Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Автоматизована система планування і управління ресурсами при виконанні IT-проєктів

Виконав: студент 2 курсу, групи
3АКІТ-22м спеціальності 151 –
Автоматизація та комп'ютерно-інтегровані
технології

Євгеній СИРЦОВ
Ім'я ПРІЗВИЩЕ

Керівник:
доцент, доцент кафедри АІІТ
степінь, звання, посада

Роман МАСЛІЙ
Ім'я ПРІЗВИЩЕ

«15» грудня 2023 р.

Опонент:
к.т.н., професор, професор кафедри КН
степінь, звання, посада

Олег КОЛЕСНИЦЬКИЙ
Ім'я ПРІЗВИЩЕ

«15» грудня 2023 р.

Допущено до захисту
Зав. кафедри АІІТ
Олег БІСІКАЛО
«15» грудня 2023

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітня програма – Інформаційні системи і Інтернет речей

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
Олег БІСІКАЛО
"20" вересня 2023 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Сирцову Євгенію Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Автоматизована система планування і управління ресурсами при виконанні ІТ-проєктів
керівник роботи Маслій Роман Васильович
затверджені наказом ВНТУ від "18" вересня 2023 року №247
2. Термін подання студентом роботи "12" грудня 2023 року
3. Вихідні дані до роботи: програмне забезпеченя з допомогою якого можна планувати і управляти ресурсами при виконанні ІТ-проєктів, можливість побудови розкладу з 100 завданнями.
4. Зміст текстової частини: вступ, огляд існуючих рішень по управлінню ІТ-проєктами, розв`язання задачі планування ресурсів ІТ-проєктами, опис програмного та технічного забезпечення, висновки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Схематичне зображення процесу Scrum, Порівняльний аналіз управління ресурсами та фінансами, Схематичне зображення виділення з беклогу продукту беклогу спринта, Виділена підмножина завдань (беклог спринта), Приклад розв'язання задачі: Вхідні дані, Функціонально-логічна структура програмного забезпечення, Фрагмент діаграми класів програмного забезпечення, Декомпозиція цілей системи, Схема обробки інформації у розробленій системі, Структурна схема програмного забезпечення, Екранна форма для формування розкладу задачі (жадібний алгоритм), Екранна форма для формування розкладу задачі (алгоритм локального пошуку), Екранна форма проведення експериментів.

Консультанти розділів роботи

Розділ	Ім`я, Прізвище та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Роман МАСЛІЙ доцент, доцент кафедри АІІТ	01.10.2023	29.11.2023

Дата видачі завдання "20" вересня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Прим
		початок	закінчення	
1	Огляд існуючих рішень управління IT-проєктами	20.09.2023	30.09.2023	
2	Огляд та розв`язання задачі планування ресурсів	01.10.2023	10.10.2023	
3	Написання коду програмного забезпечення	11.10.2023	09.11.2023	
4	Здійснення тестування системи	10.11.2023	19.11.2023	
5	Складання тексту магістерської роботи. Редагування та форматування	20.11.2023	12.12.2023	

Студент (підпис) Євгеній СИРЦО
(Ім'я ПРІЗВИЩЕ

Керівник роботи (підпис) Роман МАСЛІ
(Ім'я ПРІЗВИЩЕ)

АНОТАЦІЯ

УДК 004.7

Автоматизована система планування і управління ресурсами при виконанні ІТ-проєктів зі спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології, освітня програма – Інформаційні системи і Інтернет речей. Вінниця: ВНТУ, 2023 118с.

На укр. мові. Бібліогр.: 81 назв; рис.:28; табл.:25.

Ця магістерська робота присвячена розробленню системи планування і управління ресурсами ІТ-проєктів

Комплексний підхід до управління ІТ-проєктами вимагає ефективної організації, планування та управління ресурсами. Робота висвітлює методи управління проектами та пропонує вдосконалені алгоритми для планування ресурсів ІТ-проєктів, зокрема управління людськими ресурсами та часовими параметрами. Основною метою є розробка інформаційної системи планування ресурсів ІТ-проєкту для мінімізації часу реалізації. Робота включає аналіз існуючих підходів, розробку алгоритмів та програмної реалізації системи. Експериментальні дослідження підтверджують ефективність запропонованих методів та інформаційної системи.

Ключові слова: управління проєктами, планування ресурсів, ІТ-проєкти, управління людськими ресурсами, ефективність планування.

ABSTRACT

Automated Planning and Resource Management System for IT Projects in the field of 151 - Automation and Computer-Integrated Technologies, educational program - Information Systems and Internet of Things. Vinnytsia: VNTU, 2023. 118 p.

In Ukrainian. Bibliography: 81 titles; figures: 28; tables: 25.

This master's thesis is dedicated to the development of a planning and resource management system for IT projects. A comprehensive approach to managing IT projects requires effective organization, planning, and resource management. The work highlights project management methods and proposes improved algorithms for planning resources in IT projects, particularly managing human resources and time parameters. The main goal is to develop an information system for planning IT project resources to minimize implementation time. The work includes an analysis of existing approaches, the development of algorithms, and the software implementation of the system. Experimental studies confirm the effectiveness of the proposed methods and information system.

Keywords: project management, resource planning, IT projects, human resource management, planning efficiency.

ЗМІСТ

ВСТУП

Актуальність. Кожен проект має свою унікальність та вимагає індивідуального підходу до управління. Проєкт можна розглядати як об'єкт управління, для якого необхідна чітка координація протягом всього його існування. В процесі виконання можуть виникнути непередбачені обставини, які можуть завдати ушкодження успішному завершенню проєкту. Для швидкого знаходження оригінальних рішень часто використовується проєктне управління.

Проєктне управління є популярною системою, яка дозволяє контролювати вирішення задач та досягнення цілей проєкту. Вона визначає цілі, обмеження ресурсів, параметри, забезпечує вирішення непередбачених ситуацій та встановлює точки початку та завершення проєкту. Сучасне управління проєктами неможливо уявити без проєктного управління, яке раніше використовувалося винятково, але зараз стало стандартним методом ведення проєктів. Цей метод постійно розвивається, розширюючи свої можливості, та зараз широко використовується для скорочення життєвого циклу проєкту.

Для кожного проекту визначені цілі, результати, компоненти та обсяг робіт. Усі компоненти можуть бути змінені протягом життя проекту, і вони можуть бути змінені або вдосконалені під час розробки проекту та під час досягнення проміжних результатів.

Незважаючи на різноманітність існуючих автоматизованих систем управління проектами (Microsoft Project, Concerto, JIRA, Serena Team Track, Primavera Project Planner, Spider Project), багато важливих процесів і завдань управління проектами (включаючи планування) не мають необхідної комп'ютерної підтримки.

Метою даного дослідження є підвищення ефективності реалізації IT-проектів в компанії, оскільки створена інформаційна система планування

ресурсів IT-проекту, заснована на методі, що дозволяє мінімізувати час реалізації проекту.

Для досягнення цієї мети необхідно виконати наступні завдання:

- Аналітичний огляд і детальний аналіз існуючих підходів до систем планування ресурсів, управління проектами, обмежень і управління ризиками в рамках проектів, графіків планування завдань в рамках обмежень ресурсів;

- Розробити алгоритми вирішення завдань підтримки функціонування систем планування ресурсів IT-проектів;

- Ефективність розроблених інститутом алгоритмів;

- Розробка програмної реалізації інформаційних систем планування ресурсів IT-проектів;

- Проводити експериментальні дослідження ефективності розроблених інформаційних систем, методів вирішення завдань і використовуваних алгоритмів.

Об’єктом дослідження є процес управління IT-проектами.

Предметом дослідження є методи планування ресурсів IT-проектів.

Методи дослідження. Для вирішення поставленої задачі в роботі використовуються такі методи: системного аналізу (при проектуванні інформаційних систем), теорії планування, дослідження операцій, теорії складності (при розробці методів вирішення задач розподілу ресурсів), комп’ютерного моделювання (при розв’язуванні задач експериментальним шляхом). вивчає ефективність підходу до проблем розподілу ресурсів).

Новизна отриманих результатів полягає в побудові методів планування ресурсів з урахуванням їх обмежень, ризиків та аналізу, у розробці та вдосконаленні алгоритмів вирішення задач шляхом експериментальних досліджень.

Практична цінність. Отримані результати можуть бути застосовані як

на практиці, для розвитку існуючих рішень, так і у майбутніх дослідженнях у сфері менеджменту.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ПО УПРАВЛІННЮ ІТ-ПРОЄКТАМИ

1.1 Підходи до управління ІТ-проєктами

За час діяльності людини накопичилося багато успішно реалізованих проектів, у тому числі ІТ-проектів. Від будівництва пірамід Гізи до першої пілотованої посадки на Місяць такі сміливі ідеї вимагають спільних зусиль великої кількості людей.

Усі проекти різні, і кожен приносить у світ щось своє. Не існує ідеальної системи управління проектами, яка однаково підходила б і використовувалася для кожного типу проекту. Необхідно розуміти, що жодна система не підходить кожному керівнику на даному проекті. Проте за час існування поняття «управління проектами» було відкрито багато ефективних способів, методів і моделей, які можна використовувати для управління проектами [2-5].

В даний час в основному існують наступні методи управління проектами:

- класичний менеджмент проектів (водоспад);
- scrum;
- нахил;
- канбан;
- шість сигм;
- принц 2.

Методи управління проектами використовуються вже давно. Єгипетські піраміди та Велика китайська стіна є реалізованими проектами, і автори використовували різні методи управління проектом. На жаль, документація про те, як реалізовувалися та керувалися такими проектами,

втрачена, а сучасне управління проектами значно відрізняється від знань минулих століть.

Найефективніший спосіб реалізації проекту – це розділити його на фази або окремі завдання різного розміру. Одним із найпростіших інструментів є список дій, які ви повинні виконати для досягнення своїх цілей.

Далі ми більш детально розглянемо деякі методи управління проектами.

Водоспад. Класичний підхід до управління проектами базується на так званому каскадному циклі або «водоспаді», в якому завдання рухаються через ітераційну послідовність, подібну до процесу [6-7].

Цей підхід рекомендується для проектів, які мають жорсткі обмеження на порядок виконання певних завдань і обмеження ресурсів. Наприклад, будівництво нових житлово-громадських комплексів – якщо не закладено фундаменти, перекриття будівлі не можна зводити.

Відповідно до поетапності методу існує 5 основних фаз управління проектом за допомогою водоспадного методу (хоча можуть бути додані й інші).

5 етапів водоспаду:

Етап 1: Ініціалізація. На цьому етапі визначаються вимоги до проекту. Це робить керівник проекту та команда. Під час процесу ініціалізації часто проводиться так званий «мозковий штурм», щоб швидше та легше зрозуміти, яким має бути кінцевий продукт.

Етап 2: планування. На етапі планування команда визначає, які завдання необхідно виконати для досягнення поставлених цілей. Проектні завдання та результати проекту чіткі та детальні. На основі зібраної інформації команда складає календарний план, формує бюджет, оцінює ризики та можливі зміни в робочому процесі.

Етап 2: розвиток. Основними особливостями технологічних проектів на етапі розробки є конфігурація та технічні вимоги майбутніх проектів. спосіб її досягнення.

Етап 4: Впровадження та тестування. За цей період була завершена основна робота - написання коду. Контроль за дотриманням заздалегідь встановленого календарного графіка формує наповнення проектів, контрольованих за стандартами. У другій половині цього етапу проводиться тестування продукту, щоб порівняти, чи відповідає продукт вимогам замовника. Після перевірки дефекти товару виявляються та усуваються.

Етап 5: Моніторинг і завершення проекту. На цьому етапі все залежить від типу проекту, оскільки ця фаза може включати передачу клієнту результатів, отриманих від реалізації проекту, або досить тривалий процес взаємодії з клієнтом для покращення результатів проекту та підвищення рівня відповідності. запит клієнта.

Очевидно, що класичний метод управління проектами суворо пов'язаний з тривалістю завдання, а тривалість завдання визначається на етапі планування, то в рамках цього методу методи календарного та мережевого планування дуже підходять для реалізації проекту. [8-11] .

Переваги водоспадів:

- визначити потреби та вимоги клієнта на першому етапі проекту;

- фаза ініціалізації покращує стабільність роботи проекту;

- чітке планування дає можливість спростити реалізацію проектних завдань;

- ці етапи включають важливий етап моніторингу та тестування отриманого продукту.

- керівники проекту мають повну інформацію про ресурси проекту і, навіть якщо їхні оцінки неточні, завжди враховують ризик порушення цих оцінок.

Недоліки водоспадів:

- Будь-які зміни серйозно порушуватимуть і відхилятимуться від календарного плану, тобто відсутність гнучкості для змін під час подальшої реалізації проекту.

Agile. Гнучкий, ітеративний та поступовий підхід до управління проектом, основний фокус якого зосереджений на динамічному формуванні вимог, можливості змінювати вимоги на льоту та забезпеченні виконання цих вимог через постійну взаємодію між самоорганізованими членами команди [12-17].] .

Не всі проекти можуть бути розроблені для реалізації за допомогою класичних методів управління проектами. Відповідно до цього підходу проект ділиться не на послідовні фази, а на невеликі підпроекти, які потім призводять до готового продукту.

Переваги Agile:

- гнучкість і здатність до змін;
- адаптуватися до будь-яких умов і процесів;
- можливість реалізації проекту «пакетами»;
- відстежуйте виконання завдань за допомогою основ методу.

Недоліки Agile:

– Необхідно адаптувати цей підхід до кожної команди виконавців окремо, що є складним і досить тривалим процесом, який потребує окремого часу на переналаштування.

1.2 Використання Scrum методології у рамках управління проєктом

Scrum розбиває проект на підпроекти (частини основного проекту), які клієнт може фактично використати негайно. Потім ці частини розділяються за пріоритетом завдання (зазвичай виконується власником продукту).

Основні «шматки» реалізуються спочатку в спринтах (ітераціях у Scrum), які тривають від 2 до 4 тижнів.

Наприкінці спринту замовнику представляють робочі частини продукту — ті «шматки» проекту, які готові до використання в роботі. Після першого спринту команда впровадження починає наступний спринт. Згідно з роботами [19-22], тривалість спринту завжди фіксована, але команда визначає її самостійно, виходячи зі складності завдання та наявних можливостей команди.

Scrum складається з команд Scrum і пов’язаних з ними ролей, подій, артефактів і правил. Кожен компонент у структурі служить певній меті та є важливим для неї. Правила Scrum пов’язують ролі, події та артефакти, керуючи зв’язками та взаємодією між ними.

Сьогодні Scrum є однією з найпопулярніших методологій розробки програмного забезпечення та методів управління проектами. За визначенням, Scrum – це макет розробки, який дозволяє людям успішно та продуктивно вирішувати виявлені проблеми, а також виробляти якісні та значущі продукти (з точки зору замовника – прим. автора) [19].

Це пояснюється тим, що в Scrum практично неможливо знайти відповіді на всі запитання та інструкції щодо дій у деяких випадках (наприклад, в офіційному описі Scrum вказується лише оцінка часу, необхідного для виконання завдання). завдання, але не зазначено, як це має виглядати.

Коли говорять про методології та методології Scrum, зазвичай мають на увазі гнучкий підхід до розробки програмного забезпечення, який побудований на правилах і практиках Scrum.

Автори Scrum виділяють такі характеристики:

- легкий (англ. Lightweight);

- чіткий і зрозумілий;

- важкі для освоєння (фактично взаємовиключні уривки). У класичному Scrum є 3 основні ролі:

- власник продукту (Product Owner);

- scrum master;

- команда розробників.

Власник продукту (PO) є ключовим «перемикачем» між командою розробників і стороною клієнта. Мета посади віце-президента полягає в тому, щоб додати якомога більше цінності продукту, що розробляється, і команді розробників за допомогою будь-якого доступного методу в рамках роботи над проектом.

Те, що допомагає віце-президенту працювати над проектами, щоб задовольнити потреби клієнтів, — це відставання продуктів. Беклог продукту містить усі робочі завдання, необхідні для примусового виконання (наприклад, історії користувачів, помилки, завдання тощо), і кожне таке завдання містить пріоритет (пріоритет виконання).

Scrum Master є лідером групи розробників і відповідає за використання методології. Мета роботи Scrum Master — допомогти команді розробників підвищити ефективність шляхом правильного налаштування робочих процесів, взаємодії між членами команди, навчання та мотивації команди. Зміцнювати командний дух.

Команда розробників (development team) складається з різних експертів, які працюють над виробленим продуктом.

Загалом, усі виконавці, які працюють із Scrum (власник продукту, Scrum-майстер і команда розробки), називаються командами Scrum.

Якщо ви посилаєтеся на документ «The Scrum Guide» (який є офіційним поясненням автора щодо всіх функцій Scrum), команда розробників повинна мати наступні якості та характеристики:

- самоорганізація. Ніхто з команди Scrum не може сказати команді розробників, як втілити беклог продукту в робочий продукт, який очікує клієнт;

- Багатофункціональний, багатозадачний. Це навички, необхідні для опанування командної роботи, продуктивності та реалізації робочого продукту;

- відповідальність. За виконану роботу (навіть якщо були допущені помилки) відповідає вся команда Scrum, а не окремі члени команди. Рекомендований розмір команди Scrum – 7-8 осіб (плюс-мінус 2 людини). Відповідно до класичних характеристик Scrum, великі команди вимагають занадто багато ресурсів для спілкування, але в той же час малі команди збільшують ризик провалу проекту (через достатню кількість необхідних навичок і кваліфікації) і зменшують обсяг роботи над проектом. невдача. В цілому команда може виконувати роботу за одиничний термін реалізації проекту [5-6].

Процес Scrum

Основою Scrum є його поділ на спринти, під час яких виконується робота над обраними завданнями для часткової реалізації продукту. Після завершення спринту необхідно представити роботу, виконану під час спринту (нову робочу версію продукту або часткову функцію). Спринти завжди обмежені часом: зазвичай спринти тривають 1-4 тижні і стільки ж тривають до кінця реалізації проекту.

На початку кожного спринту здійснюється планування спринту, оцінюється вміст беклогу продукту та визначається (призначається) спринтський беклог, який містить різні типи завдань, які необхідно реалізувати в поточному спринті, а також їх результати та запропоновані плани реалізації. У кожного спринту є своя конкретна ціль (цель), яка повинна служити мотивуючим фактором і може бути досягнута шляхом виконання завдань у резерві спринту.

У класичному Скрамі щодня проводиться щоденна зустріч команди (Daily Scrum), на якій всі учасники команди відповідають на три основні запитання: «Що було зроблено вчора?», «Що я буду робити сьогодні?», «Що в мене є». робити із завданням?" Питання, може мене щось стримує?".

Метою таких зустрічей є визначення поточного стану команди, розуміння ходу роботи над впровадженням спринту, проблеми, виявлені на ранній стадії або проблеми, що виникли під час роботи, а також підготовка рішень щодо зміни окремих робочих моментів, необхідних для досягнення поточна мета спринту. Після закінчення спринту відбувається підготовка до спринту (Sprint Review) і ретроспектива спринту (Sprint Retrospective), основна мета яких – оцінити ефективність (або продуктивність) Робота Scrum-команди в попередньому спринті, прогнозування очікуваної ефективності в наступному спринті на основі отриманих результатів, виявлення незрозумілих завдань або проблем, оцінка ймовірності виконання дорученої роботи над проектом тощо.

Як працює Scrum, показано на малюнку 2.2:

Рисунок 1.1 – Схематичне зображення процесу Scrum

Scrum часто кажуть, що він не працює або що він працює набагато гірше, ніж спочатку очікувалося. Слід зазначити, що найчастіше ситуація виникає через одну з наступних причин:

- неправильне або неповне застосування Scrum.

На думку кількох авторів Scrum, досвід є основним джерелом надійної інформації щодо його роботи та правильного використання. Посібник Scrum пише про необхідність повноти та точності при використанні Scrum через організацію процесу, яка не є типовою для класичної структури, оскільки немає формальних експертів, таких як менеджери чи лідери;

Один із головних принципів Scrum — це самоорганізація міжфункціональних команд. Згідно з дослідженнями соціологів, кількість ініціатив співробітників зі здібностями до самоорганізації та багатозадачності. Цей показник не перевищує 15% працездатного населення [23-25].

Звідки це береться, без суттєвих змін у ролях Scrum Master і Product Owner лише невелика група експертів може ефективно працювати в Scrum-команді, що фактично суперечить основній ідеї Scrum і має Зміни у використанні Результатом може стати Scrum, а саме:

- Scrum використовується для продуктів, вимоги яких суперечать основній ідеї scrum.

Scrum належить до гнучкого сімейства, тому Scrum вітає зміни у вимогах у будь-який час (залишок продукту може змінитися в будь-який час). Це створює певні незручності для використання Scrum в проектах з фіксованим бюджетом (або часом). Згідно з основною ідеєю Scrum неможливо заздалегідь передбачити і прорахувати всі зміни, тому немає сенсу заздалегідь планувати весь проект, обмежтеся плануванням тільки в даний момент, тобто враховуйте тільки ті завдання, які необхідно реалізувати в поточному спринті [26]. Звичайно, є й інші обмеження.

Переваги та недоліки

Scrum має багато переваг. Він є клієнтоорієнтованим, адаптивним і гнучким, що дає клієнтам можливість вносити будь-які зміни до вимог проекту в будь-який час (але немає гарантії, що ці нові зміни будуть реалізовані в проекті). Можливість змінювати вимоги та умови поставлених завдань приваблює багатьох клієнтів.

Scrum легко «вивчити», і він економить час, віддаляючись від некритичних видів діяльності. Scrum забезпечує гнучкість, щоб мати потенційно робочий продукт або систему в кінці кожного спринту.

Філософія Scrum підкреслює, що самоорганізована, багатофункціональна та відповідальна команда може вирішувати необхідні завдання з мінімумом активності. Рекомендується невеликим компаніям, а також новачкам, оскільки дає можливість повністю відмовитися від необхідності спеціалізованого навчання співробітників і менеджерів.

