МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки Факультет

Центр післядипломної освіти

Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
рівень вищої освіти

Дослідження методів оптимізації при розробці програмного забезпечення для планування ресурсів IT проєктів

Виконав:
студент групи ІПЗздм-21-2
Ласіцина Ю.М.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Керівник доц. Назаров О.С.
(посада, прізвище)

Допускається до захисту
Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
« 23 » травня 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Ласіциної Юлії Миколаївни
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів оптимізації при розробці програмного забезпечення для планування ресурсів ІТ проєктів»
затверджена наказом університету від 03.04.2023 № 83 Стз

2. Термін подання студентом роботи до екзаменаційної комісії «08» травня 2023 р.

3. Вихідні дані до роботи *оптимізація ресурсів при розробці програмного забезпечення для планування ІТ проєктів*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз предметної галузі і постановка задачі, оптимізації ресурсів при розробці програмного забезпечення для планування ІТ проєктів.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	25.01.23 – 19.02.23	*Виконано*
2	Огляд існуючих методів та алгоритмів	20.02.23 – 10.03.23	*Виконано*
3	Алгоритми захисту ПЗ	11.03.23 – 25.03.23	*Виконано*
4	Підготовка пояснювальної записки	26.03.23 – 15.04.23	*Виконано*
5	Спецчастина	15.04.23 – 17.04.23	*Виконано*
6	Підготовка презентації та доповіді	18.04.23 – 07.05.23	*Виконано*
7	Попередній захист	16.05.23	*Виконано*
8	Нормоконтроль, рецензування	01.05.23 – 15.05.23	*Виконано*
9	Занесення диплома в електронний архів	22.05.23	*Виконано*
10	Допуск до захисту у зав. кафедри	22.05.23	*Виконано*

Дата видачі завдання 25 січня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Назаров О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 88 с., 24 рис., 25 табл., 46 джерело, 5 додатків..

SCRUM, БЕКЛОГ, МІНІМІЗАЦІЯ ЧАСУ ВИКОНАННЯ ЗАВДАНЬ, РЕСУРС, СПРИНТ, ТЕОРІЯ РОЗКЛАДІВ, УПРАВЛІННЯ ПРОЄКТАМИ.

Об’єктом дослідження є процес управління IT-проєктами.

Предметом дослідження є методи планування ресурсів IT-проектів.

Метою дослідження є підвищення ефективності реалізації IT-проекту всередині організації шляхом розробки інформаційної системи для планування ресурсів IT-проекту, яка базується на техніках, які скорочують час реалізації проекту.

Методи дослідження. Завдання вирішувалися з використанням наступних прийомів: системний аналіз (для побудови інформаційної системи); теорія планування, дослідження операцій, теорія складності (для створення методів вирішення питань розподілу ресурсів); комп'ютерне моделювання (в експериментальних дослідженнях ефективності методів розв'язування задач розподілу ресурсів).

Зараз автоматизація підприємств швидко зростає в багатьох галузях. Це пов'язано з тим, що клієнтам потрібні інформаційні системи для задоволення все більш конкретних потреб бізнесу. Одним із найбільш перспективних секторів людської діяльності сьогодні є IT-бізнес. Важливим елементом є виконання численних IT-проектів від клієнтів з усього світу. Адміністрування проектів має бути добре виконане, щоб вони були успішно завершені.

Наукова новизна одержаних результатів полягає у побудові підходу до планування ресурсів, який враховує обмеження, ризики та аналіз; алгоритми вирішення проблеми розробляються та вдосконалюються шляхом використання експериментальних досліджень.

SCRUM, BACKLOG, MINIMIZATION OF TASK PERFORMANCE TIME, RESOURCE, SPRINT, SCHEDULE THEORY, PROJECT MANAGEMENT.

The object of research is the IT project management process.

The subject of the research is the methods of resource planning for IT projects.

The purpose of the study is to increase the effectiveness of the implementation of the IT project within the organization by developing an information system for planning IT project resources, which is based on techniques that reduce the time of project implementation.

Research methods. Tasks were solved using the following methods: system analysis (for building an information system); planning theory, operations research, complexity theory (to create methods for solving resource allocation issues); computer modeling (in experimental studies of the effectiveness of methods of solving resource allocation problems).

Today, enterprise automation is growing rapidly in many industries. This is due to the fact that customers need information systems to meet increasingly specific business needs. One of the most promising sectors of human activity today is the IT business. An important element is the implementation of numerous IT projects for clients from all over the world. Project administration must be well-executed for successful completion.

The scientific novelty of the obtained results lies in the construction of an approach to resource planning that takes into account limitations, risks and analysis; problem-solving algorithms are developed and improved through the use of experimental studies.

Я, Ласіцина Юлія Миколаївна студент(ка) групи ІПЗздм-21-2 здобувач вищої освіти на другому (магістерському) рівні кафедра програмної інженерії, (повна назва кафедри) заявляю: моя кваліфікаційна робота на тему

Дослідження методів оптимізації при розробці програмного забезпечення для планування ресурсів ІТ проєктів, що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може

бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений (а) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Управління проектами [1] – це методологія організації, планування, управління та координації людських, фінансових і матеріально-технічних ресурсів протягом усього циклу проекту з метою ефективного досягнення його цілей шляхом використання сучасних методів, технік і технологій управління для досягнення визначених результатів. з точки зору обсягу, вартості, часу, якості та задоволеності учасників.

Мета, кінцеві результати, обсяг роботи та обсяг роботи вказуються для кожного проекту. Усі компоненти можуть змінюватися, коригуватися або вдосконалюватися протягом життя проекту, як під час розробки проекту, так і під час досягнення проміжних результатів.

Планування, організація, координація, активізація та контроль – це п’ять видів управлінської діяльності, що складають функцію управління [1].

Якість проекту, час, контракти та постачання, комунікація проекту, ризики, персонал і багато видів ресурсів, включаючи людські, фінансові та технологічні ресурси, є одними з функцій управління. Давайте розглянемо деякі з них.

Пошук і відбір кандидатів, найм і звільнення співробітників, планування і розподіл працівників на робочі місця, організація навчання і підвищення кваліфікації, встановлення обов’язків, створення робочого середовища, яке сприяє командній роботі, запобігання і вирішення конфліктів, які можуть виникнути, вирішення проблем з оплатою праці тощо в управлінні людськими ресурсами проекту.

Крім того, під час роботи слід враховувати тайм-менеджмент, зокрема: визначення діяльності та її тривалості; визначення дат початку та закінчення проекту; мінімізуючі (оптимізуючі) характеристики; прогнозування термінів завершення заходів, етапів і проекту в цілому; та прийняття рішень щодо усунення небажаних часових відхилень. Для реалізації тайм-менеджменту використовуються процедури аналізу часу проекту та його складових, планування робіт, контролю за розкладом, їх актуалізації та корекції. Під час реалізації проекту тимчасові ресурси порушуються

через погане планування людських ресурсів, що призводить до зриву термінів проекту та втрати грошових ресурсів проекту.

Створення та використання системи підтримки прийняття рішень, яка дозволяє виконувати складні операції планування та формує вплив на виконання проекту на основі аналізу поточного статусу проекту, є одним із методів покращення управління проектами розробки (програмного забезпечення).

Незважаючи на широкий спектр автоматизованих систем управління проектами, доступних на даний момент (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 або реалізовані не повністю.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ПО УПРАВЛІННЮ ІТ-ПРОЄКТАМИ

1.1 Підходи до управління ІТ-проєктами

Протягом історії людства було створено багато успішних ініціатив, у тому числі ІТ-проектів. Такі зухвалі плани потребували спільних зусиль великої кількості людей, від будівництва пірамід у Гізі до першої подорожі людини на Місяць.

Кожна ініціатива є унікальною та вносить щось унікальне у світ. Немає ідеального рішення для управління проектами, яке б однаково добре працювало на всіх типах проектів. Важливо розуміти, що не існує системи, яка працює для кожного керівника проекту. Проте в ході управління проектами було розроблено багато успішних підходів, методологій і моделей [2].

На даний момент доступно наступні основні методи управління проектами:

– Waterfall Model;
– Scrum;
– Kanban;
– Lean;
– PRINCE2;
– шість сігм (Six Sigma).

Техніки управління проектами використовуються вже деякий час. Велика китайська стіна та єгипетські піраміди – це актуалізовані проекти, у яких автори застосували різні методи управління проектами. На жаль, більше немає жодних письмових записів про конкретні методи, які використовуються для реалізації та управління цими проектами, і наше поточне розуміння управління проектами дуже відрізняється від розуміння попередніх століть.

Найкращий спосіб виконати проект – розділити його на менші, легші завдання або фази. Контрольний перелік завдань, які необхідно виконати для досягнення цілей, є одним із основних інструментів для цього.

Давайте розглянемо додаткові методи управління проектами більш детально.

Waterfall. Традиційний «Waterfall» підхід до управління проектами, при

якому дії завершуються послідовно через ітерації, щоб нагадувати потік [3].

Цей метод підходить для проектів із суворими обмеженнями на порядок певних дій і обмеженими ресурсами. Наприклад, при будівництві нового житлово-експлуатаційного комплексу не можна влаштовувати перекриття будинків, поки не буде закладено фундамент.

Існує п’ять основних етапів управління проектом з використанням Waterfall підходу відповідно до його поетапності. (хоча можливе доповнення додатковими фазами).

5 етапів Waterfall:

Етап 1. Ініціалізація. На цьому етапі встановлюються потреби проекту. Керівник проекту та команда відповідають за це. Під час ініціалізації часто проводяться так звані «мозкові штурми», під час яких бажаний результат розуміється швидше та простіше.

Етап 2. Планування. На етапі планування команда вирішує, що потрібно зробити для досягнення цілей. І цілі проекту, і його результати визначені та вичерпні. На основі отриманої інформації команда розробляє графік, визначає бюджет, оцінює ризики та розглядає можливі зміни завдання.

Етап 3. Розробка. Конфігурація майбутнього проекту та технологічні підходи до його виконання вибираються на етапі розробки, що є ключовою характеристикою технологічних проектів.

Етап 4. Реалізація та тестування. На цьому етапі основне завдання, розробка коду, завершено. Зміст проекту та критерії розробляються шляхом контролю за дотриманням попередньо встановленого графіку. Другу половину цього етапу займає тестування продукту, де продукт порівнюється та перевіряється на відповідність вимогам клієнта. Дефекти товару виявляються та усуваються після перевірки.

Етап 5. Моніторинг і завершення проєкту. Цей етап може відрізнятися залежно від типу проекту, оскільки він може включати надання клієнту результатів проекту або він може бути результатом тривалого процесу роботи з клієнтом для

покращення результатів проекту та підвищення ступеня відповідності критеріям клієнта.

Техніки календарно-мережевого планування ідеально підходять для реалізації проекту в рамках традиційного методу управління проектами, оскільки зрозуміло, що тривалість діяльності, визначена на етапі планування, міцно пов’язана з цим методом [4].

Переваги Waterfall:

– на початковому етапі проекту визначення потреб і вимог клієнта;

– підвищується стабільність проекту на етапі ініціалізації;

– ретельна підготовка дозволяє своєчасно завершити проектну діяльність;

– основні етапи включають важливий етап, який включає тестування та моніторинг готового продукту.

– навіть якщо їх оцінка відхилена, керівник проекту добре поінформований про ресурси проекту, і небезпека перевищення бюджету постійно береться до уваги.

Недоліки Waterfall:

Найбільшим недоліком будь-яких змін є те, що вони викликають серйозні збої та відхилення від розкладу, що ускладнює адаптацію до змін у реалізації проекту пізніше.

Agile. Гнучкий ітераційно-інкрементальний підхід до управління проектами, де основними цілями є динамічне формування вимог, можливість зміни цих вимог під час роботи та забезпечення відповідності цим вимогам у результаті постійної взаємодії членів самоорганізованої команди [5].

Не кожен проект можна спланувати до виконання за допомогою традиційного підходу до управління проектами. Цей метод ділить проект на менші підпроекти, які пізніше будуть об’єднані, щоб створити кінцевий продукт, а не розбивати його на послідовні етапи.

Переваги Agile:

– гнучкість і здатність до змін;

– гнучкість у всіх ситуаціях і процедурах;

– можливість реалізації проекту «шматками»;

– відстеження завдань із використанням фундаментальних принципів підходу.

Недоліки Agile полягає в тому, що цей метод має бути індивідуально адаптований до кожної команди розробників, що є складним і трудомістким процесом, який потребує окремих часових зобов'язань для реконфігурації.

1.2 Використання Scrum методології у рамках управління проєктоми

Scrum розділяє проект на менші підпроекти (компоненти більшого проекту), які майже відразу можна використовувати. Потім для розрізнення цих компонентів використовується пріоритет завдання (зазвичай власник продукту). Основні «компоненти» спочатку поділяються на спринти, або ітерації Scrum, які тривають від двох до чотирьох тижнів.

Робоча частина продукту – ті «шматочки» проекту, які вже можна використовувати в роботі – доставляється замовнику після завершення спринту. Загін починає наступний спринт після початкового. Згідно з роботами [6], тривалість спринту завжди вказується, але команда приймає рішення самостійно, враховуючи складність завдань і можливості команди.

Scrum складається з команд Scrum і ролей, подій, артефактів і правил, які з ними пов'язані. Кожен елемент каркаса має окрему функцію і є для нього вирішальним. Рекомендації Scrum керують зв'язками та взаємодією між ролями, подіями та артефактами, об'єднуючи їх.

Scrum є однією з найбільш широко використовуваних технік і процесів управління проектами сьогодні. Згідно з визначенням, Scrum – це структура розробки, яка дозволяє людям успішно та продуктивно вирішувати відомі проблеми, а також пропонувати високоякісні та цінні продукти (з точки зору замовника – прим. автора) [7].

Це пов’язано з тим, що практично неможливо знайти вирішення кожної проблеми та вказівки щодо того, як діяти в конкретних обставинах під час використання Scrum (наприклад, в офіційному описі Scrum згадується лише наявність оцінок часу, необхідного для виконання завдань, без згадки про тип оцінити так і має бути).

Найчастіше під час обговорення методів і методології Scrum згадується гнучкий процес розробки програмного забезпечення, побудований на правилах і практиках Scrum.

Творці Scrum стверджують наступні характеристики підходу:

– легкий у використанні (Lightweight);

– зрозумілий та доступний;

– важко вивчити (майже взаємовиключні параграфи).

У класичному підхді Scrum є 3 базових ролі:

– власник продукту (Product owner);

– скрам майстер (Scrum master);

– команда розробників (Development team).

Головним «перемикачем» між командою розробників і стороною клієнта є власник продукту. Метою ВП є використання всіх підходів у межах параметрів проектної роботи, щоб максимізувати цінність продукту, що розробляється, і команди розробників.

Беклог продукту (Product Backlog) є прикладом, який допомагає ВП у його зусиллях завершити проект і задовольнити потреби клієнта. Будь-які робочі завдання (такі як User Story, Bug, Task тощо), які потрібно завершити, перераховуються в журналі продукту з пріоритетом, призначеним для виконання кожного завдання.

Scrum-майстер, який контролює використання командою розробників методології, є її капітаном. Робота scrum-майстра полягає в тому, щоб допомогти команді розробників стати більш продуктивною за допомогою правильного

налаштування робочого процесу, залучення команди, навчання та стимулювання для виховання почуття єдності.

Команда розробників складається з багатьох спеціалістів, які працюють над наступним продуктом.

Загалом, команда scrum – це будь-яка група осіб, які використовують Scrum для співпраці, включаючи власника продукту, майстра scrum і команду розробників.

Команда розробників повинна володіти такими атрибутами та рисами, згідно з Посібником Scrum (який є остаточним поясненням усіх елементів Scrum його авторами):

– самоорганізованість.Жодна scrum-команда не має повноважень керувати командою розробників щодо того, як перевести невиконаний продукт у функціональний продукт, якого вимагає клієнт;

– багатофункціональність, мультизадачність. Фактичне володіння здібностями, необхідними для співпраці, виробництва та реалізації придатного результату;

– відповідальність. Навіть якщо сталася помилка, за виконане завдання відповідає вся команда scrum в цілому, а не окремий член команди. Команда scrum повинна складатися з 7-8 (плюс-мінус 2) учасників. З іншого боку, менші команди збільшують ризик провалу проекту (через недостатні необхідні навички та кваліфікацію) і зменшують обсяг роботи, яку команда в цілому може виконати за одиницю проектного часу, відповідно до класичних функцій Scrum [8]. З іншого боку, більші команди потребують занадто багато ресурсів для спілкування.

