Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Розробка елементів системи автоматизованого проєктування технологічного розділу проєктів вагоноремонтних підприємств»
за освітньою програмою: «12 Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1912»

_____ /Богдан САВЧЕНКО/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Вадим ГОРЯЧКІН/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /Світлана ВОЛКОВА/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems» Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis

on the topic: « Development of elements of an automated design system for the technological section of reilcar repair enterprises.»
according to educational curriculum «Software engineering»in the Speciality: «121 Software engineering»

Done by the student of the group PZ1912: /Bohdan SAVCHENKO/

Scientific Supervisor: /Vadym HORIACHKIN/

Normative controller: /Vadym HORIACHKIN/

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____/Вадим ГОРЯЧКІН/
(підпис)
Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра студенту Савченку Богдану Станіславовичу
1. Тема роботи: «Розробка елементів системи автоматизованого проєктування технологічного розділу проєктів вагоноремонтних підприємств»
Керівник роботи: Горячкін Вадим Миколайович, доцент
затверджені наказом № 1209 ст від 07.12.2022
2. Строк подання студентом роботи: __.__.202_ р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1			
2			
3			
4			
5			
6			
7	Подання кваліфікаційної роботи до кафедри		
8	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії		

Студент _____ Богдан САВЧЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Вадим ГОРЯЧКІН
(підпис) (Ім'я ПРІЗВИЩЕ)РЕФЕРАТ

РЕФЕРАТ

Пояснювальна записка складається з 8 розділів:

– вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;

– збір вимог до програмного забезпечення – у цьому розділі описуються аналоги програми та література по даній предметній області. Складається з 11 сторінок;

– зовнішнє і внутрішнє проектування – у цьому розділі проведений огляд вхідних і вихідних даних, формалізація задачі, розробка фізичного проекту, приводиться опис об’єктно-орієнтованого проектування, проектування інтерфейсу користувача, ескізи форм, аналіз проекту, проектування динаміки системи. Складається з 14 сторінок;

– розробка програми – включає в себе вибір мови програмування та розробку алгоритмів необхідних для реалізації проекту. Складається з 3 сторінок;

– тестування та налагодження – включає в себе вибір стратегії тестування, опис тестів методом «білої» скриньки. Також аналіз помилок їх вплив на систему та вирішення проблеми. Складається з 10 сторінок;

– висновки. Складається з 1 сторінки;

– список літератури – включає в себе бібліографічний список використаної літератури. Складає 2 сторінки;

– додатки – містить технічне завдання і робочий проект.

Кількість таблиць: 8 штук.

Кількість рисунків: 13 штук.

Ключові слова: вагон, персонал, ремонт, модуль, економічний, параметри.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Ремонтна група вагонів - поняття, яке означає певну кількість вагонів, які поступають разом на ремонт або знаходяться в одному місці з певною метою або для певного завдання.

Види ремонту:

К (Капітальний ремонт) - повна або значна реконструкція вагона з метою відновлення його працездатності та продовження терміну його служби.

М (Модернізація) - процес вдосконалення або оновлення існуючого вагону з метою покращення його функціональності, ефективності, безпеки або комфорту.

У (Утилізація) - остаточне припинення експлуатації вагону. Утилізація може включати: розбирання, розчавлення або переробку вагона, щоб вилучити корисні матеріали та складові частини.

ВСТУП

Ця дипломна робота присвячена розробці елементів системи автоматизованого проєктування технологічного розділу проєктів вагоноремонтних підприємств. У сучасних умовах, коли ефективність та точність проєктування мають велике значення для успішної роботи, автоматизація цього процесу є надзвичайно важливою.

Розробка системи автоматизованого проєктування технологічного розділу дозволить вдосконалити та прискорити процес проєктування, забезпечити високу точність та надійність результатів, а також знизити витрати робочого часу та ресурсів. Застосування автоматизованої системи дозволить ефективно планувати та координувати роботу в технологічному розділі, спростити процес прийняття рішень та забезпечити високу якість проєктів.

Основна мета цієї роботи полягає в розробці ключових елементів системи автоматизованого проєктування, таких як інтерфейс користувача, база даних, модуль управління та алгоритми оптимізації. При цьому враховується специфіка вагоноремонтних підприємств та їх потреби, забезпечуючи функціональність та зручність використання системи.

Розробка елементів автоматизованої системи проєктування технологічного розділу проєктів вагоноремонтних підприємств має великий потенціал для підвищення ефективності роботи та покращення якості виконуваних проектів. Окрім того, вона може сприяти зменшенню затрат і підвищенню конкурентоспроможності підприємств вагоноремонтної галузі.

У даній роботі будуть вивчені сучасні підходи та методики автоматизованого проєктування технологічного розділу проєктів вагоноремонтних підприємств. Будуть проведені дослідження для визначення оптимальних рішень щодо реалізації системи та виконання аналізу її переваг та можливих викликів. Результати цієї роботи будуть корисними для практиків та фахівців у галузі вагоноремонту, а також внесуть свій внесок у розвиток сучасних технологій у сфері виробництва та ремонту залізничного складу.

1 ЗБІР ТА АНАЛІЗ ВИМОГ

Аналіз вимог пов’язаний з цілями й потребами замовника. Під час аналізу вимог слід визначити:

- спосіб використання ПЗ;
- вхідні та вихідні дані;
- спосіб зображення інформації;
- наскільки продуктивним повинен бути додаток;
- надійність додатка;
- інструменти для реалізації проекта.

Тобто результатом виконання збору та аналізу вимог, повинно бути вирішено, що повинен виконувати програмний продукт, як це повинно виглядати та за допомогою яких інструментів виконано.

1.1 Огляд аналогів

В ході дослідження виявлено лише один аналог для моделювання гнучких потокових виробництв для ремонту рухомого складу – **Conveyers Model**.

Інформацію щодо функціонала додатка було надано замовником. Далі розглянемо отриману інформацію.

Conveyers Model – перша версія додатку, яка була розроблена з метою моделювання гнучких потокових виробництв для ремонту рухомого складу. Але він дозволяє лише прорахувати коефіцієнти використання й навантаження модулів.

На рис. 1.1. зображена форма для введення конфігурації моделі, а саме: обрати які методу будуть використані, обрати кількість позицій, кількість модулів на цих позиціях, час переміщення до них та коефіцієнт надійності(передбачається можливість виходу зі строю). Також є можливість розгляду моделі з більше ніж одним трансбордером і щоб вони обслуговували різні позиції.

На рис. 1.2 зображені результати обчислення коефіцієнтів використання та навантаження для усіх модулів.

А на вкладці “Программа ремонта” (рис 1.3) є можливість переглянути номери вагону коли він надійшов на ремонт та під яким номером був відремонтований, тип вагону, вид та час ремонту, час простою та знаходження у цеху.

