ДОДАТОК А

Лістинг програми

```
import math
import random
import os
import timeit

import pylab as pl
from minisom import MiniSom
import numpy as np

import matplotlib.pyplot as plt
from matplotlib.patches import RegularPolygon
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib import cm, colorbar
from sklearn.preprocessing import MinMaxScaler


class Node:
    def __init__(self):
        self.predecessors = []
        self.successors = []
        self.renewableResourceRequirements = []
        self.startTime = None
        self.finishTime = None
        self.duration = None
        self.name = None
        self.es = None
        self.ls = None
        self.ef = None
        self.lf = None
        self.prioValue = None
        self.started = False
        self.finished = False
        self.scheduled = False
        self.selectionProbability = None

    def __repr__(self):
        return str(self.name)


class Project:
    def __init__(self):
        self.renewableResourceAvailability = []
        self.numberOfJobs = None
```

```
        self.numberOfNondummyJobs = None
        self.numberOfRenewableResources = None
        self.nodes = {}

        self.horizon = None
        self.population = []

    def readProject(self, instanceFilepath, instanceName):
        fullpath = os.path.join(instanceFilepath, instanceName)

        with open(fullpath, "r") as file:
            file = list(file)
            for lineIndex, line in enumerate(file):
                if "projects" in line:
                    dummy = file[lineIndex + 1].split(" ")
                    self.numberOfJobs = int(dummy[-1])
                    self.numberOfNondummyJobs = self.numberOfJobs - 2

                    dummy = file[lineIndex + 2].split(" ")
                    self.horizon = int(dummy[-1])

                    dummy = file[lineIndex + 4].split(" ")
                    self.numberOfRenewableResources = int(dummy[-4])

                    for i in range(self.numberOfJobs):
                        dummy = Node()
                        dummy.name = i
                        self.nodes[i] = dummy

                elif "PRECEDENCE RELATIONS:" in line:
                    startingLineIndex = lineIndex + 2

                    for i in range(self.numberOfJobs):
                        dummy = file[startingLineIndex + i]
                        dummy = dummy.rstrip()
                        dummy = dummy.split(" ")
                        dummy = list(filter(None, dummy))
                        dummy = [int(entry) for entry in dummy]

                        numberOfSuccessors = dummy[2]
                        for j in range(numberOfSuccessors):
                            successorNodeName = dummy[3 + j] - 1

self.nodes[i].successors.append(self.nodes[successorNodeName])

                    for node in self.nodes.values():
                        for succ in node.successors:
                            succ.predecessors.append(node)

                    for node in self.nodes.values():
                        node.predecessors = list(set(node.predecessors))
```

```
            elif "REQUESTS/DURATIONS:" in line:
                startingLineIndex = lineIndex + 3

                for i in range(self.numberOfJobs):
                    dummy = file[startingLineIndex + i]
                    dummy = dummy.rstrip()
                    dummy = dummy.split(" ")
                    dummy = list(filter(None, dummy))
                    dummy = [int(entry) for entry in dummy]

                    self.nodes[i].duration = dummy[2]
                    for k in range(self.numberOfRenewableResources):

self.nodes[i].renewableResourceRequirements.append(int(dummy[3 + k]))

            elif "RESOURCEAVAILABILITIES:" in line:
                startingLineIndex = lineIndex + 2

                dummy = file[startingLineIndex]
                dummy = dummy.rstrip()
                dummy = dummy.split(" ")
                dummy = list(filter(None, dummy))
                dummy = [int(entry) for entry in dummy]

                for availability in dummy:
                    self.renewableResourceAvailability.append(availability)

    def solveInstanceViaGa(self, popSize, numberOfGenerations,
mutationProbability, sgs, convergenceRate):
        incumbent = Individual()

        convergence = 0
        incumbentFitness = float('inf')

        population = self.createInitialPopulation(popSize)

        for indiv in population:
            sgs(indiv)

        populationForSom = population
        skipNeighbors = 0

        for gen in range(numberOfGenerations):
            print("gen: " + str(gen))
            print("incumbent fitness: " + str(incumbentFitness))
            print("--")

            if incumbent.fitness != float('inf') and incumbentFitness >
incumbent.fitness:
                incumbentFitness = incumbent.fitness
```

```
            elif incumbentFitness == incumbent.fitness:
                convergence += 1

            if gen != 0 and gen % 5 == 0:
                populationForSom = populationForSom + list(set(population))
                populationForSom = list(set(populationForSom))

            if convergence == convergenceRate:
                incumbentIdx = 0
                for ind in populationForSom:
                    if ind.activityList == incumbent.activityList:
                        incumbentIdx = populationForSom.index(ind)
                som = Som(len(set(populationForSom)))
                result = som.trainSom(list(set(populationForSom)), incumbentIdx,
skipNeighbors)
                skipNeighbors += 2
                print(f"res: {result}")
                randomIndices = np.random.choice(len(set(population)), 20,
replace=False)
                ii = 0
                for r in randomIndices:
                    if ii >= len(result):
                        ii = 0
                    population[r] = result[ii]
                    ii += 1
                convergence = 0

            offsprings = self.crossover(population)
            offsprings = self.mutate(offsprings, mutationProbability)

            population, incumbent = self.rankAndReduce(population, offsprings,
incumbent, sgs)

        print("finished GA with fitness of: " + str(incumbent.fitness))
        return incumbent.fitness

    def createInitialPopulation(self, popSize):
        self.forwardBackwardScheduling()
        population = []

        dummy = Individual()
        nodesSortedByMinLft = list(self.nodes.values())
        nodesSortedByMinLft.sort(key=lambda x: x.lf)

        dummy.activityList = nodesSortedByMinLft
        population.append(dummy)

        for i in range(1, popSize):
            dummy = Individual()

            unselectedNodes = list(self.nodes.values())
```

```
            selectedNodes = []

            startingNode = unselectedNodes[0]
            selectedNodes.append(startingNode)
            unselectedNodes.remove(startingNode)

            while unselectedNodes:
                possibles = []
                for node in unselectedNodes:
                    if all(predecessor in selectedNodes for predecessor in
node.predecessors):
                        possibles.append(node)

                maxLft = max(_.lf for _ in possibles)