Процес Scrum

Фундаментальною складовою Scrum є розподіл роботи на спринти, протягом яких опрацьовуються певні завдання, щоб частково реалізувати продукт. По завершенню спринту повинна бути презентація роботи (нова робоча версія продукту або частини функціоналу). Спринт завжди обмежений у часі; зазвичай він триває 1-4 тижні і залишається таким до завершення проекту.

Кожен спринт починається зі спринту планування спринту (Sprint Planning), під час якого оцінюється вміст резерву продукту, обирається (призначається) беклог спринту (Sprint Backlog), який містить різні типи завдань, які потрібно виконати в поточному спринті, а результатом їх виконання є показано. Кожен спринт має чітку мету або цілі, які можна досягти, виконавши залишені дії спринту, і вони повинні служити мотивацією.

Процес Scrum проілюстровано на рисунку 1.1.

Рисунок 1.1 – Схематичне зображення процесу Scrum

У традиційній методології Scrum щодня проводиться щоденна зустріч команди, або «Daily Scrum», під час якої всі члени команди відповідають на три прості запитання: «Що було зроблено вчора?», «Що я буду робити протягом дня?» Які у мене проблеми, пов'язані з завданням, і чи може щось заважати моїй роботі? Ці сесії служать для оцінки поточної ситуації в команді, відстеження виконання роботи спринту, виявлення будь-яких проблем або питань, що виникли, і розробки рішень для модифікації окремих робочих моментів, які можуть знадобитися для досягнення цілей спринту. Підготуйтеся до огляду спринту (Sprint Review) та ретроспективи спринту (Sprint Retrospective) після завершення спринту. Ці зустрічі в першу чергу призначені для оцінки ефективності (або продуктивності) команди

scrum під час попереднього спринту, прогнозів щодо очікуваної ефективності команди під час наступного спринту, виявлення будь-яких незрозумілих завдань або проблем, визначення ймовірності того, що проекту виділено робота буде завершена тощо.

Часто кажуть, що Scrum не працює або що він працює набагато гірше, ніж очікувалося.

Згідно з численними публікаціями на цю тему, основним джерелом надійних знань для належного застосування та функціонування Scrum є досвід. Потреба в ретельності та точності в застосуванні Scrum була окреслена в Посібнику Scrum і є результатом незвичайної організації процесу традиційних структур, де немає формальних спеціалістів, таких як менеджери чи керівники. Самоорганізовані команди та багатофункціональні команди є одними з основних принципів Scrum. Кількість ініціатив співробітників, здатних виконувати багато завдань і самоорганізовуватися, згідно з дослідженням соціальних наук. Лише 15% зайнятого населення представлено цією кількістю [9].

Це демонструє, що лише невеликий відсоток спеціалістів може добре працювати в Scrum-командах без суттєвих коригувань обов’язків Scrum-майстра та власника продукту, що насправді суперечить фундаментальному принципу Scrum і може призвести до модифікованого застосування Scrum, тобто.

Scrum приймає зміни вимог у міру їх виникнення, оскільки він є членом сімейства Agile (залишок продукту можна змінити в будь-який час). Це ускладнює використання Scrum у проектах із встановленими лімітами витрат (або термінами виконання). Оскільки важко заздалегідь передбачити та врахувати кожну зміну, центральний принцип Scrum полягає в тому, що планування має здійснюватися лише в відповідний час, тобто брати до уваги лише ті завдання, які потрібно виконати протягом поточного спринту [10] . Звичайно, існують додаткові обмеження.

Переваги і недоліки. Scrum пропонує значну кількість переваг. Він орієнтований на клієнта, гнучкий і адаптований, що дозволяє клієнту коригувати вимоги проекту, коли вони вважають за потрібне (але немає гарантії, що ці нові

зміни будуть реалізовані в проекті). Багато споживачів вважають привабливою гнучкість для зміни умов і потреб завдань.

Завдяки своєму дослідженню та здатності економити час, уникаючи некритичних процесів, Scrum досить легко «адаптувати». Наприкінці кожного спринту Scrum дозволяє надати систему чи продукт, які можуть працювати.

Відповідно до ідеології scrum, команда, яка є самоорганізованою, багатофункціональною та підзвітною, може виконувати необхідні завдання, скорочуючи будь-яку кількість операцій. Це чудово підходить як для малого бізнесу, так і для стартапів, оскільки повністю усуває потребу в спеціалізованій підготовці менеджерів і персоналу.

Безсумнівно, у Scrum є певні недоліки. Scrum накладає певну кількість жорстких, неприємно строгих обмежень через простоту використання та мінімалістичний дизайн. Клієнта не хвилюють внутрішні правила та процедури команди scrum, особливо якщо вони певним чином обмежують клієнта, тому це стає менш очевидним, якщо всі в команді знають і приймають концепцію «орієнтованості на клієнта». Наприклад, навіть незважаючи на те, що він повністю суперечить стандартам scrum, спринт відставання може бути скоригований майже повністю за бажанням клієнта, якщо це необхідно.

Оскільки Scrum є членом сімейства Agile, проблема значно гірша, ніж має бути, тому що в Scrum неприйнятно, наприклад, організовувати внутрішні комунікації або використовувати конкретні сценарії, коли присутні певні ризики [11]. У результаті виникла офіційна опозиція, яка порушила всі закони Scrum.

Постійний акцент Scrum на самоорганізованій, багатофункціональній команді є ще одним недоліком. Незважаючи на те, що це значно знижує вартість командної роботи, це також ставить більше вимог до вибору та мотивації працівників. Залежно від стану ринку праці створити повну та ефективну команду Scrum може бути майже неможливо.

Ми порівняємо управління ресурсами з фінансовим менеджментом, щоб краще його зрозуміти, і порівняння буде показано в таблиці 1.1.

Таблиця 1.1 – Порівняльне дослідження управління ресурсами та управління фінансами

Фінанси	Ресурси
Коли на проект закінчуються гроші (наприклад, на виплату зарплати – є фінансовий дефіцит), ви можете взяти кредит / займ.	Потрібен певний ресурс на короткий проміжок часу (наприклад, щоб виконати термінову роботу, не порушуючи основну діяльність) – погодьтеся «поширити» ресурс на сусідній проект, де він недостатньо використовується.
Якщо є додаткові гроші, покладіть їх у банк або використайте для фінансування проекту, який буде успішним.	Затягування клієнтом вимог «блокує» розробку; ми позичаємо людей для проекту там, де вони найбільше потрібні. Ваш бюджет проекту зберігається, оскільки витрати за цей час переносяться на суміжний проект.
На ринку з'явилися дешеві гроші – перекредитуємо	Ми забезпечуємо заміну після того, як сусідній проект звільняє своїх розробників за нижчою ціною, ніж аутсорсинговий ресурс, використаний для його власного проекту.

Було виявлено, що зазвичай завжди існує порівняння між фінансовим менеджментом та управлінням ресурсами, і навпаки. Проте різниця між ресурсами та грошима, яка характеризує особливості цього регіону, досить значна.

1.3 Задача планування ресурсів із застосуванням методології Scrum

Конкретний IT-проект отримав зелене світло. Цей проект пройшов дослідження (аналіз), знайдено інструменти та передумови (умови) для його виконання. Гнучкий підхід, який є членом сімейства Agile, методологія Scrum слугуватиме основою для всього процесу управління реалізацією проекту.

За результатами дослідження умов проекту визначено перелік завдань (завдань), які необхідно виконати в ході проекту; отже, створюється резерв продукту. Виконавець бенчмарку визначає тривалість і пріоритетність виконання кожного завдання (під пріоритетом ми маємо на увазі важливість виконання завдання) (під виконавцем бенчмарку ми маємо на увазі виконавця, який працює з коефіцієнтом продуктивності 1).

Процес пріоритезації завдань призводить до відставання спринту, яке показано на рисунку 1.2 у порядку завдань з високим, середнім і низьким пріоритетом.

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 – Виділення беклогу продукту з резерву спринту на схематичному прикладі

Призначимо наступні рівні пріоритету: високий, середній і низький (високопріоритетні завдання реалізуються першими, відповідно низькопріоритетні завдання реалізуються в останню чергу, можуть не бути реалізовані взагалі або можуть бути відкладені до впровадження в наступних версіях продукту).

Для поступового впровадження проекту Scrum використовує ітерації фіксованої довжини, відомі як спринти, які зазвичай охоплюють 2-4 тижні. Беклог спринту створюється з резерву продукту та містить певну кількість перевірених елементів, щоб розпочати реалізацію спринту.

Необхідно контролювати кількість завдань, щоб не перевищувати продуктивність команди. Це завдання підмножини додається до резерву виконання першого спринту.

Даний ІТ-проект реалізує команда виконавців, кожен з яких має унікальний коефіцієнт ефективності за посадою. У наказі вказується склад команди, місця та кількість учасників. Ми поділяємо виконавців на три категорії: молодші, середні та старші. Виконавець середньої позиції є еталонним виконавцем, оскільки він має коефіцієнт продуктивності 1. Коефіцієнти продуктивності для молодших і старших виконавців різні.

У такій постановці питання ми враховуємо сценарій, за якого спеціалісти змінні та здатні працювати паралельно. Залежно від ролі, яку вони займають, коефіцієнти продуктивності виконавців змінюються.

Відставання продукту змінюється, пріоритети завдань можуть змінюватися, самі завдання можуть змінюватися, а завдання можуть бути розділені на менші, коли створюється відставання для наступного спринту.

Таким чином, до кінцевого терміну політики, який є часом, до якого вся робота зі спринтом на рисунку 1.3 має бути завершена, реалізовано відставання спринту.

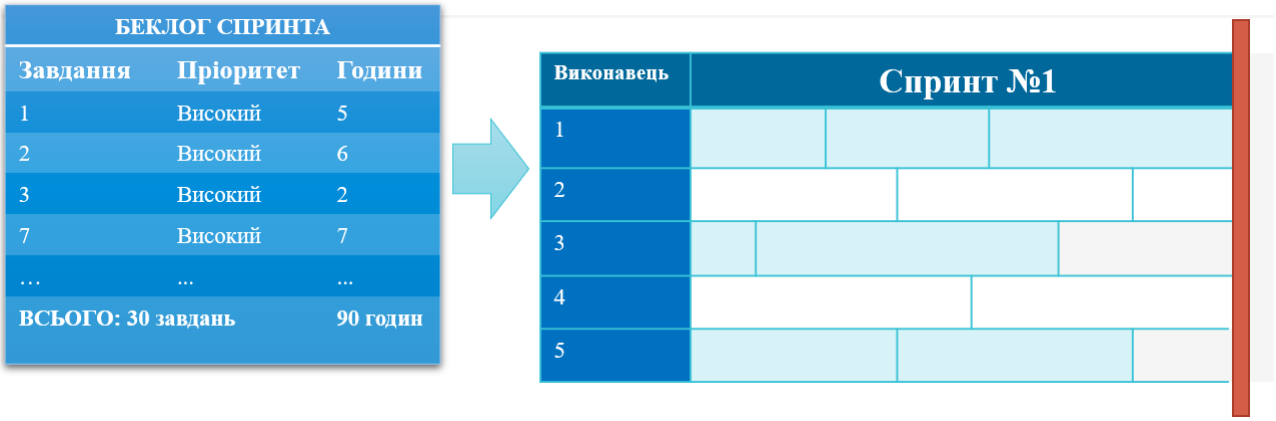

Рисунок 1.3 – Підмножина завдань (беклог спринта) для розподілення між учасниками команди

Теорія розкладу може допомогти визначити характеристики та особливості цієї проблеми, оскільки метою проблеми є скорочення часу виконання проекту та ефективне планування ресурсів (у цій задачі були обрані час і людські ресурси). Ця NP-повна задача є членом класу задач теорії планування.

Класифікація викликів і чітке визначення їх складності були однією з головних проблем нового підходу до проблем планування. Грем запропонував загальноприйняту таксономію для питань TR. Огляди проблем TR та їх складності були запропоновані в працях Гері та Джонсона, Ланстри, Лоулера, Танаєва, Брукер та інших [12] і були сповнені конструктивізму та впевненості.

NP-складність характеризує переважну більшість труднощів у дослідженнях теорії планування. Незважаючи на це, вирішення цих питань є необхідним на практиці. Існують різні методи для цього. Реалізація поліноміальних евристичних алгоритмів стоїть на першому місці. Оцінки похибок результатів, досягнутих для кількох евристичних методів і алгоритмів, уже відомі. Їх називають наближеннями. Також існують методи апроксимації, які забезпечують як відносну похибку, так і абсолютну неточність результатів [13]. Кілька питань NP-комплексу демонструють існування певного методу апроксимації. За допомогою цього методу, який є повністю поліноміальним методом апроксимації, можна знайти наближений розв’язок задачі з відносною похибкою в межах будь-якого заданого значення $\varepsilon >$

0 за час, за час, який поліноміально залежить від $1 / \varepsilon$ та розміру вхідних даних дані для проблеми (FPTAS).

Знаходження так званого граничного значення – для якого прийнятно ідентифікувати оцінене рішення за поліноміальний час – схема поліноміальної апроксимації має вирішальне значення для задач без схеми апроксимації (PTAS).

Евристичні алгоритми в даний час зазвичай використовуються для визначення оптимальної відповіді, і це рішення є максимально близьким до оптимального рішення за прийнятний проміжок часу.

Однак недоліком цих та споріднених алгоритмів є те, що вони не забезпечують оптимальність кінцевого продукту.

У найгіршій ситуації неможливо оцінити, наскільки відповідь відхиляється від ідеального рішення.

Техніці та підходам розв’язання задач NP-комплексу приділено багато уваги в роботах з TR. Найпоширенішими техніками зараз є методи гілок і границь або методи стисненого перебору [14]. Оцінки нижньої межі цільової функції обчислюються (під час її мінімізації), а комбінаторні якості проблем використовуються для зменшення пошуку. Часто задачу TP розв’язують методом динамічного програмування [15].

Часто буває, що питання теорії планування можна виразити як питання цілочисельного лінійного програмування. Вирішенню таких питань присвячені численні дослідження та роботи, наприклад [16].

Підхід до програмування з обмеженнями в даний час є широко використовуваною технікою.

Теорія декомпозиції є ще однією областю, де вона може бути ефективно використана [17].

Алгоритми, які поєднують аспекти різних підходів, можуть впоратися з деякими складними проблемами TR. «Гібридні алгоритми» – це їхня назва[18]. Це, на наш погляд, один із найперспективніших напрямків.

1.4 Огляд існуючих рішень задачі планування ресурсів

До цього моменту кілька вчених замислювалися над проблемою побудови інформаційної системи планування ресурсів IT-проектів на основі проблеми теорії розкладів.

Ключові переваги та застосування інформаційних технологій обговорюються авторами в [19]. Для аналізу діяльності та компонентів системи управління проектами використовувалася автоматизована система. Розглянуто ІС, які реалізують сервіси календарного планування та контролю якості реалізації проекту, наголошено на їх перевагах та практичному застосуванні. Досконально досліджено ефективність таких методів управління проектами, які передбачають планування пакетів робіт і безпосередній контроль за їх виконанням. Досить детально розглянуто методологію «Total Cost Of Oversight», яка дає можливість побудувати оцінку економічної ефективності проектів.

Стаття [20] описує фундаментальні ідеї термінів «проект», «план» і «програма» та пояснює, чим вони відрізняються одна від одної. Визначено методи, які використовуються для здійснення управління проектами (управління проектами). Основні переваги впровадження інформаційних систем управління проектами (ІСУП) були виявлені після дослідження ролі та значення інформаційних технологій в управлінні проектами. Проводиться оцінка ефективності використання систем управління проектами в Америці з виділенням відомих інформаційних систем управління проектами.

У роботі [21] розглядається багатовимірне представлення даних для управління IT-проектами. Стаття підтверджує теоретичні положення та пропонує методологічні та корисні поради щодо підвищення ефективності функціонування інформаційної системи. Представляючи результати дослідження фундаментальних концепцій і методик управління проектами інформаційних процесів, підтримується методологія управління проектами корпоративної інформаційної системи. Враховується процес розробки IT-проекту, який дозволяє розглядати широкі

стратегічні цілі розробки, інтеграції та проектування операційної моделі системи підтримки прийняття рішень.