Звичайно, Scrum також має певні недоліки. Завдяки простоті використання та мінімалізму Scrum накладає багато суворих і незручних правил. Але якщо всі в команді розуміють і приймають концепцію «клієнтоорієнтованості», тоді це стає менш помітним, оскільки клієнта не хвилюють внутрішні правила та процеси команди Scrum, особливо якщо вони певним чином обмежують слова клієнта. . Наприклад, при необхідності, виходячи з вимог замовника, спринтський беклог може бути практично повністю змінений, незважаючи на те, що він повністю суперечить правилам scrum.

Проблема набагато більша, ніж є насправді, тому що scrum належить до гнучкого сімейства, тому в scrum такі речі, як планування внутрішніх комунікацій або використання певних сценаріїв, коли є певні ризики, неприйнятні [25]. Таким чином, виникло офіційне заперечення, тобто порушення всіх законів Scrum.

Іншим недоліком Scrum є його акцент на самоорганізації, багатофункціональних командах. Хоча це дає можливість істотно знизити витрати на командну роботу, це також призводить до підвищення вимог до відбору та мотивації працівників. За певних умов на ринку праці створити зрілу ефективну команду Scrum практично неможливо.

Щоб краще зрозуміти природу управління ресурсами, ми проведемо аналогію з фінансовим менеджментом, і результати наведені в таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз управління ресурсами та фінансами

Фінанси	**Ресурси**
В певний момент виникає нестача коштів проєкту(наприклад, щоб виплатити зарплату – касовий розрив) – беремо кредит / займ	Потрібен конкретний ресурс на невеликий період часу (наприклад, щоб зробити термінову роботу без шкоди для основного бізнесу) - домовляємося «поширити» ресурс у сусідньому проєкті, де ресурс недозавантажений
Є надлишок грошей - кладемо їх в банк або інвестуємо в прибутковий проєкт.	Розробка "заблокована" тривалим узгодженням вимог замовником – віддаємо людей у тимчасове користування проєкту, де вони найпотрібніші. При цьому, витрати на цей період переносяться на сусідній проєкт, економимо бюджет свого проєкту
На ринку з'явилися дешеві гроші – перекредитуємо	На сусідньому проєкті звільнилися свої розробники, вартість яких нижче, ніж використовуваний на власному проєкті аутсорсний ресурс - робимо заміну

Загальноприйнятою закономірністю є те, що майже в усіх випадках управління ресурсами можна провести аналогії з фінансовим менеджментом і навпаки. Проте між грошима та ресурсами є дуже суттєва різниця, яка й визначає специфіку даної предметної сфери.

1.3 Задача планування ресурсів із застосуванням методології Scrum

Отримано замовлення на реалізацію IT-проекту. Проект вивчається (аналізується) і визначаються ресурси та вимоги (умови), необхідні для його реалізації. Весь процес управління проектом здійснюватиметься за допомогою методології Scrum, гнучкої методології, яка є частиною сімейства Agile.

За результатами дослідження умов реалізації проекту визначається набір завдань (задач), які необхідно виконати до реалізації всього проекту, що формує продуктовий беклог. Кожному завданню присвоюється пріоритет щодо його виконання (під пріоритетом мається на увазі пріоритет виконання завдання) та виконання еталонного виконавця (еталонний виконавець відноситься до робочого часу виконавця з коефіцієнтом продуктивності 1).

Ми встановлюємо такі класифікації пріоритетів: високий, середній і низький (відповідно, завдання з високим пріоритетом виконуються першими, завдання з низьким пріоритетом виконуються в останню чергу і можуть не виконуватися взагалі або можуть бути відкладені до наступної версії продукту)).

У Scrum ітерації фіксованої довжини (так звані спринти), як правило, 2-4 тижні, використовуються для реалізації проекту крок за кроком. Щоб розпочати впровадження в спринті, призначте резерв спринту, який містить певну кількість передбачуваних завдань із резерву продукту. Беклог спринту формується на основі пріоритету завдань: спочатку всі завдання з високим пріоритетом, потім завдання із середнім пріоритетом, потім завдання з низьким пріоритетом, як показано на малюнку 1.2.

БЕКЛОГ ПРОДУКТУ		
Номер завдання	Пріоритет	Тривалість (години)
1	Високий	4
2	Високий	7
3	Низький	3
4	Високий	2
5	Середній	5
6	Середній	7
7	Високий	9
8	Низький	2
9	Середній	1
10	Низький	4
…		
80	Середній	8
Всього: 80 завдань		200 годин

БЕКЛОГ СПРИНТА		
Номер завдання	Пріоритет	Тривалість (години)
1	Високий	4
2	Високий	7
4	Високий	2
7	Високий	9
…		
Всього: 30 завдань		90 годин

Рисунок 1.2 – Схематичне зображення виділення з беклогу продукту беклогу спринта

Кількість завдань не повинна перевищувати значення продуктивності команди. Цей піднабір завдань стає резервом для першого спринту.

Над реалізацією цього IT-проекту працює команда виконавців, кожен з яких має свій коефіцієнт продуктивності за посадою. Склад, посади та чисельність бригади вказуються в наказі. Ми визначаємо посади виконавців як: молодший, середній, старший. Коефіцієнт продуктивності виконавця середньої посади дорівнює 1, тобто виконавець середньої посади є еталонним виконавцем. Існують відповідні показники продуктивності для молодших і старших виконавців.

У цій постановці задачі ми розглядаємо випадок, коли експерти взаємозамінні і можуть працювати паралельно. Виконавці мають різні коефіцієнти продуктивності праці, які визначаються займаною посадою.

У міру формування резерву для наступного спринту, резерв продукту змінюється, пріоритети завдань можуть змінюватися, а самі завдання можуть змінюватися та розділятися на менші завдання.

Таким чином, реалізація відставання спринту триватиме до кінцевого терміну директиви, тобто часу, до якого має бути завершена вся робота спринту на рисунку 1.3.

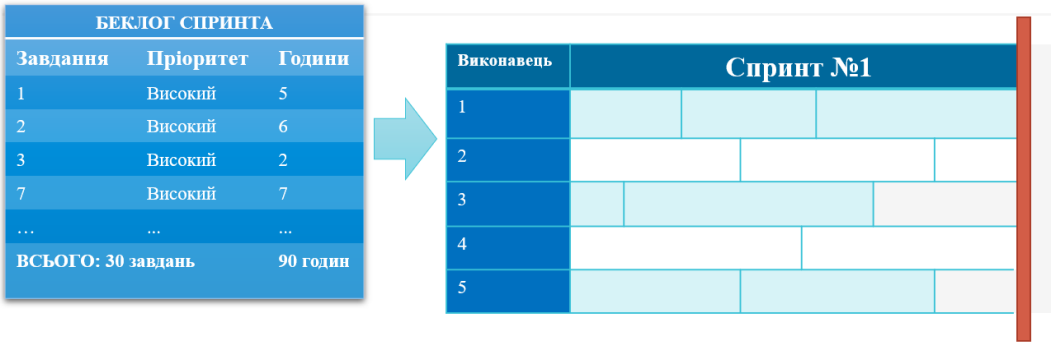

Рисунок 1.3 – Виділена підмножина завдань (беклог спринта) призначається на учасників команди

Оскільки цілями даного завдання є мінімізація часу виконання проекту та ефективне планування ресурсів (вибір часу та людських ресурсів у цьому завданні), які обмежені, теорія розкладу може допомогти визначити властивості та характеристики цього завдання. Ця задача належить до класу задач теорії розкладу і є NP-повною.

Одним із головних питань при плануванні нових напрямків виконання завдань є класифікація завдань і чітке визначення їх складності. Досить стабільну на сьогодні класифікацію задач TP запропонував Грем. Проблема TR та її складність розглядаються в роботах Гарі та Джонсона, Ленстри, Лоулера, Танаєва, Брукер та ін., повних структур та визначень [27].

Більшість проблем, які вивчаються в теорії розподілу, є NP-складними. Тим не менш, практика потребує вирішення таких питань. Є кілька способів досягти цього. Перший – реалізація поліноміального евристичного алгоритму. Для багатьох евристик і алгоритмів відома оцінка похибки отриманих результатів. Їх називають апроксимаціями. Існують також деякі алгоритми апроксимації, які гарантують як відносну похибку результатів [28], так і абсолютну похибку результатів [29]. Деякі NP-складні задачі доводять існування деякого наближеного розв’язку. Використовуючи цю схему, можна визначити наближений розв’язок задачі за час, поліноміально залежний від 1/ε та розміру вхідних даних, з відносною похибкою, яка не перевищує жодного заданого значення ε > 0. Задача - Схема повністю поліноміальної апроксимації (FPTAS).

Для завдань без апроксимаційних схем важливо визначити так звані є граничні значення, для яких можна визначити є наближений розв’язок за поліноміальний час – Polynomial Approximation Schemes (PTAS).

В даний час широко використовуються метаевристичні алгоритми, які дозволяють знайти найкраще рішення за прийнятний час і яке максимально наближене до оптимального рішення. Але недоліком цього та подібних алгоритмів є відсутність гарантії оптимальності отриманого рішення.

Немає способу визначити, наскільки найгірше рішення відрізняється від найкращого.

Також було витрачено багато досліджень і часу на роботу з TR, щоб з’ясувати точні методи та підходи до вирішення NP-складних проблем. Найбільш популярним став метод скороченого сортування або метод гілок і зв'язків [22, 24]. Щоб скоротити пошук, обчисліть нижчу оцінку цільової функції (у випадку, коли вона мінімізована) і застосуйте комбінаторні властивості задачі. Для розв’язування задач ТР широко використовуються методи динамічного програмування [1, 6, 19].

Як правило, задачі теорії планування можна розглядати як задачі цілочисельного лінійного програмування. Вирішенню таких проблем присвячено багато досліджень і робіт, наприклад, роботи [3, 4, 9, 19].

Сьогодні широко використовуються методи програмування з обмеженнями. Теорія розкладу також є однією із сфер її ефективного використання [5].

Деякі складні задачі TR можна розв’язувати за допомогою алгоритмів, які використовують елементи кількох методів одночасно. Їх називають «гібридними алгоритмами» [8]. Ми вважаємо цей напрямок одним із найперспективніших.

1.4 Огляд існуючих рішень задачі планування ресурсів

Дотепер проблема проектування інформаційних систем планування ресурсів ІТ-проектів на основі теоретичних проблем планування розглядалася багатьма вченими.

Тому в роботі [30] автори розглядають основні переваги та недоліки використання інформаційних технологій. Завдання та елементи системи управління проектами аналізуються, зокрема, з точки зору систем автоматизації. Проаналізовано реалізацію ІК, у тому числі функції календарного планування та контролю якості виконання проекту, висвітлено плюси та мінуси їх використання на практиці. Проведено детальний аналіз ефективності впровадження такої системи управління проектами, що включає планування інженерного комплексу та безпосередній контроль його реалізації. Подано детальний огляд підходу «Надлишкова загальна вартість», який дає можливість оцінити економічну ефективність проекту.

У роботі [31] визначено сутність таких понять, як «проект», «план», «програма» та пояснено відмінності між ними. Визначено заходи щодо проведення проектного менеджменту (управління проектами). Досліджено

роль і значення інформаційних технологій в управлінні проектами та визначено основні переваги використання інформаційної системи управління проектами (ІСУМ). Він присвячений відомій інформаційній системі управління проектами та розглядає ефективність використання систем управління проектами в Сполучених Штатах.

У [32] досліджується багатовимірне представлення даних управління ІТ-проектами. У роботі підтверджено теоретичні положення та подано системні та практичні рекомендації, що дозволяють підвищити ефективність функціонування інформаційної системи. Наведено результати аналізу основних принципів і методів управління проектами інформаційних процесів, продемонстровано методологію формування інформаційної системи управління проектами підприємства. Розглянуто процес створення ІТ-проектів, які можуть враховувати загальні стратегічні цілі розробки, уніфікації та формування операційної моделі системи підтримки прийняття рішень.

Робота [33] присвячена розробці моделі розподілу ресурсів для ІТ-проектів. Проаналізовано існуючі економічні та математичні моделі та можливість їх застосування для прийняття рішень у системах управління персоналом ІТ-підприємства. Створено багатокритеріальну економіко-математичну модель динамічної оптимізації для оптимального розподілу ІТ-проектів ІТ-командами. Запропонована методика пошуку оптимальних рішень з використанням розроблених економіко-математичних моделей.

У роботі [34] розкрито принципи використання інформаційних технологій в управлінні проектами підприємства. Автори розглядають особливості використання інформаційних технологій в управлінні проектами підприємства. Показано необхідність використання інформаційних технологій та досліджено механізм впровадження інформаційних технологій в управління проектами.

У роботі [35] запропоновано використання інформаційних технологій у системах управління проектами. Автори особливо підкреслюють, що цільове

використання інформаційних технологій для управління ресурсами пояснюється широким впровадженням, високими фінансовими резервами, досить вузькою спеціалізацією та закритістю відомих рішень.

Управління ресурсами таких проектів має базуватися на поєднанні процедурних і ресурсних методів, що відрізняються від існуючих методів універсальністю використання, орієнтованими на застосування складних математичних моделей і методів, спрямованих на вдосконалення моделей використання проектів інформаційних технологій, врахування основних факторів, що впливають на ефективність проектної діяльності та планування ресурсів і якість послуг.

Щодо розв’язання проблем планування, то тут є кілька робіт (статей), які висвітлюють основні проблеми цих розв’язків або неможливість розв’язання даної проблеми.

Розроблено нові фундаментальні принципи управління та контролю IT-проектів. Опублікував велику кількість науково-популярних матеріалів, що стосуються IT-індустрії, та провів різноманітні тренінги [29]. Така ситуація в IT-індустрії була створена для того, щоб істотно покращити економічний стан країни та рівень життя всієї країни.

Робота [36] описує методи планування проекту з обмеженими ресурсами для різних цілей планування. Результати показують, що вибір відповідних завдань планування є одним із головних рішень, які необхідно прийняти на етапі планування проекту, оскільки він визначає вигляд і форму плану проекту та ефективність використання ресурсів. Через невід'ємну складність планування проектів з відновлюваних ресурсів в умовах обмеженої доступності було використано багато швидких і простих евристик, але вони не дають точних результатів.

У роботі [37] оцінено можливість збалансування ресурсів трьох програмних пакетів управління проектами на двох практичних задачах. Виявляється, тривалість проекту залежить від програмного забезпечення чи методу, який використовується. Це підтверджується тим фактом, що для

задач такого розміру, складності та цільової функції метод виконує трансляційну мінімізацію. Спостереження підтверджують, що розвиток виробництва з необмеженими ресурсами планування збільшується з 41,11% до 167,79% і що Primavera P6 перевершує MS Project і Open Workbench. Через широкий діапазон результатів керівники проектів не повинні «довіряти» першому отриманому результату, а спробувати інші правила або навіть програмне забезпечення, якщо це можливо.

У [38], щоб оцінити кожне потенційне рішення, тобто пріоритет завдань, відповідно до переваг і обмежень ресурсів, використовується паралельна схема для перетворення цієї частки в можливий граф. Обчислювальний аналіз показує, що цей метод має потенціал для знаходження глобального оптимуму (тобто оптимального графік з найкоротшою тривалістю проекту) і є більш ефективним, ніж евристика, завдяки таким властивостям, як односторонній механізм для обміну досвідом під час пошуку рішення. Цей підхід забезпечує ефективну та просту в застосуванні альтернативу для аналізу та отримання дійсних результатів.

У першому розділі цієї статті розглядаються різні методи управління проектами, представлені конкретні моделі та висвітлюються їхні переваги та недоліки. Детально розглянуто метод управління IT-проектами – Scrum. Дає повний опис бізнес-процесів, які відбуваються під час управління проектом.

Розглянуто задачу планування ресурсів у структурі управління проектами з використанням методології Scrum. Визначено основні етапи під час планування ресурсів. Детально проаналізовано існуючі рішення подібних проблем планування. Для кожного існуючого рішення дається опис, його основна мета та результати, яких вдалося досягти автору при дослідженні певної проблеми.

2 РОЗВ'ЯЗАННЯ ЗАДАЧІ ПЛАНУВАННЯ РЕСУРСІВ ІТ-ПРОЄКТІВ

2.1 Математична постановка задачі

Дано набір завдань проекту J = {1,..., j,..., n} і набір виконавців (членів команди) M = {1,...,i,...,m}.

У системі всі завдання надходять одночасно в нульовий час, і всі завдання мають загальний елемент інструкції d.

Процес обслуговування кожного завдання виконується без перерви до завершення виконання.

Припустимо, що кожен член команди має свій коефіцієнт виконання завдання - ki > 0, кожне завдання має свій пріоритет - pj > 0, а початковий час виконання tj > 0, j = {1, n} (початковий - тобто , еталонна тривалість, еталонний коефіцієнт продуктивності виконавця ki = 1) може виконати завдання, і тоді тривалість виконання і-м членом бригади j-го завдання з урахуванням коефіцієнта продуктивності учасника становить:

$$t_{ij} = t_j * k_i \,. \qquad (2.1)$$

Нехай Cj – час виконання j-го завдання, Cmax = max{Cj} – максимальний час виконання завдання (час виконання останнього завдання), Ci – час виконання і-го завдання. виконується третім виконавцем, Zj = max {0,Cj - d}, − завдання запізнюється; Ej = {0, d - Cj} − хід виконання j-го завдання; E = SUN(Ej) – загальна затримка; Z = SUM(Z j) – загальний аванс.

Критерій 1. Необхідно розробити план виконання завдання, в якому максимальний час виконання завдання мінімізований:

$$C_{\max} \to \min \,. \qquad (2.2)$$

Критерій 2. Необхідно розробити графік виконання завдань, в якому завдання будуть розподілені порівну між виконавцями з урахуванням їх коефіцієнтів продуктивності:

$$\max_j \{E_j, Z_j\} \to \min . \qquad (2.3)$$

2.2 Дослідження властивостей задачі

Передбачається, що процес обслуговування кожного завдання в даному завданні триває без зупинки, доки завдання не буде завершено.

По-перше, визначити питання про можливість одночасного виконання завдань декількома виконавцями. У цьому випадку це неможливо, тоді завдання зводиться до призначення n завдань m виконавцям, причому ці виконавці не перетинаються.

Завдання може бути доручено тільки одному виконавцю в певний момент часу і буде виконуватися ним до моменту виконання завдання.

Для систем, де кожне завдання виконується одним виконавцем і тривалість завдань змінюється в довільному порядку від одного виконавця до іншого, загальна продуктивність завдань залишається невідомою. Але якщо виконавців можна впорядкувати за швидкістю виконання завдання (для даного завдання - за коефіцієнтом їх продуктивності), і цей порядок однаковий для всіх завдань, то існує алгоритм, який мінімізує середню тривалість виконання завдання.

Тоді кожен виконавець має коефіцієнт продуктивності виконання завдання – ki > 0, кожне завдання має свій пріоритет – pj > 0, а початковий час виконання tj > 0, j = {1, n}. . Початкове значення — це еталонна тривалість завдання, протягом якої завдання може бути виконане еталонним виконавцем з коефіцієнтом продуктивності, що дорівнює 1 (ki = 1).

Оскільки введені певні символи, ми розглянемо їх більш детально нижче:

- Cj - момент виконання j-го завдання;

- Cmax = max{Cj} - час виконання останнього завдання (максимальний час виконання завдання);

- Ci - момент виконання завдання і-м виконавцем.

Вважається, що кожне завдання має термін виконання (або розклад)

- d. Значення директивного терміну - це момент, коли завдання має бути виконано і не пізніше заданого терміну.

При аналізі цього завдання постає питання визначення нижньої межі загальної тривалості завдання (нижньої межі критерію оптимальності, згаданого вище):

$$C^* = \frac{T}{K}, \qquad (2.4)$$

де T = SUM(tj) – загальна кількість завдань у системі, а K = SUM(ki) – загальний коефіцієнт продуктивності.

Це дає розклад для всіх членів команди для виконання своїх завдань у час C* відповідно до вищезазначених критеріїв.

Після визначення терміну команди наведемо моменти затримки та випередження, що використовуються для формування j-го завдання критерію, а також їх сумарне значення:

- Z j = max{0,Cj - d}, − Запізнення j-го завдання;

- Ej = max{0, d - Cj } − Хід виконання j-го завдання;

- E = SUM(Ej) – загальний час затримки завдання;

- Z = SUM(Zj) – загальний прогрес виконання завдання.

2.3 Алгоритми розв’язання задачі

Для вирішення задач теорії планування, які належать до класу NP і для яких існує багато різних критеріїв оптимальності, існує багато різних алгоритмів вирішення, кожен зі своїми перевагами та недоліками. Але для задач теорії розподілу рекомендується використовувати евристичні алгоритми вирішення. Хоча немає гарантії, що буде знайдено оптимальне рішення, ці алгоритми значно скорочують час, необхідний для вирішення таких проблем, забезпечуючи швидкі результати.

Тому для вирішення вищезазначених задач у роботі було обрано алгоритм локального пошуку [40] та жадібний алгоритм [41].

жадібний алгоритм

Жадібний алгоритм простий для розуміння та реалізації, і працює дуже швидко.Є багато задач різної складності, які можна вирішити за допомогою жадібного алгоритму. Алгоритм обробляє завдання у довільному порядку. Він намагається доручити кожне завдання першому виконавцю, який може призначати завдання на основі його виконання. На малюнку нижче показано, як працює жадібний алгоритм.

Рішення жадібного алгоритму

Крок 1. Відсортувати масив завдань у порядку спадання тривалості виконання завдання t1 > t2 > ... > tj > ... > tn.

Крок 2. Визначити i-го виконавця в момент виконання завдання Ci = Cmin, де i = {1,..., m}, і призначити йому j-те завдання з найбільшою тривалістю t1.

Крок 3. Повторювати крок 2, поки n завдань із множини J = {1,..., j,..., n} не буде призначено m виконавцям.

Крок 4. Отримайте додаток G.

алгоритм локального пошуку

Навпаки, алгоритм локального пошуку довів, що є однією з найкращих евристик для розв’язання задач, точне розв’язання яких вимагає експоненціального часу.