                total = sum((maxLft - _.lf + 1) for _ in possibles)

                for node in possibles:
                    node.selectionProbability = (maxLft - node.lf + 1) / total

                selectedNode = random.choices(possibles, [_.selectionProbability for
_ in possibles])[0]

                selectedNodes.append(selectedNode)
                unselectedNodes.remove(selectedNode)

            dummy.activityList = selectedNodes
            population.append(dummy)

        return population

    def forwardBackwardScheduling(self):
        self.nodes[0].es = 0
        self.nodes[0].ef = 0

        for node in self.nodes.values():
            if node == self.nodes[0]:
                continue
            node.es = max(_.ef for _ in node.predecessors)
            node.ef = node.es + node.duration

        finishNode = self.nodes[self.numberOfJobs - 1]
        finishNode.lf = self.horizon
        finishNode.ls = self.horizon

        nodes = list(self.nodes.values())
        nodes.reverse()

        for node in nodes:
            if node == finishNode:
                continue
            node.lf = min(_.ls for _ in node.successors)
```

```
            node.ls = node.lf - node.duration

        return None

    def crossover(self, population):
        random.shuffle(population)
        mothers = population[:int(len(population) / 2)]
        fathers = population[int(len(population) / 2):]

        q = random.randint(1, self.numberOfNondummyJobs - 1)

        offsprings = []

        for mother, father in zip(mothers, fathers):
            daughter = Individual()
            motherInput = mother.activityList[:q]
            fatherInput = [_ for _ in father.activityList if _ not in motherInput]
            daughter.activityList = motherInput + fatherInput
            offsprings.append(daughter)

            son = Individual()
            fatherInput = father.activityList[:q]
            motherInput = [_ for _ in mother.activityList if _ not in fatherInput]
            son.activityList = fatherInput + motherInput
            offsprings.append(son)

        return offsprings

    def mutate(self, offsprings, mutationProbability):
        for indiv in offsprings:
            for index, activity in enumerate(indiv.activityList[:-1]):
                if random.random() < mutationProbability:
                    storedCurrentActivityList = indiv.activityList[:]
                    indiv.activityList[index + 1], indiv.activityList[index] =
indiv.activityList[index], \
                                                        indiv.activityList[index + 1]
                    if not indiv.checkPrecedenceFeasibility():
                        indiv.activityList = storedCurrentActivityList
        return offsprings

    def rankAndReduce(self, population, offsprings, incumbent, sgs):
        population.extend(offsprings)

        for indiv in population:
            sgs(indiv)

        population = sorted(population, key=lambda x: x.fitness)
        population = population[:len(population) - 1]

        if incumbent.fitness > population[0].fitness:
            incumbent = population[0]
```

```
            print("new incumbent:" + str(incumbent.fitness))

        return population, incumbent

    def executeGa(self, instanceName):
        instanceDirectory = "./instances/"
        instancePath = instanceDirectory + instanceName
        self.readProject(instancePath, instanceName)

        result = self.solveInstanceViaGa(200, 1000, 0.01,
self.forwardBackwardScheduling, 20)
        return result


class Individual:
    def __init__(self):
        self.activityList = []
        self.fitness = float('inf')

    def __eq__(self, other):
        return self.activityList == other.activityList

    def __hash__(self):
        return hash(tuple(self.activityList))

    def evaluateFitness(self):
        startTimes = []
        finishTimes = []

        for node in self.activityList:
            total = 0
            for resource in range(self.numberOfRenewableResources):
                availability = self.renewableResourceAvailability[resource]
                required = node.renewableResourceRequirements[resource]

                for job in self.activityList:
                    if node != job and node.startTime < job.finishTime and
job.startTime < node.finishTime:
                        if node.renewableResourceRequirements[resource] != 0 and
job.renewableResourceRequirements[
                            resource] != 0:
                            total = 1
                            break

                if total == 0:
                    minStartTime = self.horizon

        if minStartTime < self.horizon:
            if node != self.activityList[0] and node != self.activityList[-1]:
                startTimes.append(minStartTime)
                finishTimes.append(minStartTime + node.duration)
```

```
                node.startTime = minStartTime
                node.finishTime = minStartTime + node.duration

        for i in range(len(startTimes)):
            for j in range(i + 1, len(startTimes)):
                if startTimes[i] > startTimes[j]:
                    startTimes[i], startTimes[j] = startTimes[j], startTimes[i]
                    finishTimes[i], finishTimes[j] = finishTimes[j], finishTimes[i]

        for index, node in enumerate(self.activityList):
            node.startTime = startTimes[index]
            node.finishTime = finishTimes[index]

        self.fitness = 0

        return self.fitness

    def checkPrecedenceFeasibility(self):
        feasible = True

        for node in self.activityList:
            for successor in node.successors:
                if self.activityList.index(node) > self.activityList.index(successor):
                    feasible = False

        return feasible


class Som:
    def __init__(self, population):
        self.population = population
        self.weights = np.random.random((2, self.population))
        self.winner = None

    def trainSom(self, population, incumbentIdx, skipNeighbors):
        self.winner = self.population[incumbentIdx]

        for i in range(len(self.population)):
            self.population[i].activityList = population[i].activityList

        for i in range(1000):
            for individual in self.population:
                winner = self.determineWinner(individual)
                if self.population.index(winner) >=
self.population.index(self.winner):
                    self.winner = winner
                    return self.population

        def determineWinner(self, individual):
            self.winner = self.population[0]
            minDistance = float('inf')
```

```
            for currentIndividual in self.population:
                distance = sum(abs(x - y) for x, y in zip(individual.activityList,
currentIndividual.activityList))

                if distance < minDistance:
                    self.winner = currentIndividual
                    minDistance = distance

            return self.population
```

ДОДАТОК Б

Апробарація Результатів Кваліфікаційної Роботи

(Міжнародний молодіжний форум «Радіоелектроніка і молодь в XXI столітті», який відбувся 16-18 квітня 2024р.)

