одержані моделі дали змогу побудувати алгоритми обходу мережевого графа для оцінки тривалості виконання проєктних робіт з подальшою автоматизацією процесів короткострокового планування за умови введення будь-яких змін до змісту та обсягу окремих спринтів проєкту або складу команди розробників.

Висновки. Шляхом імітаційного моделювання з використанням понятійного апарату мереж Петрі отримано уніфіковану модель процесів обходу мережевих графів, яка полягає у циклічності процедури встановлення зв'язку між попередньою і поточною подією та уможливлює її адаптивне застосування не залежно від складності та конфігурації мережевих графів.

Шляхом алгоритмічного відтворення розроблених імітаційних моделей побудовано чітку процедуру розрахунку раннього та пізнього термінів виконання подій, що дає змогу провести автоматизацію цих процесів для вирішення задач короткострокового планування проєктів з розроблення спеціалізованого програмного забезпечення.

Ключові слова: безпеко-орієнтовані системи, мережі Петрі, мережевий граф.

Yu. S. Nazar, O.V. Prydatko
Lviv State University of Life Safety, Lviv, Ukraine

MODELLING THE NETWORK GRAPH TRAVERSAL PROCESS FOR SOLVING SHORT-TERM IT PROJECT PLANNING TASKS

Introduction. The practice of using network graphs is known for their effectiveness in project management for long-term planning and assessment of the project's current status. The network graph allows you to calculate the main indicators for finding the critical path of work, as well as time reserves for each work separately. This method has proven

itself in managing large-scale projects, but the effectiveness of its application for short-term software development planning in dynamic conditions (conditions of constant changes in content, human and time resources) has not been sufficiently studied. In addition, the difficulty of using standard project management approaches to managing the process of developing specialised software in a dynamic environment (conditions of constant change) leads to the urgent task of researching new, scientifically based methods and approaches for estimating the duration of specialised software development, which will be adapted to the specifics of work in dynamic conditions with a significant degree of uncertainty and resource constraints (human and time resources).

Purpose. Simulating the processes of network graph traversal to calculate the main parameters of the network model will allow for quickly and accurately assessing the duration of project work in the context of dynamic resources and work content.

Methods. To achieve this goal, we used the simulation modelling methods of discrete systems using a Petri net. Given that the object of modelling is the process of traversing a network graph, the conceptual framework of graph theory is used to describe its key properties.

Results. Unified models of network graph traversal processes of varying complexity and configuration have been obtained, which allows solving the problems of short-term planning of projects for the development of safety-oriented services in dynamic conditions by calculating the early and late times of events and obtaining characteristics of the duration of project work based on them. The obtained models made it possible to build algorithms for traversing the network graph to estimate the duration of project work with subsequent automation of short-term planning processes, subject to any changes to the content and scope of individual project sprints or the composition of the development team.

Conclusions. A unified model of network graph traversal processes was obtained through simulation modelling using the conceptual apparatus of Petri nets. It consists of the cyclic procedure for establishing a connection between the previous and current events and allows its adaptive application regardless of the complexity and configuration of network graphs.

Through algorithmic reproduction of the developed simulation models, a clear procedure for calculating the early and late deadlines for events has been built, which allows for the automation of these processes to solve the problems of short-term planning of specialised software development projects.

Keywords: safety-oriented systems, Petri nets, network graph.

Вступ. Оцінка часу створення спеціалізованого програмного забезпечення є важливою задачею у процесах управління IT-проєктами, адже дає змогу оцінити тривалість робіт за умови визначеного переліку завдань та ресурсного обмеження. Проте процеси розроблення спеціалізованого програмного забезпечення безпеко-орієнтованого спрямування характеризуються динамічним оточенням. Постійні зміни до вимог, динамічність часових та матеріальних ресурсів та супутні ризики – це ті фактори, які мають значний вплив на терміни реалізації та випуск нового релізу спеціалізованого програмного продукту. Сьогодні в галузі менеджменту IT-проєктів відомі два основних підходи до управління процесами розроблення програмного забезпечення – гнучкий та каскадний, кожен з яких володіє своїми принципами та обмеженнями. Зокрема, при гнучкій розробці фіксованими є команда розробників та час виконання, а вимоги – змінні. При каскадній – навпаки фіксованим є чіткий перелік робіт (вимог до продукту), а команда і час здебільшого можуть змінюватись [11].

З досвіду розроблення спеціалізованих безпеко-орієнтованих сервісів та програмного забезпечення, які реалізуються працівниками Державної служби України з надзвичайних ситуацій, було виявлено проблему неефективності згаданих моделей управління життєвим циклом програмного забезпечення (у тому числі на етапі розроблення), оскільки динамічністю характеризуються не лише обсяги і зміст робіт, а також і часові ресурси [12]. Саме тому актуальності набуває науково-прикладна задача дослідження нових, науково-обґрунтованих методів та підходів для оцінки тривалості розроблення спеціалізованого програмного забезпечення, які будуть адаптованими під специфіку роботи у динамічних умовах із значною часткою невизначеності та ресурсних обмежень (людських та часових).

Аналіз наукових досягнень в предметній області. У світовій та національній науковій літературі трапляється значна кількість праць, присвячених питанню короткострокового планування процесу розробки програмного забезпечення. Зокрема, у роботі [1] доведено, що планування займає центральне місце у Agile проектах, а всі інші сфери обертаються навколо нього. Роботи [2, 3, 4] спрямовані на використання експертного методу оцінки часових ресурсів за Agile методологією. У статті [5] запропоновано використання інструменту блок-схеми для прийняття рішень у міждисциплінарному дослідницькому співробітництві, а у роботі [6] було удосконалено регресійне рівняння для оцінювання трудомісткості розробки програмного забезпечення, створеного за гнучкою методологією. Наукова праця [7] присвячена оцінюванню користувацьких історій на основі декомпозиції складності за допомогою байєсівських мереж. У роботі [14] автори представили метод синхронного виконання етапів життєвого циклу програмного забезпечення в рамках DevOps процесів.