Алгоритм локального пошуку — це метаевристичний алгоритм, заснований на виконанні локального пошуку в певній околиці, щоб запобігти досягненню тупика локального оптимуму, не допускаючи рухів, які призвели б до повернення до раніше знайденого рішення та повторення процесу. Алгоритм працює циклічно. Алгоритм локального пошуку починається з остаточного рішення. У кожній ітерації генерується рішення для певної околиці, і найкраще рішення в цій околиці стає новим рішенням. Вибір найкращого доступного рішення в сусідстві здійснюється таким чином, щоб не застосовувати жодних заборонених властивостей. Якщо нове рішення є кращим і прийнятним, оновіть найкраще прийнятне рішення. Процес не завершується, доки не буде виконано одну з двох умов зупинки алгоритму: досягнуто максимальної кількості виконаних ітерацій і максимально дозволеної кількості ітерацій без зміни поточного рішення.

Локальний пошук – це пошук за допомогою алгоритму локального пошуку, де пошук виконується тільки за рахунок поточного стану рішення, а результати попередніх виконань не враховуються і не запам’ятовуються. Основною метою пошуку є не лише знаходження найкращого шляху до цільової точки, а й оптимізація цільової функції задачі.Тому фактично задача, Завдання, які розв’язуються подібними або схожими алгоритмами, називаються задачами оптимізації.

Цей алгоритм зарекомендував себе як одна з найкращих евристик для розв’язання задач, для точного розв’язання яких потрібен експоненціальний час.

Першим кроком у побудові плану місії є побудова початкового плану за допомогою жадібного алгоритму.

Рішення алгоритму локального пошуку:

Крок 1. Відсортувати масив завдань у порядку спадання тривалості виконання завдання t1 > t2 > ... > t j > ... > tn.

Крок 2. Визначити i-го виконавця i = {1, m} у час виконання завдання Ci = Cmin і призначити йому j-е завдання з найбільшою тривалістю t.

Крок 3. Повторювати крок 2, поки n завдань із множини J = {1,..., j,..., n} не буде призначено m виконавцям.

Крок 4. Отримайте додаток G.

Крок 5. На основі графіка G визначити значення Ci серед усіх виконавців. Виберіть у ньому Cmax і Cmin.

Крок 6. Повторіть кроки 1, 2 і 3 для вибраних виконавців із часом завершення Cmax і Cmin, щоб скласти графік G'.

Крок 7. Отриманий розклад G' додати до загального початкового розкладу G замість виконавця і зі значеннями Cmax і Cmin - отримати розклад для ітерації 1 - G1.

Крок 8. Повторіть кроки 5, 6 і 7 для розкладу G1. Ми отримаємо розклад G2.

Крок 9. Повторювати крок 8 до досягнення умови припинення пошуку: коли оптимальний розклад усіх попередніх ітерацій Gq q , q = 1, 2,..., l і наступних l, l = 1,.. . ., ітерації з використанням відповідного розкладу Gl дають гірші (або такі самі) результати, ніж ітерація Gq. Значення l наведено на початку алгоритму.

Вивчаючи схеми та властивості цих алгоритмів, ми очікуємо, що алгоритми локального пошуку нададуть рішення, які є швидшими, точнішими та більш узгодженими зі встановленими критеріями, ніж жадібні алгоритми.

2.4 Приклад розв’язання задачі

Давайте розв’яжемо цю задачу (критерій 1) за допомогою жадібного алгоритму та покажемо крок за кроком, що відбувається на кожному кроці.

Вихідні дані наведемо в таблиці 2.1.

Таблиця 2.1 – Вхідні дані

Назва	**Значення**
Множина значень тривалості завдань	1, 1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 6, 6, 7, 8, 9, 10
Множина коефіцієнтів продуктивності виконавців	1, 1, 2, 3

Набір завдань повинен бути впорядкований за зменшенням тривалості завдання, а набір коефіцієнтів ефективності виконавців може залишатися незмінним. Початковий момент, коли кожен виконавець виконує завдання, визначаємо в таблиці 2.2.

Таблиці 2.2 – Момент, коли кожен виконавець виконує завдання

Виконавець (№)	**Момент завершення виконання завдань**
1	0
2	0
3	0
4	0

Етап 1. Візьмемо для прикладу завдання 1 тривалістю 10 і розрахуємо Момент, коли кожен виконавець виконує завдання, як показано в таблиці 2.3.

Таблиця 2.3 – Визначити час виконання завдання №1 для кожного виконавця

Виконавець (№)	**Момент завершення виконання завдання №1**
1	10
2	10
3	20
4	30

Далі визначаємо найменшого виконавця на момент виконання завдання №1. Тобто завдання можна передати виконавцю №1 або виконавцю №2, як показано на малюнку 2.1

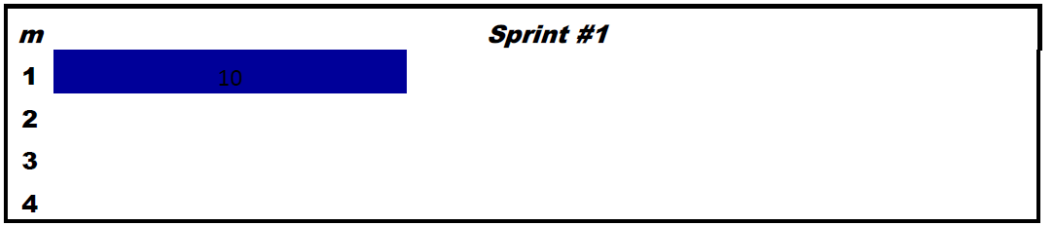

Рисунок 2.1 – Поставити завдання виконавцям №1

Ми збережемо останні зміни в таблиці та визначимо Момент, коли кожен виконавець виконує завдання, а надалі будемо записувати завдання, поставлені кожному виконавцю, Таблиця 2.4.

Таблиця 2.4 – Момент, коли кожен виконавець виконує завдання

Виконавець (*m*)	**Моменти завершення виконання завдань по кожному виконавцю (C_i)**	**Етап 1**
1	10	1(10)
2	0	-
3	0	-
4	0	-

Примітка: Надалі для зручності заповнення форми в заголовку будемо використовувати символи m, Ci та номер етапу (1, 2, 3 і т.д.).

другий етап.

Візьмемо для прикладу завдання 2 тривалості 9 і розрахуємо Момент, коли кожен виконавець виконує завдання з урахуванням результатів етапу 1. Результати наведені в таблиці 2.5.

Таблиця 2.5 – Етап 2: Визначити час виконання завдання №2 для кожного виконавця

Виконавець (№)	**Момент завершення виконання завдання №2**
1	19
2	9
3	18
4	27

Далі визначаємо найменшого виконавця за часом виконання завдання №2. На етапі 2 завдання можна призначити Виконавцю №2, як показано на малюнку 2.2.

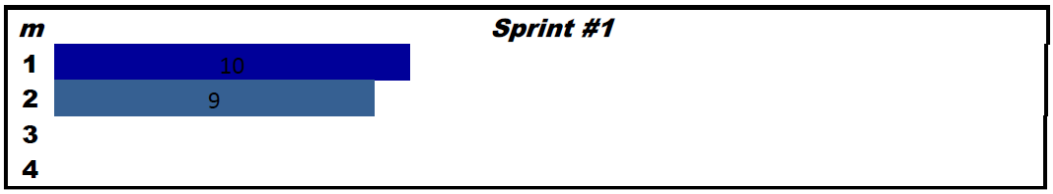

Рисунок 2.2 – Поставити завдання виконавцям №2

Занесемо отримані результати в таблицю 2.6.

Таблиця 2.6 – Момент, коли кожен виконавець виконує завдання

m	C_i	**Ет. 1**	**Ет.2**
1	10	1(10)	-
2	9	-	2(9)
3	0	-	-
4	0	-	-

Третя фаза.

Візьмемо для прикладу завдання 3 тривалості 8 і розрахуємо Момент, коли кожен виконавець виконує завдання з урахуванням результатів етапу 2. Результати наведені в таблиці 2.7.

Таблиця 2.7 – Етап 3: Визначити час виконання завдання №3 для кожного виконавця

Виконавець (№)	**Моменти завершення виконання завдання №3**
1	18
2	17
3	16
4	24

Далі визначаємо найменшого виконавця за часом виконання завдання №3. На етапі 3 завдання можна призначити виконавцю №3, як показано на малюнку 2.3.

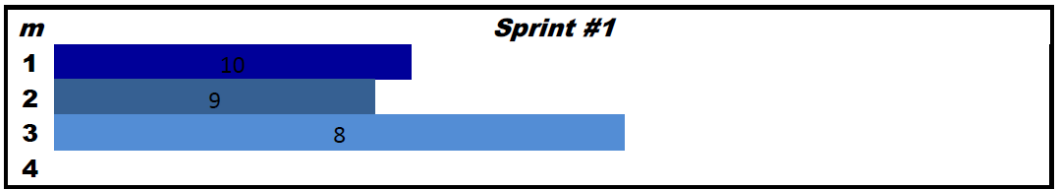

Рисунок 2.3 – Поставити завдання виконавцям №3

Отримані результати заносимо в таблицю 2.8.

Таблиця 2.8 – Момент, коли кожен виконавець виконує завдання

m	C_i	**Ет. 1**	**Ет.2**	**Ет. 3**
1	10	1(10)	-	-
2	9	-	2(9)	-
3	16	-	-	3(8)
4	0	-	-	

Четвертий етап.

Візьмемо для прикладу завдання 4 тривалістю 7 і розрахуємо Момент, коли кожен виконавець виконує завдання з урахуванням результатів етапу 3. Результати наведені в таблиці 2.9.

Таблиця 2.9 – Етап 4: Визначити час виконання завдання №4 для кожного виконавця

Виконавець (№)	**Моменти завершення виконання завдання №4**
1	17
2	16
3	30
4	21

Визначаємо найменшого виконавця на момент виконання завдання. На етапі 4 завдання можна призначити виконавцю №2, як показано на малюнку 2.4.

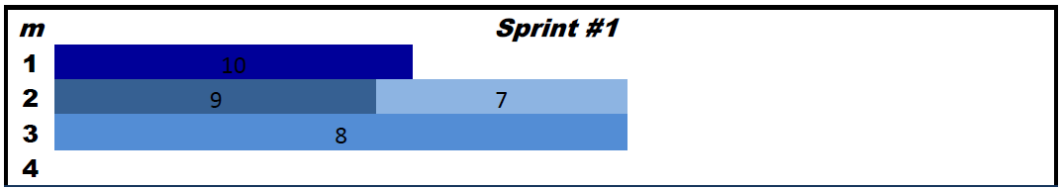

Рисунок 2.4 – Поставити завдання виконавцям №2

Отримані результати заносимо в таблицю 2.10.

Таблиця 2.10 – Момент, коли кожен виконавець виконує завдання

m	C_i	**Ет. 1**	**Ет.2**	**Ет. 3**	**Ет.4**
1	10	1(10)	-	-	-
2	16	-	2(9)	-	4(7)
3	16	-	-	3(8)	-
4	0	-	-		-

П'ятий етап.

Візьмемо для прикладу завдання 5 тривалості 6 і розрахуємо Момент, коли кожен виконавець виконує завдання з урахуванням результатів етапу 4. Результати наведені в таблиці 2.11.

Таблиця 2.11 – Етап 5: Визначити час виконання завдання №5 для кожного виконавця

Виконавець (№)	Моменти завершення виконання завдання №5
1	16
2	22
3	28
4	18

Визначаємо найменшого виконавця на момент виконання завдання. На етапі 5 завдання можна призначити виконавцю №1, як показано на малюнку 2.5.

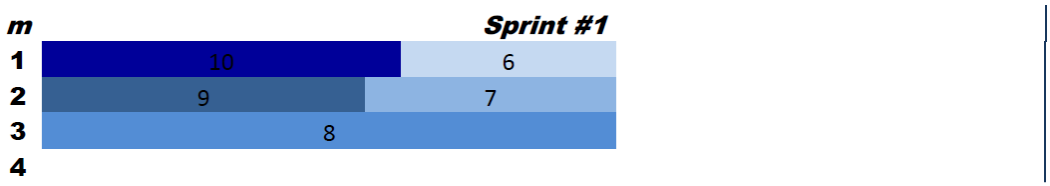

Рисунок 2.5 – Поставити завдання виконавцям №1

Отримані результати заносимо в таблицю 2.12.

Таблиця 2.12 – Момент, коли кожен виконавець виконує завдання

m	C_i	Ет. 1	Ет.2	Ет. 3	Ет.4	Ет.5
1	16	1(10)	-	-	-	5(6)
2	16	-	2(9)	-	4(7)	-
3	16	-	-	3(8)	-	-
4	0	-	-		-	-

Етап шостий.

Розглянемо виконання наступного завдання 6 тривалості 6 і, враховуючи результати етапу 5, розрахуємо Момент, коли кожен виконавець виконує завдання. Результати наведені в таблиці 2.13.

Таблиця 2.13 – Етап 6: Визначити час виконання завдання №6 для кожного виконавця

Виконавець (№)	**Моменти завершення виконання завдання №6**
1	22
2	22
3	28
4	18

Визначаємо найменшого виконавця на момент виконання завдання. На етапі 6 завдання можна призначити виконавцю №4, як показано на малюнку 2.6.

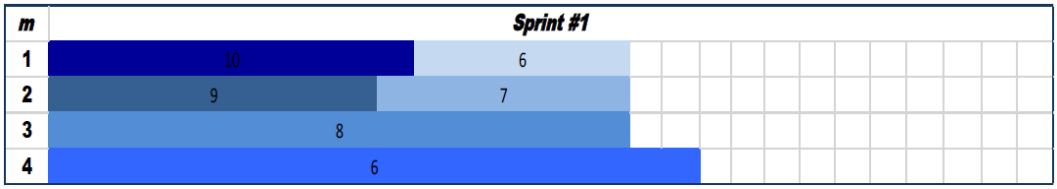

Рисунок 2.6 – Поставити завдання виконавцям №4

Отримані результати заносимо в таблицю 2.14.

Таблиця 2.14 – Момент, коли кожен виконавець виконує завдання

m	C_i	**Ет. 1**	**Ет.2**	**Ет. 3**	**Ет.4**	**Ет.5**	**Ет.6**
1	16	1(10)	-	-	-	5(6)	-
2	16	-	2(9)	-	4(7)	-	-
3	16	-	-	3(8)	-	-	-
4	18	-	-		-	-	6(6)

*Примітка: ми повторюємо аналогічні дії для кожного з наступних 10 етапів. Далі детально розглянемо останні два етапи.

Етап 16.

Візьмемо для прикладу завдання 16 тривалості 1 і розрахуємо Момент, коли кожен виконавець виконує завдання з урахуванням результатів етапу 15. Результати наведені в таблиці 2.15.

Таблиця 2.15 – Етап 16: Визначити час виконання завдання №16 для кожного виконавця

Виконавець (№)	**Моменти завершення виконання завдання №16**
1	28
2	29
3	28
4	27

Визначаємо найменшого виконавця на момент виконання завдання. На етапі 16 завдання можна призначити виконавцю №4, як показано на малюнку 2.7.

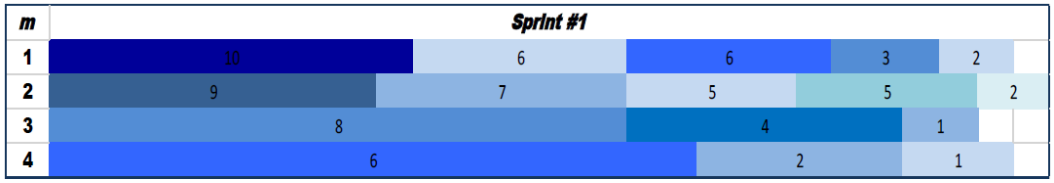

Рисунок 2.7 – Поставити завдання виконавцям №4

Отримані результати заносимо в таблицю 2.16.

Таблиця 2.16 – Момент, коли кожен виконавець виконує завдання

m	C_i	**Ет. 1**	**Ет.2**	**Ет. 3**	**Ет. 4**	**Ет. 5**	…	**Ет. 12**	**Ет. 13**	**Ет. 14**	**Ет. 15**	**Ет. 16**
1	27	1(10)	-	-	-	5(6)	…	-	13(2)	-	-	-
2	28	-	2(9)	-	4(7)	-	…	-	-	14(2)	-	-
3	26	-	-	3(8)	-	-	…	-	-	-	15(1)	-
4	27	-	-		-	-	…	12(2)	-	-	-	16(1)

Етап 17.

Візьмемо для прикладу завдання 17 тривалості 1 і розрахуємо Момент, коли кожен виконавець виконує завдання з урахуванням результатів етапу 16. Результати наведені в таблиці 2.17.

Таблиця 2.17 – Визначення навантаження машин

Виконавець (№)	**Моменти завершення виконання завдання №17**
1	28
2	29
3	28
4	30

Визначаємо найменшого виконавця на момент виконання завдання. На етапі 17 завдання можна призначити виконавцю №3, як показано на малюнку 2.8.

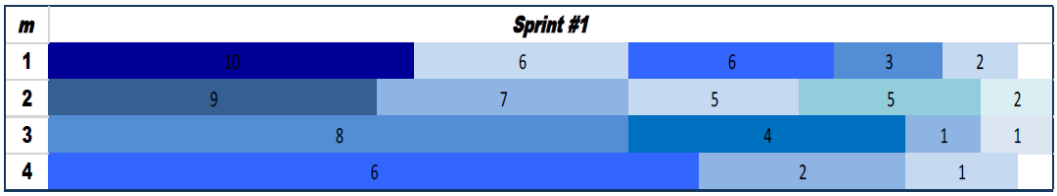

Рисунок 2.8 – Призначити останнє завдання виконавцю №3

Отримані результати заносимо в таблицю 2.18.

Таблиця 2.18 – Момент, коли кожен виконавець виконує завдання

m	C_i	**Ет. 1**	**Ет. 2**	**Ет. 3**	**Ет. 4**	**Ет. 5**	…	**Ет. 12**	**Ет. 13**	**Ет. 14**	**Ет. 15**	**Ет. 17**
1	27	1(10)	-	-	-	5(6)	…	13(2)	-	-	-	-
2	28	-	2(9)	-	4(7)	-	…	-	14(2)	-	-	-
3	28	-	-	3(8)	-	-	…	-	-	15(1)	-	17(1)
4	27	-	-		-	-	…	-	-	-	16(1)	-

Ми отримали результати жадібного алгоритму, де цільова функція має значення 28 (для виконавців №2 і №3).

У цій частині магістерської роботи запропоновано математичну формулу задачі, встановлено математичну модель та визначено критерій оптимальності задачі.

Вивчається характер завдання, характеристики завдань і виконавців у системі. Представлено обґрунтування вибору алгоритму розв’язання задачі, його переваги та недоліки, а також розв’язок алгоритму.

Наведено детальний приклад роботи жадібного алгоритму з таблицями, що записують дані після виконання кожного етапу. Наведені результати роботи цього прикладу, тобто отримані в кінці розрахунку значення цільової функції.

2.5 Спосіб управління проєктами на основі мережевого планування

Для розробки алгоритмів оптимізації систем управління проектами будемо використовувати систему методів управління проектами, а саме мережеве планування. Ці методи використовуються для розробки проектних календарів, управління проектними завданнями та запобігання всім можливим ризикам, які можуть спричинити затримки або переривання проекту, оптимізації тривалості завдань, визначення необхідних ресурсів та розподілу працівників [2]. Системи методів зазвичай використовують два елементи побудови: гістограму та мережу. Діаграми Ганта використовуються як гістограми для відображення хронологічного порядку проектних завдань. Ця діаграма допомагає візуально розподілити завдання та їх ресурси. Кожна панель представляє завдання, а її розмір – тривалість завдання. Крім того, кожна смуга має свій опис і прив’язку до іншої смуги. Гістограми дуже прості у використанні, але вони не дуже зручні у великих проектах. Боже, мережу важче налаштувати, але практичніше. Загалом, мережі використовуються для аналізу часу виконання проектних завдань, а сучасні програми поєднують ці два методи, а потім використовують мережеві діаграми для визначення критичного шляху проекту, який потім виділяється на гістограмі. Цей тип програми призначення співробітників використовує гістограми та дозволяє користувачам самостійно призначати співробітників на основі завдань. У свою чергу, ми пропонуємо створити нові оптимізації, які дозволять користувачеві автоматично призначати співробітників і дозволяють програмі виконувати розрахунки, які допомагають йому в плануванні проекту. Суть нашої концепції полягає в пріоритезації всіх вершин графа, автоматичному розподілі співробітників і визначенні мінімальної кількості співробітників, які можуть завершити проект в найкоротші терміни і завершити проект в мінімальний час. Кількість персоналу обмежена.

Мережна діаграма - це зважений граф, що складається з послідовності завдань і їх зв'язків. Для створення діаграм використовуються чотири елементи, а саме:

• Подія – це етап реалізації проекту, який може означати як початок, так і кінець роботи. Можна сказати, що подія відбувається відразу, тому що не потребує ресурсів і часу. Це показано на графі вершинами (колами).

• Роботи — це процеси, для виконання яких потрібні ресурси та час. Ресурси можуть бути у формі працівників, заробітної плати, обладнання тощо. Робота зображена на схемі у вигляді дуг (стрілок).

• Очікування – це процес, який не вимагає витрат ресурсів, але він невіддільний від витрати часу. На схемі це показано так само, як і робота.

• Залежності — це особливий вид творів, які є логічними зв’язками між творами. Залежності дозволяють дізнатися, після якої події може початися та чи інша робота, Це також вимагає вкладення часу та ресурсів. Це позначено пунктирною стрілкою.

Для системи управління проектами будемо використовувати модифікований мережевий граф, який складатиметься з вершин (завдань), дуг (зв’язків між завданнями), ваг вершин (час виконання завдання). Змінена діаграма мережі також успадкує правила діаграми мережі, а саме:

• На схемі не повинно бути петель і петель.

• Граф має початкову вершину і кінцеву вершину. Початкова вершина - це вершина, яка не з'єднана з попередньою вершиною, але з'єднана з наступною вершиною. Кінцевою вершиною вважається вершина, яка має з’єднання з попередньою вершиною і не має зв’язку з наступною вершиною.

• Кожна вершина повинна бути з'єднана принаймні з однією іншою вершиною.