УДК 004.89

Дослідження методів ШІ для планування айті проектів

Москаленко М.В.
Науковий керівний – професор кафедри ПІ Качко О.Г.
Харківський національний університет радіоелектроніки,
каф. ПІ,
м. Харків, Україна
тел. +38(099) 557-72-92, e-mail:
mykhailo.moskalenko@nure.ua.

This article is dedicated to exploring Artificial Intelligence (AI) methods used for planning IT projects. It provides an overview of existing project management methods in Information Technology (IT) and analyzes the possibilities and effectiveness of utilizing AI in this field. The authors provide examples of applying machine learning algorithms to forecast task completion times and optimize resources in projects. The conclusions allow us to conclude about the potential of using AI methods to improve the efficiency of IT project planning and offer recommendations for future research and practical application.

У сучасну цифрову епоху, де інформаційні технології відіграють переважну роль у просуванні підприємств і організацій на ринку, планування айті проектів стає важливою ланкою у їх стратегічному керівництві. Протягом останніх років, завдяки стрімкому розвитку технологій, особливо активно досліджується сфера штучного інтелекту (ШІ), який потенційно може революціонізувати підхід до управління проектами. Ця стаття має на меті детально проаналізувати та охопити широкий спектр методів ШІ, що застосовуються у плануванні айті проектів, зокрема зосереджуючись на їхній ефективності та можливостях застосування. Висвітлення цих аспектів є важливим для розуміння того, як ШІ може змінити підхід до управління ІТ-проектами та забезпечити їх успішне виконання у сучасному швидкозмінному середовищі.

Перед висновками та рекомендаціями щодо використання методів ШІ для планування айті проектів, важливо розглянути існуючі підходи. В літературі з питань управління проектами

можна знайти різноманітні методи, такі як методологія Agile, каскадний метод, Scrum тощо. Кожен з цих підходів має свої переваги та недоліки, які важливо враховувати при виборі методу для конкретного проекту.

Штучний інтелект дозволяє використовувати комп'ютерні алгоритми та моделі для прийняття рішень у реальному часі. У контексті планування айті проектів, методи ШІ можуть виявитися корисними для автоматизації процесів планування, прогнозування та оптимізації ресурсів. Наприклад, алгоритми машинного навчання можуть аналізувати історичні дані проектів, щоб передбачити терміни виконання завдань та ресурси, необхідні для їх виконання.

Методи ШІ мають потенціал покращити ефективність планування айті проектів шляхом автоматизації та оптимізації процесів. Проте важливо враховувати контекст конкретного проекту, а також обмеження та можливості використання штучного інтелекту. Для успішного впровадження методів ШІ у практику планування айті проектів необхідно провести додаткові дослідження та експерименти, щоб визначити найбільш ефективні та придатні для використання методи.

Список використаних джерел:

1. Smith, J. (2020). "Application of Artificial Intelligence in IT Project Management." Journal of IT Management, 15(2), 45-60.
2. Johnson, A. (2019). "Comparative Analysis of Project Management Methodologies in IT Projects." International Journal of Project Management, 25(3), 110-125.
3. Chen, L., & Wang, Y. (2018). "Machine Learning Techniques for Project Time Estimation in IT Projects." IEEE Transactions on Software Engineering, 42(4), 320-335.
4. Gupta, S. (2017). "Integration of Artificial Intelligence in Agile Project Management." International Conference on Information Systems, 78-92.
5. Lee, H., & Kim, S. (2016). "Scrum Adoption in IT Projects: A Case Study Analysis." Journal of Software Engineering Research and Development, 10(1), 55-70.

