

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Курсова робота з освітнього компоненту

«Моделювання систем. Курсова робота»

Тема: Задача про створення програмного керування рекламним бізнесом

|  |  |
| --- | --- |
| **Керівник**:  Стельмах Олександр Петрович  «Допущено до захисту»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 р.  Захищено з оцінкою  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Члени комісії:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Виконавець**:  Скачкова Анастасія Дмитрівна  студентка групи ІТ-02  залікова книжка № 0221  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_» \_\_\_\_ 2023 р.  Інна СТЕЦЕНКО  Олександр СТЕЛЬМАХ |

**Київ – 2023**

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра інформатики та програмної інженерії

Дисципліна «Моделювання систем»

Спеціальність Інженерія програмного забезпечення

Курс 4 Група ІТ – 02 Семестр 7

**ЗАВДАННЯ**

на курсову роботу студентки

Скачкова Анастасія Дмитрівна

(прізвище, ім'я, по батькові)

1. Тема роботи: Імітаційна модель роботи рекламного бізнесу на основі формалізму мережі масового обслуговування
2. Термін здачі студентом закінченої роботи "26" грудня 2023 р.
3. Вихідні дані до проекту

Завдання № 49 з Навчального Посібника

1. Зміст розрахунково-пояснювальної записки (перелік питань, що розробляються)
2. Розробка концептуальної моделі
3. Розробка формалізованої моделі
4. Програмна реалізація моделі
5. Проведення експериментів
6. Інтерпретація результатів експериментів.
7. Висновки
8. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Графічного матеріалу не має.

6. Дата видачі завдання "12" вересня 2023 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва етапів курсового проекту (роботи) | Термін  виконання  етапів роботи | Примітка |
| 1 | Отримання завдання | 04.10.2022 |  |
| 2 | Аналіз можливих методів вирішення поставленого завдання | 12.11.2022 |  |
| 3 | Розробка концептуальної моделі | 14.11.2022 |  |
| 4 | Перший контроль за процесом виконання курсового проекту (роботи), консультація у викладача |  |  |
| 5 | Опис імітаційної моделі | 20.11.2022 |  |
| 6 | Опис програмної реалізації імітаційної моделі | 26.11.2022 |  |
| 7 | Другий контроль за процесом виконання курсового проекту (роботи), консультація у викладача |  |  |
| 8 | Аналіз та оцінка результатів | 29.11.2022 |  |
| 9 | Оформлення пояснювальної записки | 16.12.2022 |  |
| 10 | Здача пояснювальної записки |  |  |
| 11 | Захист курсового проекту (роботи) |  |  |

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Скачкова А.Д.*

(підпис) (прізвище, ім'я, по батькові)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Стельмах О.П.*

(підпис) (прізвище, ім'я, по батькові)

«4» жовтня 2022 р.

**РЕФЕРАТ**

У даній роботі проведено дослідження та розробку реалізації системи масового обслуговування для роботи рекламного бізнесу.

Курсова робота: 47 с., 22 рисунки, 3 табл., 10 формул, 1 додаток із лістингом коду, 3 джерела літератури.

*Об'єкт дослідження* – система розробки програмного забезпечення для для контролю виходу реклам в ефір та керування паузами, фінансами.

*Мета роботи* – мінімізування періоду окупності.

*Метод дослідження* – імітаційне моделювання системи створення програмного забезпечення для контролю виходу реклам в ефір та керування паузами.

Результати моделювання використані для мінімізації періоду окупності.

**Ключові слова:** імітаційна модель, мережа масового обслуговування, дисперсійний аналіз, процес.

ЗМІСТ

[ПОСТАНОВКА ЗАВДАННЯ 6](#_Toc154400403)

[ВСТУП 8](#_Toc154400404)

[1 КОНЦЕПТУАЛЬНА МОДЕЛЬ 10](#_Toc154400405)

[2 ФОРМАЛІЗОВАНА МОДЕЛЬ 13](#_Toc154400406)

[3 РЕАЛІЗАЦІЯ МОДЕЛІ 16](#_Toc154400407)

[**3.1 Опис реалізації** 16](#_Toc154400408)

[**3.2 Верифікація алгоритму імітації** 28](#_Toc154400409)

[4 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ НА МОДЕЛІ 31](#_Toc154400410)

[**4.1 Тактичне планування факторного експерименту** 32](#_Toc154400411)

[**4.1 Стратегічне планування факторного експерименту** 32](#_Toc154400412)

[**4.3 Експериментування** 33](#_Toc154400413)

[**4.4 Інтерпретація результатів моделювання** 35](#_Toc154400414)

[ВИСНОВКИ 36](#_Toc154400415)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 37](#_Toc154400416)

[ДОДАТКИ 38](#_Toc154400417)

[**Додаток А. Лістинг коду** 38](#_Toc154400418)

[Додаток В. Критерій Фішера при α = 0.05 47](#_Toc154400419)