Створення моделі розподілу ресурсів для ІТ-проектів є центром роботи [22]. Проведено дослідження сучасних економіко-математичних моделей та можливості їх використання в системі управління персоналом ІТ-підприємства. Найкращий розподіл ІТ-проектів між ІТ-командами досягається за допомогою багатокритеріальної економіко-математичної моделі динамічної оптимізації. Запропоновано спосіб використання побудованої економіко-математичної моделі для отримання найкращої відповіді.

У роботі автор [23] розкрив принципи використання інформаційних технологій в управлінні проектами підприємства. Автор врахував особливості використання інформаційних технологій в управлінні бізнес-проектами. Було доведено, що використання інформаційних технологій є важливим, і були розглянуті методи їх інтеграції в управління проектами.

Використання інформаційних технологій в управлінні проектами було запропоновано в [24]. Автори підкреслюють, що широке впровадження, значні фінансові резерви, відносно обмежена спеціалізація та конвергенція перевірених і вірних методів пояснюють потенційне застосування інформаційних технологій для управління ресурсами.

Управління ресурсами таких проектів має базуватися на поєднанні процедурних і ресурсних методів, і воно відрізняється від існуючих підходів тим, що наголошується на застосуванні набору математичних моделей і методів, спрямованих на покращення способів використання ІТ-проектів під час прийняття враховувати ключові змінні, що впливають на продуктивність проекту, планування ресурсів і якість послуг.

Нижче наведено багато документів (статей), щоб висвітлити ключові проблеми цих методів або складність вирішення проблеми, оскільки вона стосується методу точного вирішення проблеми планування.

Розроблено нові фундаментальні концепції управління та контролю за станом ІТ-проекту. Існує чимала кількість науково-популярних книг, пов’язаних з

IT-бізнесом, проведено багато тренінгів [25]. Щоб підняти економіку країни та загальний рівень життя, ця ситуація була реалізована в IT-секторі.

Резюме методів планування проекту з обмеженими ресурсами для різних цілей планування було надано в [26]. Було продемонстровано, що вибір прийнятної цілі планування є одним із найважливіших виборів, який необхідно зробити на етапі планування проекту, оскільки він впливає на те, наскільки ефективно використовуються ресурси, а також на вигляд і форму плану проекту. Було застосовано декілька швидких і простих евристик через природну складність планування проектів з відновлюваними ресурсами, які мають обмежену доступність, але точних результатів роботи не було опубліковано.

Можливості вирівнювання ресурсів трьох пакетів програмного забезпечення для управління проектами порівнювалися з використанням двох проблем реального світу в [27]. Отримані дані показують, що програмне забезпечення або метод, що використовується, впливають на тривалість проекту. Це підтверджується тим фактом, що цей метод забезпечує мінімізацію панорамування для проблем такого розміру, складності та цільової функції. Спостереження показало, що Primavera P6 працює краще, ніж MS Project і Open Workbench, і що виробничі розробки зросли з 41,11% до 167,79% необмеженого ресурсу розкладу. Керівники проектів повинні спробувати різні правила або навіть програмне забезпечення, якщо це можливо, а не «довіряти» першим результатам, які вони отримують через величезну різноманітність результатів.

У роботі [28] використовується паралельний підхід для перетворення цієї частинки в перспективний розклад відповідно до ресурсної переваги та ресурсних обмежень, щоб оцінити кожне потенційне рішення, тобто запропонований пріоритет завдання. Завдяки таким властивостям, як механізм одностороннього обміну досвідом у вирішенні проблем, обчислювальний аналіз демонструє, що ця стратегія ефективніша за евристичний підхід і має здатність розкривати глобальні оптимуми (тобто оптимальні графіки з найкоротшою тривалістю проекту). Цей метод пропонує практичну та просту альтернативу для проведення аналізу та отримання успішних результатів.

Висновки до розділу

Було розглянуто кілька методологій управління проектами, надано конкретні моделі, а їх переваги та недоліки висвітлено в першому розділі цієї дипломної роботи. Метод Scrum для управління IT-проектами був ретельно вивчений. Було описано весь обсяг бізнес-операцій, залучених до управління проектом.

Планування ресурсів в рамках управління проектами з використанням методології Scrum має детальну постановку проблеми. Описано основні етапи планування ресурсів. Проведено ретельну оцінку існуючих підходів до відповідного питання планування. Надано опис кожного існуючого рішення разом з інформацією про основну мету рішення та результати, які автор зміг отримати за допомогою аналізу конкретної проблеми.

2 РОЗВ'ЯЗАННЯ ЗАДАЧІ ПЛАНУВАННЯ РЕСУРСІВ ІТ-ПРОЄКТІВ

2.1 Математична постановка задачі

Дано множину завдань проєкту $J = \{1, \ldots, j, \ldots, n\}$ та множину виконавців (учасників команди) $M = \{1, \ldots, i, \ldots, m\}$.

Усі завдання в системі надходять у нульовий момент часу одночасно та ділять директивний період d .

Кожна процедура обслуговування завдань триває безперервно, доки не буде завершена.

Припускаючи, що кожен член команди має коефіцієнт ефективності виконання завдання $k_i > 0$ і що кожне завдання має пріоритет $p_j > 0$ і початкову тривалість виконання $t\,j > 0\ j = \overline{1,n}$ (початкова – тобто, еталонна тривалість, за яку завдання може бути виконане еталонним виконавцем з коефіцієнтом ефективності $k_i = 1$), то тривалість виконання *j*-ого завдання *i*-м членом команди, з огляду на результативну складову учасника вірно таке:

$$t_{ij} = t_j * k_i \tag{2.1}$$

Нехай Cj представляє точку, в якій *j*-те завдання завершено, $C_{max} = \max_j\{C_j\}$ позначає максимальний момент виконання завдання (момент виконання останнього завдання), $Z_j = \max_j\{0, d - C_j\}$ – затримку завдання; $E_j = \max_j\{0, d - C_j\}$ – час виконання завдання, C_i – момент виконання завдання для i -го виконавця, $Z = \sum_{j-1}^{n} Z_j$ – сумарне випередження, і $E = \sum_{j-1}^{n} E_j$ -загальна затримка.

Критерій 1. Найкраще створити розклад завдань, який мінімізує максимальний час, необхідний для виконання кожного завдання:

$$C_{max} \to min \tag{2.2}$$

Критерій 2. Необхідно скласти план завдань, який рівномірно розподіляє навантаження на виконавців з урахуванням рівня їх продуктивності:

$$\max_j\{E_j, Z_j\} \to min \tag{2.3}$$

2.2 Дослідження властивостей задачі

Передбачається, що процес виконання кожного завдання всередині даного завдання триватиме безперервно, доки не будуть виконані всі завдання.

Потім з'ясовується, чи може кілька виконавців виконувати завдання одночасно. Якщо в цій ситуації це неможливо, завдання обмежується наданням n завдань m виконавцям, щоб вони не збігалися.

Тільки один виконавець може отримати завдання в певний момент, і воно може виконуватися лише до моменту завершення.

Загальна продуктивність завдань досі невідома для систем, де кожна робота виконується одним виконавцем, а тривалість завдань змінюється між виконавцями у випадковому порядку. Наявний підхід, однак, мінімізує середній час виконання завдання, якщо можна відсортувати виконавців за швидкістю виконання завдання (для даного завдання – за коефіцієнтом продуктивності).

Тоді кожен виконавець має коефіцієнт виконання завдання $k_i > 0$, пріоритет завдання $p_j > 0$ і початкову тривалість виконання $t_j > 0\ j = \overline{1,n}$ для кожної роботи. Орієнтовна тривалість, протягом якої робота може бути завершена контрольним виконавцем, який, у свою чергу, має коефіцієнт ефективності одиниці $(k_j = 1)$., є початковою тривалістю.

Нижче ми розглянемо деякі додаткові позначення більш детально, оскільки вони були зроблені:

– C_j – момент закінчення j -ого завдання;

– $C_{max} = \max_j\{C_j\}$ – момент закінчення останнього завдання (максимальний момент виконаних завдань);

– C_i – момент закінчення виконання завдань i -им виконавцем.

Вважається, що кожне завдання має директивне слово (або термін планування), що позначається літерою d. Значення терміну директиви вказує, коли завдання має бути завершено та не пізніше зазначеного терміну.

Пошук нижньої межі загальної довжини завдання (нижньої межі вищезгаданих критеріїв оптимальності) стався на початку дослідження цієї проблеми.

$$C^* = \frac{T}{K} \tag{2.4}$$

де $T = \sum_{j-1}^{n} t_j$ – об'єм завдань у системі, $K = \sum_{i-1}^{m} k_i$ – загальний коефіцієнт продуктивності.

З цього випливає, що розклад, за яким кожен член команди виконує свою роботу в час C^* відповідно до вищезазначених стандартів, є ідеальним за двома факторами.

2.3 Алгоритми розв'язання задачі

Існує безліч можливих стратегій вирішення, кожна зі своїми перевагами та недоліками, для розв'язання задачі теорії планування класу NP із кількома критеріями оптимальності. Однак має сенс використовувати евристичні методи розв'язування задач теорії розкладу. Хоча пошук найкращого рішення не гарантований, ці алгоритми значно скорочують час, необхідний для вирішення цих проблем, гарантуючи швидкий результат.

Таким чином, для вирішення вищезгаданої проблеми вибрано жадібний алгоритм [29] і алгоритм локального пошуку [30].

Жадібний алгоритм простий для розуміння та використання, працює швидко та може використовуватися для вирішення широкого спектру завдань різної складності. Завдання виконуються за алгоритмом у довільному порядку. Він прагне дати кожне завдання першому виконавцю, який відповідає кваліфікації на основі

продуктивності цього виконавця.

Нижче наведено схему, яка показує, як працює жадібний алгоритм.

Крок 1. За тривалістю завдання відсортуйте завдання $t_1 > t_2 > \cdots > t_j > \cdots > t_n$ в масиві в порядку спадання.

Крок 2: Виберіть виконавця з часом виконання завдання C i=C min і дайте йому j-те завдання, яке має найбільшу тривалість t 1.

Крок 3: Повторення Кроку 2, поки m виконавців не отримають n завдань із набору $J = \{1, \dots, j, \dots n\}$.

Крок 4: Отримайте розклад G.

Натомість одним із найкращих евристичних алгоритмів для вирішення проблем, точні рішення яких займають експоненціально довгі періоди часу, є алгоритм локального пошуку.

Алгоритм локального пошуку – це мета-евристичний алгоритм, зосереджений на роботі з локальним пошуком у певному районі, щоб уникнути його потрапляння в тупики в локальних оптимумах, запобігаючи тим рухам, які призводять до повернення до раніше встановлених рішень, і повторення циклічної операції алгоритму. Остаточна відповідь полягає в тому, звідки береться алгоритм локального пошуку. На кожній ітерації формується конкретне ребро рішення, і найкраще рішення вздовж цього ребра стає новим рішенням. Щоб уникнути включення будь-яких заборонених характеристик, вибирається найкращий варіант у безпосередній близькості. Якщо нове рішення є кращим і підпадає під дозволені рішення, найкраще існуюче рішення оновлюється. Поки не буде виконано один із двох критеріїв для завершення алгоритму – максимальна кількість виконаних ітерацій або максимальна кількість ітерацій, дозволених, коли поточне рішення не змінилося, – процедура триватиме.

Локальний пошук – це тип пошуку, який виконується за допомогою алгоритмів локального пошуку, у якому попередні пошуки не враховуються та не запам’ятовуються, а пошук повністю базується на поточному стані рішення. Проблеми, які обробляються такими або подібними методами, називають проблемами оптимізації, оскільки основною метою пошуку є визначення

найкращого шляху до цільової точки, а також оптимізація цільової функції проблеми.

Цей один із найефективніших евристичних методів для вирішення проблем, точні відповіді на які потребують надзвичайно багато часу.

Першим кроком у створенні плану виконання завдання є використання математичної процедури для створення початкового розкладу.

Схема алгоритму локального пошуку:

Крок 1. Відсортуйте завдання в масиві $t_1 > t_2 > \cdots > t_j > \cdots > t_n$ у порядку зменшення часу виконання.

Крок 2. Призначити j-завдання з найбільшою довжиною t_1 і-му виконавцю $i = \overline{1,m}$, час виконання завдання якого $C_i = C_{min}$.

Крок 3. Повторити Крок 2, поки m виконавців не буде розподілено n завдань із набору $J = \{1, \dots, j, \dots n\}$.

Крок 4. Отримано розклад G.

Крок 5. Визначити значення $J = \{1, \dots, j, \dots n\}$ і для кожного виконавця за розкладом G. Серед них вибрати C_{max} та C_{min} .

Крок 6. Коли G' закінчить завдання, повторіть Кроки 1-3 для обраних виконавців. Використання C max і C min для побудови розкладу.

Крок 7. Розклад ітерації $1 - G_1$ створюється шляхом додавання отриманого розкладу G' до загального початкового розкладу G на місці виконавців зі значеннями C_{max} та C_{min}

Крок 8. Повторіть кроки 5-7, щоб деконструювати G_1 і отримати розклад G_2.

Крок 9. Повторити Крок 8, коли найкращий розклад G_q з усіх попередніх ітерацій $q, q = 1, 2, \dots, l$ і наступних ітерацій $l, l = 1, \dots, f$ із відповідними розкладами G_l забезпечує гіршу (або однакову) результат як ітерація G_q, пошук припиняється.

На початку алгоритму дано значення l.

Ми передбачаємо, що алгоритм локального пошуку, на відміну від жадібного алгоритму, дасть більш точну відповідь, яка більш узгоджується із встановленими критеріями після вивчення схем і атрибутів обох алгоритмів.

2.4 Приклад розв’язання задачі

Давайте використаємо жадібний метод для вирішення цієї проблеми (критерій 1) і покажемо, що відбувається на кожному етапі цього шляху.

Вихідні дані наведені в таблиці 2.1.

Таблиця 2.1 – Вхідні дані

Назва	Значення
Множина значень тривалостізавдань	1, 1, 1, 2, 2, 2, 3, 4, 5, 5, 6, 6, 6, 7, 8, 9,10
Множина коефіцієнтів продуктивності	1, 1, 2, 3

Набір завдань повинен бути організований відповідно до робочого часу, а коефіцієнти продуктивності виконавців можуть залишатися незмінними. Визначимо початкову точку виконання роботи кожного виконавця в таблиці 2.2.

Таблиці 2.2 – Моменти завершення виконання завдань по кожному виконавцю

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

Етап 1. Обчисліть час, необхідний для виконання Завдання 1, яке має тривалість 10, для кожного з виконавців, наведених у таблиці 2.3.

Таблиця 2.3 – Розрахунок часу, який знадобився кожному виконавцю для виконання завдання 1.

Виконавець (№)	Момент завершення виконаннязавдання №1
1	10
2	10
3	20
4	30

Знаходимо виконавця, чий час на виконання завдання №1 є найкоротшим. Тобто, як показано на рисунку 2.1, завдання може бути призначено Виконавцю 1 або Виконавцю 2.

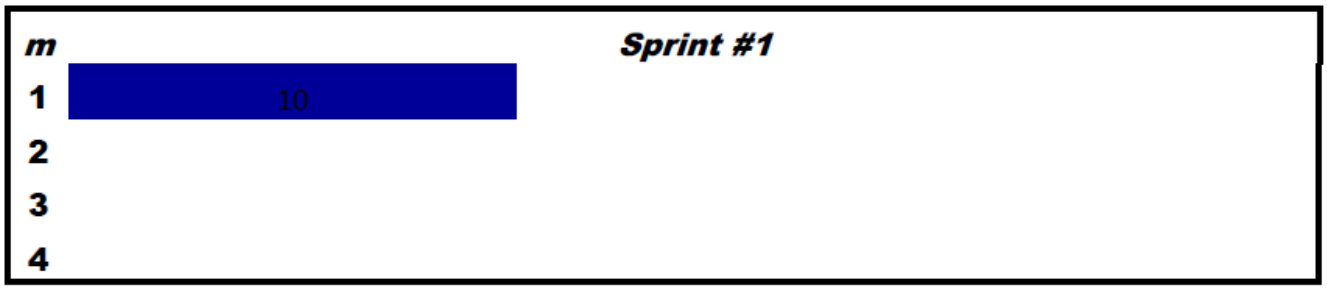

Рисунок 2.1 – Призначення виконавця №1 на завдання

Збережемо останні зміни до таблиці 2.4, де ми будемо оновлювати призначені завдання для кожного виконавця в майбутньому, і яка визначає час виконання роботи для кожного виконавця.