ДОДАТОК В

Слайди презентації

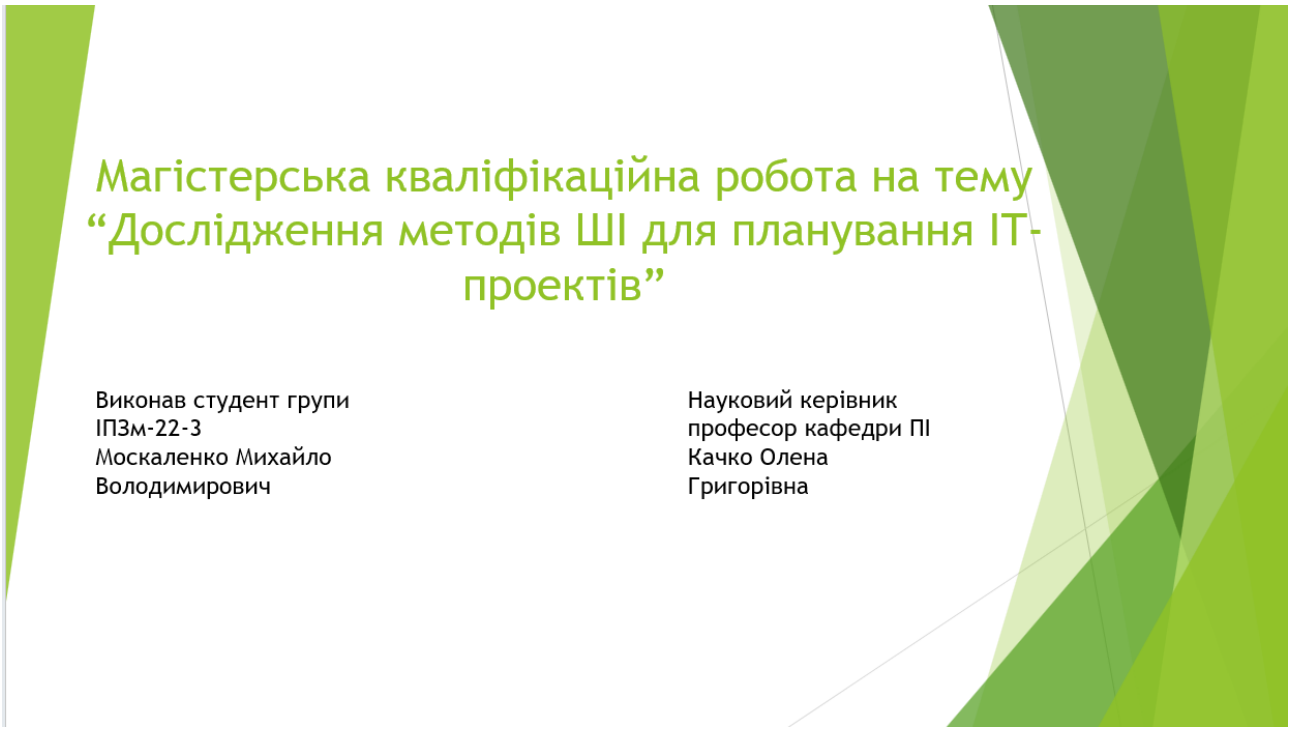


Рисунок В.1

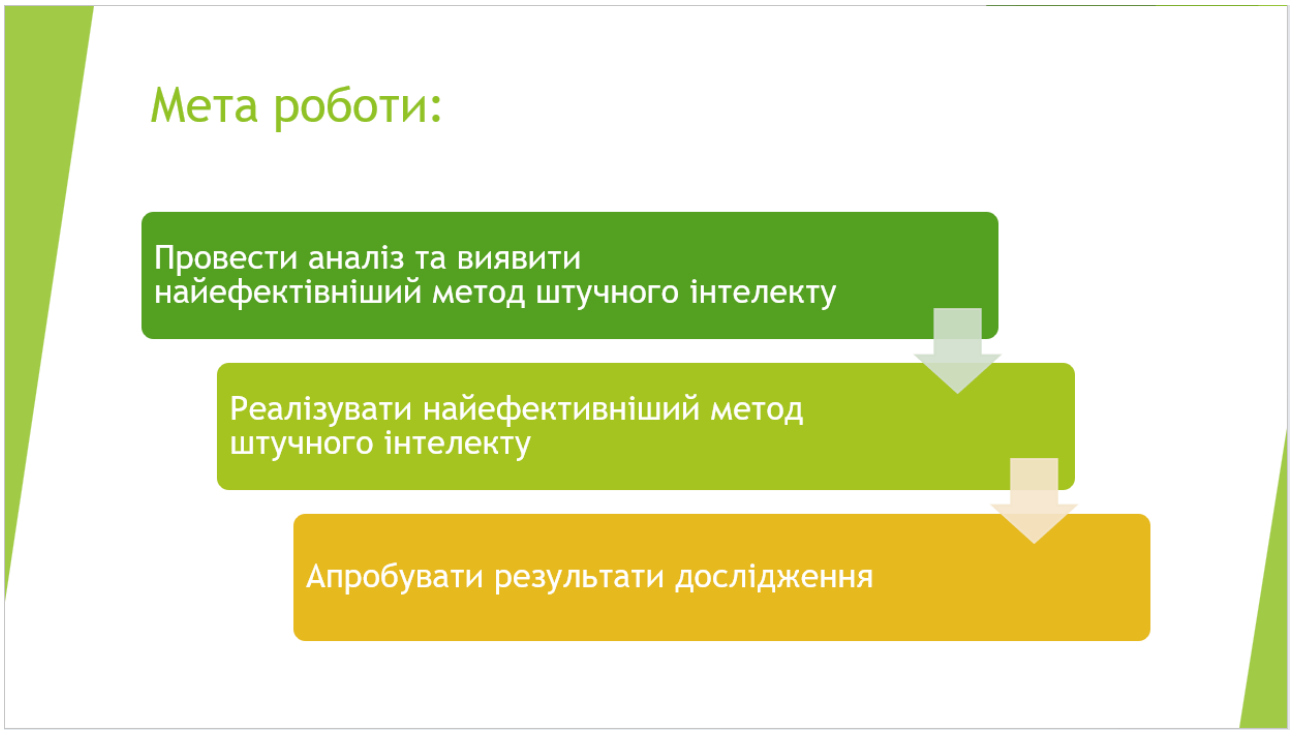


Рисунок В.2

Рисунок В.3

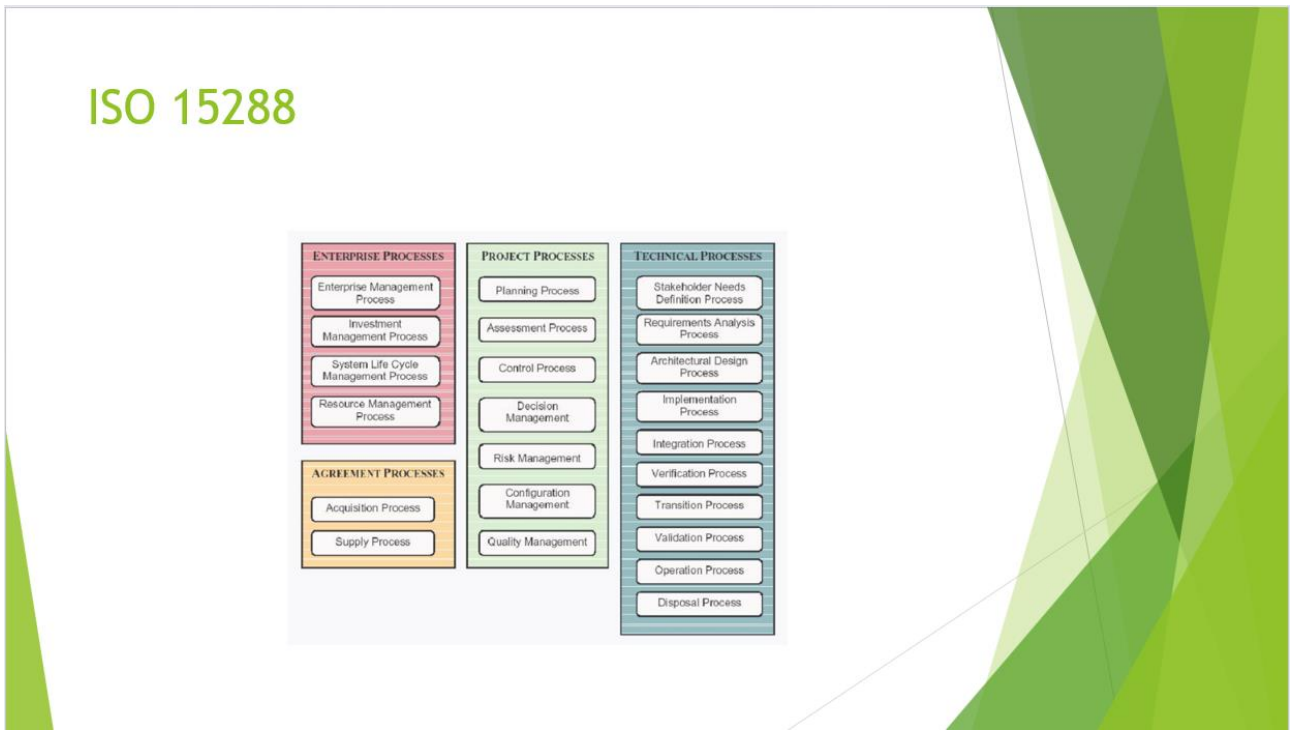


Рисунок В.4

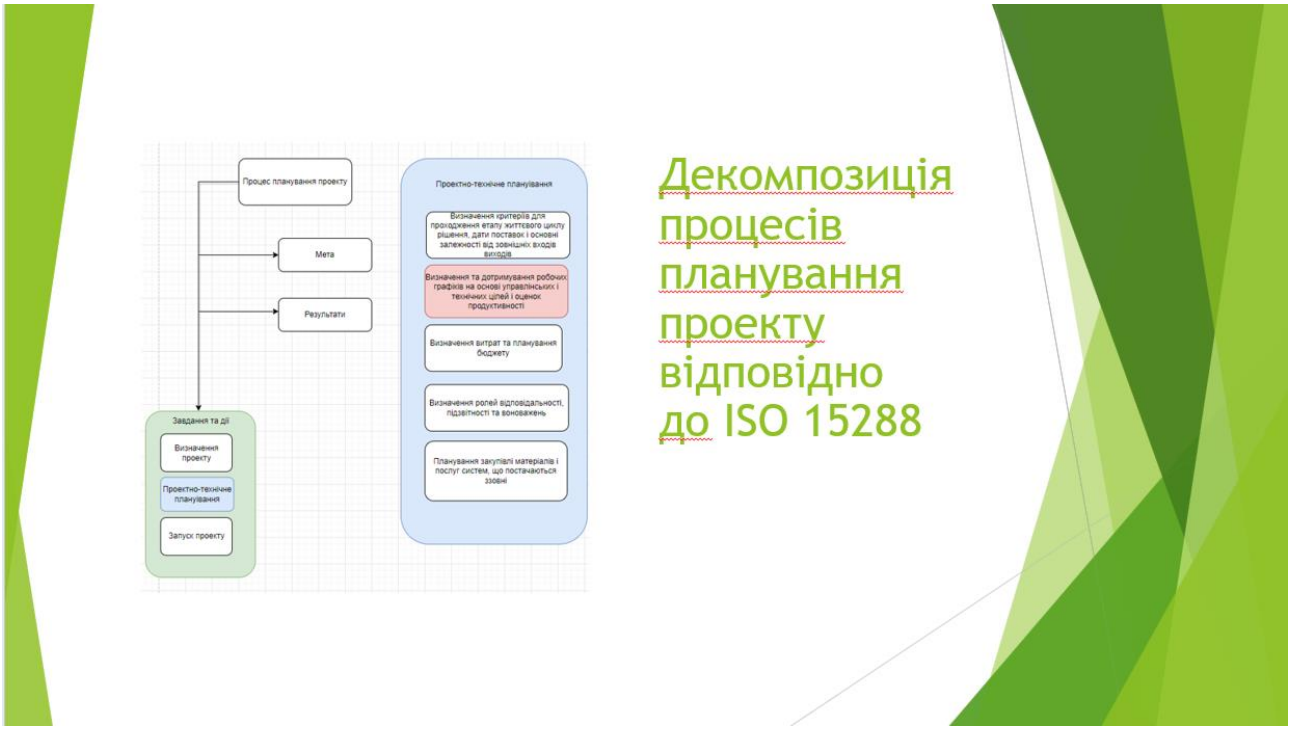


Рисунок В.5

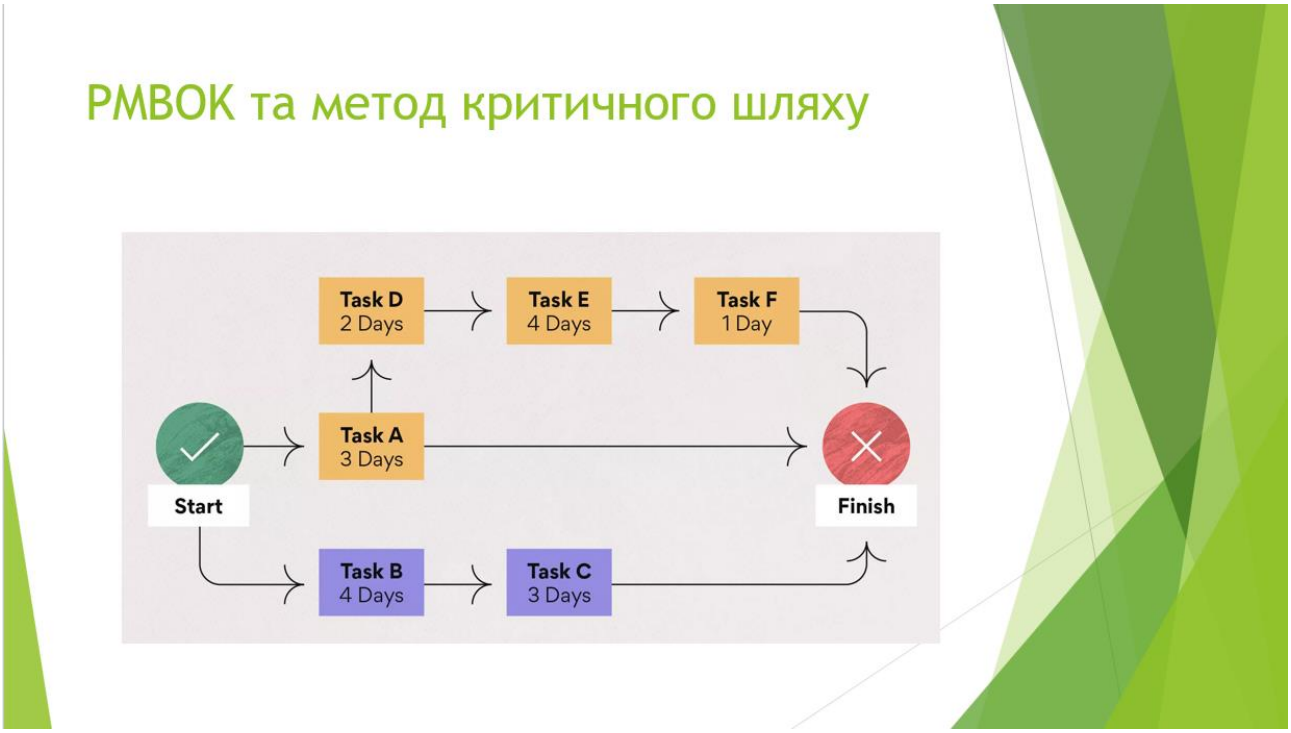


Рисунок В.6

Метод критичних ланцюгів

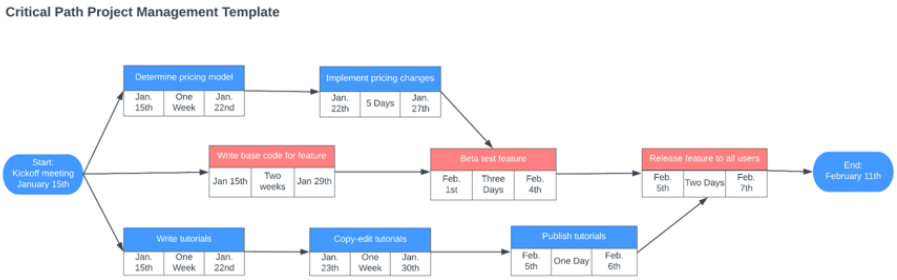


Рисунок В.7

Генетичний алгоритм

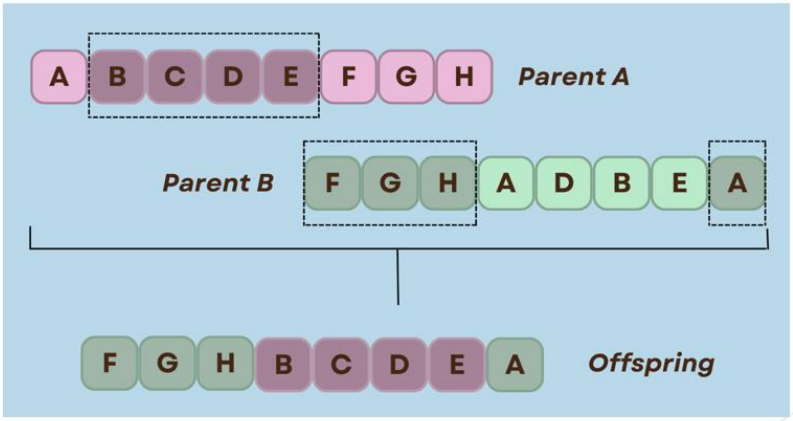


Рисунок В.8

Порівняння підходів

Критерій	Метод критичного шляху	Методи критичних ланцюгів	Генетичні алгоритми
Застосування	Прості та середньо складні проекти	Складні проекти зі значними обмеженнями на ресурси	Проекти з високою складністю та динамікою