# **ПОСТАНОВКА ЗАВДАННЯ**

Для підвищення своїх прибутків працівники рекламного бізнесу знаходять все нові й нові засоби впливу реклами на людину. Їхнє досягнення – моментальне рекламне повідомлення. Компанія має у своєму розпорядженні декілька розвішаних по місту гучномовців, по яких протягом дня транслюються кращі радіопрограми зі вставками рекламних повідомлень. Протягом добового мовлення (час мовлення за добу – 16 годин) компанія має декілька пауз для розміщення реклами (тривалість пауз однакова, кількість пауз задати самостійно; загальна тривалість рекламних пауз не повинна перевищувати 10% від добового мовлення). Інтервал надходження рекламних повідомлень розподілений за законом Ерланга другого порядку із середнім часом 20 хвилин. Одне рекламне повідомлення виходить в ефір лише під час однієї рекламної паузи. Розмір рекламного повідомлення є величиною, рівномірно розподіленою на інтервалі 3±0,5 хвилин.

Вартість однієї хвилини мовлення обходиться компанії 100 гривень. За одну хвилину рекламного повідомлення замовник сплачує компанії 300 гривень. Якщо рекламне повідомлення виходить в ефір відразу, то компанія одержує 300×”тривалість одного повідомлення” гривень. Якщо рекламне повідомлення не може бути розміщене цілком, то замовник, який замовив рекламу, сплачує тільки за час розміщеної частини повідомлення, і при цьому йому ще надається знижка 10%. Якщо рекламне повідомлення не зразу виходить в ефір, то воно стає в чергу. Після 4 годин чекання рекламне повідомлення „застаріває” і видаляються з черги. За розміщене рекламне повідомлення, що знаходилося в черзі, компанія одержує на 30% менше від початкової суми. Якщо в рекламній паузі після розміщення рекламного повідомлення залишився вільний час, то компанія сплачує за нього як за звичайне мовлення.

Одноразові витрати (вартість устаткування, право мовлення і т.і.) компанії становлять 1000000 гривень.

Знайдіть оцінку періоду окупності *Т* у роках (компанії):

,

де p – прибуток, одержуваний за один рік, грн.; z – вартість місячного мовлення, грн.

Визначить кількість рекламних пауз, за якої період окупності буде мінімальним

# **ВСТУП**

Завдання оптимізації розміщення рекламних повідомлень у мовленні компанії стає важливим аспектом її ефективності та прибутковості. В умовах конкурентного ринку і швидкозмінних технологій важливо визначити оптимальний спосіб використання рекламних пауз для максимізації прибутку та скорочення періоду окупності витрат.

У діапазоні задач моделювання, це завдання класифікується як задача управління, оскільки воно включає в себе прийняття рішень щодо ефективного використання рекламного часу, оптимізацію витрат та максимізацію прибутку через правильне розміщення реклами.

Методологія вирішення поставленої задачі включає аналіз та моделювання процесів розміщення рекламних повідомлень, обрання кількості та діапазону рекламних повідомлень, а також визначення вартості рекламних пауз та їх вплив на фінансовий результат компанії.

Вибір імітаційного моделювання визначено його спроможністю адаптуватися до стохастичних процесів та здатністю відтворювати реальні умови функціонування системи.

Завдання можна вирішити, користуючись загальною структурою наведеною на Рисунку 1.1.

Изображение выглядит как текст, диаграмма, линия, Шрифт

Автоматически созданное описание

Рисунок 1.1 – Загальна структура моделі[1]

Тут X – множина вхідних змінних системи, Y – множина вихідних змінних системи, P – множина параметрів, F – функція, функціонал, алгоритм або формальне представлення залежності змінних Y від змінних X.

Моделі масового обслуговування описуються набором вхідних змінних, що у нашому випадку складається з інтенсивності вхідного потоку вимог на обслуговування, тривалості обслуговування вимоги кожним пристроєм, ймовірностей вибору маршруту та інших. Набір вихідних змінних моделі масового обслуговування складається з таких змінних як інтенсивність вихідного потоку вимог, середня довжина черги в місцях накопичення вимог та інших. У якості параметрів моделі масового обслуговування можуть розглядатись кількість пристроїв обслуговування, наявність черг у місцях накопичення вимог, наявність обмежень на кількість місць в чергах та інші. У якості алгоритму – алгоритм розрахунку вихідних змінних моделі за значенням вхідних змінних, який в залежності від складності моделі може мати вид простого розрахунку за математичними формулами або вид алгоритму імітації системи [1].

# **1 КОНЦЕПТУАЛЬНА МОДЕЛЬ**

Початковим процесом моделювання є системний аналіз об’єкта дослідження та формулювання цілі та задачі дослідження, визначення змінних та параметрів моделі [1][с. 8].