Таблиця 2.4 – Моменти завершення виконання завдань по кожному виконавцю

Виконавець (m)	Моменти завершеннявиконання завдань по кожному виконавцю (C_i)	Етап 1
1	10	1(10)

Кінець таблиці 2.4.

Виконавець (m)	Моменти завершеннявиконання завдань по кожному виконавцю (C_i)	Етап 1
2	0	
3	0	
4	0	

Примітка: Для зручності заповнення далі будемо використовувати позначення m, C_i таблиці в етапі 2.

Розглянемо завдання 2, яке має тривалість дев'ять хвилин. Виходячи з результатів етапу 1, визначте, скільки часу знадобиться кожному виконавцю для виконання цієї роботи. У таблиці 2.5 представлені результати.

Таблиця 2.5 – Етап 2: визначення моменту завершення виконання завдання №2 для кожного виконавця

Виконавець (№)	Момент завершення виконаннязавдання №2
1	19
2	9
3	18
4	27

Знайдіть виконавця, якому потрібно найменше часу, щоб завершити роботу 2. Як видно на рисунку 2.2, виконавець 2 може отримати завдання на етапі 2.

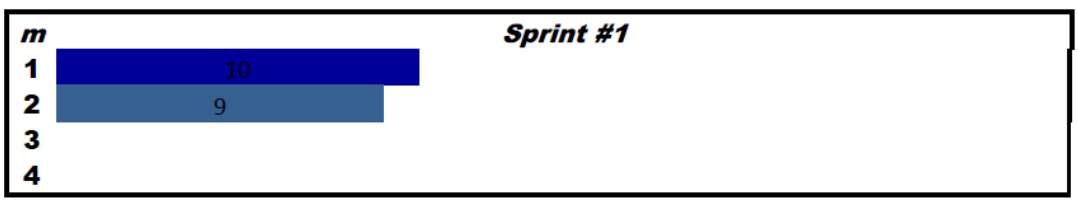

Рисунок 2.2 – Призначення виконавця №2 на завдання

Отримані результати потрібно додати до таблиці 2.6.

Таблиця 2.6 – Моменти завершення виконання завдань по кожному виконавцю

Виконавець(m)	Моменти завершення виконання завдань по кожному виконавцю (C_i)	Етап 1	Етап 2
1	10	1(10)	
2	9		2(9)
3	0		
4	0		

Етап 3.

Розглянемо завдання 3 тривалістю 8 хвилин. Виходячи з результатів етапу 2, визначте, скільки часу знадобиться кожному виконавцю для виконання завдання. У таблиці 2.7 представлені результати.

Таблиця 2.7 – Етап 3: визначення моменту завершення виконання завдання №3 для кожного виконавця

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

Знайдіть виконавця, якому потрібно найменше часу, щоб завершити завдання 3. Як показано на рисунку 2.3, завдання можна призначити завданню номер 3 на етапі 3.

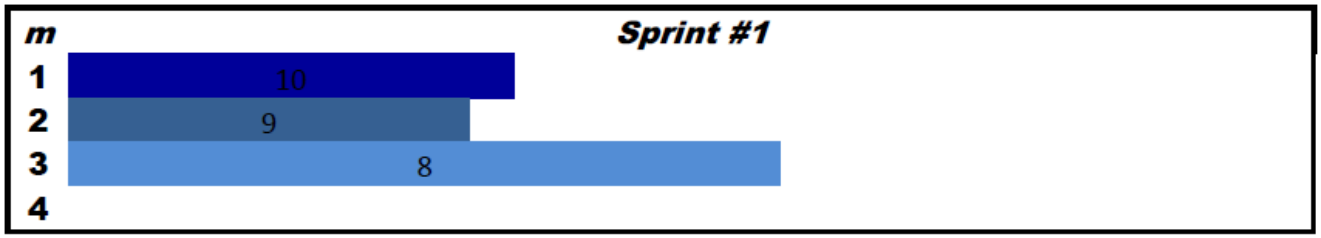

Рисунок 2.3 – Призначення виконавця №3 на завдання

Отримані результати потрібно додати до таблиці 2.8.

Таблиця 2.8 – Моменти завершення виконання завдань по кожному виконавцю

Виконавець(m)	Моменти завершення виконання завдань по кожному виконавцю (C_i)	Етап 1	Етап 2	Етап 3
1	10	1(10)		
2	9		2(9)	
3	16			3(8)
4	0			

Етап 4.

Розглянемо завдання 4, яке має тривалість 7, і визначте час виконання кожним виконавцем цієї роботи, враховуючи результати етапу 3. Таблиця 2.9 представляє результати.

Таблиця 2.9 – Етап 4: визначення моменту завершення виконання завдання №4 для кожного виконавця

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

Визначаємо виконавця, який виконає завдання за найкоротший час. Як показано на рисунку 2.4, виконавцю 2 можуть бути доручені обов'язки на етапі 4.

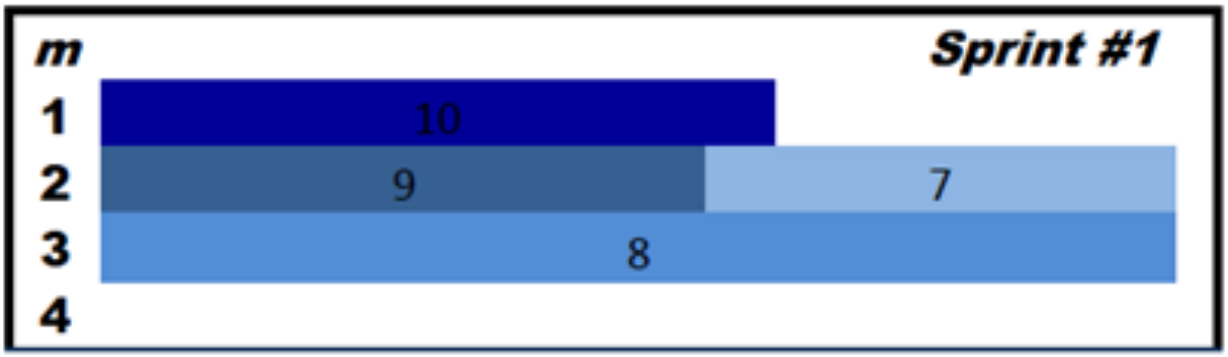

Рисунок 2.4 – Призначення виконавця №2 на завдання

Отримані результати потрібно додати до таблиці 2.10.

Таблиця 2.10 – Моменти завершення виконання завдань по кожному виконавцю

Виконавець (m)	Моменти завершення виконання завдань по кожному виконавцю (C_i)	Етап 1	Етап 2	Етап 3	Етап 4
1	10	1(10)			
2	16		2(9)		4(7)
3	16			3(8)	
4	0				

Етап 5.

Розглянемо Завдання 5, яке має тривалість шість, і визначте час виконання кожним виконавцем цієї роботи, враховуючи результати Етапа 4. Результати відображені в таблиці 2.11.

Таблиця 2.11 – Етап 5: визначення моменту завершення виконаннязавдання №5 для кожного виконавця

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

Знайдіть виконавця, який виконає завдання за найкоротший час. Як показано на рисунку 2.5, виконавцю 1 можуть бути доручені обов’язки на етапі 5.

Рисунок 2.5 – Призначення виконавця №1 на завдання

Внесемо отримані результати до таблиці 2.12.

Таблиця 2.12 – Моменти завершення виконання завдань по кожному виконавцю

Виконавець (m)	Моменти завершення виконання завдань по кожному виконавцю (C_i)	Етап 1	Етап 2	Етап 3	Етап 4	Етап 5
1	16	1(10)				5(6)
2	16		2(9)		4(7)	

Кінець таблиці 2.12.

Виконавець (m)	Моменти завершення виконання завдань по кожному виконавцю (C_i)	Етап 1	Етап 2	Етап 3	Етап 4	Етап 5
3	16			3(8)		
4	0					

Етап 6.

Розглянемо завдання номер шість, яке має довжину шість, і визначте час, необхідний кожному виконавцю для його виконання, враховуючи результати п’ятого етапу. У таблиці 2.13 представлені результати.

Таблиця 2.13 – Етап 6: визначення моменту завершення виконаннязавдання №6 для кожного виконавця

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

Знайдіть виконавця, який виконає завдання за найкоротший час. Як показано на рисунку 2.6, виконавцю 4 можуть бути доручені обов’язки на етапі 6.

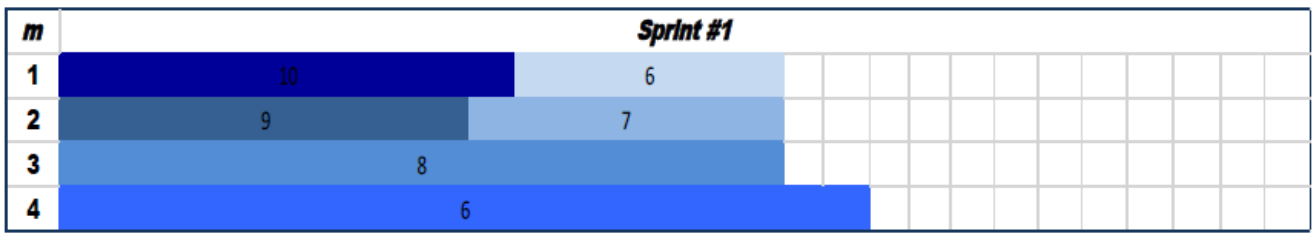

Рисунок 2.6 – Призначення виконавця №4 на завдання

Отримані результати потрібно додати до таблиці 2.14.

Таблиця 2.14 – Моменти завершення виконання завдань по кожному виконавцю

Виконавець(m)	Моменти завершення виконання завдань по кожному виконавцю (C_i)	Етап 1	Етап 2	Етап 3	Етап 4	Етап 5	Етап 6
1	16	1(10)				5(6)	
2	16		2(9)		4(7)		
3	16			3(8)			
4	18						6(6)

Примітка: для наступних 10 етапів повторюємо ті самі етапи в кожному етапі. Далі опишемо останні два етапу.

Етап 16

Розглянемо завдання 16 довжиною 1 і визначимо, скільки часу знадобиться кожному виконавцю для його виконання, враховуючи результати етапу 15. У таблиці 2.15 представлено результати.

Таблиця 2.15 – Етап 16: визначення моменту завершення виконання завдання №16 для кожного виконавця

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

Визначаємо виконавця, який виконає завдання за найкоротший час. Виконавець 4 може отримати завдання на етапі 16, як показано на рисунку 2.7.

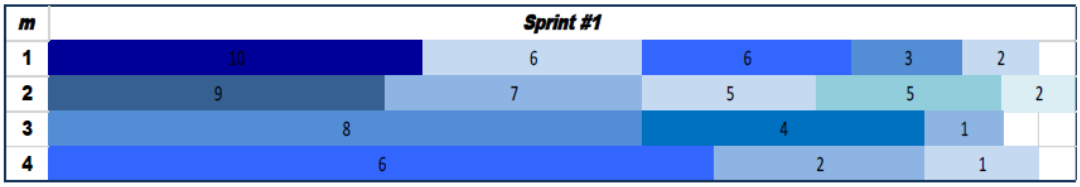

Рисунок 2.7 – Призначення виконавця №4 на завдання

Отримані результати потрібно додати до таблиці 2.16.

Таблиця 2.16 – Моменти завершення виконання завдань по кожному виконавцю

m	Ci	Етап 1	Етап2	Етап 3	Етап 4	Етап 5	…	Етап1 2	Етап1 3	Етап 14	Етап 15	Етап 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.

Розглянемо завдання 17 із тривалістю 1 і визначимо час виконання кожного виконавця для цього завдання, враховуючи результати етапу 16. Таблиця 2.17 представляє результати.

Таблиця 2.17 – Визначення завантаженості машин

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

Визначаємо виконавця, який виконає завдання за найкоротший час. Виконавець 3 може отримати завдання на етапі 17, як показано на рисунку 2.8.

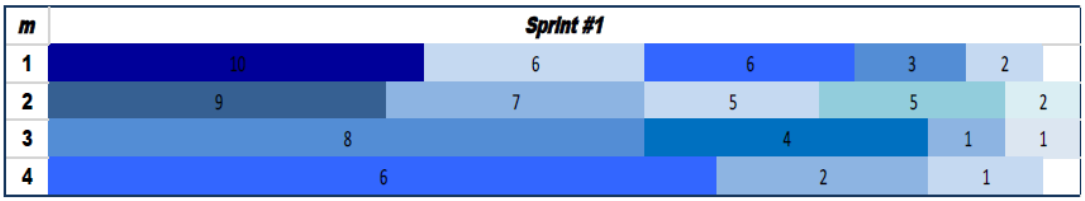

Рисунок 2.8 – Призначення виконавця №1 на останнє завдання

Отримані результати потрібно додати до таблиці 2.18.

Таблиця 2.18 – Моменти завершення виконання завдань по кожному виконавцю

m	Ci	Етап 1	Етап 2	Етап 3	Етап 4	Етап 5	…	Етап 12	Етап 13	Етап 14	Етап 15	Етап 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)	

Значення цільової функції для альфа-алгоритму, який дав результати для виконавців №2 і №3, становить 28.

Висновки до розділу

Формальна постановка задачі, математична модель і критерії оптимальності задачі були представлені в цьому розділі магістерської роботи.

Досліджено характеристики пропонованого завдання, особливості завдань та виконавців системи. Було продемонстровано, які алгоритми були обрані для вирішення задачі, їх переваги та недоліки, а також схеми роботи алгоритмів.

Наведено детальну ілюстрацію того, як алгоритм працює з таблицями, де дані записуються після кожного етапу. Надається результат прикладу або значення цільової функції, виявлене після обчислень.

3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис програмного забезпечення

Вибір найбільш ефективного середовища та технології для розробки кількох алгоритмів, що передбачають збереження всіх даних у базі даних, а також створення класифікатора завдань, що вимагає високого рівня продуктивності від обраних інструментів для його реалізації, були проблемами, які не один автор, з якими стикався під час роботи над магістерською роботою. Нижче наведено описи вибраних технологій і необхідного програмного забезпечення.

3.1.1 Вимоги до програмного забезпечення

Функціональні вимоги до програмного забезпечення наведено в таблиці 3.1.

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 дозволяє відображати отриманий графік (результат)	Середній
4	Система дозволяє проводити експерименти.	Високий
5	Система дозволяє демонструвати результати експерименту.	Високий

Кінець таблиці 3.1.

№	Назва	Пріоритет
6	Система надає можливість вводити дані для генерування задач: – кількість задач; – кількість завдань; – тип команди; – тип спринта.	Високий
7	Система пропонує можливість використання двох алгоритмів для вирішення задачі (алгоритм локального пошуку та жадібний алгоритм).	Високий
8	Система дозволяє тестувати два алгоритми.	Високий

Кожна з вищезазначених функцій виконується відповідним модулем у цій системі, і кожен із цих модулів пропонує можливість виконувати певний набір дій, таких як зміна даних, пошук даних, збереження результатів пошуку та створення звітів.

На рисунку 3.1 наведено логічну та функціональну організацію програмного забезпечення, створеного в рамках магістерської роботи.

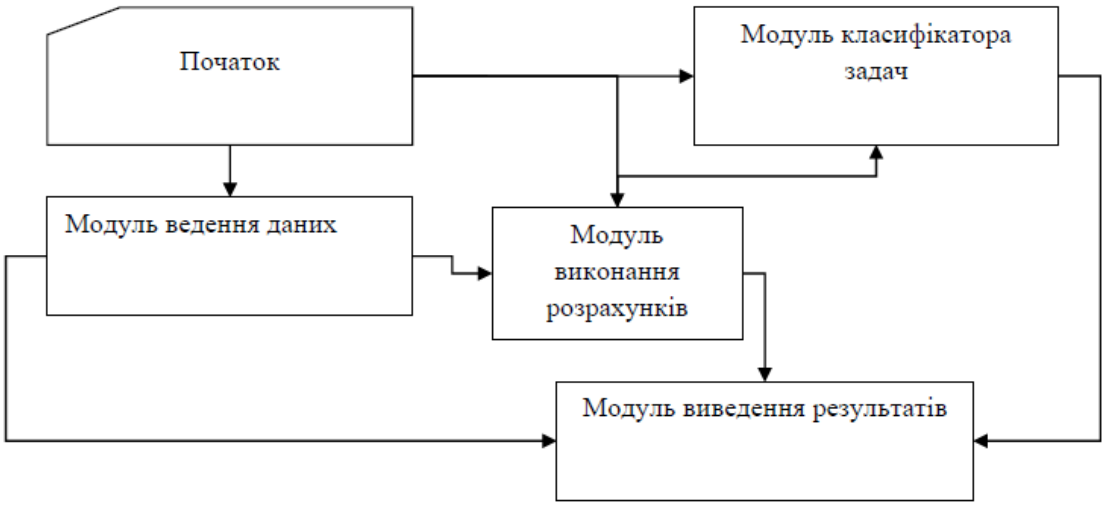

Рисунок 3.1 – Логічна та функціональна архітектура програмного забезпечення

Інформаційна система планування ресурсів ІТ-проектів пройшла успішне тестування та готова до використання.