Базовий принцип	Планування за найбільш тривалою послідовністю завдань	Планування з урахуванням обмежень ресурсів та введенням часових буферів	Використання імітації природного відбору для знаходження оптимальних рішень
Складність впровадження	Низька	Середня	Висока
Гнучкість	Низька	Середня	Висока
Оптимізація	Фокусується на оптимізації часу	Фокусується на використанні ресурсів.	Декілька параметрів
Управління ризиками	Має обмеження у врахуванні невизначеності	Обмежено, у центрі уваги — ресурси	Висока адаптивність

Рисунок В.9

Вибір підходу

- Машинне навчання
- Нейронні мережі
- Евристичний
- Гібридний

Рисунок В.10

Гібридний алгоритм

У якості методів для реалізації було використано наступні техніки:

- Ініціалізація
- Селекція
- Багатоточковий кросинговер
- Мутація
- Фунція сусідства
- Фунція навчання
- Топологія карти

Рисунок В.11

Багатоточковий кросинговер

Рисунок В.12

Вхідні дані для тестування гібридного алгоритму

Характеристика		Опис	Дані
projects		Кількість проєктів	1
jobs (incl. supersource/sink)		Кількість робіт, в тому числі фіктивні	32
horizon		Максимальний термін проєкту	166
resources	renewable	Кількість поновлюваних ресурсів	4
	nonrenewable	Кількість непоновлюваних ресурсів	0
	doubly constrained	Кількість подвійно обмежених ресурсів	0