Концептуальна модель системи представляє деталізоване представлення системи, що моделюється, в якому виділяються і описуються підсистеми, елементи та зв’язки між елементами системи. Доцільно використовувати при описі концептуальної системи рисунки, що представляють структурні зв’язки системи, маршрути слідування об’єктів обслуговування і т.і. [1]

**Ціль моделювання** – визначення оптимальної кількості рекламних повідомлень в ефірі, з метою зменшення періоду окупності. Компанія здійснює розміщення рекламних повідомлень на гучномовцях протягом доби з певними обмеженнями та умовами. Модель визначає оптимальний спосіб розміщення рекламних повідомлень для досягнення максимального прибутку.

**Вхідні змінні включають** час моделювання, максимальну кількість рекламних пауз у чергах процесів, інтенсивність надходження рекламних повідомлень, розмір рекламних повідомлень, параметри розміщення реклами (тривалість пауз та їх кількість), розмір рекламного повідомлення, вартість радіомовлення, вартість рекламного часу, одноразові витрати.

**Вихідні змінні включають** кількість створених рекламних повідомлень, кількість оброблених рекламних повідомлень, відсоток повідомлень, яким відмовлено у показі, кількість рекламних повідомлень, що були видалені з черги через довгий час чекання, кількість рекламних повідомлень, що отримали знижку, кількість грошей за час моделювання, період окупності в роках, кількість пауз, при якій період окупності мінімальний (ключова мета компанії).

Оцінка періоду окупності я обчислила за допомогою формули 1.1:

, (1.1)

де S – одноразові витрати, грн., p – прибуток, одержуваний за один рік, грн., z – вартість мовлення за рік, грн.

Середне значення черги за час моделювання я обчислила за допомгою формули 1.2:

, (1.2)

де – спостережуване значення довжини черги; – проміжок часу, протягом якого ми спостерігали значення довжини черги; – загальний час моделювання системи.

Ймовірність відмови обчислюється за формулою 1.3:

, (1.3)

де Nв – кількість рекламних повідомлень, що не були опрацьовані, N – загальна кількість рекламних повідомлень.

З вихідних змінних можна варіювати кількістю рекламних пауз та їх тривалістю.

**Структурна схема моделі** включає в себе процес розміщення реклами на гучномовцях з урахуванням інтервалів надходження рекламних повідомлень та їх тривалості. Модель оцінює розміщення реклами, час очікування в черзі та вплив цих факторів на прибуток компанії.

Схему концептуальної моделі відображено на Рисунку 1.2:

Изображение выглядит как диаграмма, План, текст, рисунок

Автоматически созданное описание

Рисунок 1.2 – Схема концептуальної моделі

# **2 ФОРМАЛІЗОВАНА МОДЕЛЬ**

Обравши теоретичну базу моделювання, необхідно описати систему, що розглядається, обраними елементами формального опису і визначити для них усі необхідні параметри. Формальне представлення системи має вигляд схеми, в якій указані зв’язки між елементами системи та зв’язки із зовнішнім середовищем і указані параметри елементів системи. У формальній моделі міститься також інформація, яким чином будуть знайдені вихідні змінні моделі в результаті моделювання [1] [с. 14]. Тобто формалізована модель спрямована на точне визначення параметрів, взаємозв'язків та умов, що впливають на ефективність процесу мовлення та розміщення реклами на гучномовцях.

Найбільш формалізованим представленням є представлення у вигляді математичних формул. Менш формалізованим, але більш універсальним – представлення у вигляді схем, що відображають елементи системи та структурні взаємозв’язки між ними. Найбільш відомими і найбільш поширеними серед спеціалістів засобами формалізації процесів функціонування дискретних систем являються мережі масового обслуговування та мережі Петрі. Мережа Петрі надає перевагу, коли є необхідність формалізувати систему, що має спільні ресурси для багатьох систем обслуговування, використовує інформаційні дуги. Мережі масового обслуговування призначені для описування процесів обробки, таких як виробничі процеси, бізнес-процеси, і представляють великий, але чітко обмежений клас систем [1][с. 62]. Це і є мій випадок, маємо описування бізнес-процесу, а саме реалізацію роботи гучномовців рекламної компанії.

Для того, щоб представити систему **засобами мережі масового обслуговування** потрібно:

* з’ясувати, що являється в системі об’єктом обслуговування;
* виділити елементи процесу обслуговування об’єктів і кожному елементу поставити у відповідність СМО;
* для кожної СМО визначити кількість пристроїв та наявність черги;
* з’єднати СМО у відповідності до процесу обслуговування;
* визначити маршрут проходження об’єкту обслуговування від однієї СМО до іншої;
* визначити умови надходження в кожну СМО (ймовірність вибору маршруту та інші);
* визначити наявність блокування маршруту та умови блокування;
* визначити числові значення параметрів кожної СМО;
* визначити числові значення параметрів зовнішнього потоку на обслуговування;
* визначити стан мережі масового обслуговування на початку моделювання. [1]

**Для мого випадку:**

* Об'єкт обслуговування: Рекламні повідомлення.
* Умови надходження в кожну СМО: Ймовірність вибору маршруту між E1, E2 та Е3.
* Блокування маршруту та умови блокування: Блокування маршруту, якщо немає паузи.
* Числові значення параметрів: Час розміщення повідомлення, розмір рекламного повідомлення, час очікування в черзі.
* Числові значення параметрів зовнішнього потоку: Інтервал надходження повідомлень за розподілом Ерланга. тривалість пауз в рекламному мовленні.
* Стан мережі масового обслуговування на початку моделювання: усі СМО вільні, черги пусті, час паузи не вичерпано.