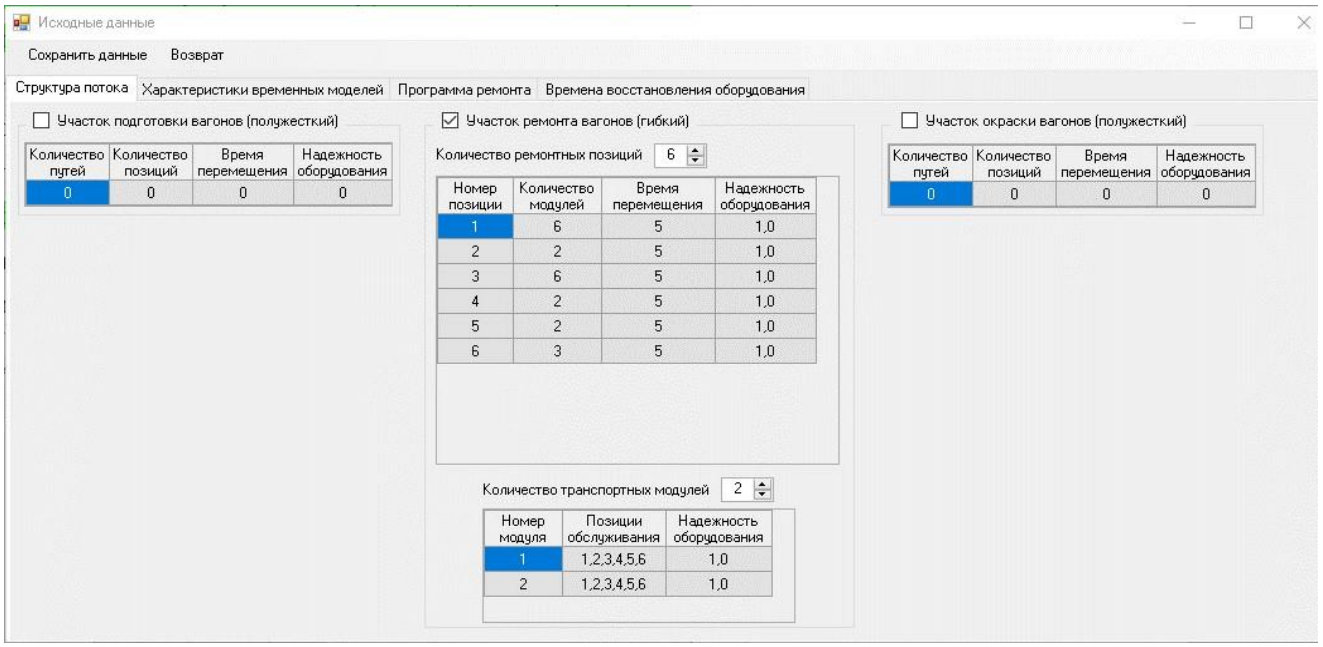

Рисунок 1.1 – Введення значень

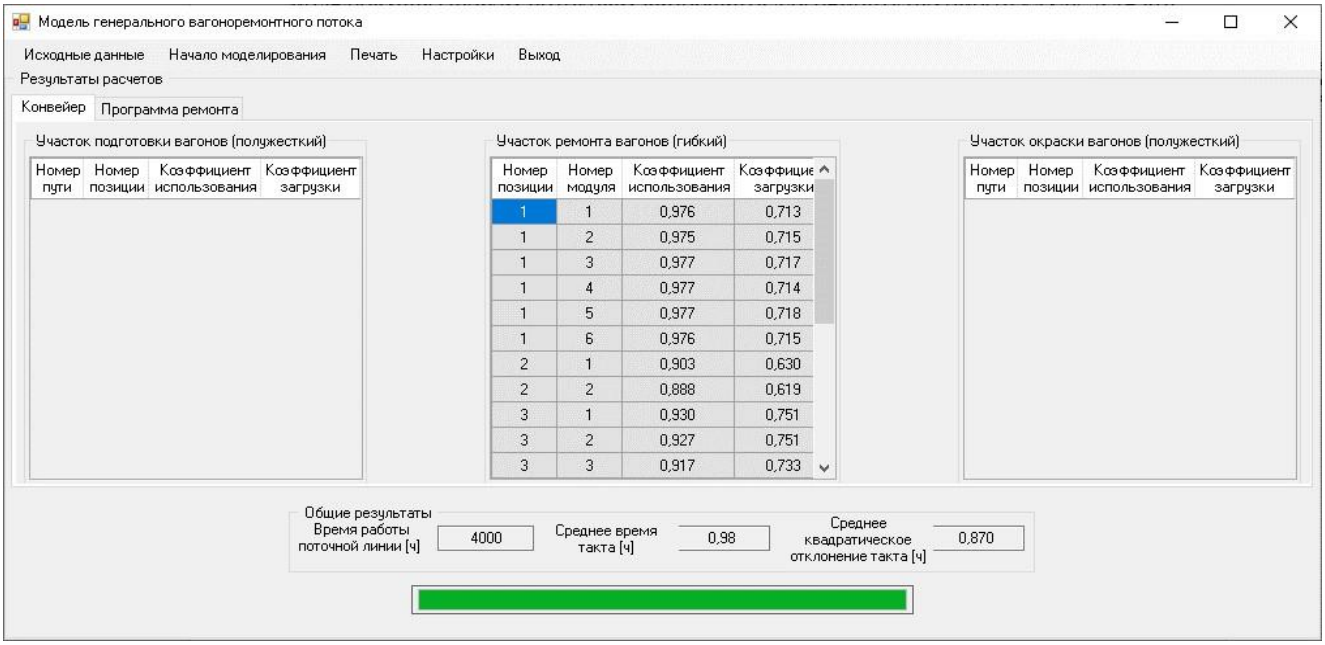

Рисунок 1.2 – Результат розрахунків для конвеєру

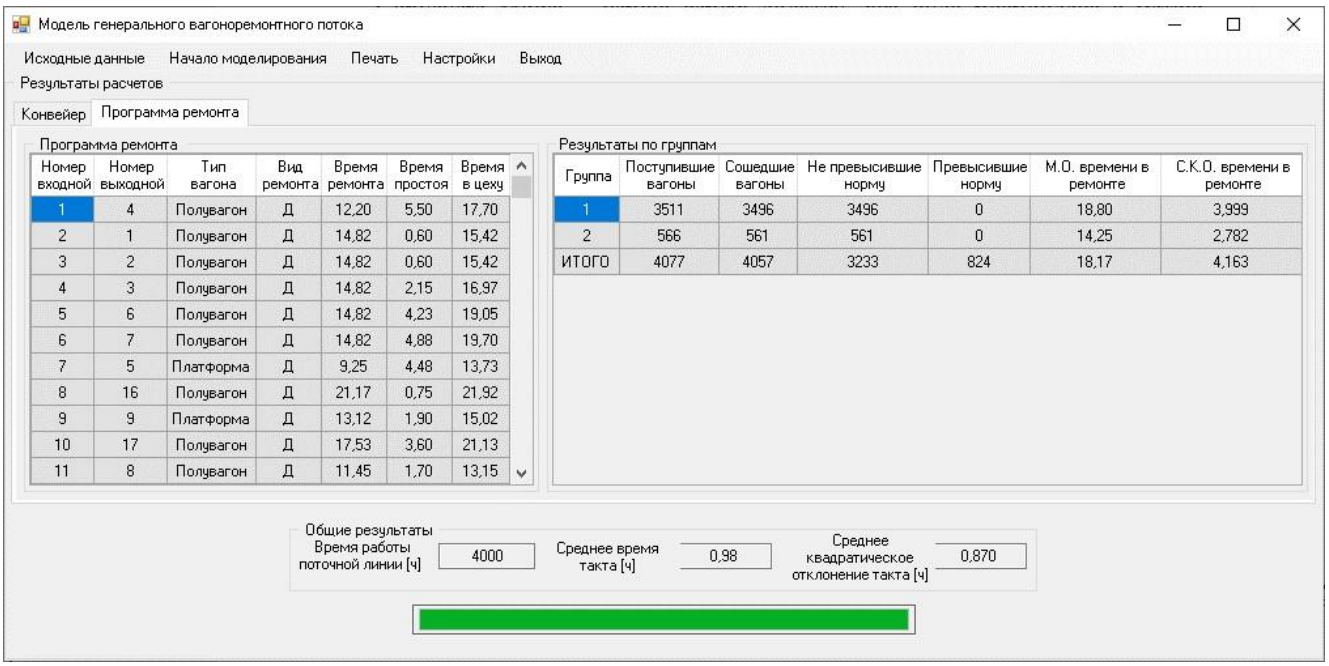

Рисунок 1.3 – Результати розрахунків програми ремонту

1.2 Огляд літератури

Далі розглянемо як відбуваються розрахунки та які формули лежать в основі програми.

1.2.1 Ремонт вагонів та персонал

Основні виробничі робітники

$$\text{Pop} = \frac{\sum_{i=1}^{i=a} N_i Q_i}{F}$$

де a – кількість ремонтних груп

Ni – кількість вагонів в і-тій групі

Qi – трудомісткість ремонту вагонів і-тої групи

Трудомісткість господарських робіт та робіт з ремонту обладнання та інструменту приймається 12% (можна задавати власне число) від кількості основних виробничих робітників.

Таблиця 1 – Професії

Найменування професії	%
Слюсарі з ремонту обладнання	**41,5**
Слюсарі з ремонту інструменту	**6,5**
Слюсарі-електрики	**11,5**
Токарі	**5,0**
Свердлувальники	**1,4**
Строгальники-фрезерувальники	**2,2**
Електрогазозварювальники	**2,5**
Ковалі	**3,1**
Маляри	**4,0**
Столяри	**8,5**
Підсобні (транспортні) робітники	**13,8**
Разом	**100**

Кількість виробничих робітників, зайнятих на господарських роботах.

Ргр = 12%

Загальна кількість виробничих робітників

Рвр = Рор + Ргр

Допоміжні робітники

Рдоп = 16% від Рвр

Інженерно-технічні робітники

Рітр = 6% від Рвр

Розрахунково-конторський персонал

Рркп = 2% від Рвр

Молодший обслуговуючий персонал

Рмоп = 2% від Рвр

1.2.2 Розрахунок рівня автоматизації та виробництва

Автоматизація виробництва розглядається як ступінь заміщення або виключення робочих функцій людини машинами у виробничих процесах.

Машини складаються з сукупності різних ланок, двигуна, передавального механізму, машини-знаряддя, контрольно-керуючого пристрою, системи автоматичного управління і змінювати програму пристрої управління сукупністю машин.

Шляхом зіставлення кількості, наявних в машині кілець з максимально можливою їх кількістю оцінюється технічний рівень машини з точки зору оснащення функцій людини в виробничому процесі.

Відповідно до методичних вказівок по оцінці ступеня і рівня автоматизації, рівень автоматизації виробництва визначається за формулою

$$K_a = \frac{\sum_{Z=3,5}^{Z=5} Zm_z}{Z_{max}\sum_{Z=0}^{Z=5} m_Z} \cdot 100\%$$

де Z - кількість кілець в машині;

$m_z = m_в K_з$ - кількість використовуваних машин;

$m_в$ - кількість встановлених машин даного типу;

$K_з$ - коефіцієнт завантаження машини;

z_{max} - максимальне число кілець в машині;

Загальна кількість робочих місць:

$$\sum_{Z=0}^{z=5} m_z = m_0 + \sum_{z=1}^{z=5} m_z$$

m_0 - кількість робочих місць ручної праці:

$$m_0 = \frac{R}{k_{зм}};$$

$k_{зм}$ - змінність роботи підприємства ;

R - явочна чисельність робітників ручної праці.

Рівень автоматизації та механізації виробництва визначається за формулою:

$$К_{ма} = \frac{\sum_{Z=1}^{Z=5} Zm_Z}{Z_{max} \sum_{Z=0}^{Z=5} m_Z} \cdot 100\%$$

Кількість кілець машин визначається з таких міркувань: ручні знаряддя праці розглядається умовно, як машини нульовий залежності, Z = 0; машини ручної дії, Z = 1; механізовано-ручні машини, Z = 2; механізовані машини, Z = 3; машини-напівавтомати, Z = 3,5; машини-автомати, Z = 4; машини гнучких виробничих модулів, Z = 4,5; машини гнучких автоматизованих ліній, Z = 4,75; машини гнучких автоматизованих дільниць і підприємств, Z = 5.

Класифікація машин по кільцевості приведена в табл. 1.

Механізовано-ручна машина - машина, яка використовує для управління енергію людей, в якій основні рухи здійснюються за рахунок енергії неживої природи, а допоміжні - за допомогою ручних пристосувань.

Механізована машина - машина виконує всі рухи за рахунок енергії неживої природи, виключаючи управління.

Машина-напівавтомат - машина, яка функціонує дискретно і керована за заданим алгоритмом з використанням енергії неживої природи, за участю людей в періодичному включенні машини (завантаження і вивантаження);

Машина-автомат - машина, яка функціонує безперервно і керована за заданим алгоритмом з використанням енергії неживої природи, без безпосередньої участі людей.

Гнучкий виробничий модуль - одиниця технологічного обладнання для виробництва виробів довільної номенклатури в установлених межах значень їх характеристик з програмним керуванням, автономно-функціонуюча, автоматично здійснює всі функції, пов'язані з їх виготовленням, має можливість вбудовування в гнучку виробничу систему.

Гнучка автоматизована лінія - гнучка виробнича система, в якій технологічне устаткування розташоване у прийнятій послідовності технологічних операцій.

Гнучка автоматизована дільниця - гнучка виробнича система, яка функціонує за технологічним маршрутом, в якому передбачена можливість зміни послідовності використання технологічного устаткування.

База даних технологічного обладнання підприємства

Таблиця 2 – Обладнання

Найменування технологічного обладнання	Кількість ланцюгів Z
Основні процеси	
Машина мийна	3
Кантувач кузова	3
Вагоноремонтна машина	3
Правильна машина	3
Візок-підйомник	2

Гайковерт	2
Газозварювальний апарат	2
Зварювальний трансформатор	2
Стенд випробування гальм	2
Машина фарбування та сушіння вагонів	3
Верстат-напівавтомат обробки колісних пар	3,5
Дефектоскоп	3
Токарний верстат	3
Стругальний верстат	3
Фрезерний верстат	3
Деревообробний станок	3
Циркулярно-маятникова пилка	2
Електродриль	2
Верстат обробки гальмівних черевиків	3
Автомат виготовлення шплінтів	4
Ковальсько-пресове обладнання	3
Прес правки кришок люків та дверей	2
Зварювальний напівавтомат	3,5
Зварювальний автомат	4
Електропіч	3
Ковальський горн	2
Допоміжні процеси	
Електрокар	3
Кран мостовий однобалковий	3
Кран козловий	3
Верстат універсальний металорізальний	3
Заточувальний верстат	3
Кривошипний механізм	2

Машина свердлильна	2
Автонавантажувач	3
Зварювальний напівавтомат	3,5
Конвеєр	3
Кран бруківка	3
Зварювальний трансформатор	2
Трансбордерний агрегат	3
Транспортний візок	1

При розрахунку автоматизації враховується тільки те обладнання в якого кількість ланцюгів більше 3.

Рівень автоматизації та автоматизації та механізації розраховується окремо для основних процесів та допоміжних. А потім ще для всього підприємства.

Ступінь автоматизації виробництва:

$$\rho_{\text{A}} = \frac{\sum_{z=3,5}^{5} n_z \, m_z}{n_{max} \sum_{z=0}^{5} m_z}$$

Таблиця 2.2 – Ступінь автоматизації

z	n_z
0	0,1
1	0,2
2	0,4
2,25	0,5
3	1,0
3,25	1,1
3,5	1,4
3,75	1,75

4,0	2,25
4,25	2,8
4,5	3,5
4,75	4,2
z_{max} =5	n_{max} =5

n_z – продуктивність праці робітників з урахуванням витрат часу на налагодження обладнання залежно від кількості кілець машини.