В якості вхідних даних будемо розглядати двовимірну матрицю зв’язків і одновимірну матрицю ваг вершин. Матриця підключення складається з 0 і INF. Для зручності ми вирішили представити нескінченність у вигляді числа 9999, так як це дозволить потім перевірити, чи пройшла

програма певний шлях. Цю матрицю необхідно перевірити за правилами мережевої діаграми. Для цього застосуємо алгоритм перевірки (рис. 2.1). Щоб перевірити, чи матриця відповідає правилам мережевого графа, ми створимо нову двійкову двовимірну матрицю status[2][N], де N – кількість вершин. Ця матриця показує, чи зв’язана вершина з якоюсь вершиною, тобто є вхідні та вихідні вершини. Щоб визначити це, потрібно пройти матрицю з’єднань, і коли з’єднання знайдено, матриці десь присвоюється справжнє значення стану. Після цього ми створюємо нове значення підрахунку, яке буде рахувати, чи є помилки в матриці статусу. Для цього нам потрібно обійти матрицю станів і застосувати правила мережевих діаграм (рис. 2.9):

• Коли status[0][i] == false і status[1][i] == true: count[0] += 1. Таким чином ми обчислюємо початкову кількість вершин;

• Якщо status[0][i] == true і status[1][i] == false: count[1] += 1. Це використовується для обчислення кінцевої вершини;

• Коли status[0][i] == false і status[1][i] == false: count[2] += 1. Це дозволить вам ідентифікувати вершини, які не мають вхідних або вихідних вершин.

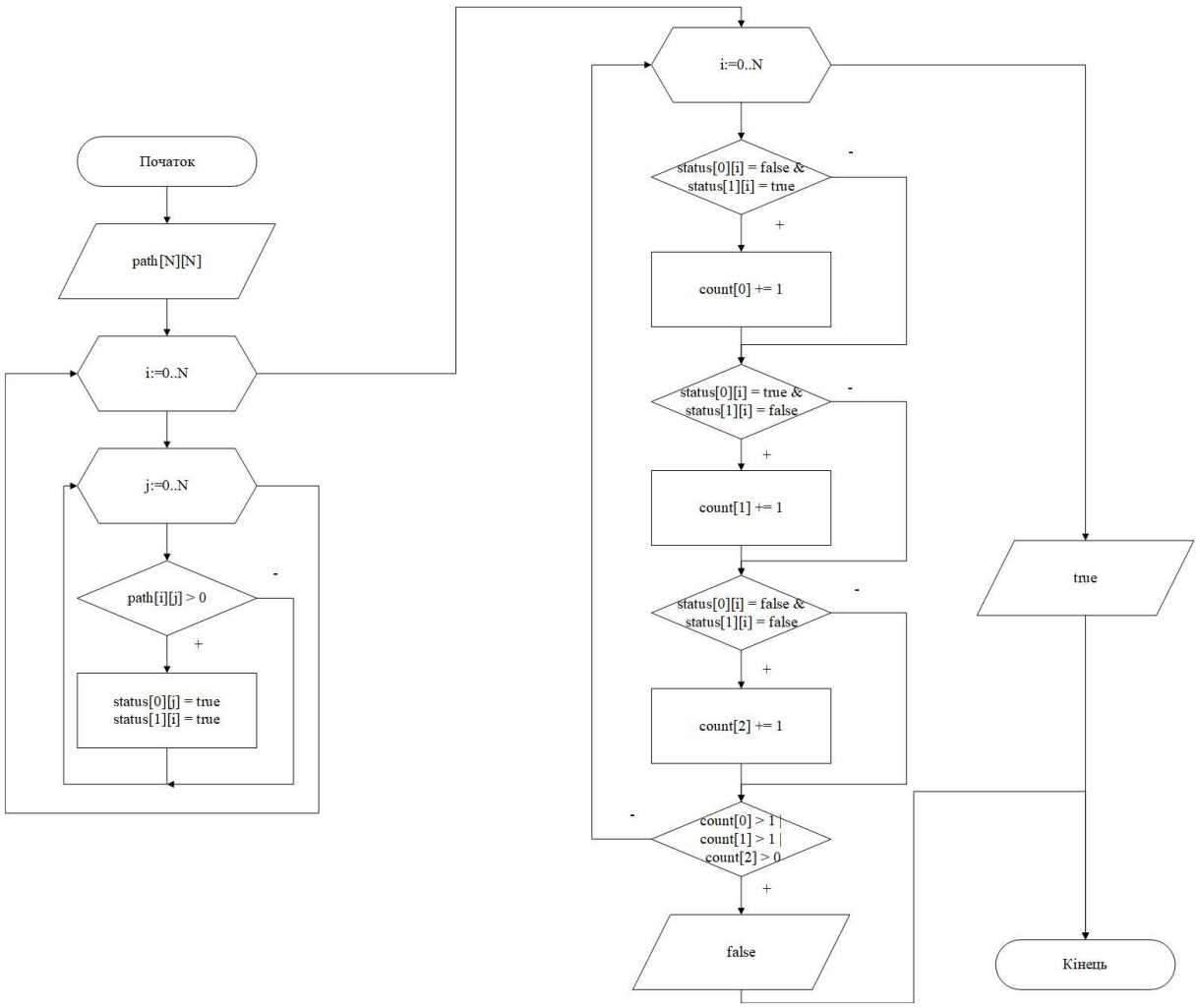

Рисунок 2.9 - Алгоритм дотримання правил мережевого графіка.

Оскільки проект має початок і кінець, можна сміливо сказати, що граф є ациклічним, що дозволяє представити граф як дерево, що складається з кількох рівнів. Спочатку потрібно знайти кінцеву вершину, яка має власну вагу, пов’язана з попереднім завданням і не має зв’язку з наступним завданням. Оскільки ми проходимо весь граф під час виконання правила мережевого графа за алгоритмом, а матриця status[2][N] ілюструє вхідні та вихідні дані вершин, ми можемо використовувати її для пошуку кінцевої вершини.

Ідея нашої оптимізації полягає в пріоритеті вершин. Крім ваг вершин, вершини також матимуть пріоритетні ваги. Для пошуку пріоритетів можна використовувати метод PERT, вхідними даними якого є найраніший і найпізніший терміни виконання завдання. Маючи ці дані, можна припустити,

що більш пріоритетними є ті завдання, у яких ранні та пізні терміни збігаються. Але ми ускладнили алгоритм і віддамо пріоритет критичному шляху вершин. Для кожної вершини необхідно знайти критичний шлях. Шлях буде визначено від кожної вершини до кінцевої вершини. Ми розглядаємо пріоритет кінцевої вершини як її власну вагу. Далі ми будемо підніматися на різні рівні дерева, які з'єднуються з останньою вершиною. У цьому випадку пріоритет вершини є сумою ваги цієї вершини (часу виконання завдання) і максимального пріоритету наступних завдань:

$$Wi + \max(CPj)$$

Де Wi — вага певної вершини, а CPj — рядок пріоритету вихідної вершини.

За знайденими критичними шляхами шукаємо максимальне значення, яке стане критичним шляхом для всього графа.

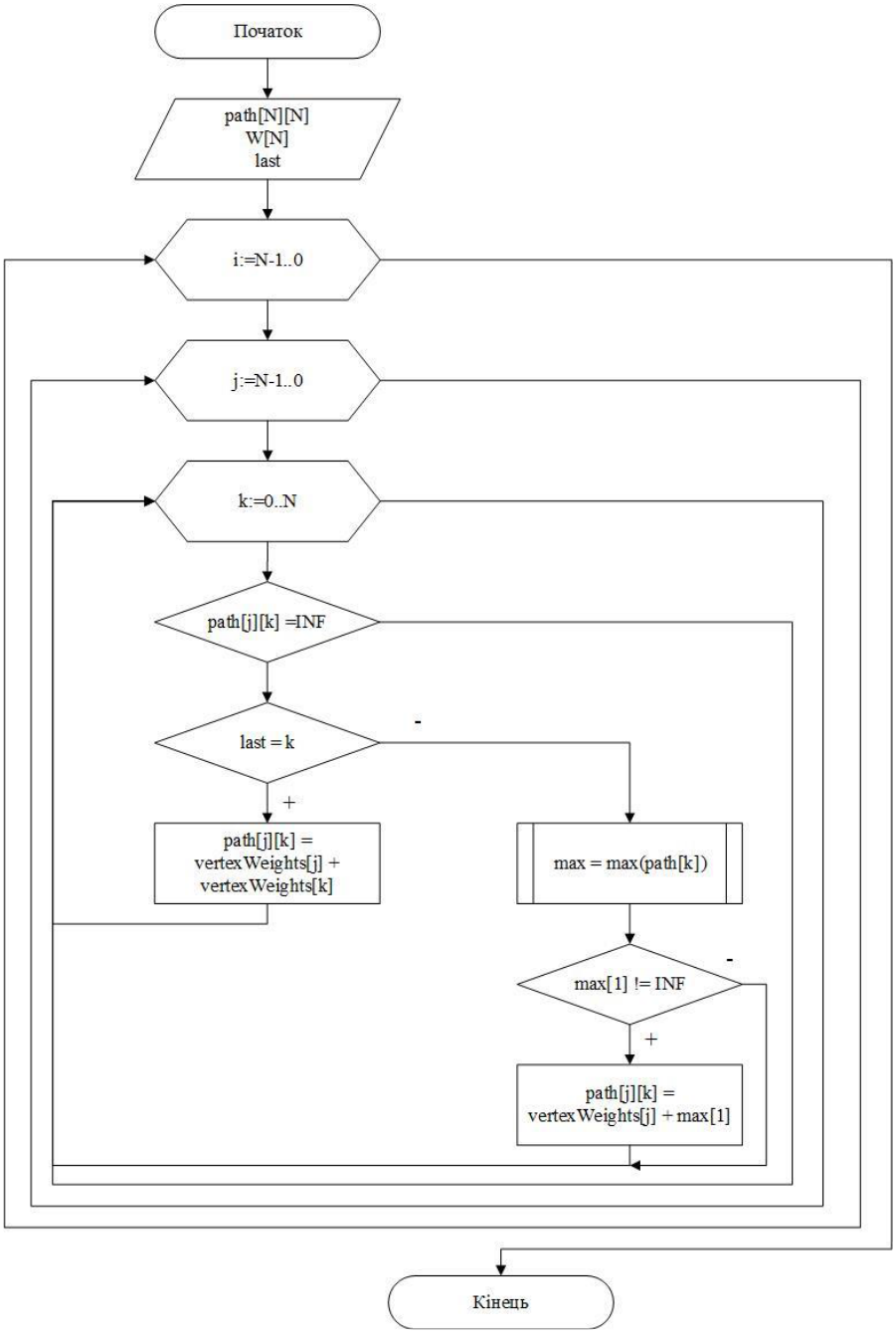

Рисунок 2.10 - Алгоритм визначення пріоритетів вершин графу.

Після того, як ми знайшли критичний шлях для вершини, ми використаємо матрицю пріоритетів для визначення критичного шляху, який буде служити вхідними даними для алгоритму. На додаток до цього нам також потрібна матриця ваг вершин.

Часто мінімальна кількість людей, необхідних для завершення проекту за найкоротший проміжок часу, дорівнює ширині діаграми. Однак це не завжди так, оскільки час виконання роботи може відрізнятися. Для цього

візьмемо загальний обсяг і розділимо його на критичні шляхи (рис. 2.10, 2.11).

$$\frac{\sum_{i=0}^{N} W_i}{\text{CP}}$$

де N – кількість вершин.

W — вага вершини.

CP — це критичний шлях проекту.

Тобто ми витратили весь час і зусилля на проект, але повністю пропустили всі завдання. Результатом суми буде час, який кожен матиме для завершення проекту. Оскільки проект не може бути завершений менш ніж за цей час, ми розділимо цей результат на критичний шлях. Дроби можуть спричинити затримки або збої в реалізації проекту, тому ми округлюємо результати. Таким чином ми виконуємо проекти в найкоротші терміни з найменшою кількістю співробітників.

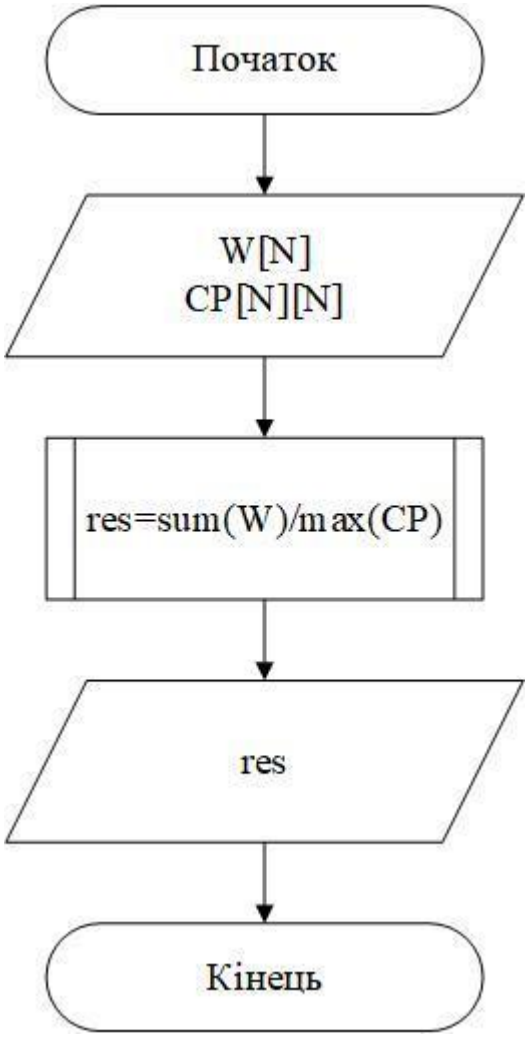

Рисунок 2.11 - Алгоритм, який використовується для визначення мінімальної кількості працівників, необхідних для реалізації проекту в найкоротші терміни

Для цього алгоритму нам потрібна матриця пріоритетів і задана кількість працівників. Спочатку розподіліть людей відповідно до завдань. Це можна зробити трьома способами:

Використовуйте матрицю пріоритетів і шукайте вакансії в порядку важливості. Ми будемо вважати важливими ті роботи найвищої ваги.

Призначте всі завдання на критичному шляху першому співробітнику та довільно розподіліть інші завдання між співробітниками.

Поєднується з попереднім способом. Дайте всі завдання на критичному шляху першому співробітнику, тому що саме ці завдання повинні бути

виконані. У свою чергу, інші працівники повинні бути призначені за пріоритетністю.

Ми вирішили вибрати перший підхід (рисунок 2.12), щоб алгоритм не був сильно навантажений і не проходив додаткові цикли. Далі ми створимо двовимірну матрицю, яка розширюватиметься під час проекту та враховуватиме дату завершення проекту. Рядки будуть відповідати за співробітників, це будуть певні процеси, на які ми завантажуємо завдання під час роботи алгоритму. Графа буде відповідати за кількість днів або годин реалізації проекту. Індекс кожного стовпця є годинником для співробітника, який виконує те чи інше завдання. Таким чином ми створимо матрицю, яка буде виглядати як діаграма Ганта. За допомогою годинника ми зможемо дізнатися, чи вільний працівник, і якщо так, підвезти його. Коли у нас є матриця, нам потрібно знати розмір стовпців, і це буде час виконання всього проекту.

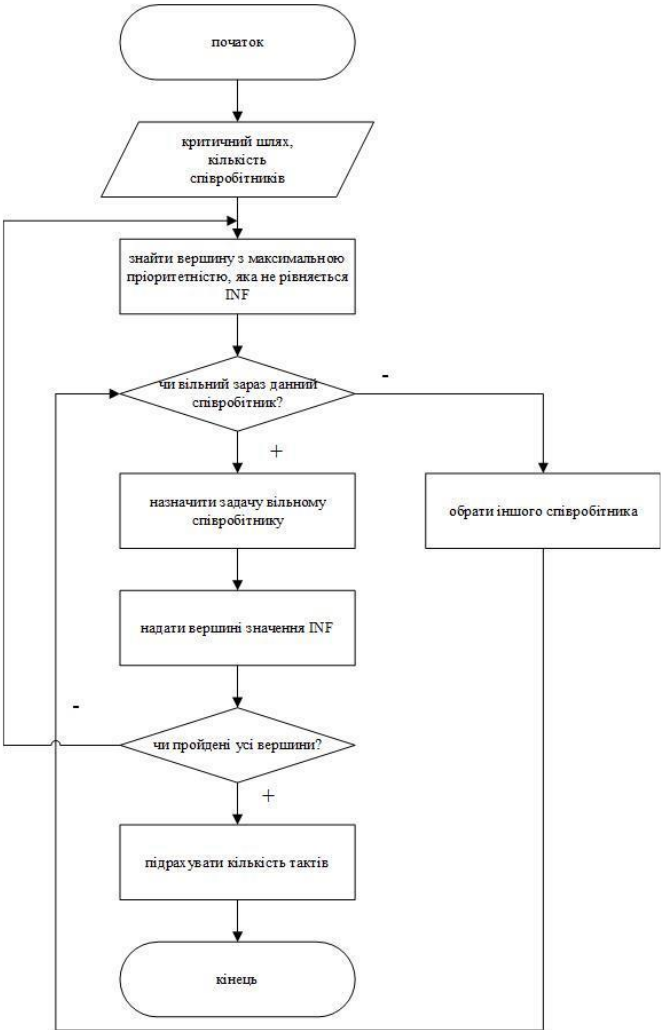

Рисунок 2.12 - Алгоритм визначення найкоротшого терміну виконання проекту при обмеженій кількості працівників.

3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис програмного забезпечення

Під час роботи над магістерською роботою було точно вибрано досить складне з точки зору реалізації рішення завдання, оскільки не один автор стикався з проблемою вибору найбільш ефективного середовища та технології розробки кількох алгоритми . Передбачає збереження всіх даних у базі даних та створення класифікаторів завдань, що вимагає високої швидкості для обраної реалізації. Далі наведено опис вимог до програмного забезпечення та обраної технології.

3.1.1. Вимоги до програмного забезпечення

У таблиці 3.1 наведено функціональні вимоги до програмного забезпечення.

Таблиця 3.1 – Функціональні вимоги до програмного забезпечення

№	Назва	Пріоритет
1	Система надає можливість вводити вхідні дані задачі: – кількість завдань; – тип команди; – тип спринта.	Високий
2	Система надає можливість генерування задач	Високий
3	Система надає можливість вводити дані для генерування задач: – кількість задач; – кількість завдань; – тип команди; – тип спринта.	Високий
4	Система надає можливість показати отриманий розклад (результат)	Середній
5	Система надає можливість проводити експерименти	Високий

№	Назва	Пріоритет
6	Система надає можливість відображати результати експериментів	Високий
7	Система надає можливість розв'язати задачу двома алгоритмами (алгоритм локального пошуку та жадібний алгоритм)	Високий
8	Система надає можливість проводити експерименти для двох алгоритмів	Високий

У цій системі кожна з вищезазначених функцій реалізована за допомогою відповідних модулів, кожен з яких містить можливість виконання певного переліку операцій, таких як редагування інформації, пошук даних і збереження результатів пошуку, формування звітів.

Функціонально-логічна структура програмного забезпечення, розробленого в рамках магістерської роботи, представлена на рисунку 3.1.

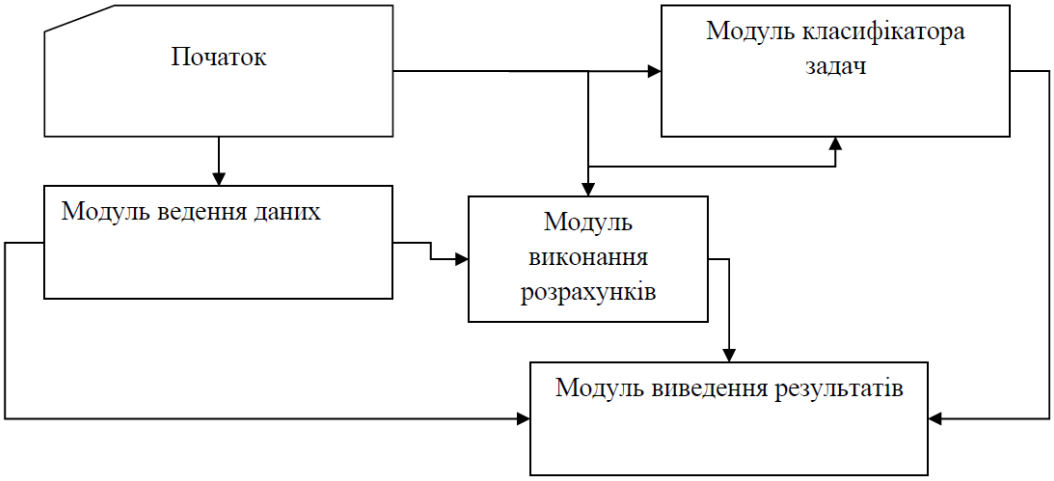

Рисунок 3.1 – Функціонально-логічна структура програмного забезпечення

Інформаційна система планування ресурсів IT-проектів успішно протестована та готова до експлуатації.

Інформаційні системи планування ресурсів IT-проектів дозволяють підвищити ефективність роботи, забезпечити простоту та зручність розрахунків, зменшити кількість помилок у плануванні ресурсів IT-проектів, пов’язаних з часовими обмеженнями, підвищити якість планування, зменшити трудовитрати на обробку інформації.

Програма може використовуватися керівниками та окремими користувачами для автоматичного планування ресурсів IT-проекту, що скоротить час розрахунку та дозволить відслідковувати зміни значень кожного окремого параметра проекту (ресурсу).

Впровадження сучасних інформаційних технологій потребує високого фінансового забезпечення. Функціонування підприємства в ринковому середовищі вимагає як мінімум аналізу економічних наслідків, а ще краще – оцінки економічної ефективності того чи іншого кроку трансформації системи.

Реалізація цієї системи залежить від багатьох факторів і є складним практичним завданням. Під час реалізації використовувався багатоетапний процес із тісною взаємодією виконавців та експертів замовника на кожному

етапі для моніторингу виконання запропонованих вимог та навчання персоналу.

Впровадження інформаційної системи планування IT-ресурсів необхідно розділити на три етапи:

дослідження. Спочатку проведіть дослідження предметної області та бізнес-процесів.

Доопрацювання системи. Програмісти налаштовують або завершують необхідні частини системи.

Запустіть систему. Почніть використовувати систему в режимі реального часу, що включає процеси навчання співробітників.