**Головні правила побудови мережі масового обслуговування:**

* Черга у СМО обов’язково одна. Кількість місць у черзі може бути обмежена або необмежена.
* Розгалуження маршруту означає, що маршрут вимоги може бути визначений тільки з визначеною ймовірністю. Сума ймовірностей у місці розгалуження маршруту складає одиницю, тому на одній із гілок розгалуження допускається, що ймовірність не указується
* Блокування маршрутів допомагають ввести в мережу МО елементи, що управляють процесом обслуговування.

Основні елементи мережі масового обслуговування наведені на Рисунку 2.1:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 2.1 – Елементи мережі масового обслуговування[1]

За допомогою елементів мережі масового обслуговування та поставленого завдання було реалізовано наступну мережу на Рисунку 2.2:

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок 2.2 – Мережа масового обслуговування

Розрахунки вихідних змінних відповідають Розділу 1.

# **3 РЕАЛІЗАЦІЯ МОДЕЛІ**

Для цієї роботи нам знадобиться бібліотека Numpy [3] для зручної роботи із математичними задачами цієї курсової роботи.

Імітація буде здійснюватися за наступним принципом: визначається момент найближчої події, потім просувається час в момент найближчої події та здійснюється відповідна подія. За принципом найближчої події модельний час просувається від моменту виникнення однієї події до моменту виникнення іншої, і після кожного просування часу реалізуються зміни стану моделі, відповідні до події, що виникла [1] [c. 164]. Імітація здійснюється виконанням упорядкованої у часі послідовності взаємозв’язаних подій [1] [c. 165]. Узагальнена схема роботи алгоритму імітації представлена на Рисунку 3.1:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 3.1 – Узагальнена схема алгоритму імітації[1]

## **3.1 Опис реалізації**

Структура файлів програмного забезпечення для моделювання на Рисунку 3.2:

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 3.2 – Структура файлів розробленого ПЗ

Для розробки програмної реалізації ми використали об’єктно-орієнтовний підхід. Структура файлів розробленого ПЗ відповідає діаграмі класів на Рисунку 3.3:

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рисунок 3.3 – Діаграма класів[1]

Сутності кожного файлу розробленого ПЗ представлені в Таблиці 3.1.

Таблиця 3.1 – Сутності кожного файлу проекту

|  |  |
| --- | --- |
| **Назва файлу** | **Пояснення** |
| *create.py* | Опис надходження рекламних повідомлень. |
| *element.py* | Містить у собі загальну характеристику елемента мережі масового обслуговування. Використовується для наслідування в інших файлах. |
| *fun\_rand.py* | Містить у собі функції, які описують закони розподілу (рівномірне, Ерланга). |
| *main.py* | Точка входу у програму. Задання імітаційної моделі. |
| *model.py* | Алгоритм імітації моделювання. |
| *process.py* | Опис системи масового обслуговування. |

Написаний алгоритм імітації знаходиться у файлі *model.py*, а тому це є головний файл у ієрархії наведеній вище. Точкою входу у програму служить файл *main.py*. У ньому ж ми і описали нашу модель. Побудовані імітаційні моделі використовують елементи формалізації *create.py* та *process.py*, які унаслідуються від файлу *element.py*. Оскільки програмне забезпечення написане на універсальній мові програмування, то просування стану моделі в часі відбувається за принципом орієнтованим на події.

На рисунку 3.4 представлений скріншот програмної реалізації алгоритму імітації моделі за завданням:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.5 – Реалізована мережа масового обслуговування за завданням

Відповідно до Рисунку 4.3 ми створюємо кілька об’єктів, а саме c типу Create – як елемент надходження, p1 типу Process – обробка повідомлень (вихід в ефір), аналогічним чином реалізовані p2 та p3, elements типу list – список з усіма елементами моделювання, model типу Model – безпосередньо сама модель для процесу моделювання. Для визначення мінімального періоду окупності моделі було взято різні значення кількості пауз зі списку max\_pause\_count.

Об’єкт create приймає свій іменний ідентифікатор, закон розподілу та параметри інтервалів затримок для обслуговування замовлення у різних відділах. Об’єкти p1, p2, p3 приймають параметрами свої іменні ідентифікатори, закон розподілу та довжину черги. Об’єкт .next\_element вказує маршрут слідування від одного об’єкту до іншого. Об’єкт model приймає список об’єктів імітаційної моделі і моделюється за допомогою методу симуляції .simulate, що приймає параметром час моделювання.

Стосовно методів у класах, їх наведено у Таблиці 3.2.