Також є низка робіт [8, 9, 10], присвячених використанню мережевих графів у процесі планування проекту. Зокрема, автори у роботі [9] обрали метод вирішення задачі планування, що заснований на застосовуванні методу мережевого планування та базується на ідеї оптимізації критичного шляху із залученням додаткових обмежених коштів. Проте описаний підхід не корелює із процесами розроблення спеціалізованого програмного забезпечення. У праці [8] описано процес розроблення комп'ютерної програми розв'язання задач мережевої оптимізації, проте розрахунок найкоротших шляхів проходить за алгоритмом Дейкстри, який не є універсальним та не може застосовуватись для мережевого планування процесу розробки програмного забезпечення. В основу наукової праці [10] закладено методи мережевого планування PERT, використання елементів теорії графів та методу діаграм Ганта. Цей підхід актуальний для каскадної моделі управління, якому притаманне послідовне виконання чітко визначених завдань за означений час.

Постановка проблеми. Формулювання мети. У працях [9, 10, 12] означено, що мережеві графи дають змогу відобразити роботи проєкту та зв'язки між ними. Такі графи зазвичай використовують в управлінні проєктами для довгострокового планування та оцінки стану виконання проєкту на поточному етапі. Найпопулярнішим алгоритмом мережевого планування є метод PERT (Project Evaluation and Review Technique). Ця мережа передбачає оцінку кожної роботи із використанням трьох тимчасових станів: оптимістичний, песимістичний та найбільш ймовірний час. Як відомо, мережевий граф дає змогу розрахувати ключові показники для знаходження критичного шляху виконання робіт, а також резерви часу для кожної роботи окремо. Цей метод добре зарекомендував себе в управлінні масштабними проєктами, проте вивчення ефективності його застосування для короткострокового планування розробки програмного забезпечення у динамічних умовах досліджено не достатньо. Крім того, попередні наукові досягнення вказують на складність використання стандартних підходів проєктного менеджменту (гнучкі та каскадні) до управління процесом розроблення спеціалізованого програмного забезпечення у динамічному оточенні. Зважаючи на зазначене, в роботі поставлено **мету** моделювання процесів обходу мережевого графа для розрахунку ключових параметрів мережевої моделі, що дасть змогу швидко та якісно оцінити тривалість проєктних робіт в умовах динамічності ресурсів та змісту робіт. Моделювання дискретних процесів мережевого планування допоможе сформувати повноцінну уяву щодо застосування PERT-підходів для задач короткострокового планування проєктів з розроблення безпеко-орієнтованих сервісів, які володіють значним показником динамічності. Одержання моделі є визначальним етапом у побудові алгоритмів обходу мережевого графа для оцінки тривалості виконання проєктних робіт. Своє чергою, розроблені алгоритми дадуть змогу надалі автоматизувати процеси короткострокового планування, а також швидко і якісно проводити переоцінку тривалості робіт за умови введення будь яких змін до змісту, обсягу, часу виконання окремих спринтів проєкту або складу команди розробників.

Методи досліджень. Для досягнення поставленої мети в роботі використано методи імітаційного моделювання дискретних систем засобами мережі Петрі. Зважаючи на те, що об'єктом моделювання є процес обходу мережевого графа, для опису його ключових властивостей використано понятійний апарат теорії графів.

Результати досліджень. Для кращого розуміння об'єкта моделювання – процесу обходу мережевого графа для визначення його ключових параметрів, варто розглянути основні елементи мережевої моделі в контексті гнучкої методології управління ІТ-проєктами. Оскільки в мережевому плануванні основними елементами моделі є роботи та події, то в контексті Agile-технології: робота – це користувацька історія або функція; подія – початок або завершення будь якої роботи (користувацької історії). Події у мережевій моделі позначаються вершинами графа, а роботи – його ребрами. За одиницю вимірювання робіт в процесі моделювання буде використано універсальну відносну величину story points (далі – s.p.), яка враховує поєднання трудомісткості завдання, його складності та ризиків, які з ним пов'язані. Такий різновид оцінки дає змогу враховувати часові та ресурсні обмеження процесу розробки специфічного програмного забезпечення залежно від досвіду команди розробників.

Для виконання процедури мережевого планування (побудови мережевого графа) необхідно [12]:

1. Пріоретизувати користувацькі історії та функції, які необхідно виконати.
2. Оцінити користувацькі історії та функції у s.p.
3. Визначити попередні користувацькі історії чи функції, які мають бути виконані, оскільки існують такі завдання, виконання яких можливе лише після завершення попередніх.

Наявність мережевого графа дає можливість сформувати уяву про орієнтовні обсяги робіт та їх черговість. Проте наявність лише самого графа не дає змоги оцінити тривалість робіт за проєктом. Побудова графа мережевої моделі в початковому

стані не дасть змоги проводити оперативний перерозподіл обсягів робіт із подальшим визначенням тривалості їх виконання в динамічному оточенні. З цією метою математичний апарат мережевого планування володіє низкою методів розрахунку параметрів мережевого графа для часової оцінки процесів розробки специфічного програмного забезпечення на стадії планування, а саме: ранній термін виконання подій T_f; пізній термін виконання подій T_l; критичний шлях виконання робіт; часові резерви $R_{подій}$, $R_{робіт}$. В математичному відношенні визначення цих параметрів не передбачає жодної складності. Проте в алгоритмічному представленні, з метою подальшої автоматизації розрахунків, можуть виникати труднощі, пов'язані з коректністю роботи алгоритму, його безвідмовністю тощо. Це може бути зумовлене складністю самої мережевої моделі, наявністю великої кількості робіт, у тому числі фіктивних, наявністю великої кількості подій, наявністю або відсутністю взаємозв'язків між ними, розгалуженістю мережевої моделі тощо.

Зважаючи на це, процедуру обходу мережевого графа для розрахунку його ключових параметрів, необхідно представити у вигляді уніфікованої імітаційної моделі. Уніфікованість моделі полягатиме у її спроможності відтворення процедури обходу будь якого мережевого графа незалежно від його складності та конфігурації. І найголовніше, уніфікована модель створить передумови для побудови відповідних алгоритмів та автоматизації процесів розрахунку параметрів мережевого графа.

Зважаючи на те, що мережевий граф та процедура його обходу є дискретною системою, для побудови імітаційної моделі процесів визначення параметрів T_f, T_l обрано математичний апарат моделювання розподілених дискретних систем мережами Петрі. Мережі Петрі обрано з точки зору зручності імітаційного відтворення паралельних (розподілених) процесів, а також їх динамічної взаємодії в системі. Саме такими характеристиками володіють мережеві моделі, які досліджуються в попередніх роботах [9, 10, 12].