Виділіть певний час на етапі дослідницького бізнес-процесу. Тут необхідно якомога точніше описати, які процеси потребують вдосконалення.

Часто функціональні можливості інформаційної системи в рази ширші, ніж реально виконувані процеси. На цьому етапі необхідно вибрати, як наявність різних можливостей буде відображатися на вартості системи, терміни виконання та впровадження, а головне, чи відповідатимуть запропоновані функції цілям компанії.

Очевидно, що надзвичайно важливо, щоб результати дослідження були представлені окремим документом, в якому детально описано хід діяльності відповідно до вимог компанії.

Після фази дослідження необхідно надати точні відповіді щодо вартості та умов впровадження та запуску інформаційної системи.

У процесі доопрацювання системи контроль процесу впровадження необхідних частин в ІС є досить точним. Для клієнтів важливо, щоб їх представляв той, хто повністю розуміє цілі, місію та бізнес-процеси компанії.

На етапі розгортання системи необхідно перевести бізнес-процеси компанії на використання впровадженої системи.

Часто розроблені системи починають працювати не відразу. Необхідно проаналізувати ступінь успішності впровадження та чи досягнуто основних цілей впровадження.

Впровадження вважається успішним лише в тому випадку, якщо система дає можливість отримати вигоду, тобто покращує робочі процеси різних підрозділів компанії, дає можливість виконувати роботу швидше, покращує якість роботи для всіх учасників і окремих процесів. Важливо постійно аналізувати показники продуктивності системи, а також рівень інтересу співробітників компанії до використання цієї конкретної системи та де вона використовується. Процес впровадження ІБ зазвичай займає від кількох місяців до року. На цьому етапі важливо зосередитися на тому, чого компанія раніше мала на меті досягти шляхом впровадження системи. Логічно мати на увазі можливі ризики та витрати різних ресурсів. По-перше, перед прийняттям рішення про запуск ІС необхідно визначити коло завдань, які повинна підтримувати інформаційна система. Для успішного використання інформаційних систем необхідно мати в наявності обладнання, яке може підтримувати роботу системи.

Технічна підтримка в першу чергу включає організаційне обладнання для впровадження систем планування ресурсів ІТ-проектів. Організаційне обладнання призначене для реалізації технологій зберігання, представлення та використання інформації, а також виконання різноманітних допоміжних операцій у рамках певних технологій інформаційної підтримки, які керують діяльністю.

3.1.2 Архітектура програмного забезпечення

Метою розробки програмного продукту є реалізація процесу планування ресурсів ІТ-проектів. На виході програмного продукту користувач отримує графік побудови (план) виконання завдань, який використовується в рамках діяльності компанії.

Щоб проілюструвати архітектуру програмного забезпечення, ось фрагмент діаграми класів на рисунку 3.2.

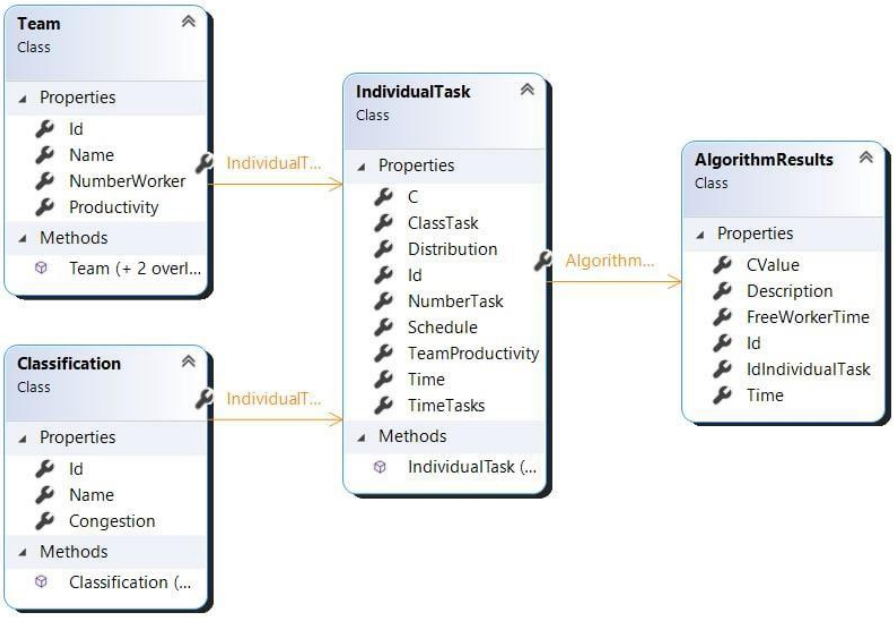

Рисунок 3.2 – Фрагмент діаграми класів програмного забезпечення

На основі результатів аналізу та врахування вищезазначених функцій було сформульовано цільове дерево системи планування ресурсів IT-проекту, як показано на рисунку 3.3.

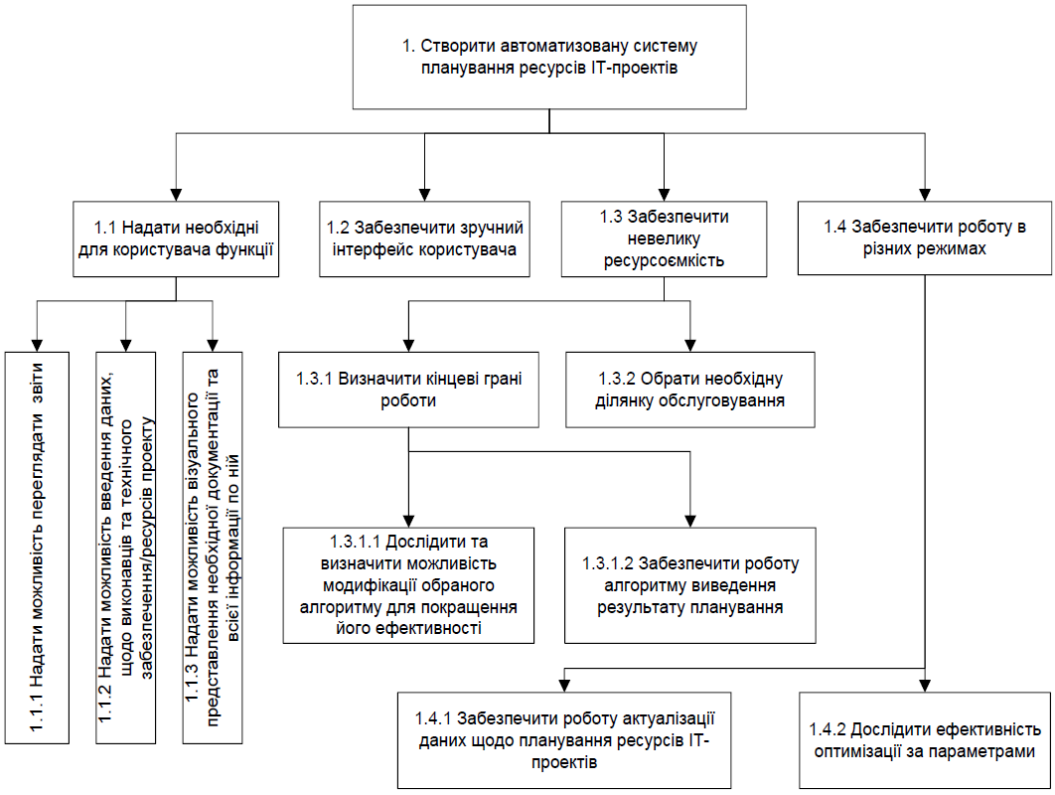

Рисунок 3.3 – Декомпозиція цілей системи

Інформаційним забезпеченням системи є єдина система класифікації та елементи єдиної системи документообігу, повторні рішення обробки інформації та методології побудови даних в організації. Нижче наведено приклад того, через які блоки обробки проходить інформація, що надходить в систему, які блоки відповідають за ту чи іншу обробку, а які взаємодіють один з одним. Схема обробки інформації наведена на рисунку 3.4.

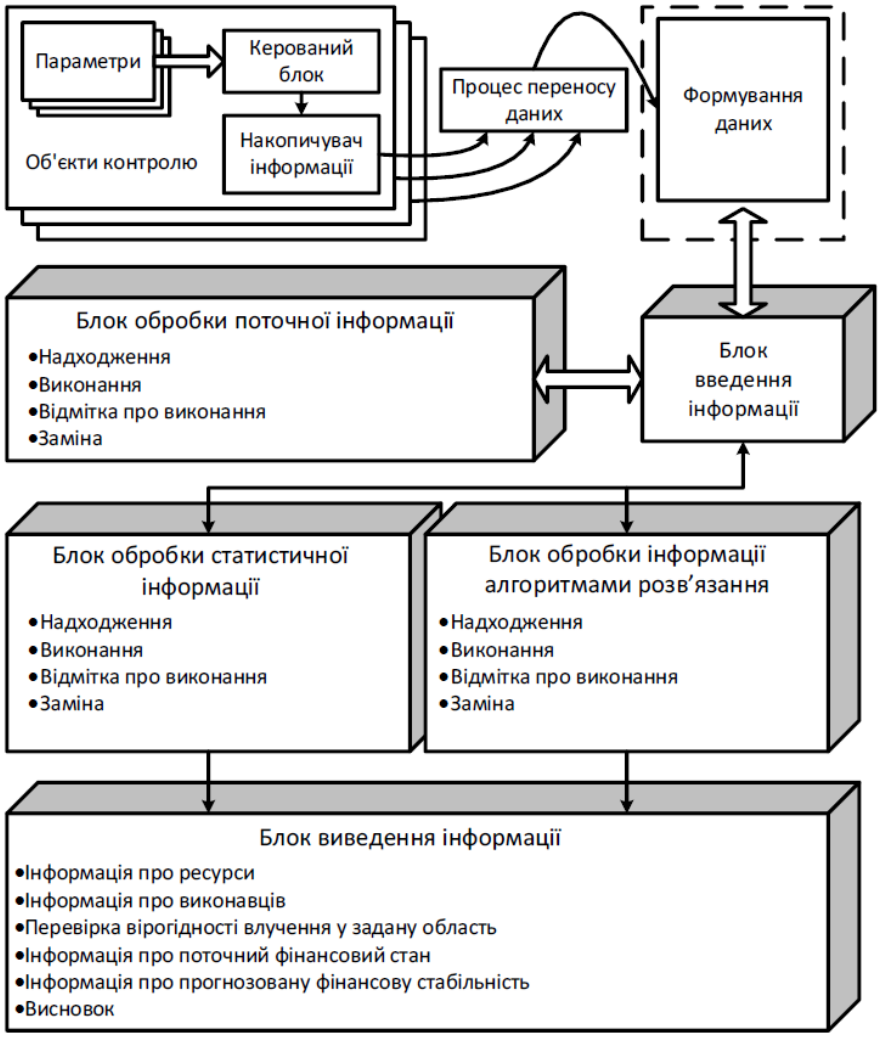

Рисунок 3.4 – Схема обробки інформації у розробленій системі

На сьогоднішній день завдання обліку документів з планування ресурсів IT-проекту на етапі експлуатації є дуже важливим через розробку та

впровадження інформаційних технологій для моніторингу документів на основі статистичних даних, реалізованих на основі визначених параметрів.

Метод моніторингу та контролю – це метод технічної діагностики, який не тільки здатний вирішити задачу визначення поточного стану та окремого планування ресурсів IT-проекту, але й здатний обробляти та оприлюднювати інформацію про стан об’єктів.

На малюнку 3.4 показано основні функціональні модулі інформаційної системи та взаємодії та зв’язки між ними. Кожен такий модуль відповідає за виконання певної функції в системі, але він не є класом програмного забезпечення.

Функціональна структура програми наведена на рисунку 3.5.

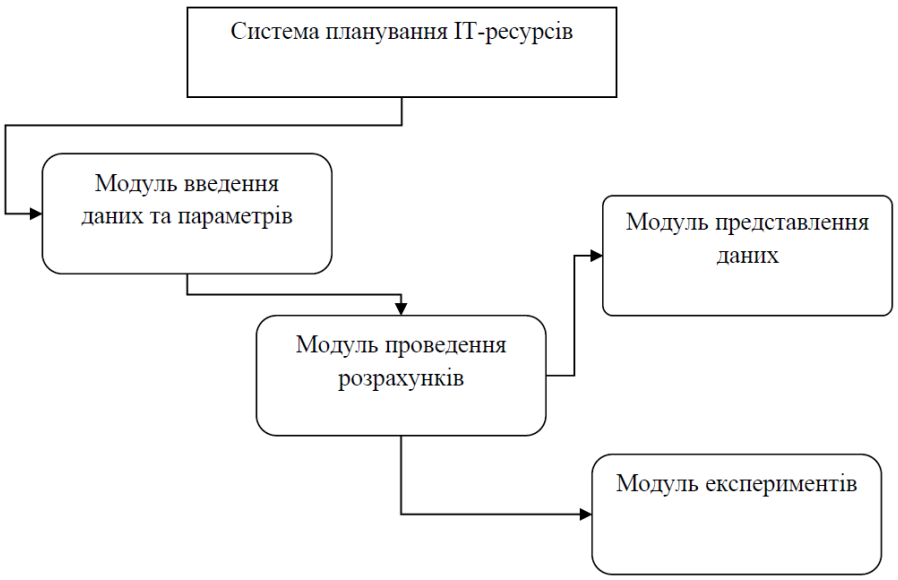

Рисунок 3.5 – Структурна схема програмного забезпечення

Модуль введення даних і параметрів

Цей модуль забезпечує введення даних про завдання та виконавців. Ця програма зберігає числові дані для кожного показника для подальшого використання системами планування ресурсів для IT-проектів

Обчислювальний модуль

Цей модуль відповідає за аналіз вхідних даних і виконання обчислень за розробленим алгоритмом для вхідної задачі.

Експериментальний модуль

Цей модуль відповідає за проведення експериментів у системі. Користувачі мають можливість проводити різноманітні експерименти над вхідними завданнями та використовувати отримані результати для досліджень. Нові показники будуть поступово доповнювати та змінювати кінцеві результати.

Модуль представлення даних

Цей модуль відповідає за рендеринг отриманих результатів і відображення їх у текстовому вигляді на екрані користувача. Чим більше вводиться вхідної інформації, тим детальніше відображаються результати.

Ця система чітко відрізняється від інших тим, що розрахована на широке коло користувачів.

3.1.3 Користувацький інтерфейс

Користувацький інтерфейс програми має бути організований зручним для користувача способом і відповідати стандартам дизайну інтерфейсу.

Інтерфейс складається з наступних частин:

- Домашня сторінка;
- Сторінка для вирішення задачі та відображення результатів (розклад);
- Сторінка з класифікатором завдань для проведення дослідів. (дослід);
- Сторінка генерації завдань (Task Generator).

Екранна форма програми, яка формує розклад після введення даних розв’язуваної задачі, наведена на рисунку 3.6-3.7 Також є різні екранні форми для роботи жадібного алгоритму та алгоритму локального пошуку.

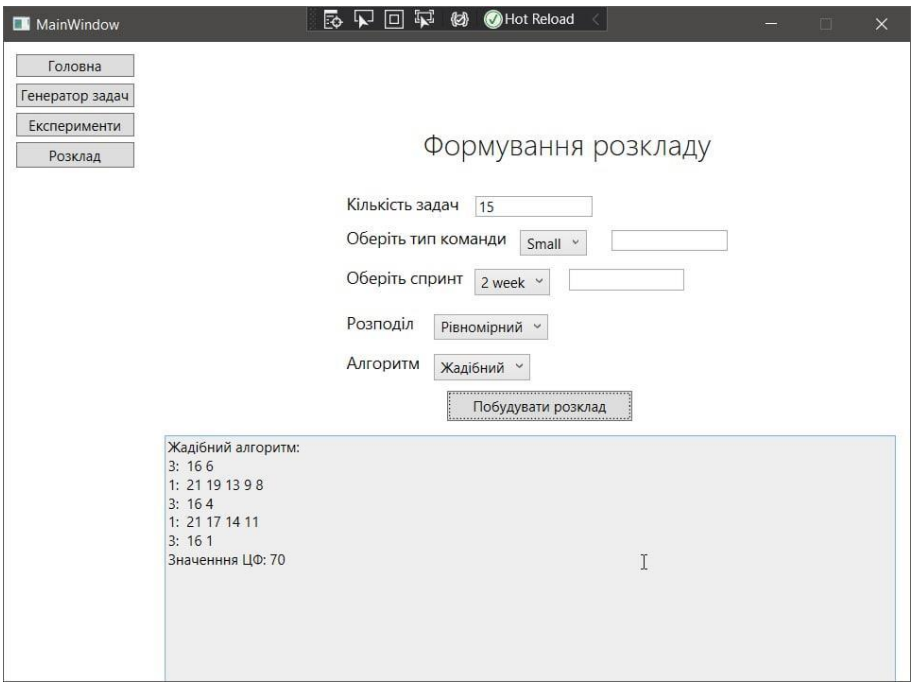

Рисунок 3.6 – Екранна форма для формаування розкладу задачі (жадібний алгоритм)

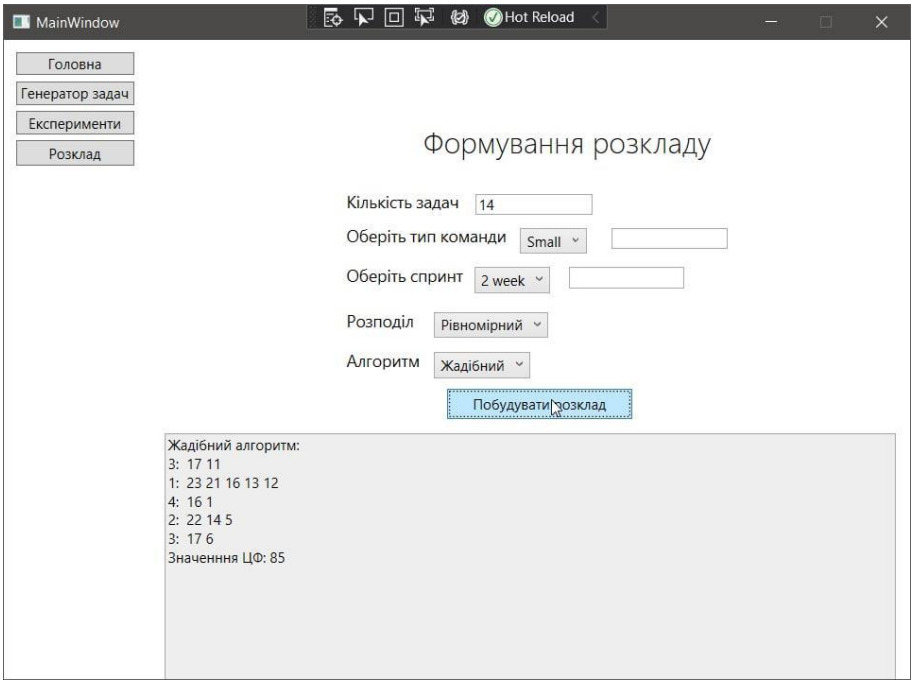

Рисунок 3.7 – Екранна форма для формування розкладу задачі (алгоритм локального пошуку)

Користувач повинен бути знайомий зі сферою управління проектами, тому що він володіє всією владою в рамках системи і його дії впливатимуть на результати роботи системи.

Система відповідає наступним вимогам і характеристикам:

- Дозволити користувачам вводити інструкції (ресурси/виконавці) у формі введення даних;

- аналізувати вхідні показники та обробляти їх, зберігаючи у формі, яка може бути оброблена програмою;

- Виконувати згортку значень для формування оптимального результату на основі набору;

- Виведення результатів роботи у формі, зручній для сприйняття та використання;

- Розрізняти вхідний доступ до різних видів інформації;

- Запобігання введенню значень, які можуть викликати збій системи;

- Зберігати та завантажувати інформацію для подальшого використання;

- кожне вікно системи має містити довідку з теоретичними відомостями та відомостями щодо використання програми;

- програмний код системи повинен відповідати стандартам кодування;

- Одночасно однією версією системи можуть користуватися кілька користувачів, які зайшли в систему з різних ПК.

Нижче наведено екранні форми для генераторів задач і експериментів на малюнках 3.8 і 3.9.