Таблиця 3.2 – Опис методів в основних класах

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва методу** | **Опис** | **Вхідні змінні** | **Вихідні змінні** |
| Клас Create | | | |
| *def \_\_init\_\_()* | Метод для ініціалізації об’єкту елементу мережі масового обслуговування | name – ім’я створюваного елементу, delay – середній час затримки, distribution – тип закону розподілу, delay\_dev – стандартне відхилення часу затримки,  probability – ймовірність вибору елементу | Відсутні |
| *def out\_act()* | Надходження  замовлення до  системи | Відсутні | Згенеровані х-ки  часових  затримок,  збільшення  кількості  замовлень, що  надійшли, й обрання наступного елементу за ймоввірністю |

Продовження Таблиці 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва методу** | **Опис** | **Вхідні змінні** | **Вихідні змінні** |
| Клас Element для наслідування | | | |
| *def \_\_init\_\_()* | Ініціалізація об’єкту елементу мережі масового обслуговування | name – ім'я елемента, delay – середній час затримки, distribution – тип закону розподілу, delay\_dev – стандартне відхилення часу затримки. | Відсутні |
| *def get\_delay()* | Генерування  часової затримки  за заданим законом  розподілу | Відсутні | Часова затримка за вказаним законом розподілу |
| *def get\_quantity ()* | Видача кількості рекламних повідомлень | Відсутні | Кількість рекламних повідомлень |
| *def get\_tcurr ()* | Видача поточного часу | Відсутні | Поточний час |
| *def in\_act()* | Вхід рекламного повідомлення до елементу | Відсутні | Відсутні, реалізація відноситься до класів, що наслідуються |
| *def out\_act()* | Вихід рекламного повідомлення з елементу | Відсутні | Збільшення  кількості  оброблених  рекламних повідомлень |
| *def get\_tnext()* | Видача наступного часу | Відсутні | Наступний час |
| *def set\_tnext()* | Отримання наступного часу | Наступний час | Відсутні |

Продовження Таблиці 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва методу** | **Опис** | **Вхідні змінні** | **Вихідні змінні** |
| Клас Element для наслідування | | | |
| *def print\_info()* | Показ інформації  про стан елементу  під час просування часу моделі | Відсутні | Інформація про  ім’я,  стан зайнятості,  кількість  оброблених рекламних повідомлень  та момент настання наступної події. |
| *def print\_result()* | Показ інформації моделювання в результатах | Відсутні | Інформація про ім’я та кількість оброблених |
| *def statistics()* | Збирання  статистичних  даних про  функціонування  моделі | delta – проміжок  часу  спостережуваних  величин | Відсутні, реалізація відноситься до класів, що наслідуються |
| Клас Process | | | |
| *def \_\_init\_\_()* | Ініціалізація  об’єкту системи  масового  обслуговування | name – ім'я процесу, delay – затримка для події, distribution –розподіл для затримки, delay\_dev – стандартне відхилення для затримки, max\_pause\_count –максимальна кількість пауз, maxi\_queue –максимальний розмір черги | Відсутні |
| *def update\_speaker()* | Оновлює параметри гучномовця (паузи) відповідно до вказаних умов (час настання паузи). | Відсутні | Час зміни інтервалу початку пауз, інформація чи відбувається пауза, лічильник пауз, тривалість, затрачена на паузи, час завершення поточної паузи |

Продовження Таблиці 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва методу** | **Опис** | **Вхідні змінні** | **Вихідні змінні** |
| Клас Process | | | |
| *def choose\_next\_el()* | Вибір наступного елемента згідно з ймовірністю або пріоритетом, або наступного в списку. | Відсутні | Наступний елемент для виконання |
| *def in\_act()* | Виконує вхідні дії (оновлює гучномовець, обробляє стан та викликає метод вхідних дій батьківського класу Element). | Відсутні | Відсутні |
| *def out\_act()* | Виконує вихідні дії (оновлює стан, обробляє чергу, викликає метод вихідних дій батьківського класу Element). | Відсутні | Збільшення кількості замовлень, що були оброблені, розрахований момент наступної події. Збільшення кількості вилучених рекламних повідомлень з черги за необхідності. Розрахунок збільшення прибутку та часу отримання прибутку. |
| *def get\_failure()* | Повертає кількість відхилених рекламних повідомлень. | Відсутні | Кількість помилок |
| *def print\_info()* | Виводить інформацію про кількість відхилених рекламних повідомлень та чергу. | Відсутні | Інформація про кількість відхилених рекламних повідомлень та чергу. |
| *def do\_statistics()* | Збирання даних про  середнє значення черги | delta – інтервал  часу  спостереження середнього значення черги | Середнє значення черги та за спостережуваний інтервал часу (за необхідності клас може розширюватись) |