Інформаційна система планування ресурсів ІТ-проектів дозволяє підвищити продуктивність, забезпечити простоту та зручність розрахунків, зменшити кількість помилок у плануванні ресурсів ІТ-проектів щодо часових обмежень, підвищити якість планування та знизити трудовитрати на обробку інформації.

Керівники та окремі користувачі можуть автоматизувати планування ресурсів для ІТ-проектів за допомогою цієї програми, що прискорить розрахунки та дасть можливість відстежувати зміни значень кожного параметра проекту (ресурсу).

Сучасні інформаційні технології необхідно впроваджувати з високим ступенем фінансового забезпечення. Компанії, що працюють у ринкових умовах, повинні принаймні аналізувати економічні наслідки та, в ідеалі, оцінювати економічну ефективність кожного кроку зміни системи.

Впровадження системи – це складна практична робота, яка залежить від багатьох змінних. Процес реалізації складається з кількох етапів, кожен з яких вимагає ретельної координації між виконавцем і замовником, контролю за виконанням вимог і навчання персоналу.

Систему планування ресурсів проекту інформаційних технологій можна реалізувати в три етапи:

Дослідження. Спочатку проводиться вивчення теми та процедур.

Доопрацювання системи. Програмісти налаштовують або покращують найважливіший компонент системи.

Запуск системи. Для початку використання системи в режимі реального часу разом із процедурами навчання співробітників.

На етап дослідження бізнес-процесів відводиться певний час. Тут вам потрібно бути якомога конкретнішим щодо процесів, які потрібно змінити.

Функціональність інформаційної системи, як правило, набагато більша, ніж фактичні процеси, які виконуються. На цьому етапі необхідно прийняти рішення щодо того, як додавання різних функцій вплине на вартість системи, її виконання

та впровадження, і, що найважливіше, чи відповідатиме запропонована функціональність потребам бізнесу.

Безсумнівно, вкрай важливо, щоб результати дослідження були представлені окремо в документі, який містить детальний опис процесу діяльності відповідно до специфікацій компанії.

Після етапу дослідження необхідно дати точні відповіді щодо ціни впровадження, термінів і запуску інформаційної системи.

Досить точно контролювати процес впровадження необхідних компонентів в ІС при доопрацюванні системи [31]. Вкрай важливо, щоб у команді клієнта був хтось, хто досконало розуміє цілі, завдання та операційні процедури компанії.

Бізнес-операції компанії повинні бути перетворені на використання системи, яка впроваджується на етапі запуску системи.

Як правило, для функціонування розробленої системи потрібен певний час. Необхідно проаналізувати ефективність впровадження та визначити, чи досягнуто основних цілей [32].

Лише тоді, коли система пропонує можливість отримати вигоду, тобто коли вона покращує робочий процес різних відділів компанії, дає змогу набагато швидше виконувати завдання, підвищує стандарти роботи всіх учасників і оптимізує різні процеси, можна впровадження вважатиметься успішним. Необхідно постійно оцінювати ефективність системи, ступінь зацікавленості працівників у використанні системи та контекст, у якому вона використовується. Час, необхідний для впровадження системи ІС, становить від кількох місяців до року. Дуже важливо зосередитись на цілях, які бізнес визначив раніше і сподівається досягти, впроваджуючи систему на цьому етапі. Має сенс враховувати потенційні ризики та витрати, пов’язані з певними ресурсами. Перш за все, перед тим, як прийняти рішення про запуск ІС, має бути встановлено коло завдань, які повинна виконувати інформаційна система. Технологія, яка може підтримувати роботу системи, необхідна для того, щоб інформаційна система була успішною [33].

Технічна допомога загалом відноситься до організаційної технології, яка використовується для впровадження системи планування ресурсів IT-проекту. Організаційна технологія створена для впровадження технологій зберігання, надання та використання інформації, а також для виконання численних завдань підтримки в контексті конкретних технологій інформаційної підтримки управлінської діяльності [34].

3.1.2 Архітектура програмного забезпечення

Впровадження процесу планування ресурсів для IT-проектів є метою розробки програмного продукту. Користувач отримує створений розклад (план) завдань на виході програмного продукту для використання в діяльності підприємства [35].

Це частина діаграми класів з рисунка 3.2, яка ілюструє архітектуру програмного забезпечення.

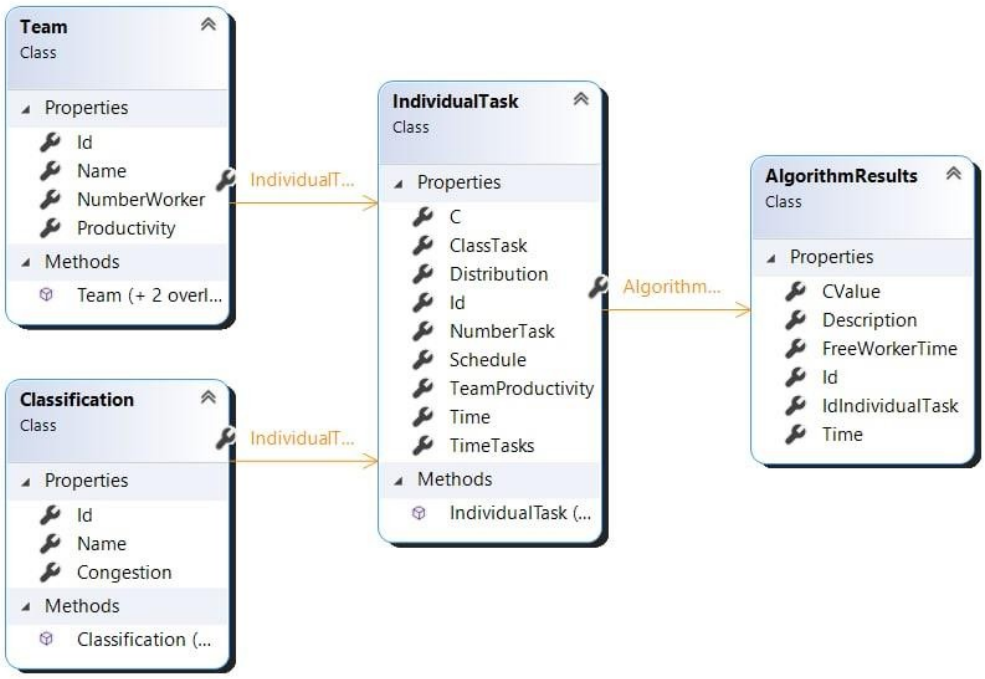

Рисунок 3.2 – Діаграма класів програмного забезпечення

Відповідно до результатів аналізу та з урахуванням вищезазначених функцій було розроблено дерево цілей системи планування ресурсів IT-проекту, як показано на рисунку 3.3.

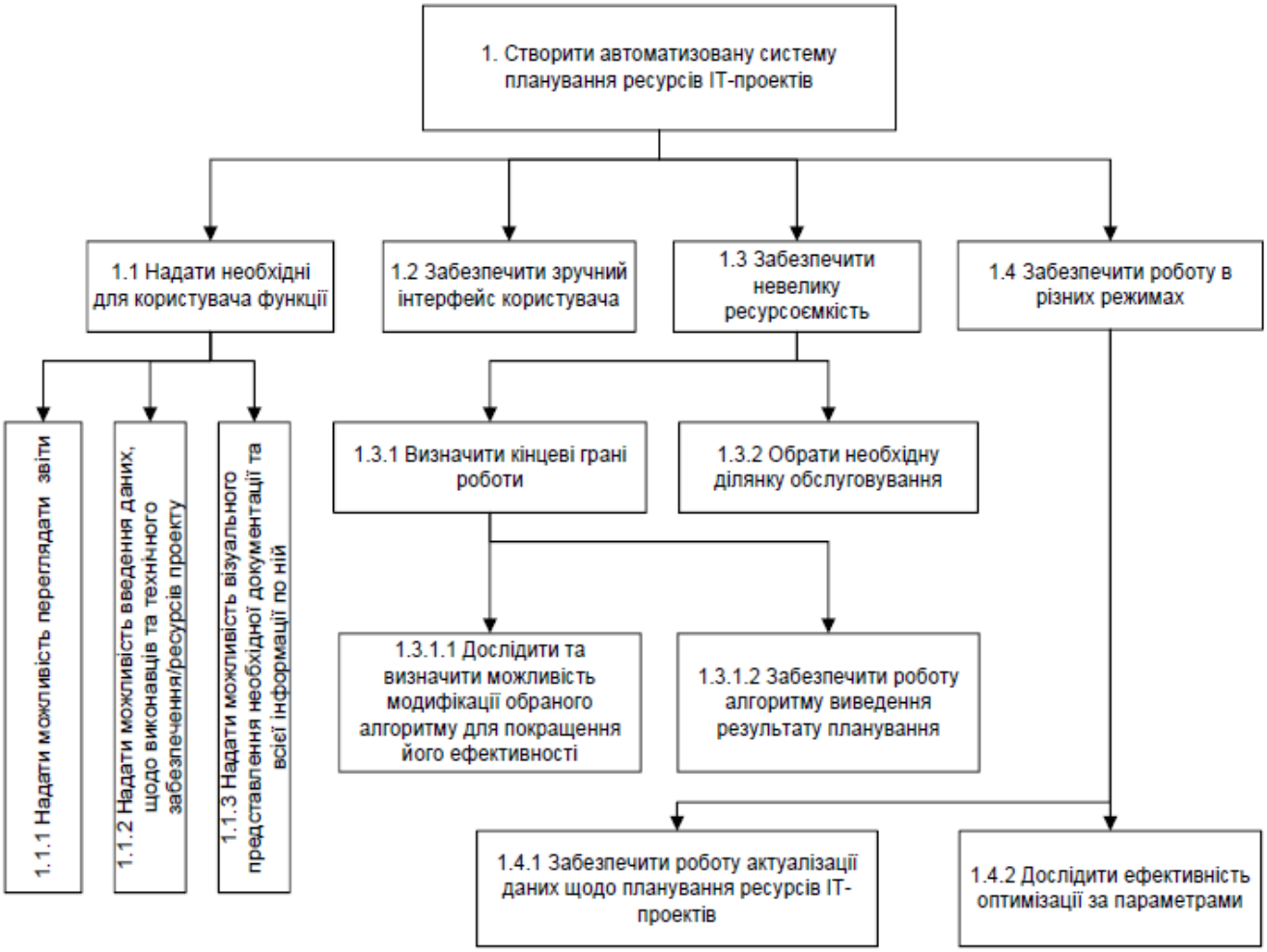

Рисунок 3.3 – Декомпозиція цілей програмного забезпечення

Компоненти єдиної системи категоризації, єдиної системи ведення обліку, повторюваних схем інформаційного процесу та методології виробництва даних складають системне інформаційне забезпечення в організації. Нижче наведено ілюстрації блоків обробки, через які проходить інформація, що надходить у систему, їхні відповідні обов’язки щодо обробки та їх взаємодію. На рисунку 3.4 зображено блок-схему обробки інформації.

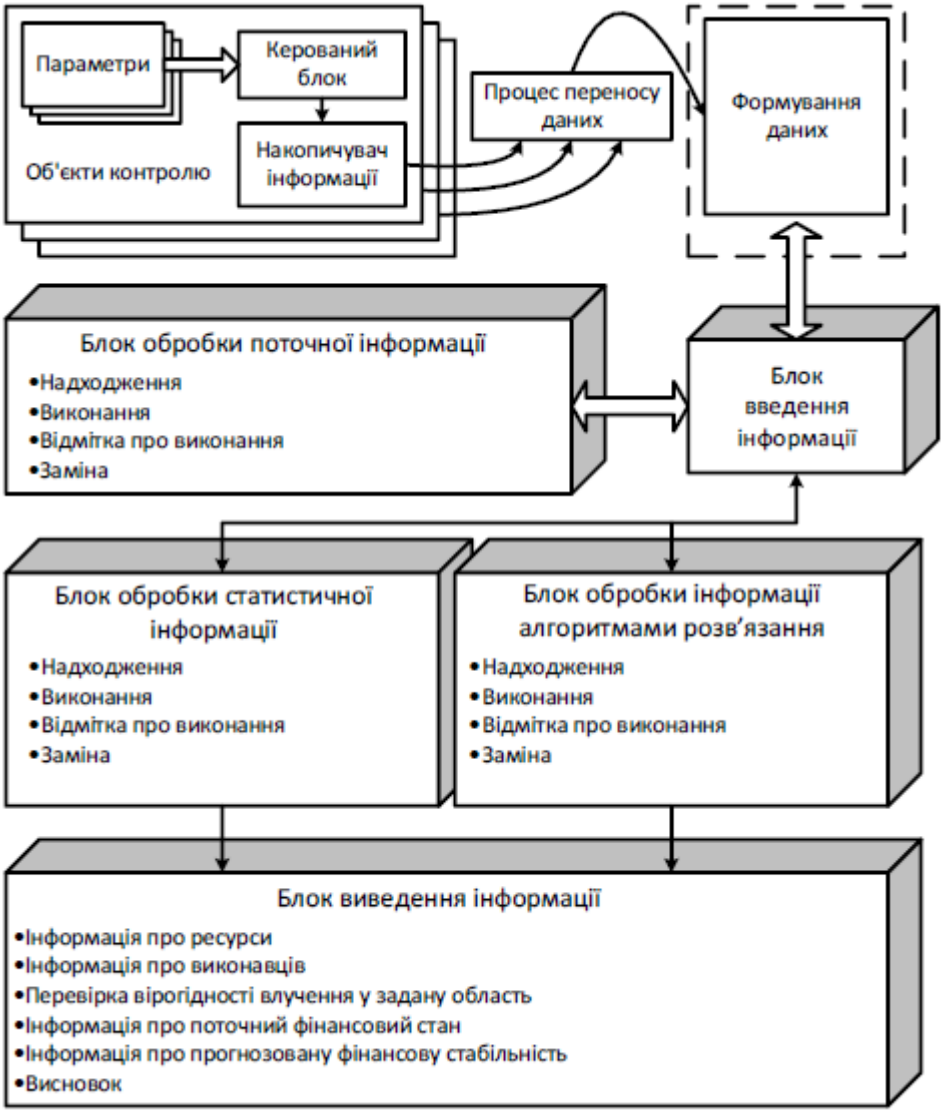

Рисунок 3.4 – Схема обробки інформації у програмному забезпеченні

В даний час дуже актуальним є створення та впровадження інформаційної технології моніторингу документів за статистичними даними реалізацій визначальних параметрів з метою фіксації документації планування ресурсів IT-проектів на стадії експлуатації [36].

Завдання визначення поточного стану та індивідуального планування ресурсів IT-проектів можуть бути вирішені за допомогою методів моніторингу та контролю, які є способами технічної діагностики. Вони також здатні обробляти та поширювати інформацію про стан об'єкта.

Основні функціональні компоненти інформаційної системи, а також те, як вони взаємодіють і поєднані, зображено на рисунку 3.4. Кожен із цих модулів

відповідає за виконання певного завдання для системи [37], але вони не становлять клас програмного забезпечення.

На рисунку 3.5 зображено функціональну організацію програмного забезпечення.