1.2.3 Економічні параметри

Чисельне значення зведенних витрат за ремонт одного вагона визначається за формулою

$$В = \frac{(С+Е_н К)}{N}$$

де **С** - річна собівартість ремонту вагонів (поточні витрати);

К - капітальні вкладення у виробничі фонди (одноразові витрати);

$Е_н$ - нормативний коефіцієнт економічної ефективності капітальних вкладень;

N - річна програма ремонту вагонів

Якщо враховувати, що на період ремонту вагони вилучаються з робочого парку, то сумарні наведені витрати, що характеризують загальні витрати на ремонт одного вагона, визначаються таким чином

$$Всум = В + Внрп$$

де **Внрп** - витрати, пов'язані з перебуванням одного вагона в неробочому парку, поки він знаходиться в ремонті

$$Внрп = t_{нрп} \cdot e_{пр}$$

де **$t_{нрп}$** - середня тривалість перебування вагона у неробочому парку, годин;

$e_{пр}$ - витратна ставка однієї години простою вагона, грн

Річна собівартість ремонту вагонів складається з різних статей витрат і може бути визначена таким чином

С=Взп + Воп + Ввб + Воб + Все + Всп + Впро + Врі + Вмзч + Вптб+Аоб+Аб

де **Взп** – загальна заробітна плата виробничих робітників

Воп- витрати на опалення будівлі депо

Ввб- витрати на вентиляцію будівлі

Воб - витрати на освітлення будівлі

Все - витрати на силову енергію

Всп - витрати на стиснене повітря

Впро - витрати на поточний ремонт обладнання та оснащення

Врі - витрати на ремонт інструменту

Вмзч - витрати на матеріали та запасні частини

Вптб - витрати на поточний ремонт будівель

Аоб - щорічні амортизаційні відрахування від вартості обладнання та оснащення

Аб - щорічні амортизаційні відрахування від вартості будівель

Одноразові витрати складаються з вартості будівельно-монтажних робіт та вартості обладнання

К = Кбмр + Кто

Кбмр - вартость будівельно-монтажних робіт;

Кто - вартість технологічного обладнання.

Витрати на опалення будівлі депо

Воп= (Роп · Vб · Кзв)/ 10

Роп - питомі річні витрати на опалення 10 м^3 будівельного об'єму будівлі

Кзв - поправочний коефіцієнт для врахування зміни вартості теплової енергії

Vб - об'єм будівлі, м^3

Річні витрати на вентиляцію

Ввб= (Рв · Vб · Кзв)/ 10

Рв - питомі річні витрати на вентиляцію 10 м^3 будівельного об'єму будівлі

Річні витрати на електричне освітлення будівлі

$$\textbf{Воб= (Ро} \cdot \textbf{Sб)/ 10}$$

Ро - питомі річні витрати на електричне освітлення фонарної будівлі, віднесені на 10 м^2 розгорнутої площі;

Sб - площа будівді

Річні амортизаційні відрахування на повне відновлення та капітальний ремонт будівлі

$$\textbf{Аб = (Vб} \cdot \textbf{В1м}^3\textbf{б} \cdot \textbf{(Нпв + Нкр))/100}$$

де В1м^{3}б- вартість 1 м^3 будівельного об'єму будівлі

Нпв- норма відрахувань на повне відновлення будівлі

Нкр- норма відрахувань на капітальний ремонт будівлі

Річні відрахування на ремонт будинку

$$\textbf{Вптб=(Vб} \cdot \textbf{В1м}^3\textbf{б} \cdot \textbf{Кпр) / 100}$$

де **Кпр** - показник річного обсягу поточних ремонтів від вартості будівлі

1.3 Постановка задачі

Необхідно розробити програмний додаток з графічним інтерфейсом, що буде обчислювати та автоматизовувати технологічні та економічні процеси пов’язані з вагоноремонтними підприємствами.

Висновки до розділу 1

Під час збору вимог було оглянуто й обрано необхідну схему виконання моделювання, визначено тип додатка та затверджено все, що необхідно прийняти на вхід та зобразити на екрані.

Найбільш інформативними джерелами інформації був огляд літератури, саме за цими методами приймалося рішення, щодо того, який додаток повинен бути та який функціонал він повинен виконувати.

2 ЗОВНІШНЄ І ВНУТРІШНЄ ПРОЕКТУВАННЯ

2.1 Зовнішнє проектування

2.1.1 Функціональне призначення

Розроблений програмний продукт має на меті реалізацію елементів системи автоматизованого проєктування технологічного розділу проекту на вагоноремонтних підприємствах на основі введених даних. Він надає зручний інструментарій для створення, модифікації та аналізу технологічних процесів, пов'язаних з ремонтом та сервісним обслуговуванням в рамках депо.

2.1.2 Експлуатаційне призначення

Програмний продукт має на меті надати працівникам, викладачам, студентам та абітурієнтам можливість наочно демонструвати взаємодію різних елементів системи вагоноремонту за допомогою введеної моделі. Це сприятиме кращому розумінню студентами процесів, пов'язаних з технологічним розділом проєктів на вагоноремонтних підприємствах.

2.1.3 Вимоги до функціональних характеристик

Повинна бути можливість задавати:

- кількість груп ремонту
- кількість робочих змін
- кількість робочих днів в році
- кількість обладнання та його коефіцієнт навантаження
- економічні параметри

Програмний продукт повинен розраховувати потрібні нам значення в залежності від обраного модуля, а саме ремонт вагонів, персонал, обладнання та економічні параметри. Процес розрахунків повинен супроводжуватися збереженням даних.

На формах, що відповідають кожному із модулів потрібно відобразити наступні елементи:

1) Форма “Ремонт вагонів”
   - основна таблиця для введення груп ремонту
   - TextBox-и для введення додаткових даних
2) Форма “Персонал”

- два елементи типу DataGridView, для відображення потрібних професій
- TextBox-и для відображення загальної кількості працівників на кож

2.1.4 Вхідні дані

Вхідними даними є:

1. Форма “Ремонт вагонів”:

- GroupNumber – зберігає інформацію про групи ремонту, значення отримуються з елементу GroupQty (типу NumericUpDown);
- CarType – обираємо тип вагону, що надійшов на обслуговування, змінна типу String;
- KindRepair – обираємо тип сервісу для вагону, змінна типу String;
- CarQty – вводимо кількість вагонів певного типу, що надійшли на обслуговування, змінна типу Int;1
- HardComplains – вводимо значення трудомісткості певного ремонту, змінна типу Int

2) Форма “Персонал”:

Даний модуль має лише вихідні дані на основі даних з модуля “Ремонт вагонів”.

3) Форма “Обладнання”:

- MainProcces - клас типу List, що зберігає в собі дані про назву та кількість ланцюгів основного обладнання, тип даних String та Double відповідно;
- supportProcces - клас типу List, що зберігає в собі дані про назву та кількість ланцюгів допоміжного добладнання, тип даних String та Double відповідно;

4. Форма “Економічні параметри”

- для кожного з видів економічних витрат задаються свої змінні, це виконується за допомогою інтсрументу List, тип змінних в списку Double

2.1.5 Вихідні дані

Вихідними даними є:

1) Форма “Ремонт вагонів”:

- відображаються нові групи ремонту

2) Форма “Персонал”:

- кількість робітників кожної професії, їхня відсоткова частка від загального числа працівників
- кількість різних типів робітників (виробничі, господарські, допоміжні, інженерні, конторські, молодший обслуговуючий персонал)

3) Форма “Обладнання”:

- кількість використовуваного обладнання для кожного з типів процесів
- рівні атоматизації та механізації виробництва

4) Форма “Економічні параметри”

- для кожного з видів економічних витрат отримуються дані, на основі раніше введених значень в поточній формі Виконаємо специфікацію функціональних вимог у вигляді прецедентів (рис. 2.1)

Рисунок 2.1 – Діаграма прецедентів

2.1.6 Опис зовнішнього інформаційного середовища

Перемикання кольору теми виконується на почтаковому екрані, в меню налаштувань;

Зберігання стану в окремий файл виконується в ручному режимі натисканням кнопки “Зберегти” в робочому середовищі програми;

Дані string, int, double вводяться з клавіатури у таблицю або TextBox-и;

За необхідності додати нову групу ремонту слід за допомогою миші(ЛКМ) натиснути на кнопку “↑” на формі, а якщо видалити – кнопку “↓“ відповідно;

Кількість вагонів вводиться з клавіатури у контрольоване поле вводу;

Для функціонування ПЗ необхідно мати: встановлену операційну систему Windows 10 чи вище, пакет SDK інструментів для .NET додатків, пакет Microsoft Visual 2015-2022 Redistributable.

2.2 Внутрішнє проектування

2.2.1 Проектування архітектури системи

2.2.1.1 Сутності програми

1. **Форма “MainForm”:**

MainForm - клас, що представляє головну форму програми. Він успадковує клас Form з простору імен System.Windows.Forms і містить різні методи та події, пов'язані з функціональністю форми.

Крім того, код використовує такі простори імен:

- SAPR.Forms - містить інші форми, зокрема RepairWagons_Form.
- System.Windows.Forms - містить класи для розробки Windows-додатків з графічним інтерфейсом користувача.
- SAPR.Models - містить моделі даних, зокрема EquipmentGridData.

У класі MainForm оголошено кілька подій, таких як NewCalculation_Btn_Click, GoOnCalculation_Btn_Click, Exit_Btn_Click, LightThemeToolStripMenuItem_Click, DarkThemeToolStripMenuItem_Click, Label1_ForeColorChanged, а також ряд методів, що виконують різні дії при взаємодії з елементами форми або зміні налаштувань.

2. **Форма “Ремонт вагонів”:**

У даній формі існує одна сутність, яка називається "RepairGroup" (Група ремонту). Вона містить наступні властивості:

- GroupNumber (номер групи) - ціле число
- CarType (тип автомобіля) - рядок
- KindRepair (тип ремонту) - рядок
- CarQty (кількість автомобілів) - ціле число
- HardComplains (складні скарги) - ціле число

Також у класі є приватний метод "GetXMLAtributes", який повертає словник з атрибутами об'єкта RepairGroup для подальшого використання при створенні XML-об'єкта. Клас містить такі методи:

- AddAsXMLObject (Додати як XML-об'єкт): Додає об'єкт RepairGroup до заданого XElement (елемента XML).

- AddAsGridViewRow (Додати як рядок DataGridView): Додає об'єкт RepairGroup як новий рядок до заданого DataGridView (таблиці з графічним інтерфейсом).
- InitializeFromGridViewRow (Ініціалізувати з рядка DataGridView): Ініціалізує властивості об'єкта RepairGroup з вказаного рядка DataGridView.
- InitializeFromXml (Ініціалізувати з XML): Ініціалізує властивості об'єкта RepairGroup з вказаного XElement (елемента XML).
- Цей клас використовується для представлення групи ремонту автомобілів і містить методи для роботи з XML-об'єктами та графічним інтерфейсом DataGridView.

3. Форма “Персонал”:

У даному модулі містяться два списки сутностей: mainWorkers і choresWorkers:
- mainWorkers: Цей список містить пари ключ-значення, де ключ - це рядок, що описує певну сутність, а значення - це числове значення, що відповідає цій сутності. Деякі ключі мають спеціальні команди, такі як "<header>", "<empty>" і "<sum>", які вказують на різні дії, які треба виконати з цими сутностями. Значення деяких сутностей можуть бути null.
- choresWorkers: Цей список також містить пари ключ-значення, але представляє інші типи робітників (обслуговуючий персонал). Ключ - це рядок, що описує певну сутність, а значення - це числове значення, що відповідає цій сутності. Список також містить спеціальні команди, такі як "<sum>", які вказують на додаткові дії з сутностями.