В дискретній моделі, описаній за допомогою мережі Петрі, події відображаються переходами t, умовами виконання яких є позиції P. На першому етапі реалізується визначення раннього терміну виконання подій. Цей параметр характеризує термін, раніше якого подія відбутись не може, та дає змогу контролювати початок виконання робіт, які не регламентовані порядковістю. Для першої події $T_{f(1)} = 0$, для всіх подальших подій мережевої моделі T_f визначається відповідно до методики.

Отже, відтворимо та опишемо модель процесу обходу мережевого графа з метою визначення раннього часу виконання робіт T_f (рис. 1).

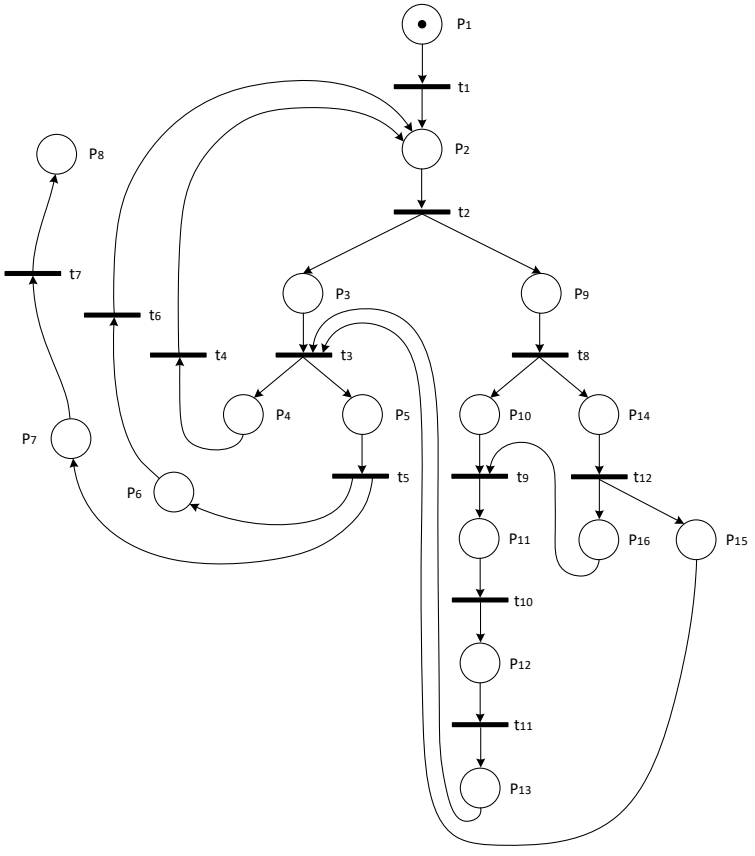

Рисунок 1 – Імітаційна модель процесу обходу мережевого графа з метою визначення раннього часу виконання робіт T_f

Процес обходу мережевого графа для розрахунку раннього терміну виконання подій починається з позиції (умови) P_1, яка відповідає за перевірку наявності початкових даних, а саме: індексу попередньої події $i = 0$; індексу наступної події $j = 1$; обсягу подій мережевої моделі n; множини робіт мережевої моделі $\{t_{i-j}\}$. Перехід (подія) t_1 відповідає за підготовку початкових параметрів циклічного обходу мережевого графа з такими значеннями: ранній термін виконання першої події $T_{f(1)} = 0$; i++; $j = i+1$. Позиція P_2 перевіряє відповідність параметрів циклу на черговій ітерації (в ході імплементації моделі буде реалізовано алгоритмічно). Власне новаційність поданої моделі обходу мережевого графа полягає у циклічності встановлення зв'язків між подіями та виконанні розрахунку раннього терміну виконання подій в межах поточної ітерації. Перехід t_2 реалізує процедуру входження у цикл та визначення поточного ребра (роботи) $i-j$. Позиція P_3 перевіряє умову чи ребро $i-j$ не належить множині $\{t_{i-j}\}$ (тобто мережевим графіком робота $i-j$ не передбачена): $i-j \notin \{t_{i-j}\}$. Перехід t_3 відповідає за визначення поточного значення події j, адже на кожній ітерації циклу значення індексу наступної події j буде змінюватись. В позиції P_4 проводиться перевірка, чи значення j менше або однакове з кількістю подій мережевої моделі n ($j \le n$). Якщо нерівність $j \le n$ виконується, здійснюється перехід t_4, де значення індексу j збільшується на 1 (j++). Після чого процедура обходу мережевого графа повертається у позицію P_2. Позиція P_5 перевіряє зворотну умову, коли значення j більше за кількість подій мережевої моделі n. За умови виконання нерівності $j > n$, здійснюється перехід t_5, де індекс i збільшується на 1 (i++). В позиції P_6 здійснюється перевірка не рівності індексу попередньої події i та кількості подій мережевої моделі n: $i \neq n$ (не досягнуто граничного значення циклу). Якщо не рівність виконується – спрацьовує перехід t_6, який відповідає за встановлення нового значення індексу $j = i+1$. Після чого здійснюється циклічне повернення до перевірки умови P_2. Позиція P_7 перевіряє рівність індексу попередньої події i та кількості подій мережевої моделі n. Якщо значення індексу i досягає n (досягнуто граничного значення циклу), то виконується перехід t_7, із виводом результатів розрахунку та подальшим завершенням обходу мережевого графа у позиції P_8.

Повернемось до розгалуження моделі після виконання переходу t_2 (вхід у цикл та перевірка поточного значення i - j). В позиції P_9 відбувається перевірка умови, чи поточна робота i - j належить множині $\{t_{i-j}\}$ (вхідними даними мережевої моделі передбачена робота $i-j$). У разі наявності відповідної роботи у заданій множині робіт $i-j \in \{t_{i-j}\}$ виконується перехід t_8, завданням якого є визначення раннього терміну виконання події на поточному етапі обходу мережевого графа для події j за залежністю:

$$T_{f(j)} = T_{f(i)} + t_{(i \to j)}, \qquad (1)$$

де $T_{f(i)}$ – ранній термін виконання попередньої події; $t_{(i \to j)}$ – обсяг роботи, яка передує події j (враховує складність та обсяги робіт у Story Point).