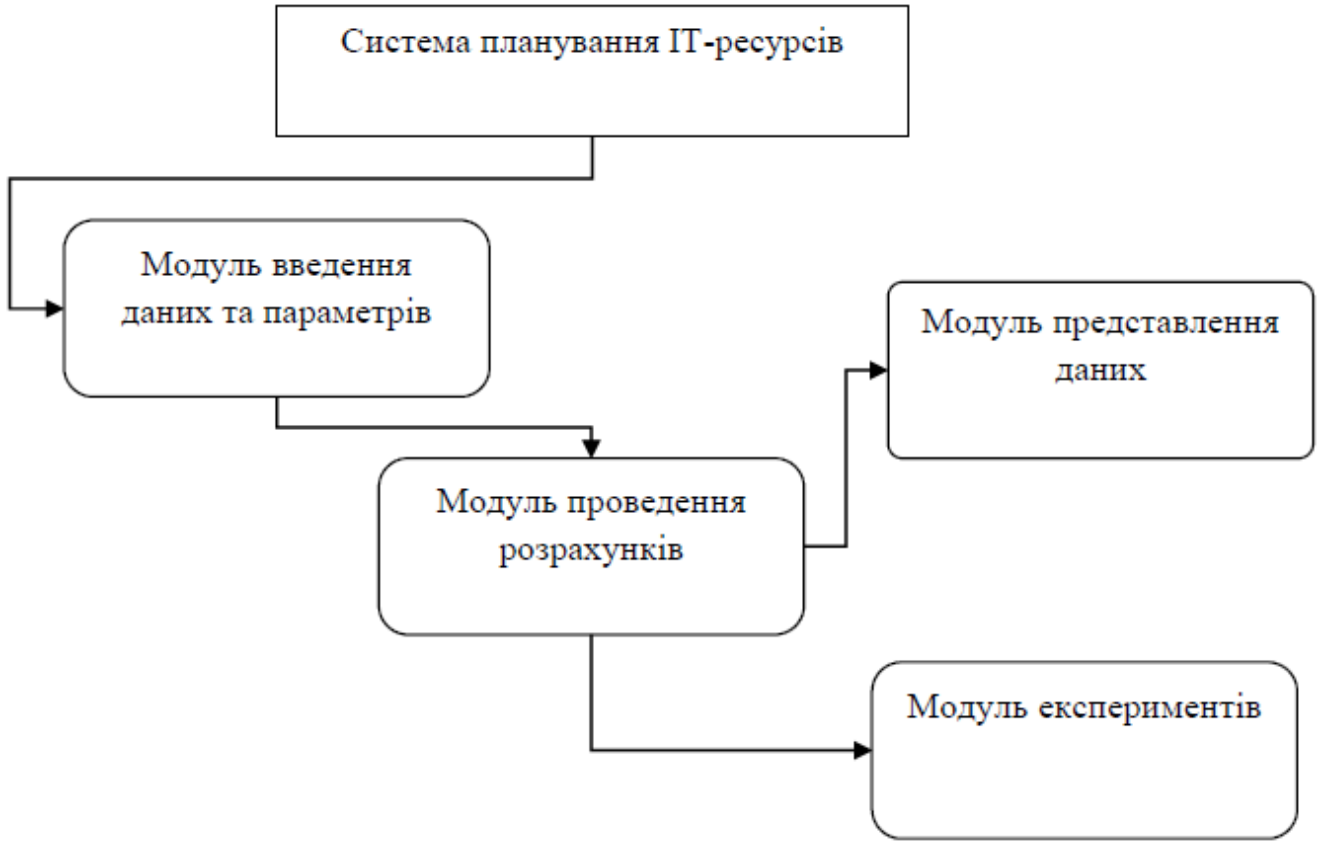

Рисунок 3.5 – Блок-схема програмного забезпечення

Модуль введення даних та параметрів

У цьому модулі пропонується введення даних за завданням і виконавцем. Для можливого використання системою планування ресурсів IT-проекту програмне забезпечення зберігає числові дані для кожного показника [38].

Модуль проведення розрахунків

Обов’язки цього модуля включають обробку обчислень на основі згенерованих алгоритмів вхідного завдання та аналіз вхідних даних.

Модуль експериментів

Через цей модуль проводяться експерименти системи. Користувач може

різними способами експериментувати з вхідними завданнями та застосовувати результати у своїх дослідженнях. Нові показники поступово доповнюватимуть і змінюватимуть результат.

Модуль представлення даних

Цей модуль відповідає за текстове представлення зібраних результатів і показ їх користувачеві. Результати будуть надані більш глибоко, чим більше вхідних даних буде надано [40].

Оскільки вона призначена для різноманітної групи користувачів, ця система суттєво відрізняється від попередніх систем.

3.1.3 Користувацький інтерфейс

Користувальницький інтерфейс програми має бути зручним для користувача та відповідати інструкціям щодо дизайну інтерфейсу.

Інтерфейс складається з таких частин:

– домашня сторінка;

– сторінка, на якій розв’язується одне завдання та відображаються результати (розклад);

– веб-сайт, що містить класифікатор для діяльності, пов’язаної з експериментом;

– сторінка для генерації завдань (генератор завдань).

На рисунках 3.6–3.7 показано компонування екрана програмного забезпечення для формування розкладу після введення даних для окремої проблеми, яку потрібно вирішити, а також інші схеми екрану для роботи алгоритму та алгоритму локального пошуку.

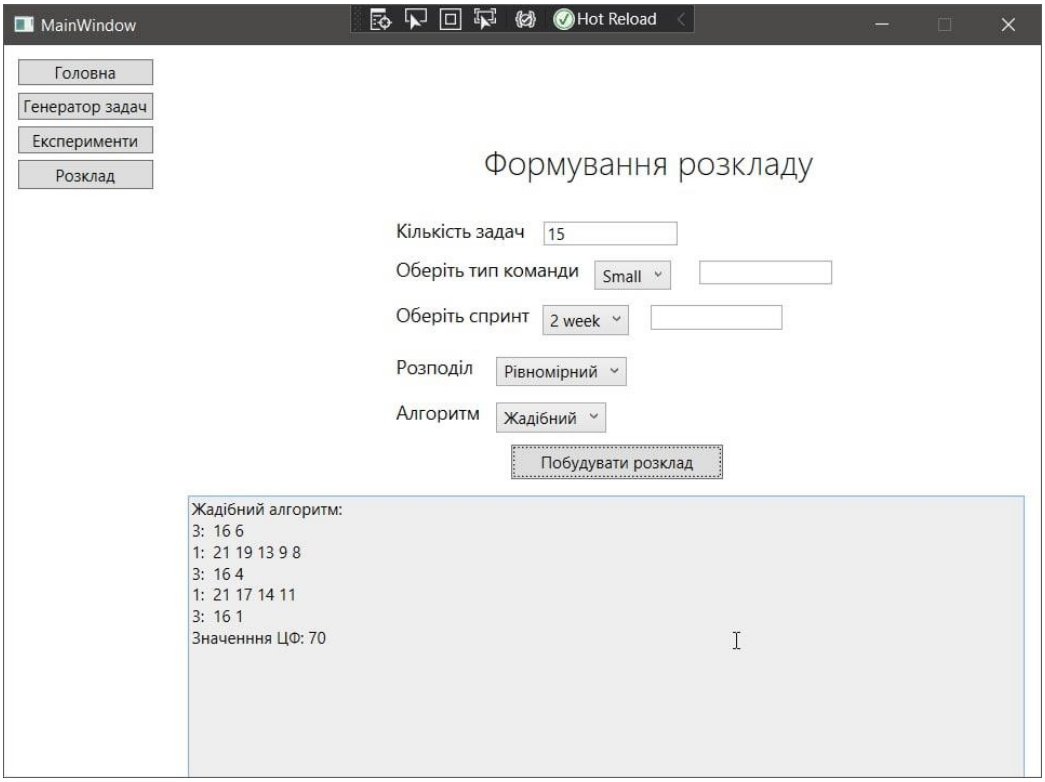

Рисунок 3.6 – Екранна форма для створення планування завдань (жадібний алгоритм)

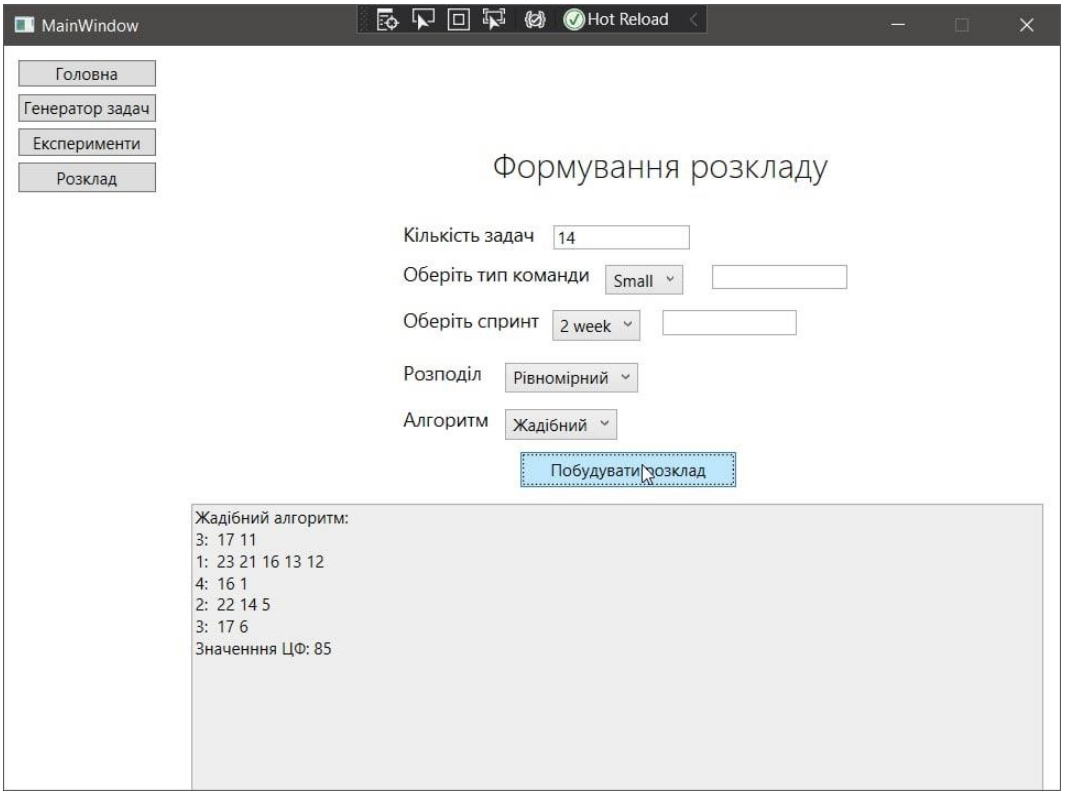

Рисунок 3.7 – Екранна форма для створення планування завдань (алгоритм локального пошуку)

Користувач повинен бути проінформований про масштаби та введення проекту, оскільки він має повний контроль над системою, і їхні рішення впливатимуть на те, як вона функціонує [41].

Система відповідає наступним умовам і характеристикам:

– забезпечити інтерфейс введення даних для користувачів для введення показань (ресурси/виконавці);

– перевірити введені показання, обробити їх і зберегти результати у форматі, який може використовувати програмне забезпечення;

– робити згортку значень, створюючи найкращі результати на основі набору;

– представити результати у зрозумілій формі для сприйняття івикористання вигляді;

– обмежити доступ до введення кількох видів інформації;

– не допускати введення значень, які можуть призвести до поломки системи;

– завантажувати та зберігати інформацію для подальшого використання;

– кожне системне вікно повинно мати розділ довідки з інформацією про роботу програми та теоретичними відомостями;

– програмний код системи повинен відповідати стандартам кодування;

– одна версія системи може використовуватися одночасно багатьма користувачами на різних ПК.

Екрани форм для генератора завдання та експерименту показані на рисунках 3.8 та 3.9 нижче.

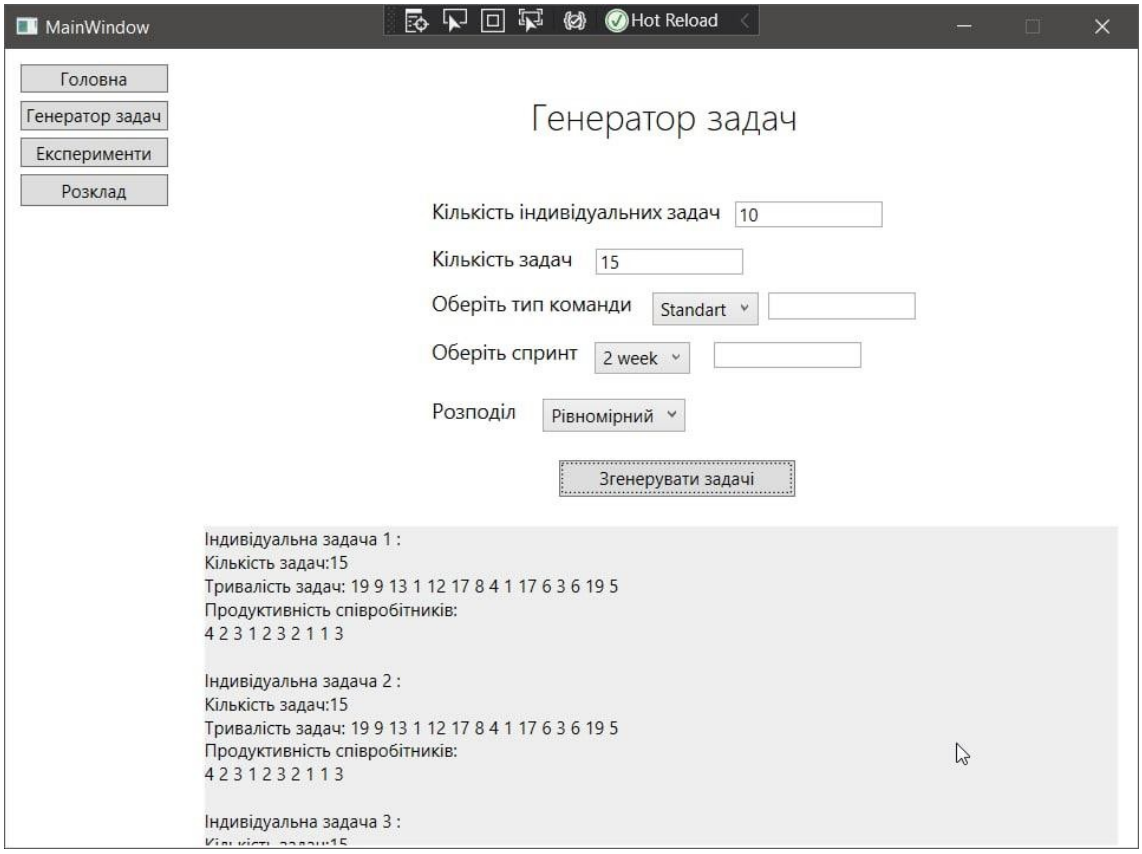

Рисунок 3.8 – Екранна форма генерації завдань

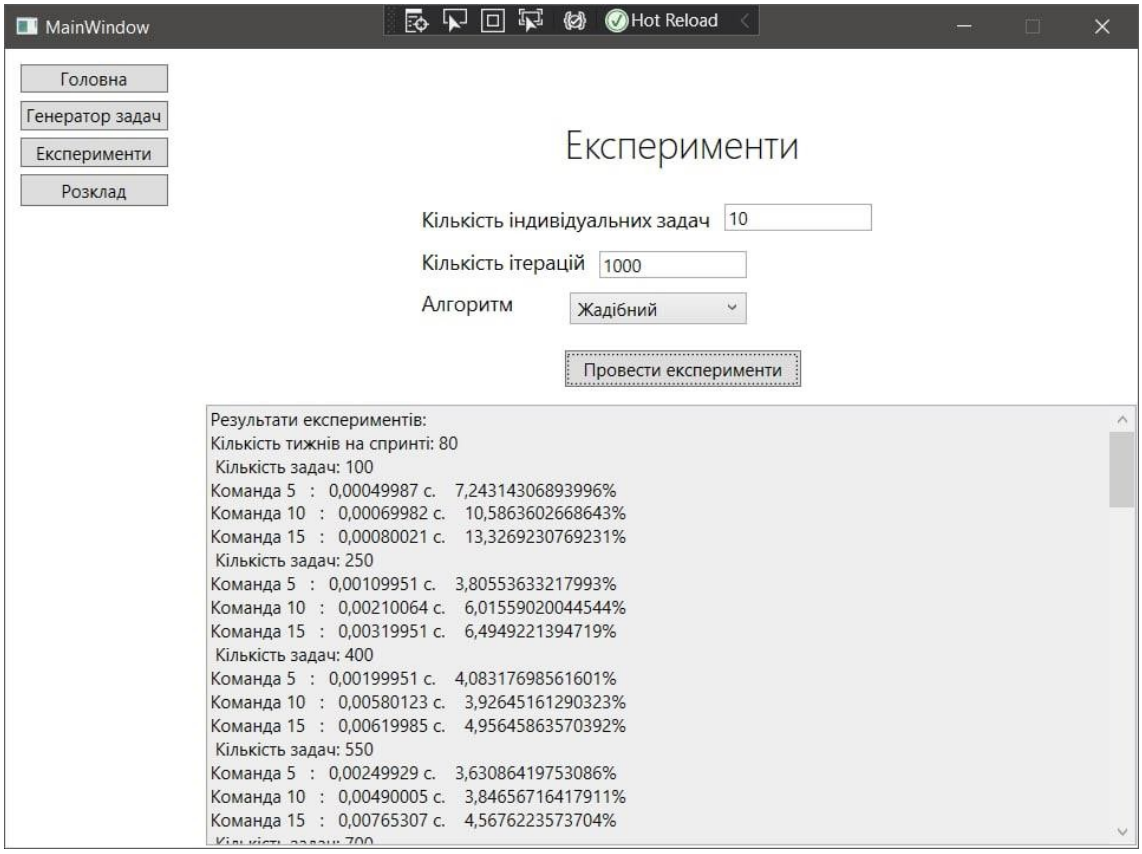

Рисунок 3.9 – Екранна форма проведення експериментів

Надійність. Програма має високий рівень надійності та здатна обробляти помилки користувача[42].