У коді також є два методи: GetMainWorkersRows і GetChoresWorkersRows, які генерують список об'єктів DataGridViewRow на основі вищезгаданих списків сутностей.

Загалом, цей код містить дані про різні типи робітників та їх відповідні значення, і використовуєся для створення рядків даних для відображення у DataGridView.

4. Форма “Обладнання”:

У даній формі існує одна основна сутність:

- EquipmentGridData - клас, що містить методи для отримання списків рядків (DataGridViewRow) для головних та допоміжних процесів.

Крім того, використовуються такі простори імен:

- System - містить базові типи та функціональні можливості .NET Framework.
- System.Collections.Generic - містить загальні типи та колекції, такі як List.
- System.Data - надає класи для роботи з даними та базами даних.
- System.Linq - надає розширення для роботи з колекціями та запитами LINQ.
- System.Text - надає класи для роботи з рядками та кодуванням.
- System.Threading.Tasks - надає підтримку асинхронного програмування.
- System.Windows.Forms - містить класи для розробки Windows-додатків з графічним інтерфейсом користувача.
- System.Xml.Linq - надає класи для роботи з XML-документами за допомогою LINQ.

Цей код визначає два списки (mainProcces та supportProcess), що містять ключ-значення пари. Кожна пара складається з рядка (string) із назвою обладнання та значення (double?) - оцінки процесу.

Також у класі є приватні методи GetMainProcessRows та GetSupportProcessRows, які повертають списки рядків з відповідними даними про процеси. Ці методи використовують метод GetProcessRowsFrom, який створює DataGridViewRow, заповнює його значеннями та повертає список таких рядків.

5. Форма “Економічні параметри”:

Клас EconomicParameters - це клас, який містить колекцію параметрів для економічного аналізу. Він містить методи та функції для обчислення значень параметрів та залежностей між ними.

Внутрішній клас Parameter - це клас, що представляє окремий параметр для економічного аналізу. Він має властивості для зберігання назви параметра, значення параметра та функції для обчислення значення параметра на основі підключених параметрів.

У класі EconomicParameters створюється колекція об'єктів типу Parameter, які відображають конкретні економічні параметри. Кожен параметр має певні залежності від інших параметрів, які вказуються у властивості ConnectedParameters. Значення параметра обчислюється за допомогою функції ValueCalculation, якщо вона вказана.

Крім того, клас EconomicParameters містить різні методи для отримання підмножини параметрів залежно від їх призначення.

Загальний зміст коду полягає в тому, щоб об'єднати різні економічні параметри та встановити їх залежності для обчислення значень.

2.2.1.2 CRC методологія

В рамках процесу проектування архітектури програми було розроблено загальну картку відповідно до методології CRC (Class, Responsibilities, Collaborators) для кожного з класів:

1. **Клас: EconomicParameters**

Відповідальності:

- Зберігати економічні параметри
- Забезпечувати доступ до економічних параметрів

Співробітники (колаборатори):

- Немає

2. **Клас: EquipmentGridData**

Відповідальності:

- Зберігати дані про обладнання у сітці
- Забезпечувати доступ до даних про обладнання у сітці

Співробітники (колаборатори):

- Клас EquipmentGroup (агрегація)

3. **Клас: RepairGroup**

Відповідальності:

- Зберігати дані про групи ремонту
- Забезпечувати доступ до даних про групи ремонту

Співробітники (колаборатори):

- Немає

4. Клас: WorkersGridData

Відповідальності:

- Зберігати дані про працівників у сітці
- Забезпечувати доступ до даних про працівників у сітці

Співробітники (колаборатори):

- Клас Worker (агрегація)

5. Клас: MainForm

Відповідальності:

- Керувати головною формою програми
- Забезпечувати взаємодію з іншими класами

Співробітники (колаборатори):

- Клас EconomicParameters (агрегація)
- Клас EquipmentGridData (агрегація)
- Клас RepairGroup (агрегація)
- Клас WorkersGridData (агрегація)

Нижче наведена таблиця залежностей класів програми:

Таблиця 2.2 – Моделювання залежностей

Клас, який зв'язується	Клас, з яким зв'язуються	Тип зв'язку
MainForm	SAPR.Forms.RepairWagons_Form	Залежність
MainForm	SAPR.Models.EquipmentGridData	Залежність
RepairWagonsForm	System.Windows.Forms.Form	Узагальнення
DataGridViewRow	System.Windows.Forms.DataGridViewCellCollection	Узагальнення

DataGridViewRow	System.Windows.Forms.DataGridView	Залежність
EquipmentGridData	EquipmentGroup	Агрегація
WorkersGridData	Worker	Агрегація
MainForm	EconomicParameters	Агрегація
MainForm	EquipmentGridData	Агрегація
MainForm	RepairGroup	Агрегація
MainForm	WorkersGridData	Агрегація

2.2.1.3 Діаграма класів

Заключний результат моделювання у вигляді діаграми класів (рис. 2.2).

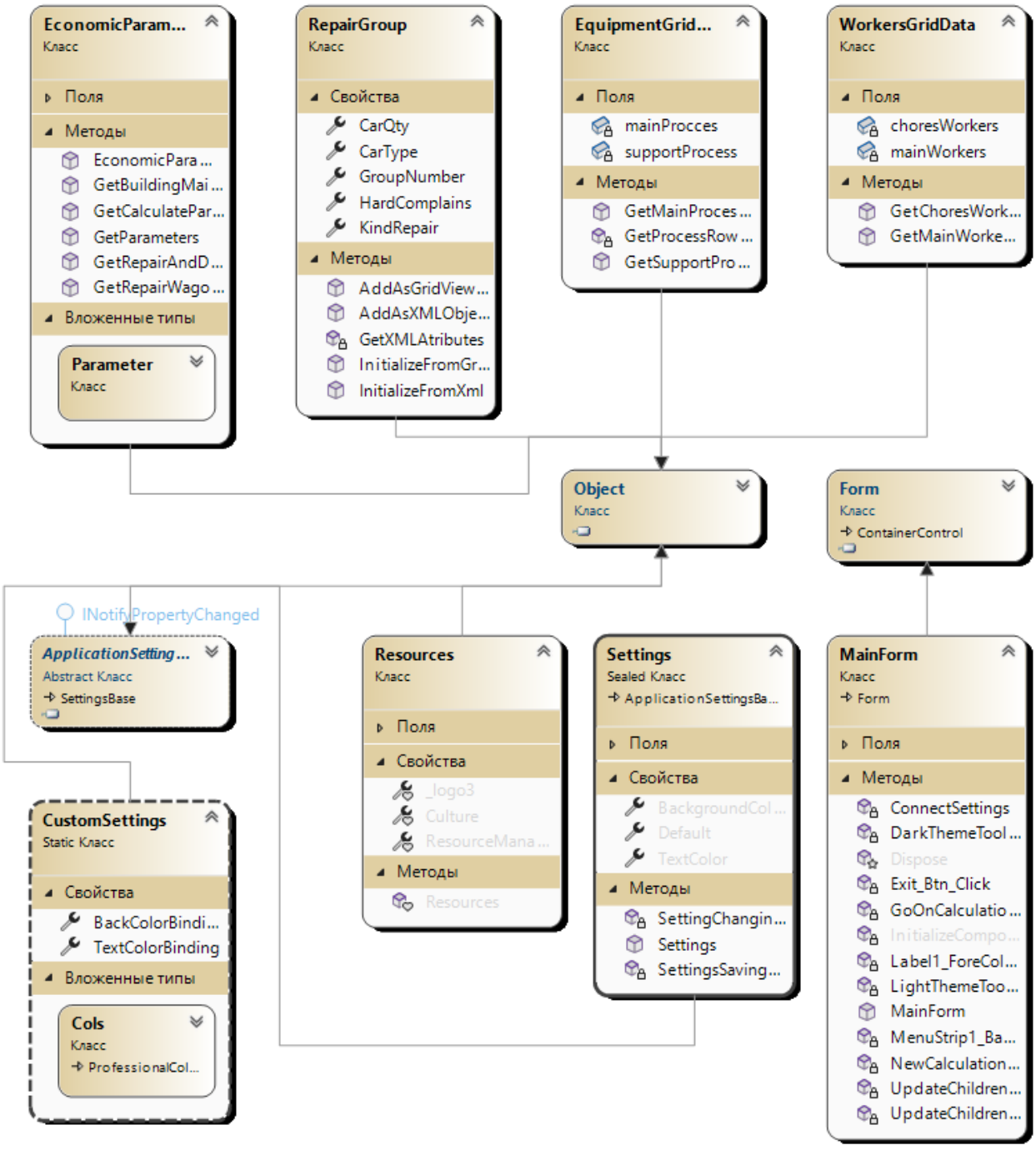

Рисунок 2.2 – Діаграма класів

2.2.2 Проектування інтерфейсу користувача

1) На стартовій формі “MainForm” необхідно відобразити:

- зображення;
- кнопку для початку розрахунку;

- кнопку для продовження розрахунку;
- меню налаштування з можливістю змінити кольорову тему програми;
- кнопка виходу;

На формі основна увага приділена саме логотипу та базовим клавішам для початку роботи з програмою:

Рисунок 2.3 – Ескіз форми “MainForm”

2) На формі “Ремонт вагонів” необхідно відобразити:

- панель управління;
- елемент NumericUpDown для можливості задавання кількості груп ремонту;
- елемент DataGriedView для введення даних для кожної із груп ремонту;
- TextBox-и для введення додаткової інформації;

Основним елементом на формі є таблиця з даними стосовно груп ремонту. розроблеа на даному етапі панель управління буде спільної для всіх подальших форм:

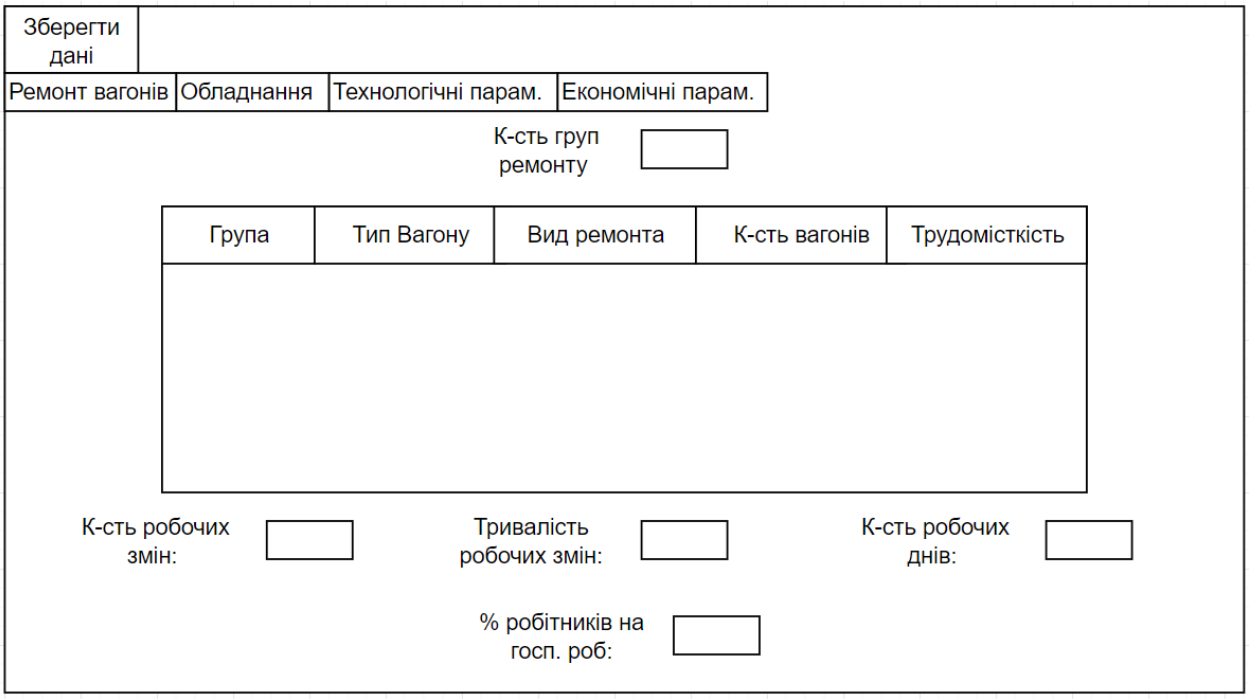

Рисунок 2.4 – Ескіз форми “Ремонт вагонів”