Далі у позиції P_{10} здійснюється перевірка, чи значення розрахованого раннього терміну $T_{f(j)}$ на поточній ітерації циклу не міститься у множині ранніх термінів виконання подій: $T_{f(j)} \notin \{T_{fj}\}$. Якщо поточне значення раннього терміну $T_{f(j)}$ не міститься у множині, це означає, що значення отримано вперше. Після чого в переході t_9 відбувається визначення попередньої події T_i для поточної події T_j: $T_{prev(j)} = T_i$, (фіксується номер попередньої події для T_j). В позиції P_{11} відбувається перевірка факту запису (наявності) $T_{prev(j)}$ до поточної події T_j (факт запису = true). В переході t_{10} відбувається запис зв'язку поточна подія T_j – попередня подія $T_{prev(j)}$, до *HashMap* $<T_j, T_{prev(j)}>$. Позиція P_{12} перевіряє факт запису зв'язку $T_j - T_{prev(j)}$ до *HashMap* $< T_j, T_{prev(j)}>$ (факт запису = true). Визначене унікальне значення раннього терміну виконання подій $T_{f(j)}$ на поточній ітерації циклу записується до множини ранніх термінів *HashSet* $\{T_{f(j)}\}$ в події t_{11}. В позиції P_{13} перевіряється чи значення $T_{f(j)}$ успішно записано до множини *HashSet* $\{T_{f(j)}\}$ (факт запису = true). За виконання умови P_{13} виконується перехід до t_3 та початок нової ітерації циклу.

Позиція P_{14} перевіряє наявність визначеного у переході t_8 значення раннього терміну виконання подій $T_{f(j)}$ у множині ранніх термінів $\{T_{fj}\}$ після чергової ітерації розрахунків (наявність $T_{f(j)}$ у множині $\{T_{fj}\}$ можлива, якщо події T_j передують декілька подій T_i, а на попередніх ітераціях циклу відбувалось визначення показника їх раннього терміну виконання). У випадку виконання умови P_{14} $T_{f(j)} \in \{T_{fj}\}$ виконується перехід t_{12}, де визначається максимальне значення серед терміну $T_{f(j)}$ поточної ітерації та попередньо записаного значення $T_{f(j)}$ до множини $\{T_{fj}\}$ (за результатами попередніх ітерацій). На цьому етапі необхідно передбачити можливість коректного розрахунку часу раннього початку для тих подій, яким передують декілька робіт з врахуванням вимоги $T_{f(j)} \to \max$, використовуючи вираз:

$$T_{f(j)} = \max \begin{cases} T_{f(k)} + t_{(k \to j)} \\ \cdots\cdots \\ T_{f(m)} + t_{(m \to j)} \end{cases}, \qquad (2)$$

де k, m – це індекси подій, що передують події j.

У позиції P_{15} відбувається перевірка нерівності $\{T_{f(j)}\} \geq T_{f(j)}$, де встановлюється, чи значення раннього терміну виконання подій, яке записано до множини $\{T_{f(j)}\}$ на попередніх ітераціях, більше або рівне поточному розрахованому значенню раннього терміну $T_{f(j)}$. Якщо нерівність виконується то перезапису, збереженого у множині значення поточним, не відбувається (записане значення більше за поточно розраховане), а система переходить на нову ітерацію циклу t_3. В позиції P_{16} відбувається зворотна перевірка нерівності $\{T_{f(j)}\} < T_{f(j)}$. Якщо поточне розраховане значення раннього терміну $T_{f(j)}$ більше за раніше записане значення до множини $\{T_{f(j)}\}$, запускається процедура перезапису більшого значення до $HashSet\{T_{f(j)}\}$ шляхом виконання переходу t_9 із подальшим проходження черговості описаних етапів.

Представлена на рисунку 1 імітаційна модель обходу мережевого графа дала змогу описати процедуру визначення раннього терміну виконання подій $T_{f(j)}$ із урахуванням усіх обмежень у вигляді алгоритму (рис. 2). Імітаційне моделювання процесу обходу мережевого графа дало змогу розробити унікальний алгоритм та адаптувати його під методологію гнучкого управління проєктними роботами. Термін раннього виконання подій для останньої події мережевої моделі n є терміном виконання усього комплексу робіт спринта.

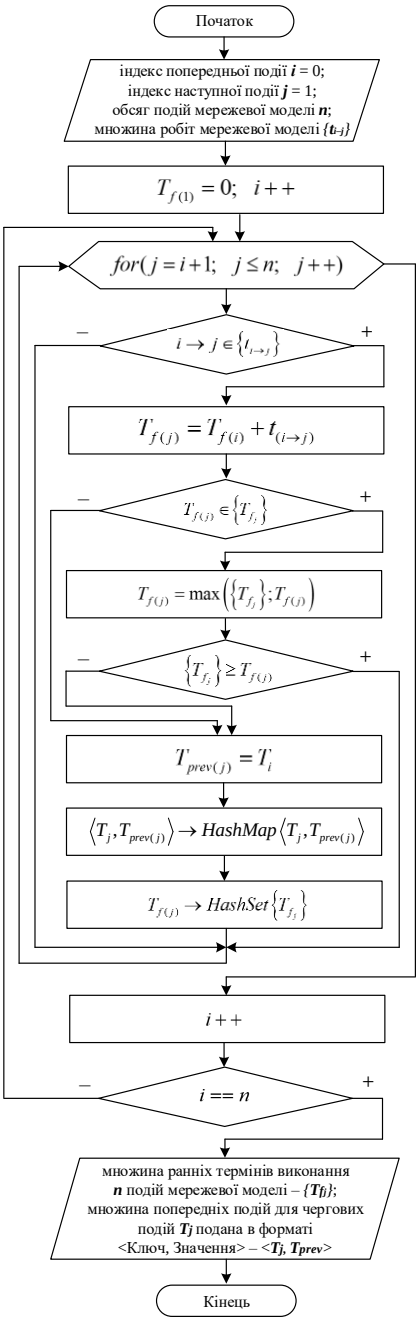

Рисунок 2 – Алгоритм обходу графа мережевої моделі для визначення показника $T_{f(j)}$

Далі проведено імітаційне моделювання процесу обходу мережевого графа з метою визначення пізнього терміну виконання робіт T_l. В основу цієї моделі також закладено принцип циклічного перебору зв'язків $i-j$, проте відмінністю є зворотній порядок обходу (від останньої до першої події). Визначення пізнього терміну виконання подій $T_{l(i)}$ реалізується з метою контролю термінів, до яких події мережевої моделі мають бути повністю виконані, задля уникнення випадків збільшення часу на реалізацію усього проєкту. Розглянемо модель цього процесу у вигляді мережі Петрі на рисунку 3.