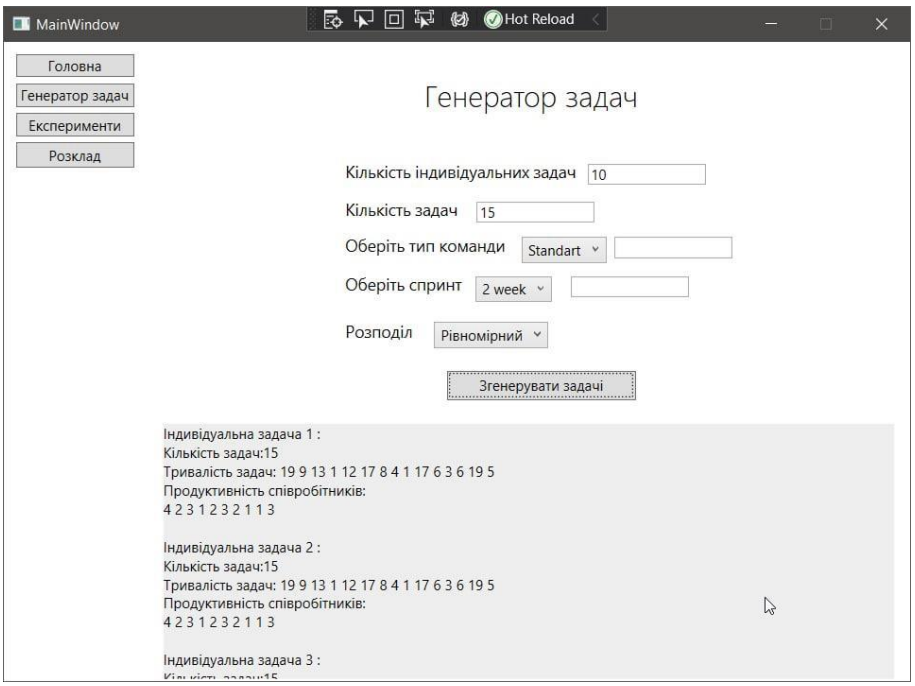

Рисунок 3.8 – Екранна форма генератора задач

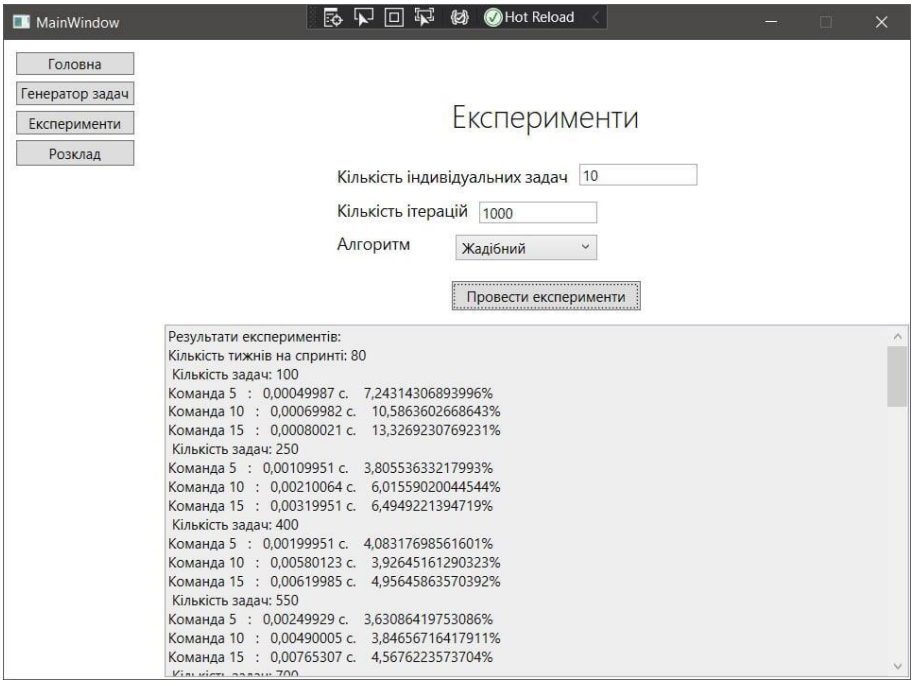

Рисунок 3.9 – Екранна форма проведення експериментів

надійність. Програма є високонадійною та може обробляти помилки користувачів.

Безпека даних. У програмі є спосіб захисту даних: шифрування файлів.

плинність. Програма працює на ПК з Windows 10/11.

Простий у використанні. Програма повинна мати зручний «дружній» інтерфейс, що дозволяє користувачеві користуватися системою без попереднього навчання її використанню.

Усі користувачі можуть використовувати базу даних, оскільки вони можуть вводити лише власну інформацію та не можуть скомпрометувати інформацію інших користувачів.

3.1.4 Засоби розробки

Вибір того чи іншого програмного засобу залежить від специфіки розробки програмного забезпечення та його популярності, а також від фінансових можливостей розробника.

Мова програмування C# є однією з найпопулярніших мов в IT-індустрії і використовується для створення настільних програм і веб-додатків. C# був створений спеціально для використання з Microsoft .NET Framework (платформа), яка пропонує багато можливостей і спрощує створення додатків. Одним із основних компонентів пакета Microsoft .NET Framework є середовище CLR, яке є загальномовним середовищем виконання, яке компілює код програми MSIL (з кодом C#, скомпільованим у нього) під час виконання, а також надає програми MSIL (Тому , а також програми, написані на мовах високого рівня, які його підтримують .NET Framework), щоб отримати доступ до бібліотеки класів .NET Framework.

Використовуючи C# і .NET Framework, ви можете створювати класичні настільні програми Windows Forms і Windows Presentation Foundation, веб-програми (з використанням технології ASP.NET), компоненти для розподілених програм і доступ до бази даних. C# надає засоби для розробки майже будь-якого типу програмного забезпечення для платформи Windows.

Дружній інтерфейс був розроблений на C# для використання з тестовою системою.

Для того, щоб програма могла виконувати перераховані вище функції, такі як обчислення, виведення результатів, реагування на дії користувача, такі як натискання кнопки, вибір рядка зі списку, необхідний програмний код.

Як інструментальне середовище проектування використовувався Rational Software Architect.

3.2 Опис отриманих результатів розв'язування та проведених експериментів

Для вирішення поставленої задачі було проведено експериментальне дослідження. Нижче наведено детальний опис отриманих результатів.

Оскільки магістерська робота пропонує багатокритеріальну задачу в теорії планування, спочатку визначається певний класифікатор задачі, на основі якого розподіл вхідних даних розподіляється через генератори різних значень.

Під час експерименту виділяються основні параметри, які впливають на результати та час роботи алгоритму.

Параметр 1 — розмір спринту. Оскільки завдання реалізуються за допомогою методології Scrum, завдання плануються за допомогою спринтів. Таблиця 3.2 показує назви та розміри спринтів відповідно.

Таблиця 3.2 – Класифікація спринтів за розміром

Позначення у класифікаторі	Тривалість спринта у тижнях	Потужність спринта на 1 людину
2 weeks	2 тижні	80 годин
3 weeks	3 тижні	120 годин
4 weeks	4 тижні	160 годин

Відповідно, потужність спринту всієї команди розраховується відповідно на основі кількості членів команди, які подали заявку на участь.

Параметр 2 – Розмір команди. Оскільки ми використовуємо Scrum, ці команди зазвичай менші, але кількість членів команди різна. Діапазон членів команди наведено в таблиці 3.3.

Таблиця 3.3 – Класифікація команд за розміром

Позначення у класифікаторі	**Кількість учасників**
Small	1-5 учасників
Standart	6-10 учасників
Big	11-15 учасників

Параметр 3 — це кількість завдань на спринт. Це значення використовувалося в діапазоні від 100 до 1000 завдань, оскільки завдання можуть виконуватися протягом тривалого часу, але їх може бути досить багато під час входу. навпаки. У таблиці 3.4 нижче наведено значення кількості завдань, використаних для експериментів.

Таблиця 3.4 – Перелік значень кількості завдань для експериментів

Кількість завдань
100
150
400
550
700
800
1000

Продовжимо експеримент.

Експеримент 1.

Для проведення експерименту командам були поставлені такі завдання: 100, 150, 400, 550, 700, 850 і 1000. Для алгоритмів жадібного та локального пошуку визначено процентне відхилення отриманого значення планування від очікуваного значення та цільову функцію.

На вході маємо:

- спринт - 2 тижні (тижні);

- Кількість членів команди змінна;

-Кількість завдань змінна. На виході отримуємо:

Відсоткове значення (%) відхилення отриманого плану від очікуваного значення та цільової функції.

Результати цього експерименту показані на малюнках 3.10 і 3.11, а отримані дані наведені в таблиці 3.5.

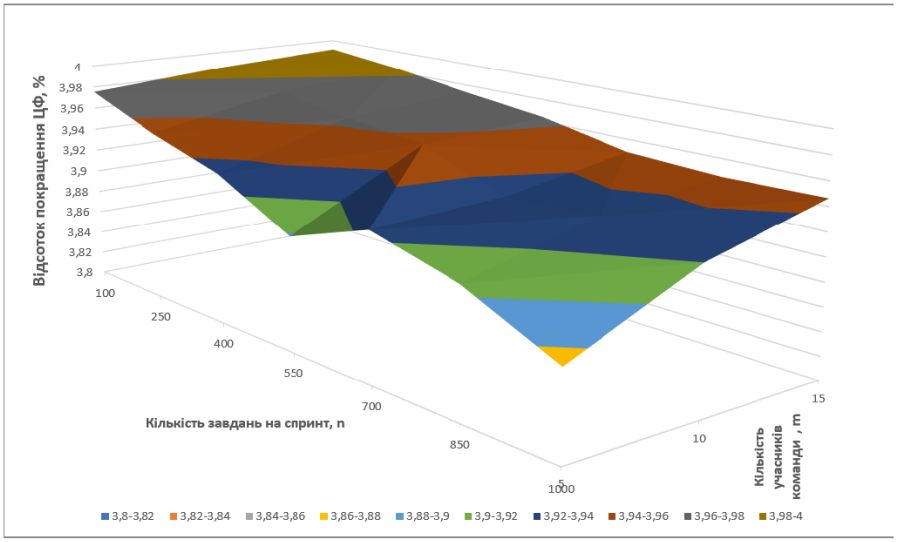

Рисунок 3.10 – Графік результатів алгоритму локального пошуку, що показує відсоткове покращення значення цільової функції

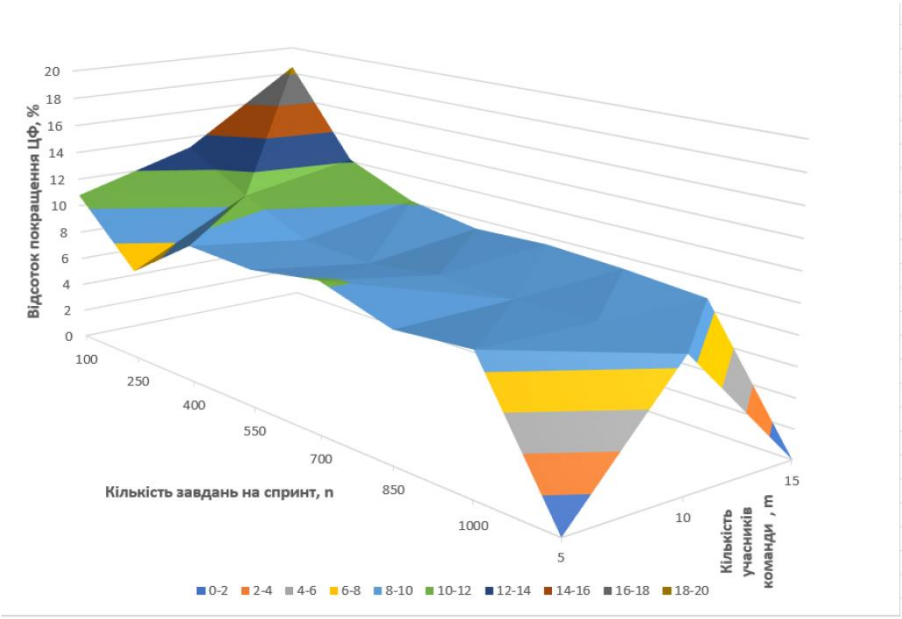

Рисунок 3.11 – Графік результатів жадібного алгоритму, що показує відсоткове покращення значення цільової функції

Таблиця 3.5 - Процентне відхилення значень CF від очікуваних результатів жадібного алгоритму

	Кількість учасників команди		
Клас спринта	5	10	15
2 weeks	10,7924	13,21922	18,48312
	6,456402	10,66193	12,14874
	9,683307	8,630869	10,13338
	9,45012	8,318408	9,310358
	10,22611	8,96475	9,515012
	8,67269	8,839579	9,227035
	9,12331	9,026354	8,759698
3 weeks	10,33026	8,703937	18,50117
	8,688955	11,33903	11,98379
	9,313208	9,281798	10,15748
	9,690909	9,111662	9,774733
	9,955739	8,622708	8,810604
	8,818788	8,69741	9,272911
	8,600035	9,312917	8,837391
4 weeks	10,37504	13,46775	17,64857
	10,03943	10,25907	12,47046
	9,189422	8,414399	9,807089
	9,139294	9,551259	9,019445
	8,05643	8,743343	9,323892
	9,269489	8,740028	8,880613
	8,810404	9,350912	8,428605

Жадібний алгоритм отримує найбільше процентне відхилення значення цільової функції, але досягає значної переваги в часі роботи алгоритму. Алгоритм забезпечує такі значення цільової функції, щоб відхилення значення цільової функції від отриманих результатів не перевищувало 3-4%.

Експеримент 2

Для проведення експерименту командам були поставлені такі завдання: 100, 150, 400, 550, 700, 850 і 1000.

На вході маємо:

- спринт – 4 тижні;

- Кількість членів команди змінна;

-Кількість завдань змінна. На виході отримуємо:

Для цього типу задачі значення часу роботи алгоритму в мілісекундах.

Результати експерименту наведені на рисунках 3.12 і 3.13, а отримані значення наведені в таблиці 3.6.

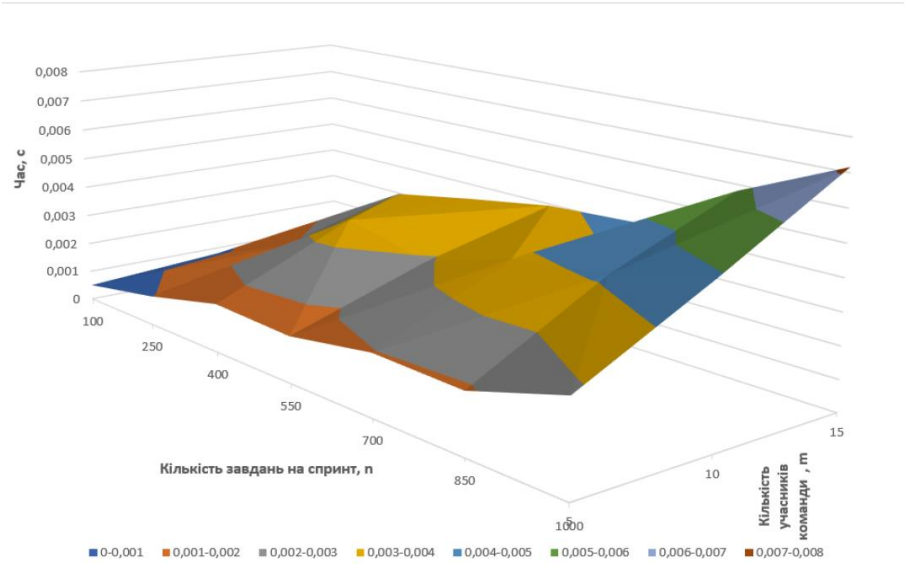

Рисунок 3.12 – Графік результатів роботи жадібного алгоритму з урахуванням параметрів

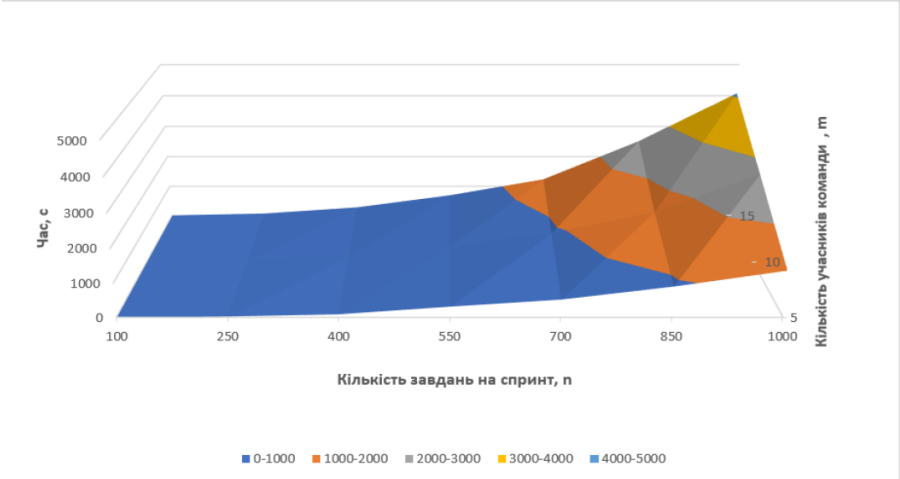

Рисунок 3.13 – Діаграма результату часу роботи алгоритму локального пошуку з урахуванням параметрів

Таблиця 3.6 – Час роботи жадібного алгоритму в секундах

	Кількість учасників команди		
Клас спринта/ кількість завдань	5	10	15
2 weeks			
100	0,000498	0,000802	0,00112
250	0,000967	0,001805	0,00297
400	0,001589	0,003515	0,003478
550	0,001483	0,002909	0,003942
700	0,001955	0,003813	0,004103
850	0,001887	0,004164	0,005809
1000	0,00282	0,004922	0,007163
3 weeks			
100	0,000248	0,000418	0,000544
250	0,00053	0,000941	0,001483
400	0,001005	0,002123	0,00305
550	0,001438	0,00256	0,004227
700	0,001767	0,003274	0,005468
850	0,002136	0,003289	0,005048
1000	0,002515	0,005245	0,006621
4 weeks			
100	0,000337	0,000529	0,00069
250	0,000683	0,001269	0,001723
400	0,001077	0,002042	0,003048
550	0,001206	0,002246	0,003228
700	0,001721	0,003332	0,005284
850	0,002067	0,004274	0,005971
1000	0,002182	0,003947	0,005576

Щодо швидкості роботи: Вище наведені результати дослідження, що відповідають швидкості роботи алгоритму.

Експерименти показують, що алгоритми локального пошуку добре працюють у пошуку кращих рішень, незважаючи на те, що виконання ітерацій для пошуку кращого рішення займає більше часу. Це також пов’язано з тим, що алгоритм локального пошуку базується на жадібному алгоритмі та повторюється за певних умов на кожній ітерації.

Дослід 3

Результати конвергенції двох алгоритмів показують, що алгоритм локальної засідки плавно сходиться після 210 вимірювань, тоді як жадібний алгоритм сходиться після 250 і 270 вимірювань відповідно. Тобто, з точки зору пошуку оптимального рішення, алгоритм локального пошуку може знайти рішення за меншу кількість ітерацій, ніж жадібний алгоритм.

Розділ 3 знайомить із програмним забезпеченням та технічною підтримкою системи планування IT-ресурсів. Розкриває структуру програми та описує процес взаємодії з програмою.

Наведено деякі експерименти, проведені під час дослідження, та їх результати. Розглянемо евристичні алгоритми вирішення: жадібний і локальний пошук. Для задач великої розмірності також існує значна різниця в часі роботи алгоритмів: жадібний алгоритм працює в 9000 разів швидше, ніж алгоритм локального пошуку. Це пояснюється тим, що для 100 завдань жадібний алгоритм будує розклад для кожного виконавця, і на цьому його робота закінчується, тоді як алгоритм локального пошуку має заданий ліміт ітерацій для повторення перестановки, що призводить до створення великої кількості операцій для 100 розклад завдань. Але в той же час використання жадібного алгоритму не гарантує кращих результатів, а алгоритм локального пошуку дає результати, які відхиляються від значення цільової функції на 3-4%. Представлена тривимірна діаграма, побудована за отриманими результатами дослідження.

Керівники та окремі користувачі можуть використовувати програму для автоматичного планування ресурсів IT-проекту, що скоротить час розрахунків і дозволить відстежувати зміни кожного окремого параметра.

ВИСНОВКИ

У магістерській роботі розроблено набір взаємопов’язаних моделей, методів і алгоритмів як основи для системи підтримки прийняття рішень, яка надає керівникам проектів своєчасну інформацію про статус проекту та підтримує рішення щодо управління ресурсами та календарного планування.

В ході роботи вирішувалися такі завдання:

- Дослідити та визначити переваги та недоліки існуючих методів управління проектами, програмних комплексів, що спеціалізуються на автоматизації процесів управління проектами та їх підтримки, алгоритмів планування проектів, використання інтелектуальних інструментів в управлінні ресурсами проекту;

- Розроблено підхід до управління ресурсами проекту, який дозволяє розподіляти завдання проекту між ресурсами, розробляти графіки проекту з урахуванням ризиків, контролювати стан проекту та приймати управлінські рішення щодо заміни ресурсів у проекті.

Розробка інформаційної системи планування ресурсів IT-проектів для підвищення ефективності роботи, забезпечення простоти та зручності розрахунків, зменшення кількості помилок при плануванні ресурсів IT-проектів, підвищення надійності інформації, зниження трудовитрат при обробці інформації.

Керівники та окремі користувачі можуть використовувати програму для автоматичного планування ресурсів IT-проекту.

Згідно з дослідженнями, основними силами, що впливають на галузеву конкуренцію, є постачальники та споживачі. Крім того, у міру розвитку ринку інтенсивність конкуренції серед існуючих конкурентів і загроза замінників стають все більш важливими.

Результати магістерської роботи:

- На основі аналізу існуючих рішень було визначено, що існуюча система надає можливість розподіляти ресурси та будувати моделі для реалізації проекту з урахуванням ресурсних обмежень, але не можна визначити загальні часові обмеження для весь проект протягом планового періоду з урахуванням індивідуальних показників кожного виконавця завдання;

- Розроблено програмний продукт, який забезпечує побудову планів розподілу завдань між учасниками на основі ефективних алгоритмів, розроблених для розв'язання задач планування ресурсів, що дозволяють будувати оптимальні або наближені до оптимальних плани. Досліджено ефективність розробленого методу. З цією метою було проведено багато експериментів на основі результатів жадібних алгоритмів і алгоритмів локального пошуку в різних вимірах проблеми.

- Було визначено, що алгоритми локального пошуку можуть знаходити кращі рішення з меншою кількістю ітерацій, ніж жадібні алгоритми. Для задач великої розмірності також існує значна різниця в часі роботи алгоритмів: жадібний алгоритм працює в 9000 разів швидше, ніж алгоритм локального пошуку. Це пояснюється тим, що для 100 завдань жадібний алгоритм будує розклад для кожного виконавця, і на цьому його робота закінчується, тоді як алгоритм локального пошуку має заданий ліміт ітерацій для повторення перестановки, що призводить до створення великої кількості операцій для 100 розклад завдань. Але в той же час використання жадібного алгоритму не гарантує кращих результатів, а алгоритм локального пошуку дає результати, які відхиляються від значення цільової функції на 3-4%.

ПЕРЕЛІК ПОСИЛАНЬ

1. Васильців Т. Г., Качмарик Я. Д., Блонська В. І., Лупак Р. Л. Бізнес-планування : навч. посіб., 2013. 173 с.

2. SGV.IN.UA. Management and Project Management: Theory and Practice [Електронний ресурс] – URL: https://sgv.in.ua/off-lifaq/25- suchasni-metodi-upravlinnya-proektami (дата звернення: 25.09.2023)

3. Бродська А. О. Використання інформаційних технологій в управлінні проєктами підприємств [Електронний ресурс] Управління розвитком складних систем. – 2013. – Вип. 13. – С. 8-11. – URL: http://urss.knuba.edu.ua/files/zbirnyk-13/8-11.pdf (дата звернення: 27.09.2023)

4. Вольфсон Б. Гибкие методологии разработки / Б. Вольфсон [Электронный ресурс] – Электрон. текстовые дан. – URL: http://adm-lib.ru/books/10/Gibkie-metodologii.pdf (дата звернення: 01.10.2023)

5. Воробець С. Й. Створення автоматизованих інформаційних систем на засадах процесного підходу / С. Й. Воробець, В. П. Кічор, А. В. Симак // Менеджмент та підприємництво в Україні: етапи становлення і проблеми розвитку : збірник наукових праць ; відп. ред. О. Є. Кузьмін. – Львів : Вид-во Львівської політехніки, 2012. С. 408–413.