3) На формі “Персонал” необхідно відобразити:

- два списки для демонастрції потрібних спеціальностей;
- текстові поля для виведення зведеної інформації по всім робіникам

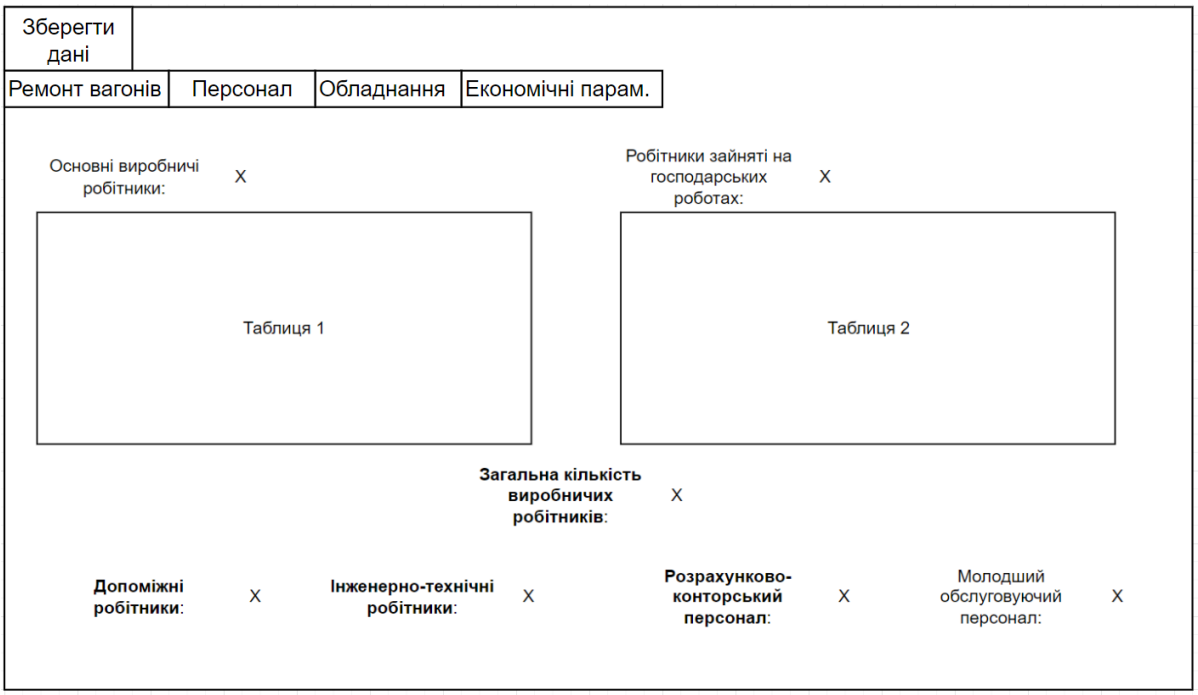

Рисунок 2.5 – Ескіз форми “Персонал”

4) На формі “Обладнання” необхідно відобразити:

- два списки для представлення основних і допоміжних процесів
- список для відображення інформації по всьому підприємству
- текстові поля для виведення додаткової інформації

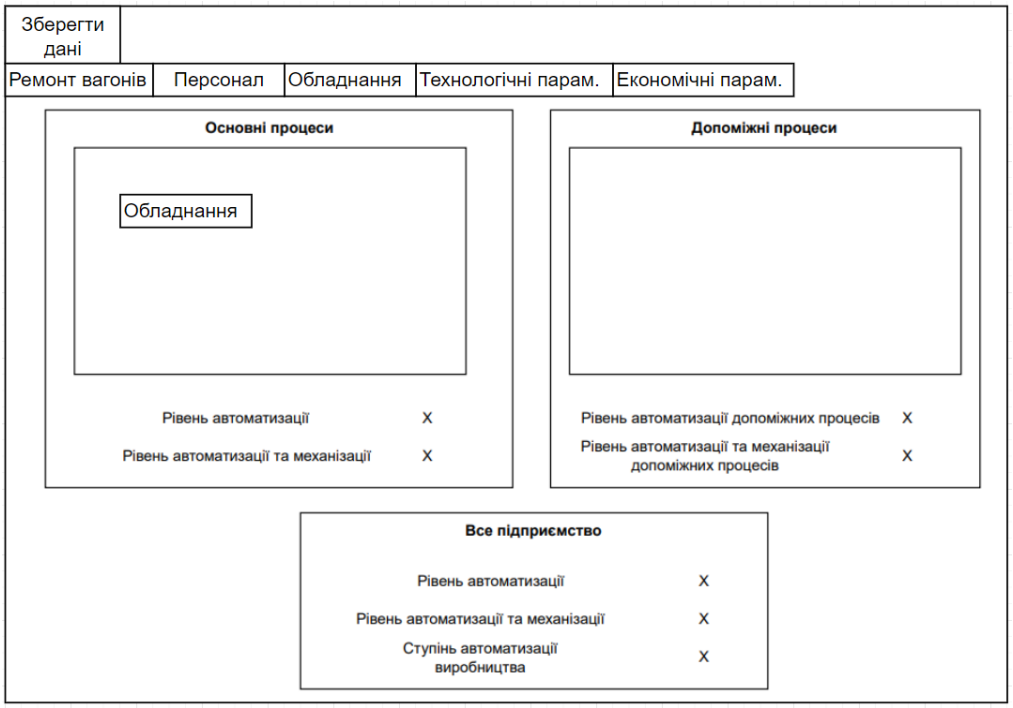

Рисунок 2.6 – Ескіз форми “Обладнання”

5) На формі “Економічні параметри” необхідно відобразити:

- 5 економічних груп, перемикання відбувається за допомогою RadioButton
- список для відображення інформації по економічній групі
- список для можливості введення змінних, що впливають на розрахунки

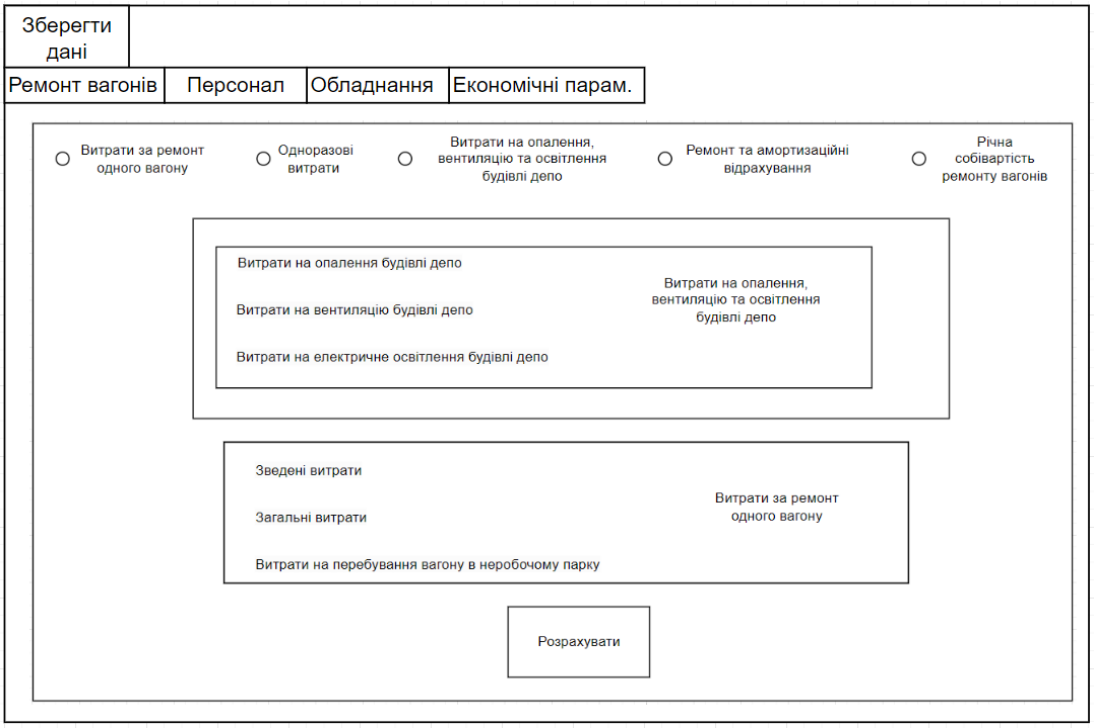

Рисунок 2.7 – Ескіз форми “Економічні параметри”

Висновки до розділу 2

Під час зовнішнього проектування було визначено, що повинне робити ПЗ, а саме, воно повинне приймати на вхід дані та повертати результат обчислень.

А під час внутрішнього проектування – розроблено архітектуру системи, визначено сутності проекту, інтерфейс користувача та як ці сутності повинні взаємодіяти між собою.

3 РОЗРОБКА ПРОГРАМИ

3.1 Вибір мови програмування

Для реалізації розроблюваного продукту було обрано мову програмування C# з використанням інтерфейсу програмування Windows Forms. Оскільки стоїть задача

обчислення, то найважливішою є задача стабільності й швидкості виконання, а саме у цьому є перевага обраної мови. Також досить важливим фактором вибору мови є те, що під час навчання я вже використовував цю мову при вивчанні комп'ютерної графіки, тобто вже маю досвід у створенні інтерфейсу програм і їх налаштування.

Для створення вікна й реалізації інтерфейсу був обраний фреймворк Windows Forms, оскільки я вже мав досвід його використання, а він дозволяє створити кросплатформовий додаток з усіма необхідними елементами інтерфейсу для розв'язання поставленої задачі.

3.2 Розробка алгоритмів

Основним є алгоритм, що повинен ініціювати виконання усіх обчислень, відповідно вхідним даним, а для спрощення цього алгоритму й уникнення дублювання коду – необхідні й другорядні алгоритми, а саме, котрі виконують одну конкретну задачу, таку як, створення масивів, ініціалізація даних з клавіатури, збереження даних та інші.

Далі описані необхідні алгоритми:

1. Додати групи ремонту та обчислити додаткові дані на формі “Групи ремонту” рис. 3.1. Якщо усі значення у рядку заповнені – залишити без змін. При виконанні алгоритму заповнюються усі поля й групи ремонту додаються до списку обчислень. В результаті обчислень отримаємо вихідні дані стосовно потрібного персоналу для виконання ремонту на формі “Персонал”.
2. Надати вхідні дані на формі “Обладнання” для основних та допоміжних процесів (рис. 3.2). Спробувати виконати обчислення по механізації та автоматизації виробництва. В результаті обчислень отримаємо вихідні дані стосовно навантаження обладнання та рівні механізації/автоматизації виробництва.