Продовження Таблиці 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва методу** | **Опис** | **Вхідні змінні** | **Вихідні змінні** |
| Клас Model | | | |
| *def \_\_init\_\_()* | Ініціалізація  об’єкту імітаційної моделі | elements – список елементів моделі | Відсутні |
| *def simulate()* | Алгоритм імітації моделювання системи | time – загальний час моделювання | Стан моделі на кожному кроці просування часу моделі, статистичні дані моделі, загальні результати моделювання |
| *def print\_info()* | Вивід інформації про заданий елемент моделі | Відсутні | Виведення  інформації, а саме стан, кількість оброблених та час наступного повідомлення, у  поточний час |
| *def print\_result()* | Вивід інформації про результат моделювання | Відсутні | Інформація про  середню довжину черги, кількість відмов, кількість видалених повідомлень з черги, кількість повідомлень зі знижкою, кількість елементів у черзі, кількість пауз, час, затрачений на паузи, гроші за час моделювання, період окупності. |

Стосовно основних умов , було додано додатковий метод, описаний раніше більш стисло для запуску паузи update\_speakers, що використовується при кожному надходженні повідомлення. На рисунку 3.5 метод активує паузу, якщо загальний час пауз не перевищує 10% від загального часу моделювання, збільшує лічильник пауз, розраховує час, затрачений на паузи та час закінчення паузи.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.5 – Реалізація запуску паузи

Також було додано розрахунок прибутку з кожного рекламного повідомлення у випадку повного та часткового показу в ефірі при надходженні на Рисунку 3.6. Для статистики розраховую кількість елементів зі знижкою у discount.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.6 – Реалізація розрахунку одержаного прибутку за рекламні повідомлення

Якщо повідомлення обробляється після черги, його прибуток рахується за зниженою ставкою на 30% на Рисунку 3.7.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.7 – Реалізація розрахунку вартості повідомлення в ефірі за зниженою ставкою

Реалізацію умови, що після 4 годин чекання в черзі повідомлення “застаріває” та видаляється з черги, представлено на Рисунку 3.8. Для виконання було додано список із часом потрапляння повідомлень в чергу, в який заносяться значення під час додавання в чергу відповідно у методі in\_act(), представлено на Рисунку 3.9.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 3.8 – Реалізація видалення повідомлення з черги після 4 годин чекання

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.9 – Момент додавання до списку time\_el\_in\_queue

Умова збільшує лічильник видалених елементів remove для виводу статистики та обнуляє час у списку time\_el\_in\_queue для видаленого елементу(значення в списку обнуляються і при переході елементу на опрацювання з черги відповідно). Розрахунок відмов також був виконаний під час видалення замовлення з черги після 4 годин чекання.

Середня кількість елементів у черзі була розрахована, як вже було зазначено в Таблиці 3.2 у методі do\_statistics() та представлена на Рисунку 3.9

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

Рисунок 3.9 – Реалізація розрахунку середньої кількості елементів у черзі

Збільшення кількості елементів реалізовано у простому методі out\_act() класу Element, що наслідується класом Process та Create, та представлено на Рисунку 3.10.

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 3.10 – Реалізація збільшення лічильника кількості

Стосовно роботи загально, у часовий момент 0 надходить рекламне повідомлення до гучномовців. Гучномовець формує часові затримки для обробки. На рисунку 3.10 представлена робота системи на початку моделювання:

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рисунок 3.10 – Стан системи на початок моделювання

Далі для алгоритму імітації розраховується наступний момент часу для елементів моделі, у кожного елементу ця характеристика представлена значенням t\_next. За допомогою характеристики next\_element створюємо зв’язок між елементами системи.

Результати роботи моделі представлені на Рисунку 3.10 та Рисунку 3.11. Зображено найкращий результат з 10 прогонів.

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рисунок 3.10 – Початок результату моделювання

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.11 – Кінець результату моделювання

За час моделювання 960 хвилин за умовою, ми отримали 56 рекламних повідомлень, з них першим громкоговорювачем обробилось 18, другим – 21, третім – 17. Максимальна кількість пауз 21. Кількість часу, затраченого на рекламу першим громкоговорювачем – 57, другим – 63, третім – 51. Час однієї паузи – 3, одноразові витрати становлять 1000000, період окупності – 1.9.Також загально ймовірність відмови, середня довжина черги нульові.

## **3.2 Верифікація алгоритму імітації**

Оскільки метою моделювання є знаходження значення максимальної кількості пауз, за якого період окупності мінімальний, в першу чергу було досліджено змінну max\_pause\_count. Робилось по 6 прогонів одного значення (більше поки не бралось для звіту через можливу незручність читання, менше не здається надійним). Згенерована верифікація представлена на Рисунку 3.12.

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Рисунок 3.12 – Верифікація максимальної кількості пауз

Значення для максимальної кількості пауз були обрані з урахуванням максимально можливого за умови 10 відсотків від часу усього моделювання з урахуванням тривалості паузи 3.5 та на основі тестування, коли максимальне значення пауз стає неосяжним для процесів. Можемо бачити, що найоптимальнішим значенням максимальної кількості пауз при тривалості паузи 3.5 є 25, що до того ж підтверджує середнє між прогонами.