6. Голоскокова А. О. Моделі та інформаційна технологія планування покращення якості процесу розробки програмного забезпечення: диссертація/ Голоскокова А. О. - Національний технічний університет" Харківський політехнічний інститут", 2018.

7. Данчук В. Д. Специфіка впровадження agile методологій для проєктів розробки програмного забезпечення. Данчук В. Д., Луцюк Д. В. - Вісник Національного транспортного університету 24, 2011 с 346-350.

8. Джулій В.М. Методи та алгоритми розробки web-додатків. Джулій В.М. - Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка, 2017 с. 107-14.

9. Кіфер В. М. СКРАМ-ефективний підхід при розробці ПЗ/ Кіфер В. М. - Зв'язок 4, 2012 с. 10-12.

10. Коляда А. С. Эффективность использовани адаптивных подходов при разработке программного обеспечения. Коляда А. С. - Інформаційні технології в освіті, науці та виробництві,2012 с 29-33.

11. Кондрин А. В., Кукарцев А. В. Стратегия внедрения CALS-технологий. Вестник СибГАУ. 2011. № 3 (36). С. 210-214.

12. Марченко А. В. Методология управления проєктами Scrum как пример внедрения методологии AGILE. Марченко А. В. - Зв'язок 4, 2016 с. 27-30.

13. Морозов В. В. Компоненти управління проєктами: навчальний посібник для самостійної роботи студентів магістратури по спеціальності 8.000003 "Управління проєктами" В.В. Морозов. – К.: Університет економіки та права "КРОК", 2005. 62 с.

14. Морозов В. В. Формування, управління та розвиток команди проєкту / В.В.Морозов, А.М.Чередніченко, Т.І. Шпильова. – Київ: Таксон, 2009. 461с.

15. Морозов В. В.Інформаційні системи і технології в управлінні проєктами. Планування проєктів у MS Project: навчальний посібник / В.В.Морозов, О.Б.Данченко, О.І. Шаров. – К. : Університет економіки та права "КРОК", 2011. 167 с.

16. Новаківський І. І. Проєктно орієнтована організаційна система управління як ціль еволюції проєктного менеджменту І. І. Новаківський. Проблеми економіки та управління. – Львів : Вид-во Львівської політехніки, 2009. – No 640. С. 163–174.

17. Ноздріна Л. В. Управління проєктами: Підручник. Л. В. Ноздріна, В. І. Ящук, О. І. Полотай; за заг. ред. Л. В. Ноздріної. – К.: Центр учбової літератури, 2010. 432 с.

18. Рач В. А. Багаторівнева системна модель виявлення специфічних проявів якостей особистостей в діяльності з управління проєктами / В. А.

Рач. Управління проєктами: стан та перспективи: матеріали IV міжнародної науково-технічної конференції. – Миколаїв: НУК, 2008. С. 131–134.

19. Самсонов А. К. Управління проєктом при створенні мобільного додатку. Самсонов А. К. - InProject, Program, Portfolio Management, 2017 — с. 79-81 ІКС ОНПУ.

20. Сванідзе Л. Г. Проєктний підхід в управлінні підприємницькою діяльністю : автореф. дис. на здобуття наук. ступеня канд. екон. наук : спец.08.06.02 “Підприємництво, менеджмент та маркетинг” Л.Г.Сванідзе. – Київ, 2001. –17 с.

21. Семенов С. Г. Усовершенствованный способ масштабирования гибкой методологии разработки программного обеспечения Семенов С. Г. - ВНТУ, 2017.

22. Семеріков С. О. Мобільне програмне забезпечення. Семеріков С. О. - Програмне забезпечення, 2010 — 156 с.

23. Столярик П. О. Особливості візуалізації розробки проєктів за Scrum методологією для розподілених команд : диссертація Столярик П. О. - ВНТУ, 2018.

24. Тернер Р.Дж. Области приложения проєктно- ориентированного управления. Р. Дж. Тернер Управление проєктами и программами, – 2017. – №3(11). – С. 220 – 236.

25. Чабанюк Я. М. Побудова і дослідження моделі надійності програмного забезпечення з індексом величини проєкту Чабанюк Я. М. - Інженерія програмного забезпечення 1.1, 2010 - № 24, с. 120-235

26. Шведа Н. М. Система управління проєктами в Україні / Н.М.Шведа, Н.Є.Юрик. Збірник тез доповідей IV Міжнародної науково- технічної конференції молодих учених та студентів «Актуальні задачі сучасних технологій», 25-26 л истопада 2015 року. – Т. : ТНТУ, 2015. – Том 2.– С. 246-247.

27. Шишкіна М. П. Якість програмних забезпечення для мобільних пристроїв Шишкіна М. П. - Науковий часопис Національного педагогічного університету імені МП Драгоманова, 2017 — 150 с.

28. Шпак Н. О. Переваги використання інформаційно-комунікаційних технологій в Україні [Електронний ресурс] / Н. О. Шпак, О. І. Венгер // Вісник Національного університету "Львівська політехніка". – 2012. – No 727. – С. 461–467. – URL: http://ena.lp.edu.ua:8080/ bitstream/ntb/13914/1/67_461-467_Vis_727_Menegment.pdf (дата звернення: 15.10.2023)

29. Каверина С.Ю., Башинская И.А. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В УПРАВЛЕНИИ ПРОЄКТАМИ / Випуск # 10 / 2017

30. Кучер О. Д, Використання ІТ в управлінні проєктами О. Д, Кучер 2019.–URL: http://dspace.mnau.edu.ua/jspui/bitstream123456789/6706/1/studentresearchjournal162-29.pdf. (дата звернення: 20.10.2023)

31. Бойко Н. І. Багатовимірне подання даних для управління ІТ-проєктами. Національний університет “Львівська політехніка”, 2015. URL: http://ena.lp.edu.ua/bitstream/ntb/29918/1/37_387-394.pdf (дата звернення: 20.09.2023) (дата звернення: 22.10.2023)

32. Івченко І. Ю., Будорацька Т.Д. Розробка моделі розподілу ІТ-проєктів на підприємствах галузі інформаційних технологій //Маркетинг і цифрові технології – ТЕС: Одеса Том 2, №3, 2018, С. 64-76 http://mdtopu.com.ua/files/download/mdt2.3.2018-16.09_1.pdf (дата звернення: 27.10.2023)

33. Філатова Т., Чернишов О. Методологія представлення соціальних проєктів ІТ-індустрії Матеріали ІІІ Міжнародної конференції«Комп’ютерна алгебра та інформаційні технології», Одеса, 20–25 серпня 2018 р./Одеський національний університет імені І.І. Мечникова, – Одеса 2018. – 198с (80-87с) http://confit.onu.edu.ua/content/CAIT-Odessa-2018.pdf (дата звернення: 30.10.2023)

34. Бродська А. О. Використання інформаційних технологій в управлінні проєктами підприємств. А. О. Бродська Управління розвитком складних систем. - 2013. - Вип. 13. - С. 8-11. - URL: http://nbuv.gov.ua/UJRN/Urss_2013_13_4. (дата звернення: 05.11.2023)

35. Project Management with Dynamic Scheduling - Baseline Scheduling, Risk Analysis and Project Control – URL: https://link.springer.com/book/10.1007%2F978-3-642-40438-2 (дата звернення: 15.10.2023)

36. A competitive genetic algorithm for resource-constrained project scheduling. Sönke Hartmann. First published: 07 December 1998 – URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/(SICI)1520-6750(199810)45:7%3C733::AID-NAV5%3E3.0.CO;2-C (дата звернення: 20.10.2023)

37. An iterative scheduling technique for resource-constrained project scheduling. K.Y.LiR.J.Willis. 2010 – URL: https://www.sciencedirect.com/science/article/abs/pii/0377221792903209 (дата звернення: 23.10.2023)

38. Abrahamsson, P., Warsta, J., Siponen, M., Ronkainen, J.; New directions on Agile methods: A comparative analysis. In proc. Of the Intl. Conf. on Software Engineering. 2003.

39. Метод підйому (локального пошуку)– URL до ресурсу: https://studfile.net/preview/3766334/page:6/ (дата звернення: 01.11.2023)

40. Жадібні алгоритми – URL до ресурсу: https://dl.sumdu.edu.ua/textbooks/95351/522264/index.html (дата звернення: 02.11.2023)

ДОДАТКИ

Додаток А

(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри АІІТ ВНТУ,
д.т.н., професор
_____Олег БІСІКАЛО

" ___ " _____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«**Автоматизована система планування і управління ресурсами при виконанні ІТ-проєктів**»

08-33.МКР.14.02.000 ТЗ

Студент групи 3АКІТ-22м

_____Євгеній СИРЦОВ___
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник доцент, доцент кафедри АІІТ

_____Роман МАСЛІЙ
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2023

1. Назва та галузь застосування

1.1. Назва – Автоматизована система планування і управління ресурсами при виконанні ІТ-проєктів

1.2. Галузь застосування – це інформаційні технології та управління проектами, зокрема в контексті розробки та впровадження інформаційних систем для автоматизації планування та управління ресурсами в ІТ-проєктах.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від "18" вересня 2023 року №247

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є розробка програмного забезпечення для планування і управління ресурсами при виконанні ІТ-проєктів.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1.Бізнес-планування : навч. посіб. / Т. Г. Васильців, Я. Д. Качмарик, В. І. Блонська, Р. Л. Лупак. –К. : Знання, 2013. – 173 с.

2. SGV.IN.UA. Management and Project Management: Theory and Practice [Електронний ресурс] – URL: https://sgv.in.ua/off-lifaq/25- suchasni-metodi-upravlinnya-proektami

3. Бродська А. О. Використання інформаційних технологій в управлінні проєктами підприємств [Електронний ресурс] / А. О. Бродська // Управління розвитком складних систем. – 2013. – Вип. 13. – С. 8-11. – URL: http://urss.knuba.edu.ua/files/zbirnyk-13/8-11.pdf

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- аналіз вхідних даних за допомогою різних методів;

- оптимізація ресурсів ІТ-проєкту;

5.2. Основні технічні вимоги до розробки:

- WINDOWS 10;
- Серидовище розробки(IDE) з підтримкою C# і .NET Framework
- ОЗУ 4 гб
- Пам`ять вбудована 500мб

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі планування і управління ресурсами. Постановка задач дослідження «20»_09__ 2023 р.
2. Розв`язання задачі планування ресурсів «01»_10__ 2023 р.
3. Розробка програмного забезпечення та аналіз результатів «11»_10__ 2023 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «10»__11_ 2023 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «21»__11_ 2023 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «20»__12_ 2023 р.

Додаток Б

(обов`язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Автоматизована система планування і управління ресурсами при виконанні ІТ-проєктів