3. Обрати перший з 5 економічних параметрів на формі “Економічні параметри”. Заповнити вхідні дані для обраного параметру. В разі успіху продовжити обчислення наступних параметрів, використовуючи вже отримані дані.

Рисунок 3.1 – Додавання і заповнення групи ремонту

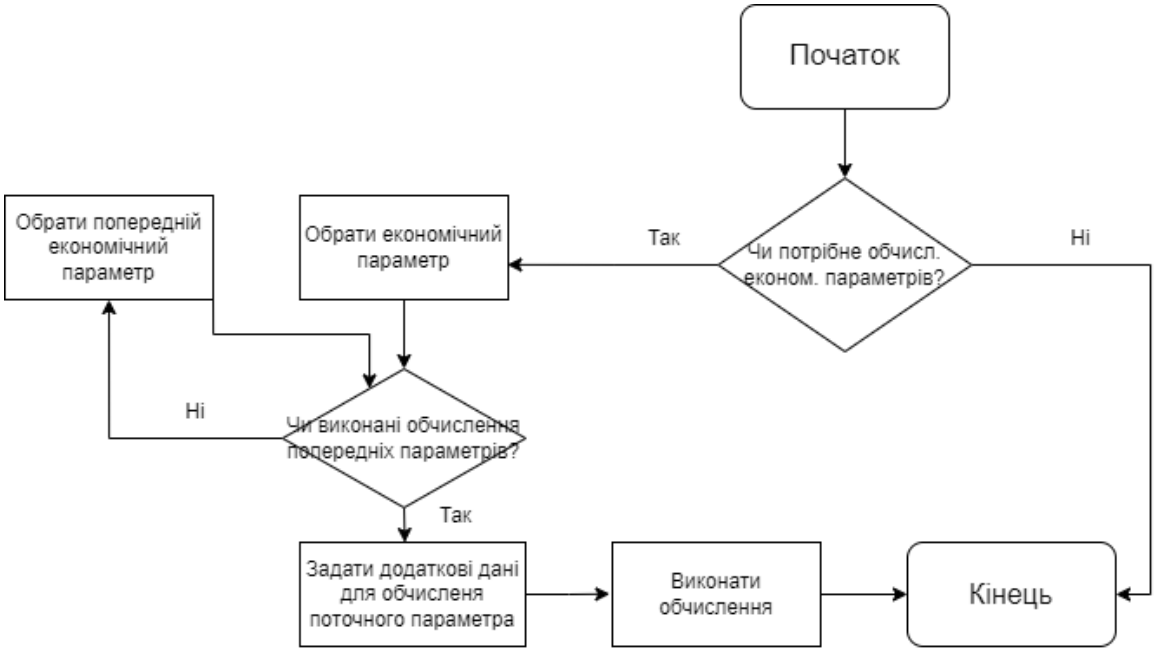

Рисунок 3.2 – Робота з обладнанням

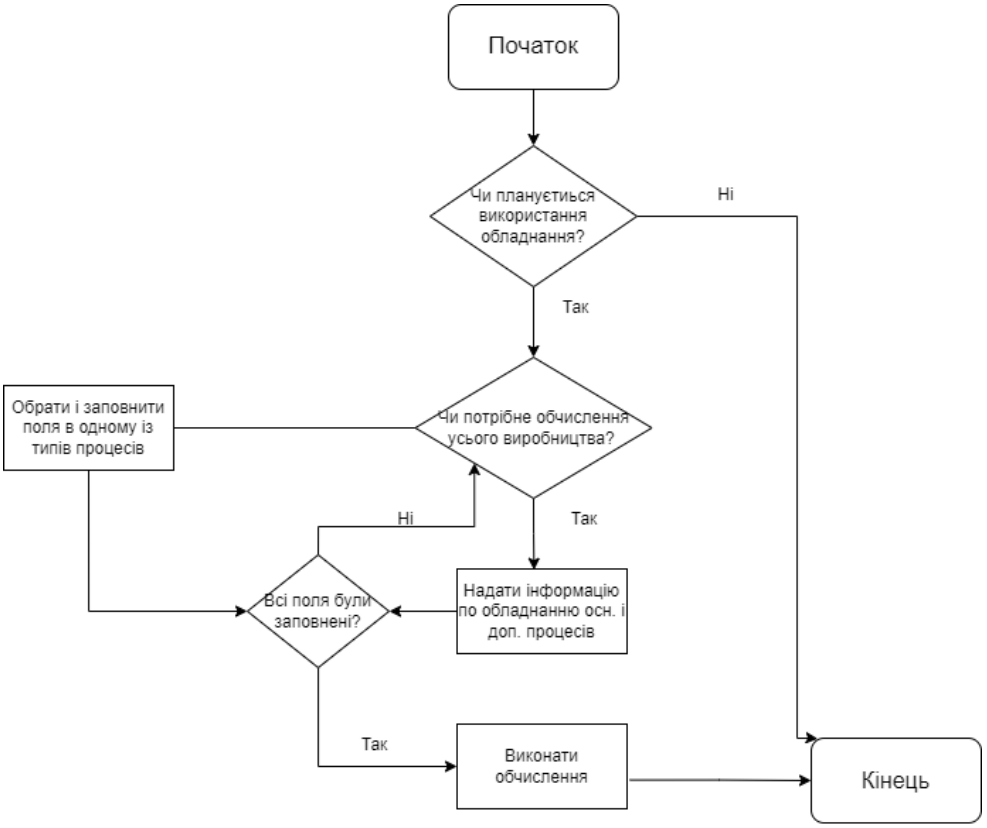

Рисунок 3.3 – Робота з економічними параметрами

Висновки до розділу 3

Було обрано мову програмування C# та фреймворк Windows Forms для розробки проекту. Крім того, було розроблено й використано три спеціалізовані алгоритми, які ефективно вирішують поставлені завдання.

Однією з переваг використання готового фреймворку Windows Forms є можливість зосередитися на розробці самої логіки програми. При цьому необхідно розмістити потрібні елементи на формі та налаштувати їх відповідно до потреб проекту.

Варто відзначити, що всі інші компоненти, які використовуються у розроблених алгоритмах, виконують функцію збереження необхідної інформації. Це допомагає підвищити ефективність та простоту реалізації поставлених завдань.

4 ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

Для забезпечення коректності функцій, їх необхідно протестувати. Для тестування були вибрані найважливіші функції проекту, які відповідають за направлення вагонів до потрібних модулів та управління всім процесом. Оскільки ці функції не планується використовувати як бібліотеки для інших проектів, тестування "чорною скринькою" не є обов'язковим, оскільки інтерфейс користувача вже перевіряє всі вхідні значення на коректність.

4.1 Специфікація функцій

Далі наведено перелік функцій, що будуть тестуватись:

1) public void AddAsGridViewRow(DataGridView dataGrid)

Ця функція виконує додавання об'єкту RepairGroup у вигляді рядка до DataGridView. У ній створюється новий рядок DataGridViewRow, ініціалізуються його комірки зі значеннями властивостей об'єкту RepairGroup та додається цей рядок до DataGridView. Значення властивостей об'єкту RepairGroup встановлюються в відповідні комірки рядка, які відповідають певним стовпцям сітки даних.

Приймає на вхід:

- об'єкт типу DataGridView, який представляє сітку даних вікна програм;

Повертає на вихід:

- не повертає значення на вихід, оскільки її основна робота полягає в додаванні рядка з даними об'єкту RepairGroup до DataGridView;

Текст функції:

```
public void AddAsGridViewRow(DataGridView dataGrid)
    {
        DataGridViewRow row = new DataGridViewRow();
        row.CreateCells(dataGrid);
        row.Cells[0].Value = GroupNumber;
        DataGridViewComboBoxCell wagonTypeCell = new
DataGridViewComboBoxCell();
        wagonTypeCell.Items.AddRange(new string[] { "Піввагон", "Цистерна",
"Критий", "Платформа", "Хопер", "Думпкар","Окатишевоз", "Вагон - бункер",
"Спецвагон 1", "Спецвагон 2" });
        row.Cells[1] = wagonTypeCell;
        row.Cells[1].Value = CarType;
        DataGridViewComboBoxCell repairTypeCell = new
DataGridViewComboBoxCell();
        repairTypeCell.Items.AddRange(new string[] { "Д", "К", "М", "У" });
        row.Cells[2] = repairTypeCell;
        row.Cells[2].Value = KindRepair;
```

```
        row.Cells[3].Value = CarQty;
        row.Cells[4].Value = HardComplains;
        dataGrid.Rows.Add(row);
      }
```

2) GetMainWorkersRows(DataGridView example)

Генерує список рядків для заповнення таблиці DataGridView з даними про робітників.

Приймає на вхід:

- об'єкт типу DataGridView, який представляє сітку даних у форматі таблиці;

Повертає на вихід:

- створює та повертає список об'єктів DataGridViewRow, які містять дані про робітників з основної групи. Кожен об'єкт DataGridViewRow представляє рядок таблиці з трьома стовпцями: назва робітника, значення та тип значення;

Текст функції:

```
public static List<DataGridViewRow> GetMainWorkersRows(DataGridView example)
    {
        List<DataGridViewRow> result = new List<DataGridViewRow>();
```

```
double sum = 0;
foreach (var record in mainWorkers)
{
        DataGridViewRow row = new DataGridViewRow();
        row.CreateCells(example);
        int specSymbol = record.Key.IndexOf(">");
        string rowName, rowValue = "";
        if (specSymbol != -1)
        {
        rowName = record.Key.Substring(specSymbol + 1);
        string commandWord = record.Key.Substring(0, specSymbol + 1);
        switch (commandWord)
        {
        case "<sum>":
                rowValue = sum.ToString();
                sum = 0;
                break;
        case "<header>":
           case "<empty>":
                break;
```

```
            }

            }

            else

            {

            rowName = record.Key;

            if (record.Value != null)

            {

                    rowValue = record.Value.ToString();

            sum += Double.Parse(rowValue);

            }

            }

            row.Cells[0].Value = rowName;

            row.Cells[1].Value = rowValue;

        row.Cells[1].ValueType = typeof(double);

            row.Cells[2].ValueType = typeof(double);

            result.Add(row);

    }

    return result;

}
```

4.2 Тестування методом Білої скриньки

4.2.1 Розроблення тестів

Функція 1(AddAsGridViewRow):

1. EmptyDataGridView

Вхідні дані:

- Порожній DataGridView

Вихідні дані:

- DataGridView з доданим одним рядком

2. ExistedDataGridView

Вхідні дані:

- DataGridView з існуючими рядками

Вихідні дані:

- DataGridView з доданим додатковим рядком, зберігаючи існуючі рядки без змін

3. NiceData

Вхідні дані:

- DataGridView з введеними дійсними даними в усіх стовпцях

Вихідні дані:

- DataGridView з новим рядком, що містить введені дані

4. BadData

Вхідні дані:

- DataGridView з деякими порожніми комірками

Вихідні дані:

- DataGridView з новим рядком, що містить порожні значення для порожніх комірок

5. WrongDataType

Вхідні дані:

- DataGridView з введеним неправильним типом даних в одній з комірок

Вихідні дані:

- Помилка або виняток, що вказує на неправильний тип даних

6. MaxSizeData

Вхідні дані:

- DataGridView з введеними значеннями максимальної довжини в усіх стовпцях

Вихідні дані:

- DataGridView з новим рядком, що містить значення максимальної довжини

7. OutLimitData

Вхідні дані:

- DataGridView зі значеннями, які перевищують допустимий діапазон або обмеження

Вихідні дані:

- Помилка або виняток, що вказує на перевищення значення

8. DuplicateValues

Вхідні дані:

- DataGridView з дублюючими значеннями в стовпці, який повинен містити унікальні значення

Вихідні дані:

- Помилка або виняток, що вказує на наявність дублюючих значень

9. BoundaryValues

Вхідні дані:

- DataGridView з граничними значеннями в різних стовпцях (наприклад, мінімальні, максимальні та крайові випадки)

Вихідні дані:

- DataGridView з новим рядком, що містить граничні значення, відображені правильно

Функція 2(GetMainWorkersRows):

1. EmptyDataGridView

Вхідні дані:

- порожній об'єкт DataGridView example;

Вихідні дані:

- порожній список рядків;

2. ValidColumn

Вхідні дані:

- об'єкт DataGridView example з відповідним числом стовпців;

Вихідні дані :

- список рядків, відповідний списку mainWorkers;

3. InsufficientColumn

Вхідні дані:

- об'єкт DataGridView example з меншою кількістю стовпців, ніж потрібно для заповнення даними;

Вихідні дані:

- список рядків з неповними значеннями, які вміщаються в наявну кількість стовпців;

4.2.2 Покриття рішень

Тестування методом покриття рішень для функції 1 (AddAsGridViewRow) представлено у табл. 4.1.