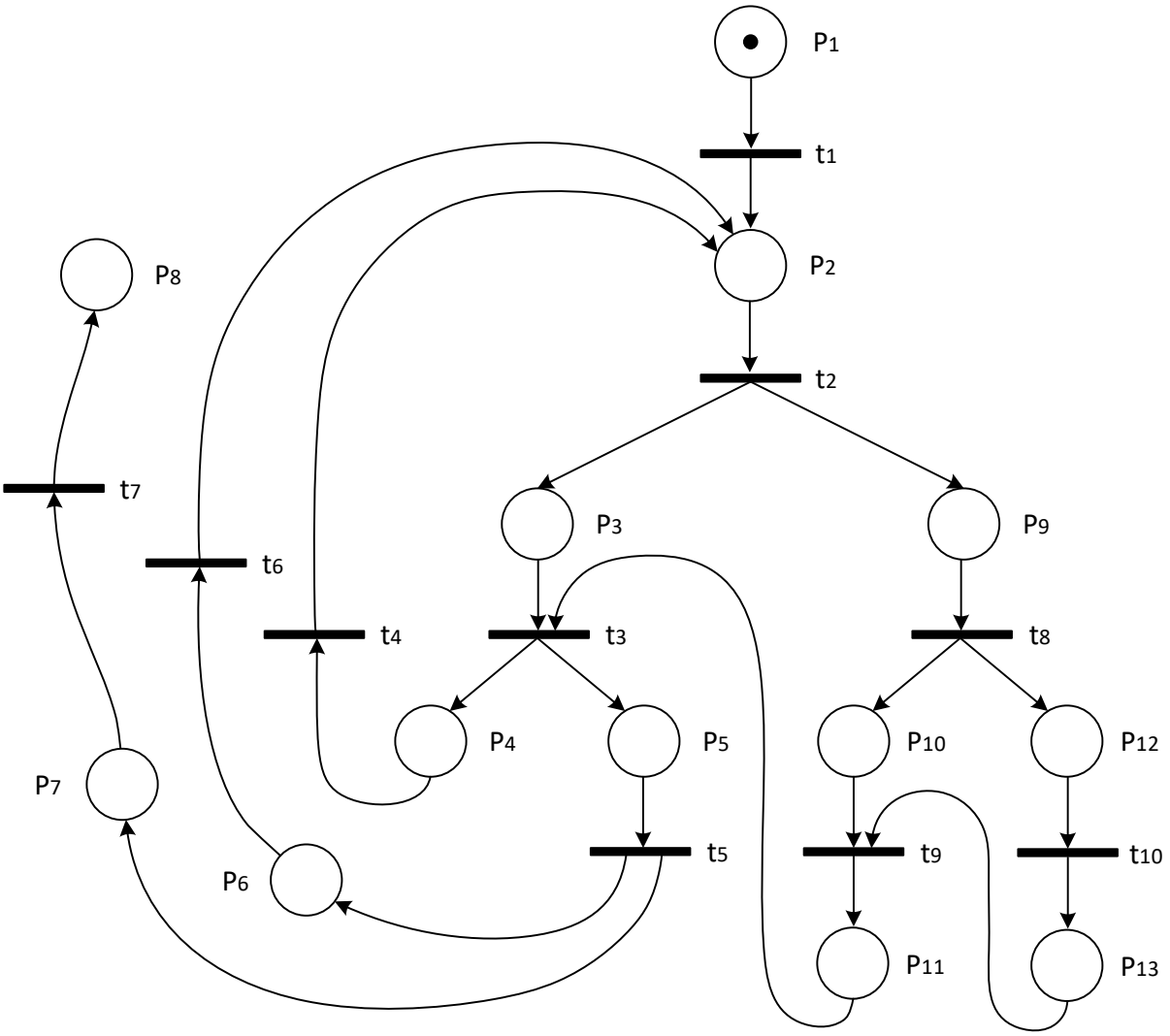

Рисунок 3 – Імітаційна модель процесу обходу мережевого графа з метою визначення пізнього часу виконання робіт T_l

Описаний імітаційною моделлю процес розрахунку пізнього терміну виконання події починається із позиції P_1, яка перевіряє введення даних: індексу попередньої події $i=0$; індексу поточної події $j = n$; обсягу подій мережевої моделі n; множини робіт мережевої моделі $\{t_{i-j}\}$; множини ранніх термінів виконання подій $\{T_{fi}\}$ (розраховано за результатами обходу графа за попередньою моделлю). За умови виконання P_1 здійснюється перехід t_1 – підготовка параметрів циклічного обходу графа, де пізній час виконання для останньої події рівний ранньому часу виконання цієї події $T_{l(n)}=T_{f(n)}$, а значення індексу $i = j–1$. Позиція P_2 перевіряє відповідність параметрів циклу на черговій ітерації. Перехід t_2 реалізує процедуру входження у цикл та визначення поточного ребра (роботи) $i-j$. Циклічна процедура обходу мережевого графа організована за прикладом попередньої моделі, відмінністю є зворотній шлях обходу графа (з кінця до початку). Позиція P_3 перевіряє умову чи ребро $i-j$ не належить множині $\{t_{i-j}\}$: $i-j \notin \{t_{i-j}\}$. Перехід t_3 відповідає за визначення поточного значення попередньої події i, адже на кожній ітерації циклу значення індексу попередньої події i буде зменшуватись. В позиції P_4 проводиться перевірка, чи значення i більше або дорівнює 1 ($i \geq 1$). Якщо нерівність $i \geq 1$ виконується, здійснюється перехід t_4, де значення індексу i зменшується на 1 (i—). Після чого процедура

обходу мережевого графа повертається у позицію P_2. Позиція P_5 перевіряє зворотну умову, коли значення $i < 1$. За умови виконання нерівності $i < 1$, здійснюється перехід t_5, де індекс j зменшується на 1 (j—). В позиції P_6 здійснюється перевірка чи значення індексу поточної події j не досягло 1: $j \neq 1$ (не досягнуто граничного значення циклу). Якщо нерівність виконується спрацьовує перехід t_6, який відповідає за встановлення нового значення індексу $i = j–1$. Після чого здійснюється циклічне повернення до перевірки умови в позиції P_2. Позиція P_7 перевіряє рівність індексу поточної події $j = 1$. Якщо значення індексу j досягає 1 (досягнуто граничного значення циклу), то виконується перехід t_7, із виводом результатів розрахунку та подальшим завершенням обходу мережевого графа у позиції P_8.