Безпека даних. Шифрування файлів є однією з функцій захисту даних програми [43].

Мобільність. На комп'ютері під керуванням операційної системи Windows 7/11 запускається програма.

Зручність використання. Програмне забезпечення повинно забезпечувати інтуїтивно зрозумілий інтерфейс користувача, який дозволяє користувачам використовувати систему без необхідності спочатку вивчати, як це робити [44].

Завдяки тому, що кожен користувач може вводити власну інформацію і не може змінювати інформацію інших користувачів, усі користувачі мають доступ до бази даних.

3.1.4 Засоби розробки

Тонкощі розробки програмного забезпечення, рівень його популярності та фінансові ресурси розробника відіграють важливу роль у виборі програмного забезпечення. Обрана мова програмування C#.

C#, одна з найбільш затребуваних мов програмування в IT-секторі, використовується для створення як настільних програм, так і веб-програм. Платформа (платформа) Microsoft.NET, яка пропонує різноманітні можливості та спрощує створення програм, була створена спеціально для роботи з C#. Common Language Runtime (CLR), загальномовне середовище виконання, яке компілює програмний код MSIL (у який скомпільовано код C#) під час виконання, а також надає доступ до програм MSIL (і, отже, до програм, написаних на мовах високого рівня, які підтримують .NET Framework). до бібліотеки класів .NET Framework [45], є однією з основних частин пакета Microsoft.NET Framework.

Можна створювати традиційні настільні програми Windows Forms і Windows Presentation Foundation, веб-програми (за допомогою технології ASP.NET), компоненти розподілених програм і доступ до бази даних за допомогою C# і .NET

Framework. З платформою Windows C# пропонує можливість створювати практично будь-який тип програмного забезпечення [46].

На C# створено зручний інтерфейс для взаємодії з системою тестування. побудова робочого простору, схожого на панель управління з «кнопками» [47].

Програма повинна мати основний комп'ютерний код для виконання визначених функцій, таких як обчислення, показ результатів і реагування на дії користувача, такі як натискання кнопок або вибір рядків зі списку.

Середовищем інструменту проектування є Rational Software Architect [48].

3.2 Опис отриманих результатів розв'язання та проведених експериментів

Щоб подолати проблему, були проведені експериментальні експерименти. Результати детально описані в розділі нижче.

По-перше, було побудовано конкретний класифікатор завдання, на основі якого визначено розподіл вхідних даних за рахунок генератора різних значень, оскільки магістерська робота представляла багатокритеріальну задачу теорії розкладів.

Ключові змінні, які впливають на результат і швидкість, з якою виконуються алгоритми, були виявлені під час досліджень.

Параметр 1 – розмір спринта. Завдання плануються за допомогою спринтів і реалізуються за допомогою методології Scrum. Назва та розмір спринту наведено в таблиці 3.2 відповідно.

Таблиця 3.2 – Класифікація спринтів за розміром

Тривалість спринта у тижнях / Позначення у класифікаторі	Інтесивність спринта на одну людину
2 тижні (2 weeks)	80 годин
3 тижні (3 weeks)	120 годин
4 тижні (4 weeks)	160 годин

Отже, потужність спринту для всієї команди визначається на основі кількості членів команди, які подали заявку на участь.

Параметр 2 – розмір команди. Наші команди часто невеликі, тому що ми використовуємо Scrum, однак існують рекомендації щодо ідеального розміру команди. У таблиці 3.3 наведено можливу кількість членів команди в діапазоні.

Таблиця 3.3 – Класифікація розмір команд та їх розмірів

Розмір команди	Кількість учасників
Маленький	3-5 учасників
Стандартний	6-10 учасників
Великий	11-15 учасників

Кількість завдань у спринті є параметром 3. У зв’язку з тим, що завдання можуть виконуватися протягом дуже тривалого часу та їх може бути кілька одночасно, це значення було використано в діапазоні від 100 до 1000 завдань. Також навпаки. Значення завдань, використаних у дослідженнях, наведено в таблиці 3.4 нижче.

Таблиця 3.4 – Список значень для підрахунку завдань експерименту

Кількість завдань
100
150
400
550
700
800
1000

Проведем експеримент №1

Експеримент 1.

Командам було призначено наступні завдання для виконання експериментів: 100, 150, 400, 550, 700, 850 і 1000. Розбіжність у відсотках між отриманим значенням розкладу та прогнозованим значенням і цільовою функцією було розраховано для жадібних і локальних методи пошуку.

На вході маємо:

– спринт – два тижні;

– кількість учасників команди – змінна;

– кількість задач – змінна.

На виході отримуємо:

Відхилення отриманого графіка у відсотках (%) від прогнозованого значення та бажаної функції.

На рисунках 3.10 і 3.11 показано результати експерименту, а в таблиці 3.5 – зібрані дані.

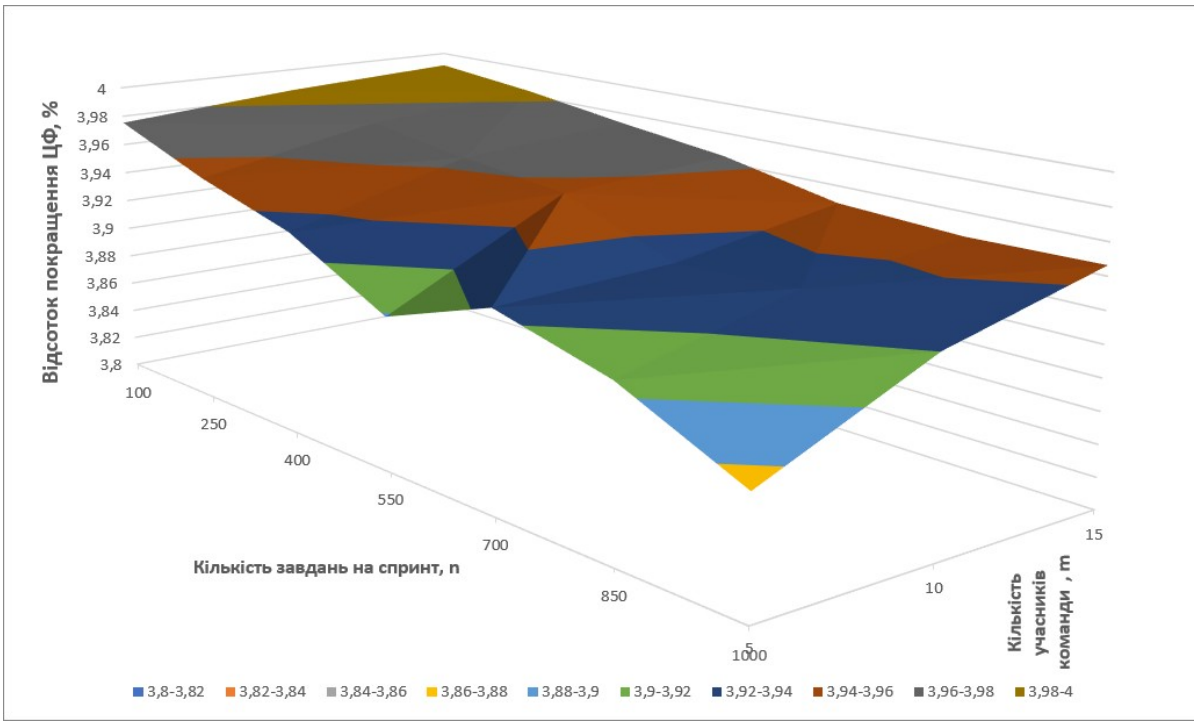

Рисунок 3.10 – Графік результатів алгоритму локального пошуку, що відображає швидкість покращення значення цільової функції

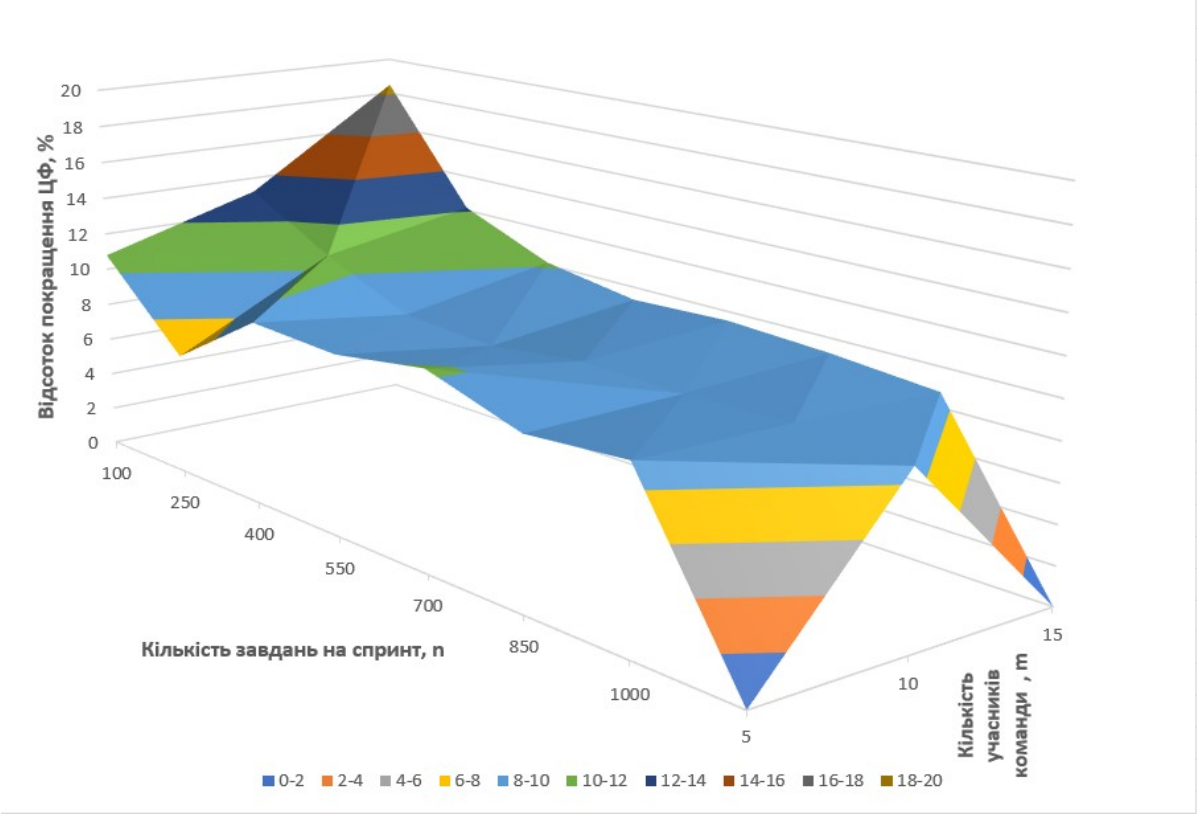

Рисунок 3.11 – Графік результатів жадібного алгоритму, що відображає відсоткове збільшення значення цільової функції

Таблиця 3.5 – Жадібний алгоритм. Відхилення у % значень ЦФ від очікуваних результатів

	Кількість учасників команди		
Клас спринту	5	10	15
2 weeks	9.124	9.027	8.758
	8.673	8.838	9.228
	10.23	8.965	9.516
	9.45	8.319	9.311
	9.684	8.631	10.134
	6.457	10.662	12.149
	10.793	13.218	18.484

Кінець таблиці 3.5.

	Кількість учасників команди		
Клас спринту	5	10	15
3 weeks	8.600	9.313	8.838
	8.819	8.698	9.273
	9.956	8.623	8.811
	9.691	9.112	9.775
	9.314	9.282	10.575
	8.689	11.338	11.984
	10.331	8.039	18.502
4 weeks	8.811	9.350	8.429
	9.270	8.740	8.881
	8.057	8.742	9.324
	9.140	9.552	9.0195
	9.189	8.415	9.808
	10.038	10.258	12.471
	10.376	13.468	17.649

Жадібний алгоритм мав найвищий відсоток відхилення від отриманого значення цільової функції, але він значно перевершував інші алгоритми з точки зору часу роботи алгоритму. Значення цільової функції, які генерує алгоритм, мають похибку значення цільової функції від отриманих результатів не більше 3-4%.

Експеримент 2

Команди отримали такі завдання для експериментів: 100, 150, 400, 550, 700, 850 і 1000.

На вході маємо:

– спринт – чотири неделі;

– кількість учасників команди – змінна;

– кількість задач – змінна.

На виході отримуємо:

Час роботи в мілісекундах алгоритму для даного класу завдань.

На рисунках 3.12 і 3.13 показано результати експерименту, а в таблиці 3.6 наведено значення результатів.

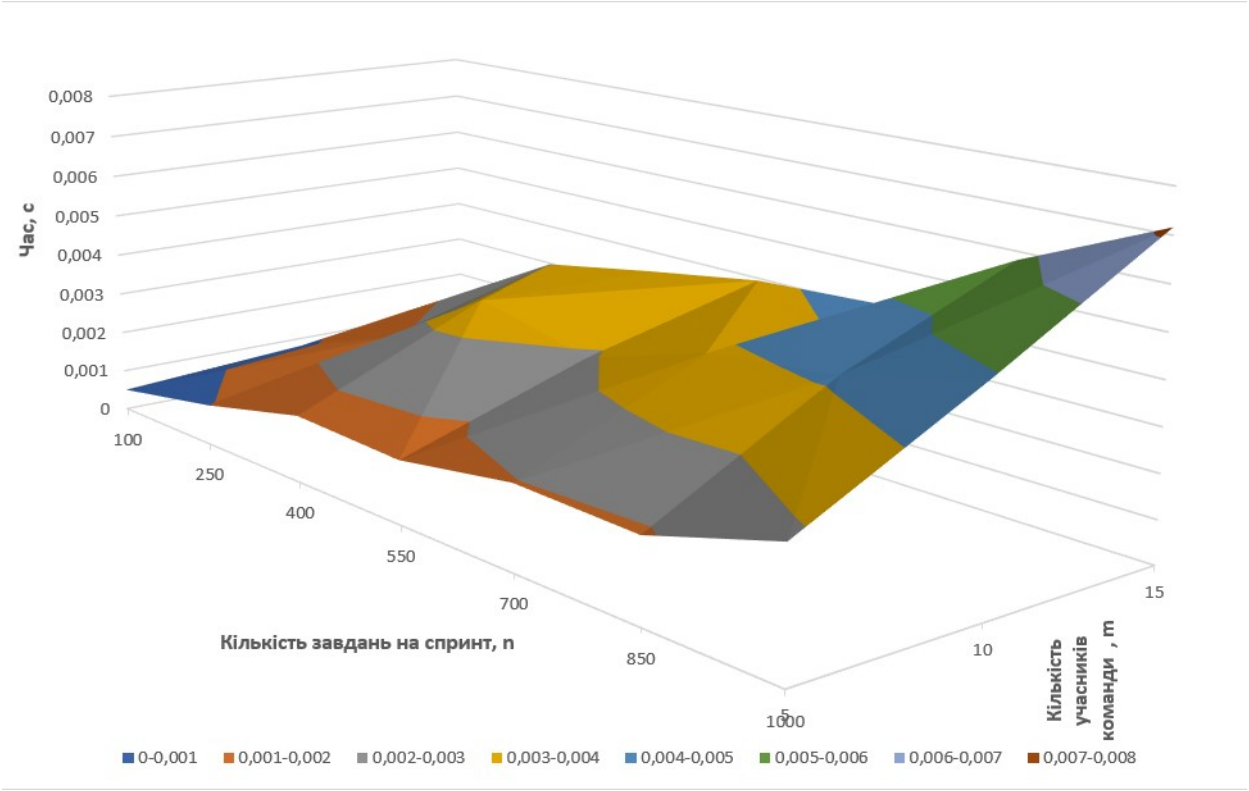

Рисунок 3.12 – Графік результатів часу роботи алгоритму з урахуванням налаштувань