Тестування методом покриття рішень для функції 2 (GetMainWorkersRows) представлено у табл. 4.2.

Таблиця 4.1 – Покриття рішень

Номер тесту	Номер рішення															
	1		2		3		4		5		6		7		8	
	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
1		*									*					
2		*										*	*		*	
3	*		*													
4		*										*		*		
5	*			*		*			*							
6	*			*	*		*									
7	*			*						*						
8	*			*	*			*								
9		*										*	*			*

Таблиця 4.2 – Покриття рішень

Номер тесту	Номер рішення							
	1		2		3		4	
	+	-	+	-	+	-	+	-
1	*							

2		*	*	*	*	*	*	*

4.2.3 Покриття умов

Результати тестування методом покриття умов для функції AddAsGridViewRow представлені в таблиці 4.3. Номер умови взято з тексту функції, а у випадку, коли рішення складається з декількох умов, кожна умова позначена у вигляді "<номер рішення>.<номер умови>".

Таблиця 4.3 – Покриття умов

Номер тесту	Номер умови																					
	1.1		1.2		2.1		2.2		3		4		5		6		7.1		7.2		8	
	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
1		*													*							
2		*														*	*		*		*	
3	*		*		*		*															
4		*														*		*				
5	*		*			*				*			*									
6	*		*			*			*		*											
7	*		*			*								*								
8	*		*			*			*			*										
9		*														*	*		*			*

Тестування методом покриття умов для функції 2 (GetMainWorkersRows)представлено у табл. 4.4.

Таблиця 4.4 – Покриття умов

Номер тесту	Номер умови			
	1	2	3.1	4

	+	-	+	-	+	-	+	-
1	*							
2		*	*	*	*	*	*	*
3		*	*	*	*	*		

Налагодження

Існують різні типи помилок, таких як:

- Синтаксичні помилки - виникають через порушення правил мови програмування. Зазвичай ці помилки виявляються під час компіляції і можуть бути виправлені досить легко. Компілятор зазвичай надає попередження про такі помилки під час трансляції програми.
- Семантичні (логічні) помилки - призводять до некоректних обчислень або помилок під час виконання програми (run-time error). Усунення семантичних помилок зазвичай вимагає виконання програми з уважно підібраними перевірковими даними, для яких відомий правильний результат.

Під час розробки програмного забезпечення були виявлені і усунені обидва типи помилок.

Синтаксичні помилки були виявлені за допомогою компілятора, який негайно показує всі синтаксичні помилки, і їх усунення зазвичай полягає у виправленні тексту команд.

Однак, семантичні помилки не так просто виявити. Вони проявляються у некоректному (неочікуваному) поведінці програми, і для їх локалізації необхідно використовувати точки зупинки або логування процесу виконання програми. Після виявлення помилок їх потрібно виправити, що вже залежить від структури алгоритму.

Висновки до розділу 4

Під час аналізу функцій методом білої скриньки виявлено деякі недоліки. У першій функції було виявлено непотрібну умову 1.2, яка не впливає на результат виконання функції. Заради забезпечення стабільності роботи додатка, було прийнято

рішення залишити умови 4.2 та 5.2, які завжди виконуються. Це дозволяє уникнути критичних помилок, якщо ці умови не виконуються. Друга функція залишилась без змін.

Тестування чорною скринькою не було необхідним, оскільки всі можливі значення, які вводить користувач, обмежені інтерфейсом програми.

Для цього проекту найбільш ефективним методом тестування виявився "Метод покриття умов". Він дозволив детальніше розглянути, які саме умови є необхідними для виконання поставленої задачі та забезпечення безпеки під час виконання програми.

ВИСНОВКИ

Розробка компонентів системи автоматизованого проектування технологічного розділу вагоноремонтних підприємств привела до створення програмного забезпечення, яке здатне моделювати процес ремонту на вагоноремонтних підприємствах. Для реалізації цього проекту було використано архітектурний шаблон "Модель-Вигляд-Контролер". Це надає зручну основу для подальшого вдосконалення програми, оскільки будь-яку складову цієї архітектури можна легко модифікувати без необхідності внесення значних змін і забезпечити відображення моделювання в різних інтерфейсах.

Проект розробки елементів системи автоматизованого проєктування технологічного розділу вагоноремонтних підприємств має важливу функцію збереження даних. Ця функція дозволяє ефективно керувати та зберігати всю необхідну інформацію, пов'язану з процесом ремонту рухомого складу. Завдяки системі збереження даних можна зберігати дані про вагони, персонал, обладнання і набір економічних параметрів. Це сприяє покращенню організації та плануванню робіт з ремонту, а також забезпечує зручний доступ до інформації для персоналу вагоноремонтних підприємств.

Програмний продукт можна розширити, включивши більш розширені моделі та інструменти аналізу ремонтних процесів. Це може допомогти покращити ефективність та оптимізацію ремонтних робіт, виявляти потенційні проблеми та забезпечувати оптимальне використання ресурсів.

Успішна реалізація цього проекту стала можливою, враховуючи знання, отримані під час курсу з комп'ютерної графіки, де вивчалися принципи розробки подібних проектів, необхідні сутності та їх взаємозв'язок. Також курс з тестування допоміг у перевірці функціональності програми навіть під час розробки, дозволяючи програмі продовжувати виконання, навіть якщо користувач вводить некоректні значення. Під час написання звіту було чітке розуміння того, яку інформацію треба демонструвати та у якому форматі, завдяки отриманим знанням з курсу з тестування.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Мямлін, В. В. Теоретичні основи створення гнучких поточних виробництв для ремонту рухомого складу [Текст]: монографія / В. В. Мямлін. – Дніпропетровськ: Вид-во ЧФ «Стандарт-Сервіс», 2014. – 380 с.
2. Мямлін, В. В. Розвиток наукових основ створення гнучких поточних технологій ремонту рухомого складу [Текст]: дисертація / В. В. Мямлін. – Дніпропетровськ: Вид-во ЧФ «Стандарт-Сервіс», 2015. – 380 с.
3. Бараш Ю. С. Раціональні шляхи розвитку технічної бази для депового ремонту вантажних вагонів [Електронний ресурс]: автореф. thesis / Бараш Юрій Савельович; Дніпропетровський інститут інженерів залізничного транспорту. – [Б. м.], 1982.
4. Gunawardhana L. K. P. D. Процес аналізу вимог, пов'язаний з розробкою програмного забезпечення [Електронний ресурс] / L. K. P. Dhananjaya Gunawardhana // Журнал інженерії програмного забезпечення та застосунків. – 2019. – Т. 12, № 10. – С. 406–422.
5. Нікандров В.В. Експериментальна психологія [Текст]: Навчальний посібник / В. В. Нікандров – СПб.: Вид-во «Мова», 2003. – 480 с.
6. Тестування програмного забезпечення [Текст]: навчальний посібник / Авраменко А.С., Авраменко В.С. Косенюк Г.В. – Черкаси : ЧНУ імені Богдана Хмельницького, 2017. – 284 с.
7. Suriya Sundaramoorthy, UML Diagramming: A Case Study Approach [Текст] / Suriya Sundaramoorthy – New York: Auerbach Publications, 2022. – 430 с.
8. Якість програмного забезпечення та тестування [Текст]: методичні вказівки до лабораторних робіт / уклад.: В. І. Шинкаренко, О. С. Куроп'ятник, Г. В. Забула, Д. О. Пєтін, Є. В. Лукін, Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во ПФ «Стандарт-Сервіс», 2018. – 50 с.
9. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн.трансп.

ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. тра-нсп. ім. акад. В. Лазаряна, 2009. - 38 с

10. Алгоритми: побудова й аналіз, 3-е видання, том 2 [Текст] / Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. – Вид-во «Діалектика-Вільямс», 2019. – 664 с.
11. ДСТУ ГОСТ 7.1-2006. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: чинний з 2007-07-01. – К.: Держспоживстандарт України, 2007. – 47 с. (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).
12. Lazar G. Mastering Qt 5: Create stunning cross-platform applications / Guillaume Lazar, Robin Penea. – [Б. м.]: Packt Publishing, 2016. – 526 с.
13. UML 2. (. D. G. UML 2002-- the Unified Modeling Language: Model engineering, concepts, and tools: 5th International Conference, Dresden, Germany, September 30-October 4, 2002: proceedings / 2002 (2002 Dresden Germany) UML. – Berlin: Springer, 2002. – 447 с.
14. Sinha A. Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry [Електронний ресурс] / Abhiup Sinha, Pallabi Das // 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 24–26 верес. 2021 р. – [Б. м.], 2021.
15. C# 9.0 in a Nutshell: The Definitive Reference “Language syntax, libraries, and advanced concepts” [Б.м.]. - John Sharp. - O'Reilly Media, 2021.

Програмний код та його опис

ДОДАТОК А

MainForm.cs

Цей програмний код представляє частину класу MainForm, який є головною формою додатку. Він відповідає за обробку подій та налаштування графічного інтерфейсу користувача. Основні елементи коду включають:

1. Оголошення та ініціалізація **класу MainForm:**

```
public partial class MainForm : Form
{
  public MainForm()
 {
       // Конструктор, де викликається ініціалізація компонентів форми та інші
налаштування.
     InitializeComponent();
     ConnectSettings();
      menuStrip1.Renderer = new ToolStripProfessionalRenderer(new
CustomSettings.Cols());
  }
  // Методи обробки подій та інші функції.
}
```

2. Метод NewCalculation_Btn_Click обробляє натискання кнопки:

```
private void NewCalculation_Btn_Click(object sender, EventArgs e)
{
   new RepairWagons_Form().ShowDialog();
    }
```

3. Метод GoOnCalculation_Btn_Click обробляє натискання кнопки "Go On Calculation" та відкриває діалогове вікно для вибору файлу. Потім створюється форма RepairWagons_Form з вибраним файлом:

```
private void GoOnCalculation_Btn_Click(object sender, EventArgs e)
```

```
{
   OpenFileDialog ofd = new OpenFileDialog
   {
      Filter = "Repair Wagon Calculation Files (*.rwcf)|*.rwcf|XML (*.xml)|*.xml|Text (*.txt)|*.txt",
      InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
   };
   if (ofd.ShowDialog() != DialogResult.Cancel)
   {
      RepairWagons_Form repairWagons = new RepairWagons_Form(new
System.IO.FileInfo(ofd.FileName));
      if (!repairWagons.IsDisposed)
      {
         repairWagons.ShowDialog();
      }
   }
```

4. **Методи LightThemeToolStripMenuItem_Click та DarkThemeToolStripMenuItem_Click** обробляють натискання пунктів меню для зміни кольорової теми програми.