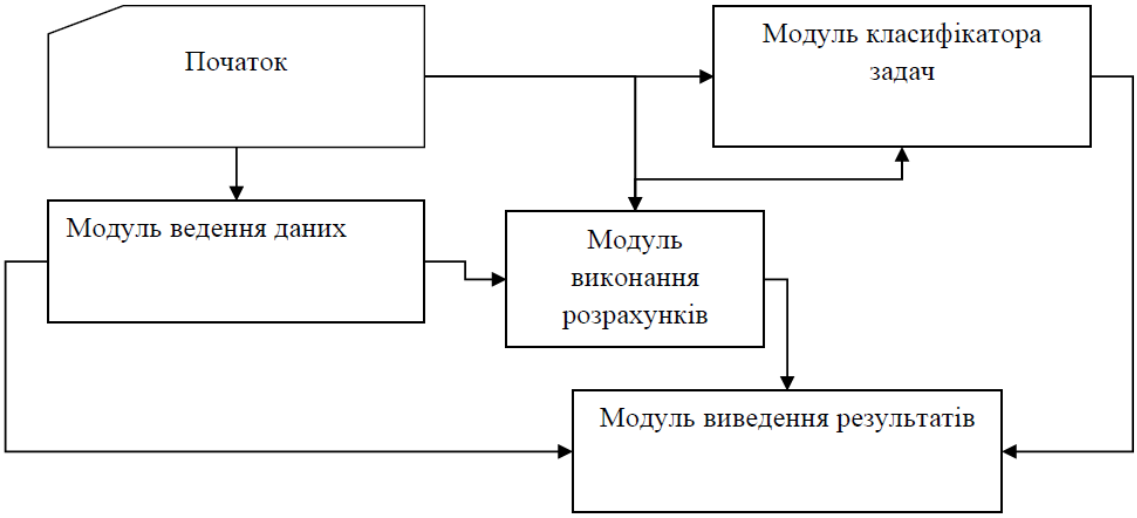

Рисунок Б.1 - Функціонально-логічна структура програмного забезпечення

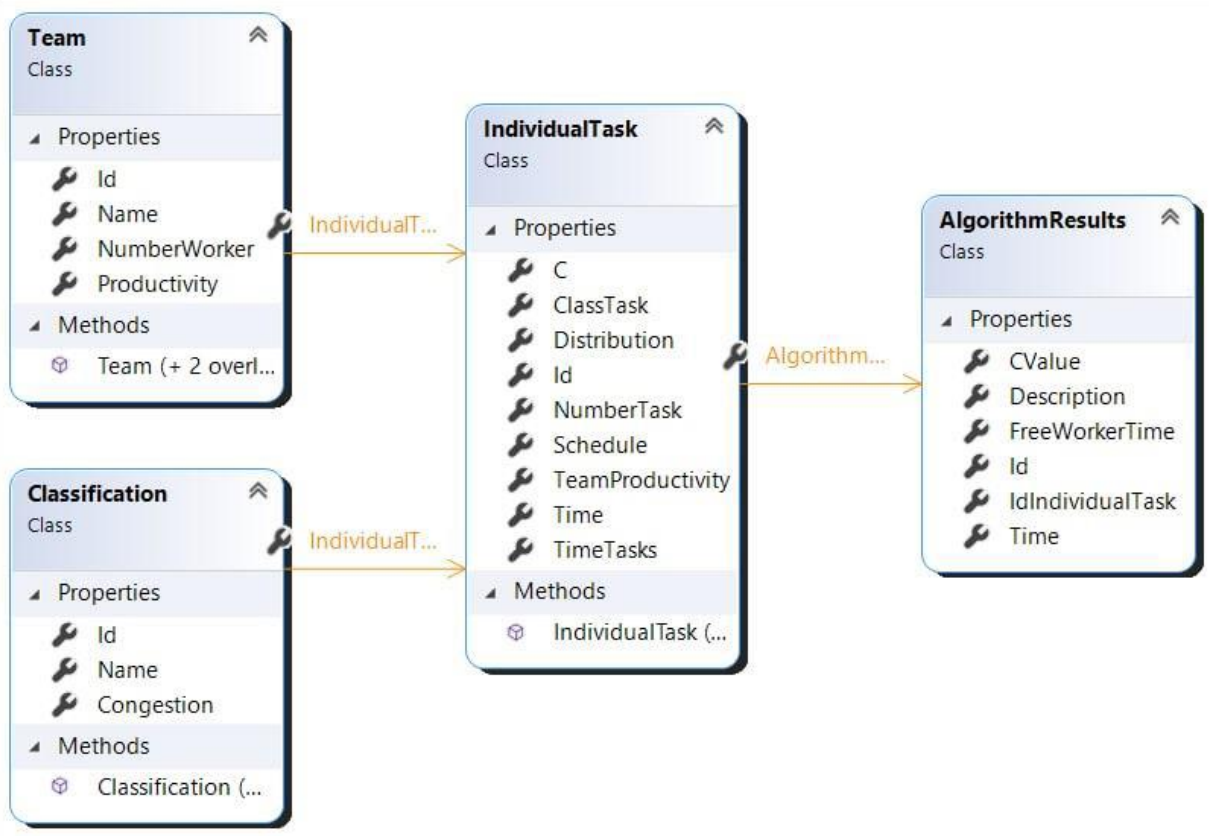

Рисунок Б.2 - Фрагмент діаграми класів програмного забезпечення

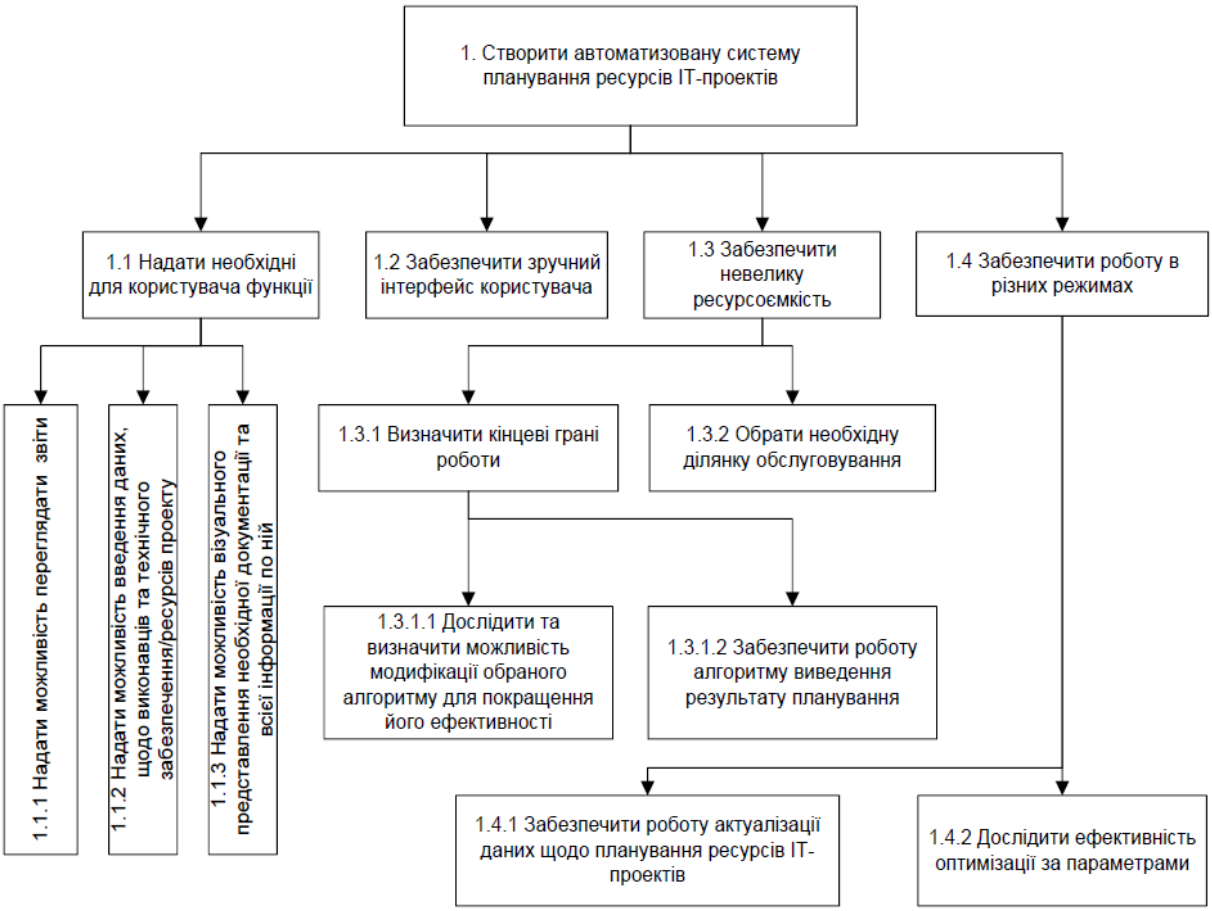

Рисунок Б.3 - Декомпозиція цілей системи

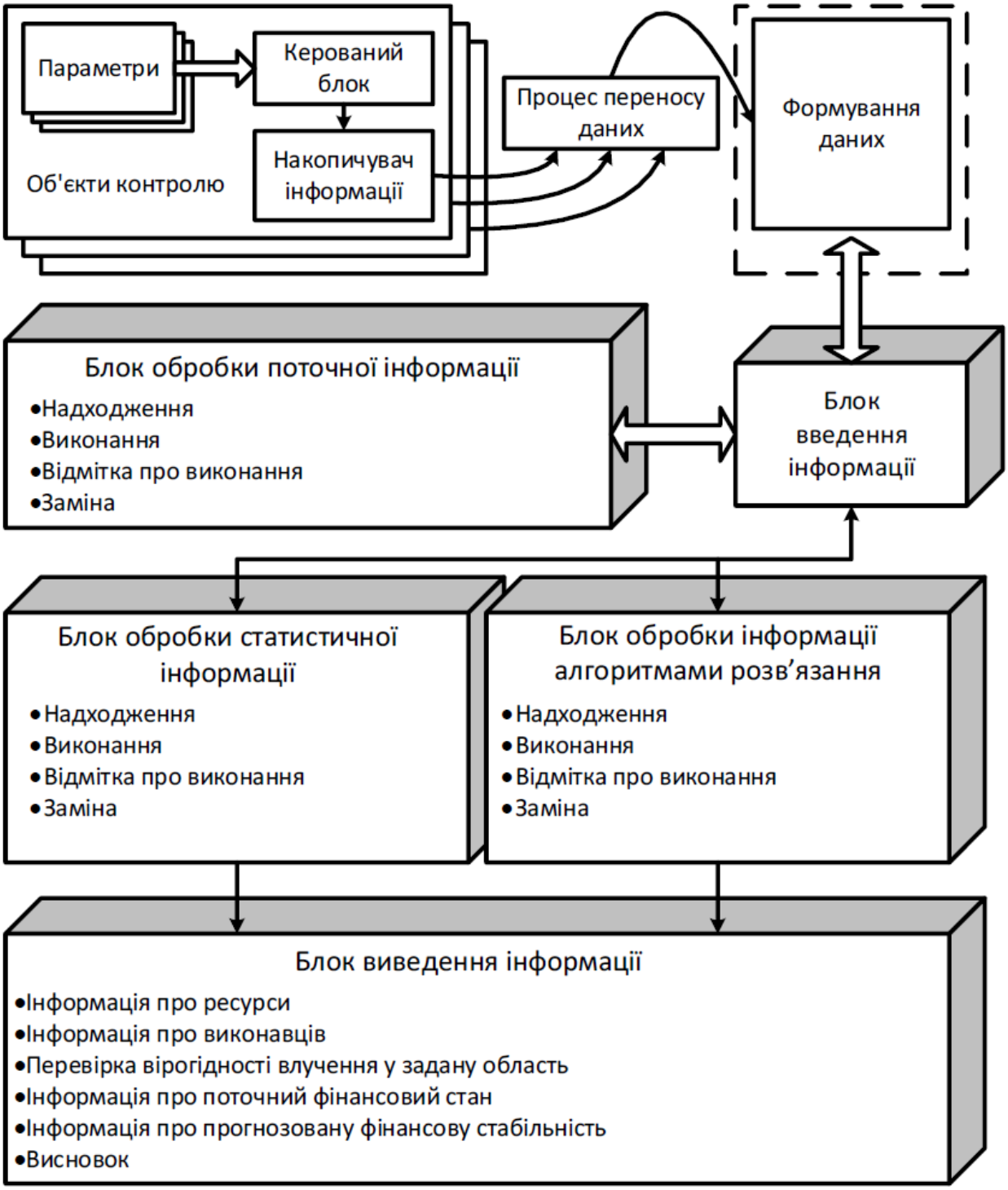

Рисунок Б.4 - Схема обробки інформації у розробленій системі

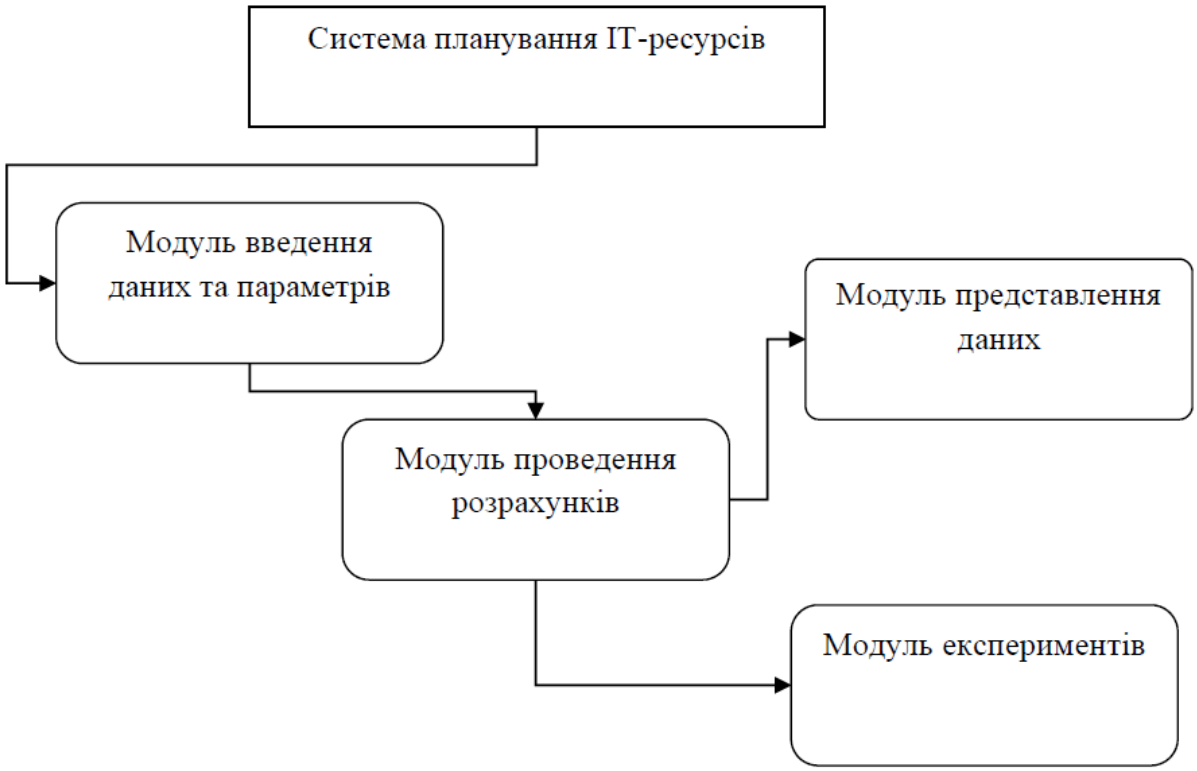

Рисунок Б.5 - Структурна схема програмного забезпечення

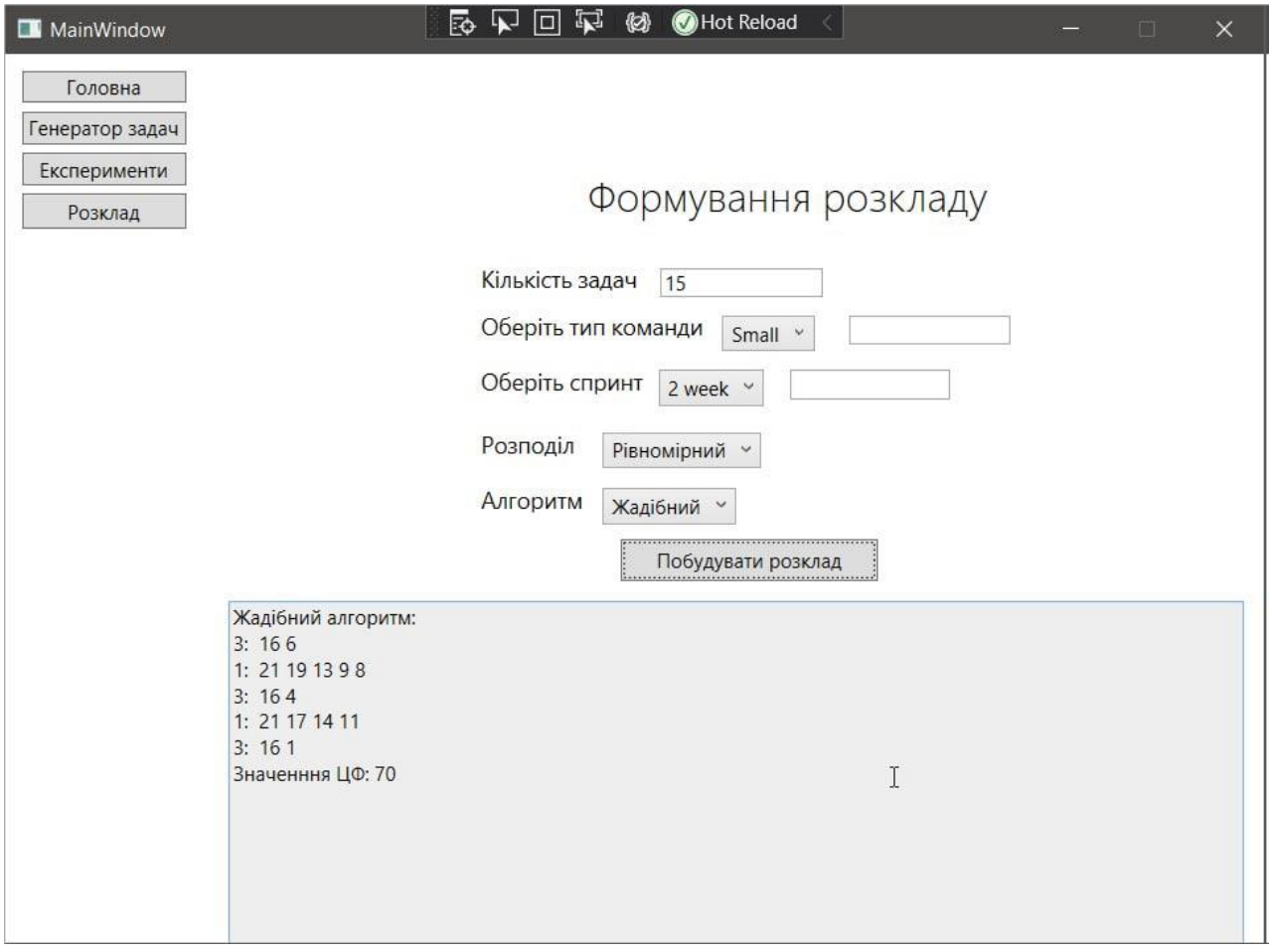

Рисунок Б.6 - Екранна форма для формаування розкладу задачі (жадібний алгоритм)

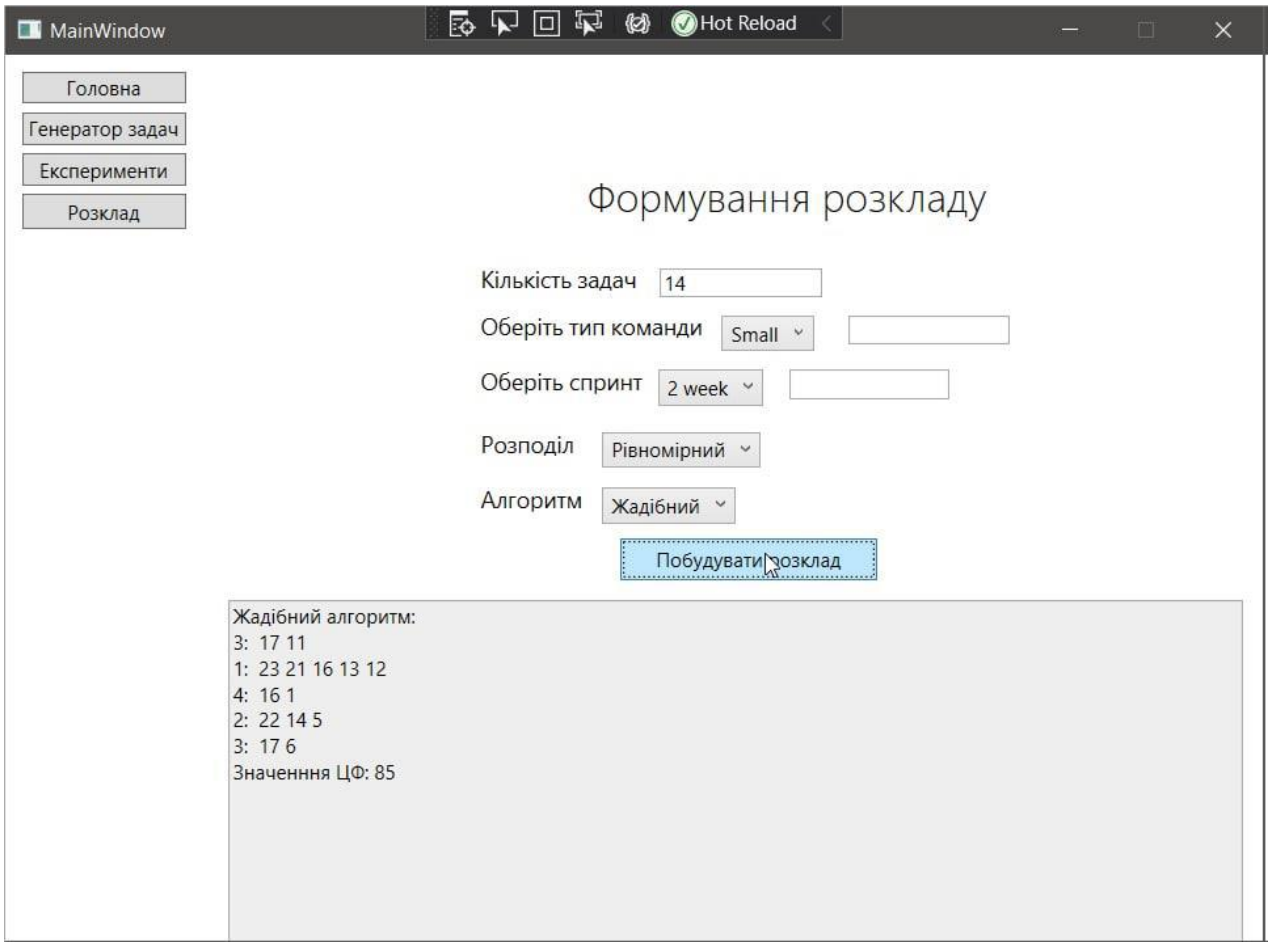

Рисунок Б.7 - Екранна форма для формування розкладу задачі (алгоритм локального пошуку)

Додаток В

Лістинг основного модуля

```
@problem_fact
class Vehicle:
    def __init__(self,
                 vehicle_num: int,
                 speed: int,
                 vehicle_capacity: int,
                 vehicle_depot_name: str,
                 shift_time: int,
                 vehicle_return_depot: str,
                 vehicle_shift_start: time):  # Change to time
        self.vehicle_num = vehicle_num
        self.speed = speed
        self.vehicle_capacity = vehicle_capacity
        self.vehicle_depot_name = vehicle_depot_name
        self.shift_time = shift_time
        self.vehicle_return_depot = vehicle_return_depot
        self.vehicle_shift_start = vehicle_shift_start
        self.vehicle_shift_end = (datetime.combine(datetime.today(), vehicle_shift_start) +
timedelta(hours=shift_time)).time()


# Crew class definition
@problem_fact
class Crew:
    def __init__(self, vehicles: List[Vehicle]):
        self.vehicles = vehicles

# WorkOrder class definition
@planning_entity
class WorkOrder:
    def __init__(self,
                 work_order_id: int,
                 task_id: int,
                 status: str,
                 created_date: datetime,
                 severity: str,
                 priority: str,
                 area_code: str,
                 sand_volume: int,
                 time_taken_for_cleaning: int,
                 crew: Optional[Crew] = None,
                 assigned_start_time: Optional[datetime] = None,
                 est_cmp_datetime: Optional[datetime] = None):

        self.work_order_id = work_order_id
        self.task_id = task_id
        self.status = status
        self.created_date = created_date
        self.severity = severity
        self.priority = priority
        self.area_code = area_code
        self.sand_volume = sand_volume
        self.time_taken_for_cleaning = time_taken_for_cleaning
        self.crew = crew
        self.assigned_start_time = assigned_start_time
        self.est_cmp_datetime = est_cmp_datetime
```

```
    # Planning variable: changes during planning, between score calculations.
    @optapy.planning_variable(Crew, ["crew"])
    def get_crew(self):
        return self.crew

    def set_crew(self, crew):
        self.crew = crew

    # Planning variable: changes during planning, between score calculations.
    @optapy.planning_variable(datetime, ["datetime"])
    def get_assigned_start_time(self):
        return self.assigned_start_time

    def set_assigned_start_time(self, assigned_start_time):
        self.assigned_start_time = assigned_start_time

@planning_solution
class Schedule:
    def __init__(self, work_orders: List[WorkOrder], vehicles: List[Vehicle], crews: List[Crew], score=None):
        self.work_orders = work_orders
        self.vehicles = vehicles
        self.crews = crews
        self.score = score

    @optapy.planning_entity_collection_property(WorkOrder)
    def get_work_orders(self):
        return self.work_orders

    @optapy.problem_fact_collection_property(Vehicle)
    def get_vehicles(self):
        return self.vehicles

    @optapy.problem_fact_collection_property(Crew)
    @optapy.value_range_provider('crew')
    def get_crews(self):
        return self.crews

    @optapy.planning_score(HardMediumSoftScore)
    def get_score(self):
        return self.score

    def set_score(self, score):
        self.score = score

@constraint_provider
def scheduling_constraints(constraint_factory: ConstraintFactory):
    return [
        # Hard constraints
        work_order_constraints(constraint_factory),
        vehicle_capacity(constraint_factory),
        shift_time(constraint_factory),
        lunch_break(constraint_factory),
        no_overlapping_jobs(constraint_factory),

        # Soft constraints
        maximize_sand_cleaning(constraint_factory),
        avoid_too_many_tasks_in_a_row(constraint_factory),
    ]

def vehicle_capacity(constraint_factory):
    return constraint_factory \
        .from_(WorkOrder) \
        .filter(lambda work_order: work_order.sand_volume > work_order.crew.vehicles[0].vehicle_capacity) \
```

```
        .penalize("Vehicle capacity", HardMediumSoftScore.ONE_HARD)


def shift_time(constraint_factory):
    return constraint_factory \
        .from_(WorkOrder) \
        .filter(lambda work_order:
                work_order.assigned_start_time.time() < work_order.crew.vehicles[0].vehicle_shift_start or
                work_order.assigned_start_time + timedelta(minutes=work_order.time_taken_for_cleaning) >
                datetime.combine(work_order.assigned_start_time.date(), work_order.crew.vehicles[0].vehicle_shift_end)
        ) \
        .penalize("Shift time", HardMediumSoftScore.ONE_HARD)



def lunch_break(constraint_factory):
    return constraint_factory \
        .from_(WorkOrder) \
        .filter(lambda work_order: 12 <= work_order.assigned_start_time.hour < 13) \
        .penalize("Lunch break", HardMediumSoftScore.ONE_HARD)



def no_overlapping_jobs(constraint_factory):
    return constraint_factory \
        .from_(WorkOrder) \
        .join(WorkOrder) \
        .filter(lambda wo1, wo2:
                wo1.crew == wo2.crew and
                wo1.assigned_start_time < wo2.assigned_start_time <
                wo1.assigned_start_time + timedelta(hours=wo1.time_taken_for_cleaning)
        ) \
        .penalize("No overlapping jobs", HardMediumSoftScore.ONE_HARD)

def maximize_sand_cleaning(constraint_factory):
    return constraint_factory \
        .from_(WorkOrder) \
        .reward("Maximize sand cleaning", HardMediumSoftScore.ONE_SOFT, lambda work_order:
work_order.sand_volume)


def avoid_too_many_tasks_in_a_row(constraint_factory):
    return constraint_factory \
        .from_(WorkOrder) \
        .join(WorkOrder) \
        .filter(lambda wo1, wo2:
                wo1.crew == wo2.crew and
                wo1.assigned_start_time + timedelta(hours=wo1.time_taken_for_cleaning) == wo2.assigned_start_time
        ) \
        .penalize("Avoid too many tasks in a row", HardMediumSoftScore.ONE_SOFT)


def work_order_constraints(constraint_factory):
    return [
        # A WorkOrder cannot start before its creation time.
        constraint_factory
            .from_(WorkOrder)
            .filter(lambda wo: wo.assigned_start_time < wo.created_date)
            .penalize("Cannot start before creation", HardMediumSoftScore.ONE_HARD),

        # A WorkOrder must finish before its estimated completion date.
        constraint_factory
            .from_(WorkOrder)
```

```
        .filter(lambda wo: wo.assigned_start_time + timedelta(minutes=wo.time_taken_for_cleaning) >
wo.est_cmp_datetime)
        .penalize("Must finish before estimated completion", HardMediumSoftScore.ONE_HARD),

    # A WorkOrder cannot be assigned to a crew if the crew's vehicle is not available.
    constraint_factory
        .from_(WorkOrder)
        .join(Crew,
            Joiners.equal(lambda wo: wo.crew),
            Joiners.lessThan(lambda wo: wo.assigned_start_time, lambda crew:
crew.vehicles[0].vehicle_shift_start),
            Joiners.greaterThan(lambda wo: wo.assigned_start_time +
timedelta(minutes=wo.time_taken_for_cleaning), lambda crew: crew.vehicles[0].vehicle_shift_end))
        .penalize("Vehicle not available", HardMediumSoftScore.ONE_HARD)
  ]


import pandas as pd

"""PREPARE DATA STAGE"""
vehicle_df = pd.read_csv('D:/CVRP_May/Scheduling/Data/crew_data.csv')
vehicle_df = vehicle_df.dropna()

work_order_df = pd.read_csv('D:/CVRP_May/Scheduling/Data/scheduling_data.csv')
work_order_df = work_order_df.dropna()

# Convert vehicle_shift_start from string to datetime.time
vehicle_df['vehicle_shift_start'] = pd.to_datetime(vehicle_df['vehicle_shift_start']).dt.time
vehicle_data = vehicle_df.to_dict('records')  # This will give a list of dictionaries

work_order_df['created_date'] = pd.to_datetime(work_order_df['created_date'])
work_order_df['est_cmp_datetime'] = pd.to_datetime(work_order_df['est_cmp_datetime'])
work_order_data = work_order_df.to_dict('records')  # This will give a list of dictionaries



# work_order_df.loc[0]['created_date'] < work_order_df.loc[0]['est_cmp_datetime']
# vehicle_df.loc[0]['vehicle_shift_start'] < vehicle_df.loc[1]['vehicle_shift_start']



vehicles = [Vehicle(**data) for data in vehicle_data]
crews = [Crew([vehicle]) for vehicle in vehicles]
work_orders = [WorkOrder(crew=None, assigned_start_time=None, **data) for data in work_order_data]


"""SOLVE STAGE"""

# Define the problem
problem = Schedule(work_orders, vehicles, crews)

# Configure and load the solver config
timeout = 10
solver_config = solver.SolverConfig()
solver_config.withSolutionClass(Schedule).withEntityClasses(WorkOrder).withConstraintProviderClass(
  scheduling_constraints).withTerminationSpentLimit(Duration.ofSeconds(timeout))


solver_factory = optapy.solver_factory_create(solver_config)
solution = solver_factory.buildSolver().solve(problem)
print("Score Explanation.")
print("Final Score: {}".format(solution.get_score()))
```

Додаток Г

(обов`язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Автоматизована система оптимізації розкладу руху маршрутного транспорту з урахуванням часу простою»
Типроботи:_____Магістерська кваліфікаційна робота_____
(БДР, МКР)

Підрозділ_____АІІТ,ФІІТА_____
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність _93,8_%_____ Схожість_6,2_%_____

Аналіз звіту подібності (відмітити потрібне)

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її
автора. Роботу направити на розгляд експертної комісії кафедри.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату
та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку_____ _____Роман МАСЛІЙ_____
(підпис) (ім`я, прізвище)

Ознайомлені з повним звітом подібності, який був згенерований системою
Unichek щодо роботи.
Автор роботи _____ _____Євгеній СИРЦОВ_____
(підпис) (ім`я, прізвище)

Керівник роботи _____ _____Роман МАСЛІЙ_____
(підпис) (ім`я, прізвище)