Перейдемо до розгалуження моделі після виконання переходу t_2 (вхід у цикл та перевірка поточного значення $i – j$). В позиції P_9 відбувається перевірка умови, чи поточна робота $i – j$ належить множині $\{t_{i\text{-}j}\}$. У разі наявності відповідної роботи у заданій множині робіт $i – j \in \{t_{i\text{-}j}\}$ виконується перехід t_8, завданням якого є визначення пізнього терміну з виразу:

$$T_{l(i)} = T_{l(j)} - t_{(i \to j)}, \tag{3}$$

де $T_{l(j)}$ – пізній термін виконання наступної (поточної) події; $t_{(i \to j)}$ – обсяг роботи, яка виконується після події i та перед подією j (враховує обмеження у Story Point).

Далі у позиції P_{10} здійснюється перевірка, чи значення розрахованого пізнього терміну $T_{l(i)}$ на поточній ітерації циклу не міститься у множині пізніх термінів виконання подій: $T_{l(i)} \notin \{T_{l(i)}\}$. Якщо поточне значення пізнього терміну $T_{l(i)}$ не міститься у множині, це означає, що розраховане значення отримано вперше. Після чого в переході t_9 відбувається запис значення пізнього терміну виконання цієї події $T_{l(i)}$ до множини $HashSet\{T_{li}\}$. В позиції P_{11} відбувається перевірка факту запису (наявності) $T_{l(i)}$ до $HashSet\{T_{li}\}$ (факт запису = true), після чого відбувається перехід на нову ітерацію циклу в переході t_3. Позиція P_{12} перевіряє наявність визначеного у переході t_8 значення пізнього терміну виконання подій $T_{l(i)}$ у множині пізніх термінів $\{T_{l(i)}\}$ після чергової ітерації розрахунків. У випадку виконання умови $T_{l(i)} \in \{T_{l(i)}\}$ виконується перехід t_{10}, де визначається мінімальне значення серед терміну $T_{l(i)}$ поточної ітерації та попередньо записаного значення $T_{l(i)}$ до множини $\{T_{l(i)}\}$, використовуючи вираз:

$$T_{l(i)} = \min \begin{cases} T_{l(k)} - t_{(i \to k)} \\ \cdots\cdots \\ T_{l(m)} - t_{(i \to m)} \end{cases}, \tag{4}$$

де k, m – це індекси похідних подій, від події i.

В позиції P_{13} відбувається перевірка факту визначення мінімального значення $T_{l(i)}$ на поточній ітерації циклу, після чого виконується його перезапис до множини $HashSet\{T_{li}\}$ шляхом повернення до переходу t_9.

Пізній термін T_l для останньої події мережевої моделі дорівнює часу раннього виконання події T_f :

$$T_{l(n\ ост.)} = T_{f(n\ ост.)}. \tag{5}$$

T_l для першої події мережевої моделі, за умови правильного виконання попередніх розрахунків, буде дорівнювати часу раннього виконання цієї події T_f:

$$T_{l(n\ поч.)} = T_{f(n\ поч.)}. \tag{6}$$

Для усіх подій мережевої моделі, крім першої та останньої, має виконуватись нерівність:

$$T_{l(n)} \geq T_{f(n)}. \tag{7}$$

Представлена на рисунку 3 імітаційна модель обходу мережевого графа дала змогу описати процедуру ітераційного визначення пізнього терміну виконання подій $T_{l(i)}$ із урахуванням усіх обмежень у вигляді алгоритму (рис. 4).

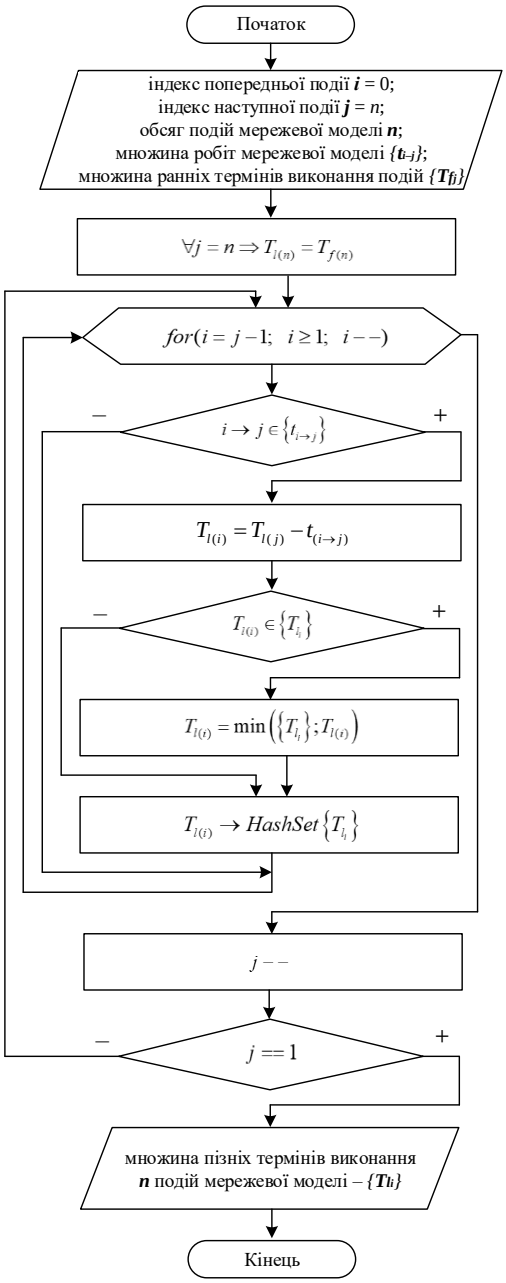

Рисунок 4 – Алгоритм обходу графа мережевої моделі для визначення показника $T_{l(i)}$

Обговорення результатів досліджень. Реалізовані імітаційні моделі та побудовані на їх основі алгоритми обходу мережевого графа для розрахунку раннього та пізнього термінів виконання подій дають змогу оцінити максимальну тривалість робіт з розроблення спеціалізованого програмного забезпечення зважаючи на наявні ресурси, необхідні для успішного виконання спринта. Отримані алгоритми на основі імітаційних моделей надають можливість автоматизувати визначення раннього та пізнього термінів виконання робіт, а відтак реалізувати процедуру оперативного перепланування тривалості проєктних робіт в динамічному оточенні (в умовах постійних змін).