5. Метод ConnectSettings налаштовує прив'язки даних для елементів у формі до CustomSettings. Це дозволяє автоматично оновлювати вигляд елементів при зміні налаштувань

RepairGroup.Cs

Клас RepairGroup визначає модель для представлення групи ремонту. Кожен об'єкт RepairGroup містить такі властивості як номер групи, тип вагону, вид ремонту, кількість вагонів і кількість важких вад.

1. Оголошення та ініціалізація властивостей:

```
public class RepairGroup
   {
```

```
    public int GroupNumber { get; set; }
    public string CarType { get; set; }
    public string KindRepair { get; set; }
    public int CarQty { get; set; }
    public int HardComplains { get; set; }
}
```

2. Метод GetXMLAtributes() - повертає словник атрибутів об'єкта у форматі XML:

```
private Dictionary<string, string> GetXMLAtributes()
    {
        Dictionary<string, string> result = new Dictionary<string, string>
        {
            { "groupNumber", $"{GroupNumber}" },
            { "carType", $"{CarType}" },
            { "kindRepair", $"{KindRepair}" },
            { "carQty", $"{CarQty}" },
            { "hardComplains", $"{HardComplains}" },
        };
        return result; }
```

3. Метод AddAsXMLObject(XElement rootElement) - додає об'єкт як елемент XML до заданого кореневого елементу:

```
public void AddAsXMLObject(XElement rootElement)
    {
        if (rootElement != null)
        {
            XElement repairGroup = new XElement("repairGroup");
            foreach (KeyValuePair<string, string> item in GetXMLAtributes())
            {
                XAttribute attribute = new XAttribute(item.Key, item.Value);
                repairGroup.Add(attribute);
```

```
            }
            rootElement.Add(repairGroup);
        }
    }
```

4. **Метод AddAsGridViewRow(DataGridView)** - додає об'єкт як рядок DataGridView. Створює новий рядок, створює комірки для кожної властивості об'єкта і заповнює їх значеннями:

```
public void AddAsGridViewRow(DataGridView dataGrid)
    {
        DataGridViewRow row = new DataGridViewRow();
        row.CreateCells(dataGrid);
        row.Cells[0].Value = GroupNumber;
        DataGridViewComboBoxCell wagonTypeCell = new DataGridViewComboBoxCell();
        wagonTypeCell.Items.AddRange(new string[] { "Піввагон", "Цистерна", "Критий",
"Плафторма", "Хопер", "Думпкар",
            "Окатишевоз", "Вагон - бункер", "Спецвагон 1", "Спецвагон 2" });
        row.Cells[1] = wagonTypeCell;
        row.Cells[1].Value = CarType;
        DataGridViewComboBoxCell repairTypeCell = new DataGridViewComboBoxCell();
        repairTypeCell.Items.AddRange(new string[] { "Д", "К", "М", "У" });
        row.Cells[2] = repairTypeCell;
        row.Cells[2].Value = KindRepair;
        row.Cells[3].Value = CarQty;
        row.Cells[4].Value = HardComplains;
        dataGrid.Rows.Add(row);
    }
```

5. **Метод InitializeFromGridViewRow(DataGridViewRow)** - ініціалізує властивості об'єкта зі значень рядка DataGridView:

```
    public void InitializeFromGridViewRow(DataGridViewRow row)
    {
        GroupNumber = int.Parse(row.Cells[0].Value.ToString());
        CarType = (string)row.Cells[1].Value;
```

```
            KindRepair = (string)row.Cells[2].Value;
            CarQty = int.Parse(row.Cells[3].Value.ToString());
            HardComplains = int.Parse(row.Cells[4].Value.ToString());
        }
```

WorkersGridData.cs:

1. Метод GetMainWorkersRows - містить список пар "назва робітника" і "кількість годин" (представлених у вигляді KeyValuePair<string, double?>):

```
public static List<DataGridViewRow> GetMainWorkersRows(DataGridView example)
{
    List<DataGridViewRow> result = new List<DataGridViewRow>();
    double sum = 0;
    foreach (var record in mainWorkers)
    {
        DataGridViewRow row = new DataGridViewRow();
        row.CreateCells(example);
        int specSymbol = record.Key.IndexOf(">");
        string rowName, rowValue = "";
        if (specSymbol != -1)
        {
            rowName = record.Key.Substring(specSymbol + 1);
            string commandWord = record.Key.Substring(0, specSymbol + 1);
            switch (commandWord)
            {
                case "<sum>":
                    rowValue = sum.ToString();
                    sum = 0;
                    break;
                case "<header>":
                case "<empty>":
                    break;
```

```
            }
        }
        else
        {
            rowName = record.Key;
            if (record.Value != null)
            {
                rowValue = record.Value.ToString();
                sum += Double.Parse(rowValue);
            }
        }
        row.Cells[0].Value = rowName;
        row.Cells[1].Value = rowValue;
        row.Cells[1].ValueType = typeof(double);
        row.Cells[2].ValueType = typeof(double);
        result.Add(row);
    }
    return result;
}
```

2. Блок коду для методу GetChoresWorkersRows виконує аналогічні дії, що і для GetMainWorkersRows, але з використанням колекції choresWorkers для створення рядків таблиці, в яких зберігаються дані про робітників, відповідальних за обов'язки:

```
public static List<DataGridViewRow> GetChoresWorkersRows(DataGridView example)
{
    List<DataGridViewRow> result = new List<DataGridViewRow>();
    double sum = 0;
    foreach (var record in choresWorkers)
    {
        DataGridViewRow row = new DataGridViewRow();
        row.CreateCells(example);
        int specSymbol = record.Key.IndexOf(">");
```

```
    string rowName, rowValue = "";
    if (specSymbol != -1)
    {
        rowName = record.Key.Substring(specSymbol + 1);
        string commandWord = record.Key.Substring(0, specSymbol + 1);
        switch (commandWord)
        {
            case "<sum>":
                rowValue = sum.ToString();
                sum = 0;
                break;
            case "<header>":
            case "<empty>":
                break;
        }
    }
    else
    {
        rowName = record.Key;
        if (record.Value != null)
        {
            rowValue = record.Value.ToString();
            sum += Double.Parse(rowValue);
        }
    }
    row.Cells[0].Value = rowName;
    row.Cells[1].Value = rowValue;
    row.Cells[1].ValueType = typeof(double);
    row.Cells[2].ValueType = typeof(double);
    result.Add(row);
}
return result;
```

```
}
```

EquipmentGridData.cs:

Даний код визначає клас EquipmentGridData, який містить два статичних методи: GetMainProcessRows і GetSupportProcessRows. Обидва методи повертають список рядків DataGridViewRow для відображення у DataGridView.

1. Метод GetMainProcessRows - повертає список рядків DataGridViewRow для основних процесів:

```
public static List<DataGridViewRow> GetMainProcessRows(DataGridView example)
{
   return GetProcessRowsFrom(example, mainProcces);
}
```

2. Метод GetSupportProcessRows - повертає список рядків DataGridViewRow для основних процесів:

```
public static List<DataGridViewRow> GetSupportProcessRows(DataGridView example)
      {
         return GetProcessRowsFrom(example, supportProcess);
      }
```

3. Метод GetProcessRowsFrom - отримує DataGridView та список джерела даних, і повертає список рядків DataGridViewRow для відображення у DataGridView. У цьому блоку виконується цикл foreach, який проходиться по кожному запису у джерелі даних source. Для кожного запису створюється новий рядок DataGridViewRow, заповнюються значення комірок та обчислюється значення nz. Рядок додається до результату result, який повертається з методу:

```
private static List<DataGridViewRow> GetProcessRowsFrom(DataGridView example,
List<KeyValuePair<string, double?>> source)
      {
         List<DataGridViewRow> result = new List<DataGridViewRow>();
         foreach (var record in source)
         {
```

```
DataGridViewRow row = new DataGridViewRow();
row.CreateCells(example);
string rowName = record.Key, rowValue = "";
if (record.Value != null)
{
   rowValue = record.Value.ToString();
}
row.Cells[0].Value = rowName;
row.Cells[1].Value = rowValue;
double nz = 0;
switch (record.Value) {
   case 0:
      nz = 0.1;
      break;
   case 1:
      nz = 0.2;
      break;
   case 2:
      nz = 0.4;
      break;
   case 2.25:
      nz = 0.5;
      break;
   case 3:
      nz = 1;
      break;
   case 3.25:
      nz = 1.1;
      break;
   case 3.5:
      nz = 1.4;
      break;
```

```
                case 3.75:
                    nz = 1.75;
                    break;
                case 4:
                    nz = 2.25;
                    break;
                case 4.25:
                    nz = 2.8;
                    break;
                case 4.5:
                    nz = 3.5;
                    break;
                case 4.75:
                    nz = 4.2;
                    break;
                case 5:
                    nz = 5;
                    break;
            }
            row.Cells[2].Value = nz.ToString();
            result.Add(row);
        }
        return result;
    }
}
```

EcomomicParameters.cs:

Основний код поділяється на два класи: EconomicParameters та Parameter. Клас EconomicParameters відповідає за розрахунок економічних параметрів, тоді як клас Parameter представляє окремий параметр.

1. Клас EconomicParameters - клас, що представляє економічні параметри:

```
class EconomicParameters:
    def __init__(self):
        self.parameters = []

    def __init__(self, parameters):
        self.parameters = []
        for parameter in parameters:
            self.add_parameter(parameter)

    def add_parameter(self, parameter):
        self.parameters.append(parameter)

    def get_parameters(self):
        return self.parameters

    def get_calculate_parameters(self):
        calculate_parameters = []
        for parameter in self.parameters:
            if parameter.is_calculate_parameter():
                calculate_parameters.append(parameter)
        return calculate_parameters

    def get_repair_wagon_parameters(self):
        repair_wagon_parameters = []
        for parameter in self.parameters:
            if parameter.is_repair_wagon_parameter():
                repair_wagon_parameters.append(parameter)
        return repair_wagon_parameters

    def get_building_maintenance_parameters(self):
        building_maintenance_parameters = []
        for parameter in self.parameters:
```

```
            if parameter.is_building_maintenance_parameter():
                building_maintenance_parameters.append(parameter)
        return building_maintenance_parameters


    def get_repair_and_depreciation_parameters(self):
        repair_and_depreciation_parameters = []
        for parameter in self.parameters:
            if parameter.is_repair_and_depreciation_parameter():
                repair_and_depreciation_parameters.append(parameter)
        return repair_and_depreciation_parameters
```

2. Клас Parameter- клас, що представляє представляє окремий параметр:

```
class Parameter:
    def __init__(self, name, value):
        self.name = name
        self.value = value
        self.connected_parameters = []


    def add_connected_parameter(self, parameter):
        self.connected_parameters.append(parameter)
```