Рисунок В.13

Вхідні дані з інформацією про проект

Характеристика	Опис	Дані
jobnr.	Ідентифікатор активності	1
#modes	Кількість режимів	1
#successors	Кількість пов'язаних наступних активностей	3
successors	Активності пов'язаних наступників	2 3 4

Рисунок В.14

Візуалізація кінцевого виклику навчання карт Кохонена

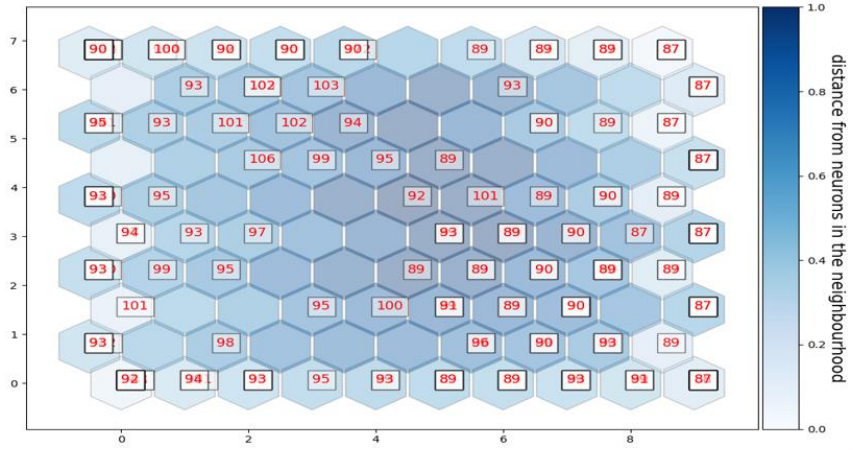


Рисунок В.15

Приклад консольного виводу

```
Quantization error: 0.8987231844731277
Topographic error: 0.12903225806451613
```

```
gen: 149
incumbent fitness: 81
--
finished GA with fitness of: 81
finished in seconds: 15.8868810999993
```

Рисунок В.16

Висновки

Усі завдання дослідження виконано – проведено аналіз методів штучного інтелекту в плануванні IT-проектів, обрано та реалізовано алгоритм, проведено його апробацію.

В результаті дослідження виявлено що використання генетичного алгоритму у парі із самоорганізаційними картами Кохонена є перспективним підходом для вирішення задачі планування IT-проектів.

Майбутнімі дослідженнями можуть бути оптимізація алгоритму, аналіз факторів що впливають на результат та ефективність гібридного алгоритму в різних умовах.

Рисунок В.17

Апробація Результатів Кваліфікаційної Роботи

(Міжнародний молодіжний форум «Радіоелектроніка і молодь в XXI столітті», який відбувся 16-18 квітня 2024р.)

УДК 004.89

Дослідження методів ШІ для планування айті проектів

Москаленко М.В.
Науковий керівний – професор кафедри ПІ Качко О.Г.
Харківський національний університет радіоелектроніки,
каф. ПІ,
м. Харків, Україна
тел. +38(099) 557-72-92, e-mail:
mykhailo.moskalenko@nure.ua.