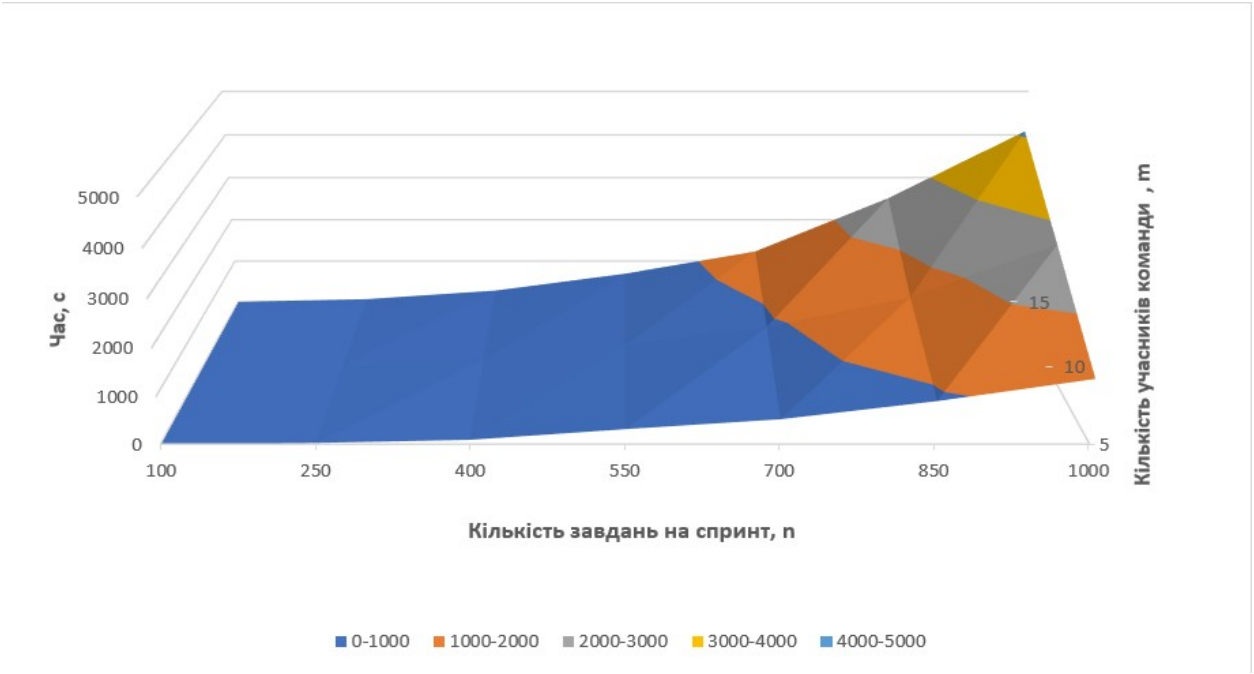

Рисунок 3.13 – Графік, що відображає результати локального алгоритму часу пошуку з урахуванням параметрів

Таблиця 3.6 – Час роботи жадібного алгоритму (у мс)

Клас спринту	Кількість завдань	Кількість учасників команди		
		5	10	15
2 weeks	100	0.00497	0.000801	0.00111
	250	0.00966	0.01804	0.0296
	400	0.01588	0.03514	0.03477
	550	0.01482	0.02908	0.03941
	700	0.01954	0.03812	0.04102
	850	0.01886	0.04163	0.05808
	1000	0.0281	0.04921	0.07162

Кінець таблиці 3.6.

3 weeks	100	0.00249	0.00417	0.00543
	250	0.0054	0.00942	0.01484
	400	0.01004	0.02124	0.0304
	550	0.01437	0.0255	0.04226
	700	0.01766	0.03273	0.05467
	850	0.02135	0.03288	0.05047
	1000	0.02514	0.05244	0.06622
4 weeks	100	0.00336	0.00528	0.0068
	250	0.00682	0.00268	0.01722
	400	0.01076	0.02041	0.03047
	550	0.01205	0.02245	0.03227
	700	0.01722	0.03331	0.05283
	850	0.02066	0.04273	0.05972
	1000	0.02181	0.03946	0.05575

Що стосується швидкості, вище наведено результати досліджень, які стосуються швидкості роботи алгоритмів.

Незважаючи на те, що ітерації для пошуку кращого рішення займають більше часу, експерименти показують, що алгоритм локального пошуку добре працює у пошуку кращих відповідей. Це також є результатом використання алгоритмом локального пошуку жадібного методу, який повторюється визначеними способами на кожній ітерації.

Експеримент 3

Алгоритм локального пошуку плавно сходиться після 210 вимірювань, тоді як жадібний підхід сходиться після 250 і 270 вимірювань, відповідно, відповідно до результатів збіжності двох алгоритмів. Іншими словами, алгоритм локального пошуку здатний визначити оптимальне рішення за меншу кількість раундів, ніж альфа-алгоритм.

Висновки до розділу

У третьому розділі розглядається програмне забезпечення та технічне забезпечення системи планування IT-ресурсів. Розкривається структура програмного додатку та пояснюється методика використання програмного забезпечення.

Дослідження включало низку експериментів, і результати цих випробувань були надані. Жадібний алгоритм і алгоритм локального пошуку є евристичними методами вирішення. Тривалість виконання методів для задач великої розмірності також істотно відрізняється: жадібний алгоритм працює в 9000 разів швидше, ніж підхід локального пошуку. Це тому, що метод локального пошуку має певну кількість ітерацій для повторення перестановок, що призводить до величезної кількості операцій для побудови розкладу для 100 завдань, а жадібний підхід будує розклад для кожного виконавця для 100 завдань і на цьому зупиняється. Хоча результати алгоритму локального пошуку показують результати з розбіжністю на 3-4% від значення цільової функції, використання жадібного підходу не гарантує кращих результатів. На основі зібраних результатів дослідження наведено тривимірні графіки.

Керівники та окремі користувачі можуть використовувати цю програму для автоматизації планування ресурсів IT-проекту, що пришвидшить обчислення та дозволить відстежувати, як змінюється кожен параметр.

ВИСНОВКИ

В якості основи для системи підтримки прийняття рішень, яка надає керівнику проекту своєчасну інформацію про стан проекту та підтримує прийняття рішень щодо управління ресурсами та календарного планування, у магістерській роботі розроблено взаємопов'язаний комплекс моделей, методів і алгоритмів.

У роботі виконуються наступні завдання:

– досліджено та визнано переваги та недоліки сучасних методів управління проектами, програмного забезпечення, що спеціалізується на автоматизації процесів управління проектами та їх підтримки, алгоритмів планування проектів та використання інструментів інтелектуалізації в управлінні ресурсами проекту;

– створено систему управління ресурсами проекту, яка дозволяє призначати завдання проекту ресурсам, створювати план проекту з урахуванням ризиків, відстежувати хід проекту та приймати управлінські рішення щодо заміни ресурсів за потреби.

Розроблена інформаційна система планування ресурсів IT-проектів дозволяє працівникам працювати ефективніше, забезпечує легкість і зручність розрахунків, знижує ймовірність помилок у плануванні ресурсів, підвищує рівень достовірності інформації, знижує трудовитрати на обробку інформації.

Керівники та звичайні користувачі можуть використовувати цей інструмент для автоматичного планування ресурсів для IT-проектів.

Проведене дослідження привело до висновку, що постачальники та клієнти є ключовими факторами, які впливають на конкуренцію в галузі. Крім того, зростання ринку збільшило як загрозу альтернативних товарів, так і рівень конкуренції серед поточних конкурентів ринку.

У результаті праці над магістерською роботою було зроблено:

– на основі аналізу існуючих рішень виявлено, що існуючі системи не здатні встановлювати загальні часові рамки для всього проекту під час планування з урахуванням індивідуальних показників кожного виконавця завдання, але вони дають можливість розподіляти ресурси та побудувати модель реалізації даного

проекту з урахуванням ресурсних обмежень;

– створено програмний продукт, який забезпечує формування плану розподілу обов'язків між учасниками на основі розроблених ефективних алгоритмів планування ресурсів, які дозволяють створювати оптимальні або наближені до ідеальних плани. Було досліджено ефективність розроблених стратегій. Для цього було проведено декілька тестів, які базувалися на результатах жадібного алгоритму та алгоритму локального пошуку в різних розмірах проблеми.

– виявлено, що жадібний алгоритм вимагає більше ітерацій, щоб знайти кращу відповідь, ніж алгоритм локального пошуку. Тривалість виконання методів для задач великої розмірності також істотно відрізняється: жадібний алгоритм працює в 9000 разів швидше, ніж підхід локального пошуку. Це тому, що метод локального пошуку має певну кількість ітерацій для повторення перестановок, що призводить до величезної кількості операцій для побудови розкладу для 100 завдань, а жадібний підхід будує розклад для кожного виконавця для 100 завдань і на цьому зупиняється. Хоча результати алгоритму локального пошуку показують результати з розбіжністю на 3-4% від значення цільової функції, використання жадібного підходу не гарантує кращих результатів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

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

2. Management and Project Management: Theory and Practice / URL: https://sgv.in.ua/off-lifaq/25-suchasni-metodi-upravlinnya-proektami (дата звернення: 11.03.2023).

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

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

5. Морозов В. В. Формування, управління та розвиток команди проєкту /– Київ: Таксон 4, 2009. – 461с.

6. Самсонов А. К. Управління проєктом при створенні мобільного додатку / InProject, Program, Portfolio Management, 2017 – с. 79-81.

7. Семеріков С. О. Мобільне програмне забезпечення / Програмне забезпечення, 2010 – 156 с.

8. Kyriy, V., Sheiko, I., Petrova, R. Optimization of Management Information Support as a Basis for Organizational Transformations at an Enterprise // Periodicals of Engineering and Natural Sciences. 2019. V.7. N. 2. pp. 679-689

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

10. Шведа Н. М. Система управління проєктами в Україні / Збірник тез доповідей IVМіжнародної науково-технічної конференції молодих учених та студентів «Актуальні задачі сучасних технологій», 25-26 листопад 2015 – Т. : ТНТУ, 2015. – Том 2. – с. 246-247.

11. Шишкіна М. П. Якість програмних забезпечення для мобільних пристроїв / Науковий часопис Національного педагогічного університету імені МП Драгоманова, 2017 – 150 с.

12. Переваги використання інформаційно-комунікаційних технологій в Україні / URL: http://ena.lp.edu.ua:8080/ bitstream/ntb/13914/1/67_461-467_Vis_727_Menegment.pdf (дата звернення: 12.03.2023).

13. Використання IT в управлінні проєктами / URL: http://dspace.mnau.edu.ua/jspui/bitstream/123456789/6706/1/studentresearchjournal162-29.pdf. (дата звернення: 13.03.2023).

14. Багатовимірне подання даних для управління IT-проєктами / URL: http://ena.lp.edu.ua/bitstream/ntb/29918/1/37_387-394.pdf (дата звернення: 14.03.2023).

15. Розробка моделі розподілу IT-проєктів на підприємствах галузі інформаційних технологій / URL: http://mdtopu.com.ua/files/download/mdt2.3.2018-16.09_1.pdf (дата звернення: 14.03.2023).

16. Методологія представлення соціальних проєктів IT-індустрії / URL: https://economics.net.ua/files/science/ek_kiber/2018/50.pdf (дата звернення: 15.03.2023).

17. Комп'ютерна алгебра та інформаційні технології / URL: http://confit.onu.edu.ua/content/CAIT-Odessa-2018.pdf (дата звернення: 15.03.2023).

18. Використання інформаційних технологій в управлінні проєктами підприємств / URL: http://nbuv.gov.ua/UJRN/Urss_2013_13_4. (дата звернення: 15.03.2023).

19. 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: 16.03.2023).

20. A competitive genetic algorithm for resource-constrained project scheduling. 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: 17.03.2023).

21. An iterative scheduling technique for resource-constrained project scheduling / URL: https://www.sciencedirect.com/science/article/abs/pii/0377221792903209 (дата звернення: 17.03.2023).

22. 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.

23. Kachko, O., Makutonina, L., Akolzina, O. NTRU Similar Algorithm Optimization for Asymmetric Encryption with the «Overstretched Parameters» // 4th International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), 2017, pp. 330-333, doi: 10.1109/INFOCOMMST.2017.8246409

24. Жадібні алгоритми / URL: https://dl.sumdu.edu.ua/textbooks/95351/522264/index.html (дата звернення: 18.03.2023).

25. Agilemanifesto / URL: http://agilemanifesto.org/iso/ru/manifesto.html (дата звернення: 19.03.2023).

26. Andrei Cristian Spataru. “Agile Development Methods for Mobile Applications”. Master of Science Thesis submitted to Computer Science School of Informatics, University of Edinburgh. 2010.

27. Atlassian JIRA Software / URL: https://atlassian.com/software/jira (дата звернення: 20.03.2023).

28. Book on a Software Engineering Model. “A Software Engineering Model for Mobile App Development” / URL: http://static.bada.com/contents/ blog/20110616/20-introduction_to_bada_app03.pdf. (дата звернення: 20.03.2023).

29. Chen G. and Kotz D., “A Survey of Context-Aware Mobile Computing Research,” Hanover, NH, USA, Tech. Rep., 2000.

30. Cohn M. User Stories Applied / Publisher: Kindle Edition, 2017 – 303 p.

31. Combining Challenge-Based Learning and ScrumFramework for Mobile Application Development Alan R. Santos, ITiCSE '15 Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education.

32. Craig Larman Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum/ Publisher: Kindle Edition, 2015 – 387 p.

33. Da, C. T. F. V., Dantas, V. L. L. Andrade, & R. M. (2011). SLeSS: A scrum and lean six sigma integration approach for the development of sofware customization for mobile phones. Proceedings of the 25th 1263 Brazilian Symposium on Software Engineering (pp. 283-292).

34. David J. S., McCarthy W. E., and Sommer B. S., "Agility – the Key to Survival of the Fittest in the Software Market", Communication of the ACM, Vol. 46, No. 5, pp.65-69. 2003.

35. Di Bella, E., Sillitti, A., Succi, G. A multivariate classification of open source developers. Inf. Sciences 221, pp. 72-83. 2013.

36. EBay Enterprise Business Taken Private, Broken Up / URL: https://fortune.com/2015/11/03/ebay-enterprise-sold/ (дата звернення: 21.03.2023).

37. Flora, H. K., Wang X., and Chande S. V., "An Investigation into Mobile Application Development Processes: Challenges and Best Practices," Int. J. Mod. Educ. Comput. Sci., vol. 6, no. 6, pp. 1–9, 2014.

38. Green M. Scrum: Novice to Ninja: Methods for Agile, Powerful Development / Publisher: Kindle Edition, 2015 – 278 p.

39. Häkkilä, J., Mäntyjärvi, J.; Developing design guidelines for context-aware mobile applications. 3rd Intl. Conf. on Mobile Technology, Applications and Systems, pp. 1-7. 2006.

40. O. Mazurova, O. Samantsov, O. Topchii, M. Shirokopetleva, A Study of Optimization Models for Creation of Artificial Intelligence for the Computer Game in the Tower Defense Genre, 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), 2020, pp. 491-496, doi: 10.1109/PICST51311.2020.9468057.

41. Harvard Business Review / URL: https://hbr.org/2016/04/the-secrethistory-of-agile-innovation (дата звернення: 22.03.2023).

42. Ken Schwaber Agile Project Management with Scrum/ Ken Schwaber – Kindle Edition, 2016 – 589 p.

43. Kniberg H. Scrum and Kanban: making the most of both/ Kniberg H. – Kindle Edition, 2016 – 243 p.

44. Kolb, D. and Fry, R..Toward an Applied Theory ofExperiential Learning. C. Cooper (ed.). Theories ofGroup Process, London: John Wiley., London, 1975.

45. Kylmäkoski, R.; RaPiD7: A collaborative method for the planning activities in software engineering: Industrial experiment. ISBN 952-15-1517-1. Nokia. 2005.

46. La, H.J., Lee, H.J., Kim, S.D.; An Efficiency-centric design methodology for mobile application architectures. 7Th International Conference on Wireless and Mobile Computing, Networking and Communications, pp. 272-279. 2011.