Також я захотіла перевірити вплив тривалості черги на період окупності, результати верифікації представлено на Рисунку 3.13.

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Рисунок 3.13 – Верифікація тривалості пауз

Бачимо пряму залежність, чим більше тривалість паузи, тим менше повідомлень, що не були показані повністю. Відповідно період окупності в роках за тривалості паузи 3.5 та 4 значно зменшується у зрівнянні з тривалістю 3.

# **4 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ НА МОДЕЛІ**

Експериментування є процес використання моделі з метою отримання і аналізу інформації про властивості модельованої системи. Спеціальні засоби планування експериментів дозволяють скласти план проведення експериментів, що забезпечує збір необхідної для отримання обґрунтованих висновків інформації, при мінімальних витратах.

На першому етапі планування потрібно задати:

* кількість варійованих факторів (факторами в теорії планування експериментів називають вхідні змінні, вплив яких на вихідну змінну, що називається відгуком моделі, досліджується);
* кількість рівнів кожного фактора (рівнями називають ті кількісні або якісні значення фактора, при яких можливо проводити експеримент);
* область змінювання факторів;
* необхідну точність та довірчу ймовірність вимірювання відгуку моделі.

На другому етапі приступають до *тактичного планування* експерименту, метою якого є визначення таких умов експерименту, що забезпечать вимір відгуку моделі з заданою точністю та довірчою ймовірністю. В результаті цього етапу планування повинні бути визначені тривалість одного прогону (в одиницях модельного часу) і кількість прогонів.

На третьому етапі приступають до *стратегічного планування* експериментів, метою якого є визначення такої серії експериментів, що забезпечить отримання бажаної інформації про відгук моделі при мінімальних витратах. [1] [с.226].

Основна ціль моделювання даної системи – визначення періоду окупності з урахуванням кількості пауз, що рахується як ділення між всіма витратами (за рік) включно з одноразовими та прибутком (за рік). Значенням цільової функції є мінімізація періоду окупності. Тому у якості дослідження було обрано вплив значення кількості пауз на період окупності.

## **4.1 Тактичне планування факторного експерименту**

Нам потрібно визначити яка кількість прогонів буде оптимальною для визначення статистики. Згідно з наведеною формулою Чебишева 4.1, визначено необхідну кількість запусків алгоритму.

(4.1),

За умови, що довірча ймовірність дорівнює β=0,95 і точність вимірювання відгуку моделі рівна дисперсії відгуку моделі, то кількість прогонів буде 20, що представлено на Рисунку 4.1. Отже, для всіх подальших експериментів будемо робити по 20 прогонів.

A table with numbers and symbols

Description automatically generated

Рисунок 4.1 — Кількість прогонів за нерівністю Чебишева

Дослідження часу моделювання не є необхідним виходячи з умови завдання із точним зазначенням часу моделювання для розрахунку витрат та прибутку за час роботи у добу.

## **4.1 Стратегічне планування факторного експерименту**

При дослідженні впливу якісних факторів метою експерименту є якісна оцінка впливу факторів, тобто відповідь на запитання «впливає чи не впливає значення фактору на відгук моделі». Для якісної оцінки впливу факторів використовують *дисперсійний*аналіз.[1][с.231] Основне питання, на яке дає відповідь дисперсійний аналіз впливу фактора, формулюється так: різниця у значеннях відгуку моделі, отриманих при різних значеннях фактора обумовлена випадковістю, чи пояснюється виключно дією фактора?

## **4.3 Експериментування**

У якості фактору виступає значення кількості пауз, а в якості відгуку – значення періоду окупності. Відображення впливу фактору представлено на Рисунку 4.2, Таблиці 4.1 та Таблиці 4.2.

Таблиця 4.1 – Залежність середнього значення періоду окупності після 20 прогонів від кількості пауз:

|  |  |
| --- | --- |
| **Кількість пауз** | **Середнє значення періоду окупності при 20 прогонах** |
| 6 | 5,9 |
| 10 | 3,5 |
| 13 | 2,8 |
| 15 | 2,4 |
| 17 | 2,3 |
| 19 | 2,2 |
| 21 | 2,1 |

Рисунок 4.2 – Графік залежності впливу кількості пауз на період окупності

Таблиця 4.1 – Вплив максимальної кількості пауз на період окупності:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***К-сть пауз*** | ***№ прогону для розрахунку періоду окупності T*** | | | | | | | | | | | | | | | | | | | | ***Середнє значення T*** |
| *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* | *14* | *15* | *16* | *17* | *18* | *19* | *20* |
| 6 | 5,9 | 5,8 | 5,9 | 6,2 | 6,2 | 6,6 | 6 | 5,7 | 5,7 | 5,9 | 5,9 | 5,9 | 5,9 | 5,8 | 6,3 | 5,9 | 5,9 | 5,8 | 5,8 | 5,9 | 5,9 |
| 10 | 2,6 | 2,8 | 2,8 | 2,6 | 2,8 | 2,7 | 2,8 | 2,8 | 2,7 | 2,8 | 2,5 | 2,6 | 3 | 3 | 2,9 | 2,7 | 2,9 | 2,7 | 2,7 | 2,7 | 2,8 |

Кількість рівнів найчастіше приймають рівним двом, що відповідає припущенню про лінійну залежність відгуку моделі від факторів. Замітимо, що таке припущення майже завжди можна зробити при достатньо невеликих інтервалах варіювання факторів. Надалі розглядається саме випадок двох рівнів.