Унікальність імітаційних моделей обходу мережевого графа полягає у циклічності процедури встановлення зв'язку між попередньою та поточною подіями $i - j$. При обході мережевого графа для визначення раннього терміну за умови фіксованого значення індексу попередньої події i здійснюється ітераційне збільшення індексу поточної події j до моменту, коли значення j буде більшим за кількість подій мережевої моделі n. На цьому етапі циклічний перебір завершується, індекс попередньої події i збільшується на одиницю та реалізується процедура повторного входу у цикл із значенням $j = i+1$. Процедура повторного виконання циклу для обходу мережевого графа виконується до моменту, коли значення індексу попередньої події i досягне значення кількості подій мережевої моделі n. При обході мережевого графа для визначення пізнього терміну, процедура має аналогічний підхід, проте у зворотньому порядку (від останньої до першої

події). За цих умов значення індексу *j* на початковому етапі дорівнює *n*, а індекс *i* ітераційно зменшується на 1. Застосування цього підходу дасть змогу почергово перебирати усі ребра (роботи) мережевої моделі із подальшим визначенням необхідних показників.

Висновок. За результатами імітаційного моделювання процесів обходу мережевих графів для визначення значень їх основних параметрів отримано такі результати:

1. Шляхом імітаційного моделювання з використанням понятійного апарату мереж Петрі отримано уніфіковану модель процесів обходу мережевих графів, яка полягає у циклічності процедури встановлення зв'язку між попередньою і поточною подіями та уможливлює її адаптивне застосування не залежно від складності та конфігурації мережевих графів.

2. Шляхом алгоритмічного відтворення розроблених імітаційних моделей побудовано чітку процедуру розрахунку раннього та пізнього термінів виконання подій, що дає змогу провести автоматизацію цих процесів для вирішення задач короткострокового планування проєктів з розроблення спеціалізованого програмного забезпечення.

Список літератури:

1. Lamsellak H., Khalil A., Belkasmi M.G., Saber M. Agile Practices in Iteration Planning Process of Global Software Development. *International Conference on Advanced Intelligent Systems for Sustainable Development. AI2SD 2022. Lecture Notes in Networks and Systems*, vol 712, 2023. Springer, Cham, https://doi.org/10.1007/978-3-031-35251-5_29

2. Lenarduzzi V., Lunesu I., Matta M., Taibi D. Functional Size Measures and Effort Estimation in Agile Development: A Replicated Study. *Agile Processes in Software Engineering and Extreme Programming. XP 2015. Lecture Notes in Business Information Processing*, 2015. vol 212. https://doi.org/10.1007/978-3-319-18612-2_9

3. Sachdeva V. Requirements Prioritization in Agile: Use of Planning Poker for Maximizing Return on Investment. *Information Technology - New Generations. Advances in Intelligent Systems and Computing*, vol 558. Springer, Cham. https://doi.org/10.1007/978-3-319-54978-1_53

4. Kula E., Greuter E., Deursen A., Gousios G. Factors Affecting On-Time Delivery in Large-Scale Agile Software Development. *IEEE Transactions on Software Engineering*, 2022. vol. 48, no. 9, pp. 3573-3592. https://doi.org/10.1109/TSE.2021.3101192.

5. Jansen U., Schulz W. FlowChart Tool for Decision Making in Interdisciplinary Research Cooperation. Digital Human Modeling. *Applications in Health, Safety, Ergonomics, and Risk Management: Health and Safety. Lecture Notes in Computer Science*, 2017. vol 10287. Springer, Cham. https://doi.org/10.1007/978-3-319-58466-9_24

6. Латанська Л. О., Балашов М. О. Удосконалення регресійного рівняння для оцінювання трудомісткості розробки програмного забезпечення, створеного за гнучкою методологією. *Матеріали XI-ої Міжнародної науково-практичної конференції «Free and Open Source Software»*, Харків, 19-21 листопада 2019 р. – Харків: Харківський національний університет будівництва та архітектури, С. 46.

7. Durán M., Juárez-Ramírez R., Jiménez S. et al. User Story Estimation Based on the Complexity Decomposition Using Bayesian Networks. *Program Comput Soft,* 2020. 46, 569–583 https://doi.org/10.1134/S0361768820080095

8. Димова Г.О., Ларченко О. В. Розробка комп'ютерної програми розв'язання задач мережевої оптимізації *Науковий журнал "Комп'ютерно-інтегровані технології: освіта, наука, виробництво"*. Луцьк, 2020. Випуск № 41 С. 143 – 151

9. Сєдих О.Л., Чобану В.В. Оптимізація мережевого графіка комплексу робіт. *Modern engineering and innovative technologies.* № 03-01, 2018 С. 61-67 https://doi.org/10.30890/2567-5273.2018-03-01-009 10.

10. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Розробка архітектури програмного засобу для управління мережевим плануванням реінжинірингу програмного проекту. *Сучасний стан наукових досліджень та технологій в промисловості.* 2019. № 2 (8). С. 25–35. DOI: https://doi.org/10.30837/2522-9818.2019.8.025

11. Кордунова Ю. С., Смотр О. О., Кокотко І. Я., Малець Р. Б. Аналіз традиційного та гнучкого підходів до створення програмного забезпечення в динамічних умовах. *Управління розвитком складних систем.* Київ, 2021. № 47. С. 71 – 77, DOI: https://doi.org/10.32347/2412-9933.2021.47.71-77

12. Kordunova Y., Prydatko O., Smotr O., Golovatyi R. Expert Decision Support System Modeling in Lifecycle Management of Specialized Software. *Lecture Notes on Data Engineering and Communications Technologies, Springer, Switzerland*. Vol. 149, 2022 pp. 367-383, https://doi.org/10.1007/978-3-031-16203-9_22

13. Kordunova Yu., Prydatko O., Smotr O., Kokotko I. The network graph traversal method for solving the problem of short-term planning of safety-oriented services development. *Monografia powstała w ramach Projektu dofinansowanego przez Ministra Edukacji i Nauki ze środków budżetu państwa w*

ramach programu „Doskonała Nauka”, Warszawa 2022. p. 172 – 181.