Рисунок В.18

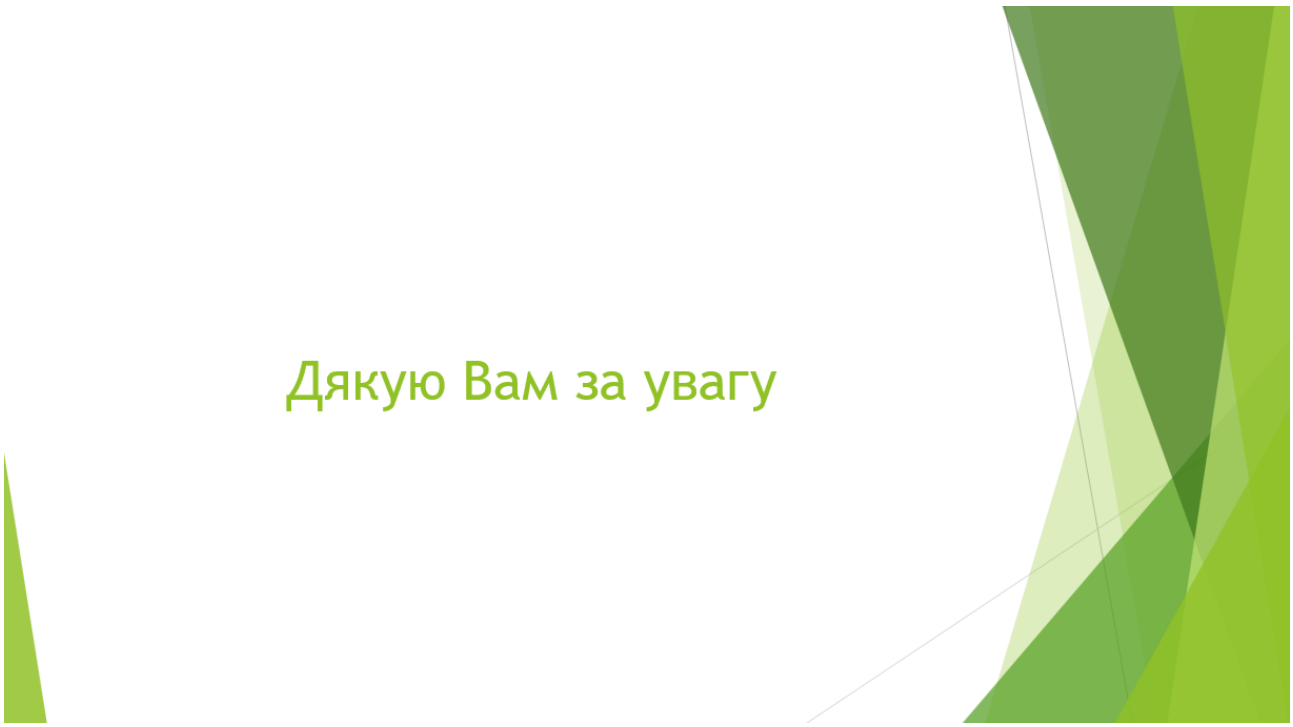
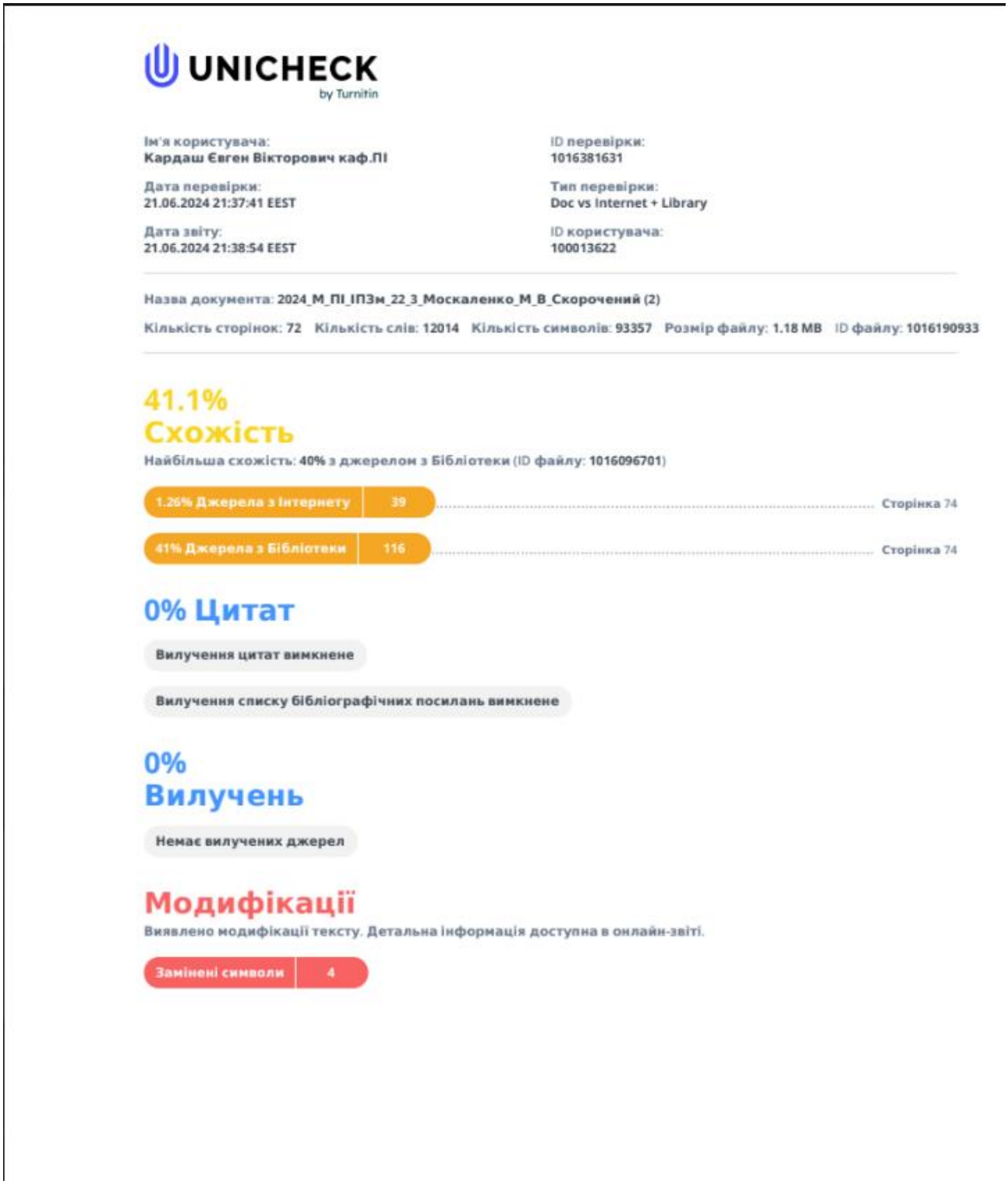


Рисунок В.19

ДОДАТОК Г

Результат перевірки на плагіат



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016381631

Дата перевірки:
21.06.2024 21:37:41 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
21.06.2024 21:38:54 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗм_22_3_Москаленко_М_В_Скорочений (2)

Кількість сторінок: 72 Кількість слів: 12014 Кількість символів: 93357 Розмір файлу: 1.18 MB ID файлу: 1016190933

41.1%
Схожість

Найбільша схожість: 40% з джерелом з Бібліотеки (ID файлу: 1016096701)

1.26% Джерела з Інтернету 39 ..... Сторінка 74

41% Джерела з Бібліотеки 116 ..... Сторінка 74

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 4

ДОДАТОК Ґ

Результат перевірки нормоконтролю

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-3 (група)

Москаленко Михайло Володимирович
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
7.3.3	Титульний аркуш входить до загальної нумерації сторінок звіту. Номер сторінки на титульному аркуші не проставляють.	1
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
7.7.4	Текст кожної позиції переліку треба починати з малої літери з абзацного відступу відносно попереднього рівня підпорядкованості.	23, далі за текстом.
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні позначки	
	7.15 Додатки	
Методичні вказівки до виконання кваліфікаційної роботи магістра… **ЗАТВЕРДЖЕНО** кафедрою ПІ протокол № 5 від 13.11.2023р. 3.2 Оформлення пояснювальної записки згідно з ДСТУ 3008:2015 Звіти у сфері науки і техніки. Структура та правила оформлювання. **Шаблон** затверджений засіданням кафедри №3 від 16.10.2023.	Увага! встановлені фіксовані береги: лівий – 25 мм., правий – 10 мм, верхній і нижній – 20 мм.	за текстом

Експерт _____ (підпис) Вадим НЕЧВОЛОД (прізвище, ініціали)

22.06.2024