14. Праворська Н., Мартинюк В. Конструювання програмного забезпечення за допомогою синхронного підходу: основні процеси та інструменти для ефективної реалізації Devops. *Вісник Хмельницького національного університету*, Том 1, №5, 2023 С. 182-191

15. Lamsellak H, Belkasm M. G. Global software development agile planning model:challenges and current trends. *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 32, No. 3, 2023, pp. 1774-1784

References:

1. Lamsellak, H., Khalil, A., Belkasmi, M.G., Saber, M. (2023). Agile Practices in Iteration Planning Process of Global Software Development. *International Conference on Advanced Intelligent Systems for Sustainable Development. AI2SD 2022. Lecture Notes in Networks and Systems*, vol 712. Springer, Cham. https://doi.org/10.1007/978-3-031-35251-5_29

2. Lenarduzzi, V., Lunesu, I., Matta, M., Taibi, D. (2015). Functional Size Measures and Effort Estimation in Agile Development: A Replicated Study. *Agile Processes in Software Engineering and Extreme Programming. XP 2015. Lecture Notes in Business Information Processing*, vol 212. Springer, Cham. https://doi.org/10.1007/978-3-319-18612-2_9

3. Sachdeva, V. (2018). Requirements Prioritization in Agile: Use of Planning Poker for Maximizing Return on Investment. *Information Technology - New Generations. Advances in Intelligent Systems and Computing*, vol 558. Springer, Cham. https://doi.org/10.1007/978-3-319-54978-1_53

4. Kula, E., Greuter, E., Deursen, A. and Gousios, G. (2022) Factors Affecting On-Time Delivery in Large-Scale Agile Software Development. *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3573-3592. https://doi.org/10.1109/TSE.2021.3101192.

5. Jansen, U., Schulz, W. (2017). FlowChart Tool for Decision Making in Interdisciplinary Research Cooperation. *Digital Human Modeling. Applications in Health, Safety, Ergonomics, and Risk Management: Health and Safety. DHM 2017. Lecture Notes in Computer Science*, vol 10287. Springer, Cham. https://doi.org/10.1007/978-3-319-58466-9_24

6. Latanska, L. O., & Balashov, M. O. (2019). Udoskonalennia rehresiinoho rivniannia dlia otsiniuvannia trudomistkosti rozrobky prohramnoho zabezpechennia, stvorenoho za hnuchkoiu metodolohiieiu [Improving the regression equation for estimating the complexity of software development created by an agile methodology]. *Materialy XI Mizhnarodnoi naukovo-praktychnoi konferentsii «Free and Open Source Software» - Materials of the 11th International Scientific and Practical Conference "Free and Open Source Software"*, Kharkiv, 19-21 novenber 2019, p. 46 [in Ukrainian].

7. Durán, M., Juárez-Ramírez, R., Jiménez, S. et al. (2020) User Story Estimation Based on the Complexity Decomposition Using Bayesian Networks. *Program Comput Soft* 46, 2020. 569–583 https://doi.org/10.1134/S0361768820080095

8. Dymova H., & Larchenko, O. (2020). Rozrobka komp’iuternoi prohramy rozv’iazannia zadach merezhevoi optymizatsii [Development of a computer program for solving network optimization problems]. *Naukovyi zhurnal "Komp’iuterno-intehrovani tekhnolohii: osvita, nauka, vyrobnytstvo" – Computer-integrated technologies: education, science, production*, (41), 143-151. https://doi.org/10.36910/6775-2524-0560-2020-41-23 9 [in Ukrainian].

9. Seidykh, O., Chobanu, V. (2017). Optimization of the network graphics of the complex of works. *Modern Engineering and Innovative Technologies*, 1(03-01), 61–67. DOI: doi.org/10.30890/2567-5273.2018-03-01-009 10.

10. Velykodniy, S., Burlachenko, Zh., Zaitseva-Velykodna, S. (2019). Rozrobka arkhitektury prohramnoho zasobu dlia upravlinnia merezhevym planuvanniam reinzhynirynhu prohramnoho proektu [Architecture development of software for managing network planning of software project reengineering]. *Suchasnyi stan naukovykh doslidzhen ta tekhnolohii v promyslovosti – Innovative Technologies and Scientific Solutions for Industries*, No. 2 (8), P. 25–35 DOI: https://doi.org/10.30837/2522-9818.2019.8.025. 11 [in Ukrainian].

11. Kordunova, Yu., Smotr, O., Kokotko, I. & Malets, R. (2021). Analiz tradytsiinoho ta hnuchkoho pidkhodiv do stvorennia prohramnoho zabezpechennia v dynamichnykh umovakh [Analysis of the traditional and flexible approaches to creating software in dynamic conditions]. *Upravlinnia rozvytkom skladnykh system – Management of Development of Complex Systems*,47, 71–77, dx.doi.org\10.32347/2412-9933.2021.47.71-77 [in Ukrainian].

12. Kordunova, Y., Prydatko, O., Smotr, O., Golovatyi, R. (2023). Expert Decision Support System Modeling in Lifecycle Management of Specialized Software. *In: Babichev, S., Lytvynenko, V. (eds) Lecture Notes in Data Engineering, Computational Intelligence, and Decision Making. ISDMCI 2022. Lecture Notes on Data Engineering and Communications Technologies*, vol 149. Springer, Cham. https://doi.org/10.1007/978-3-031-16203-9_22

13. Kordunova, Yu., Prydatko, O., Smotr, O., Kokotko, I. (2022) The network graph traversal method for solving the problem of short-term planning of safety-oriented services development. *Monografia powstała w ramach Projektu dofinansowanego przez Ministra Edukacji i Nauki ze środków budżetu państwa w ramach programu „Doskonała Nauka”*, Warszawa 2022. p. 172 – 181.

14. Pravorska, N., Martynyuk, V. (2023). Konstruiuvannia prohramnoho zabezpechennia za dopomohoiu synkhronnoho pidkhodu: osnovni protsesy ta instrumenty dlia efektyvnoi realizatsii Devops. [Developing software using a synchronous approach: key processes and tools for effective devops]. *Visnyk Khmelnytskoho natsionalnoho universytetu – Herald of Khmelnytskyi national university*, Part 1, Issue 5, (325) [in Ukrainian].

15. Lamsellak, H, Belkasm, M. G. (2023). Global software development agile planning model:challenges and current trends. *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 32, No. 3, pp. 1774-1784

Науково-методична стаття.
Надійшла до редакції 10.06.2024.
Прийнято до публікації 12.06.2024.