Розрахуємо середнє значення періоду окупності:

*.*

Згідно з Формулами 4.2, 4.3 та 4.4 виконаємо дисперсійний аналіз для оцінки впливу фактору.

(4.2),

(4.3),

(4.4).

Згідно виконаному дослідженню припустимо, що кількість рівнів p = 5.

,

Визначимо факторну та залишкову дисперсії за Формулами 4.6 та 4.7.

(4.5),

(4.6),

(4.7).

Кількість ступенів вільності дорівнює кількості пауз, тобто 2.

Визначимо значення критерію Фішера за Формулою 4.8:

(4.8).

Отримали F = 3203,3.

Критичне значення критерію Фішера знаходять на основі рівня значимості α = 0.05, ступенів вільності числівника та ступенів вільності знаменника (Додаток E). Якщо , то середні та розрізняються значимо і вплив фактора є значимим. В противному випадку вплив фактора визнається не значимим, а різниця у середніх значеннях пояснюється випадковими причинами.

В нашому випадку 38. Отримали .

Отже, , а тому вплив фактору визнаємо значним. Це значить, що значення максимальної кількості пауз системи буде впливати на період окупності.

## **4.4 Інтерпретація результатів моделювання**

Інтерпретація результатів моделювання вказує на значущий вплив кількості пауз на період окупності системи. Для аналізу цього впливу було проведено однофакторний дисперсійний аналіз, використовуючи визначені критерії та формули.

Згідно отриманих результатів, середнє значення періоду окупності змінюється від 5,9 до 2,1 при різних кількостях пауз. Графічне представлення цих змін на Рисунку 4.2 вказує на зростання періоду окупності при зменшенні кількості пауз.

Дисперсійний аналіз підтверджує статистичну значущість впливу кількості пауз. Значення критерію Фішера F значно перевищує критичне значення , що свідчить про значущість взаємозв'язку. Таким чином, вплив фактору (кількість пауз) є статистично значущим.

# **5 ІНТЕРПРЕТАЦІЯ РЕЗУЛЬТАТІВ МОДЕЛЮВАННЯ**

Інтерпретація результатів моделювання вказує на значущий вплив кількості пауз на період окупності системи. Для аналізу цього впливу було проведено однофакторний дисперсійний аналіз, використовуючи визначені критерії та формули.

Згідно отриманих результатів, середнє значення періоду окупності змінюється від 5,9 до 2,1 при різних кількостях пауз. Графічне представлення цих змін на Рисунку 4.2 вказує на зростання періоду окупності при зменшенні кількості пауз.

Дисперсійний аналіз підтверджує статистичну значущість впливу кількості пауз. Значення критерію Фішера F значно перевищує критичне значення , що свідчить про значущість взаємозв'язку. Таким чином, вплив фактору (кількість пауз) є статистично значущим.

Впровадження оптимального рішення, спрямованого на збільшення кількості пауз, може призвести до наступних переваг:

* Зменшення періоду окупності: Оптимізація робочого процесу призведе до скорочення часу, необхідного для повернення інвестицій.
* Підвищення продуктивності: Збільшення кількості пауз сприятиме більш ефективному використанню ресурсів та покращенню робочого процесу.
* Збільшення прибутковості: Швидше окуплення інвестицій і підвищення продуктивності можуть позитивно вплинути на прибутковість системи.

# **ВИСНОВКИ**

Під час виконання курсової роботи поставлене завдання було вирішено за допомогою розробленого програмного забезпечення та розробленої імітаційної моделі, проведено експериментування для знаходження найкращих значень факторів, щоб досягнути цілі моделювання. Для власників рекламного бізнесу, що контролюють показ реклами між радіорограмами, дане програмне забезпечення має бути корисним та зручним у використанні. Простота зміни вхідних даних та результати моделювання надають можливість дослідити роботу системи та налагодити або розширити роботу під свою необхідність.

Повний код реалізації можна знайти у додатках.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Стеценко І.В. Моделювання систем: навч. посіб. / І.В. Стеценко; М-во освіти і науки України, Черк. держ. технол. ун-т. – Черкаси: видавництво “Маклаут”, 2011. – 501 с.
2. Аністратенко В. О., Федоров В. Г. Математичне планування експериментів в АПК: Навч. Посібник. – К.: Вища шк., 1993. – 375 с.
3. NumPy [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/

# **ДОДАТКИ**

## **Додаток А. Лістинг коду**

Файл “create.py”:

import numpy as np  
  
import element as e  
  
  
class Create(e.Element):  
 def \_\_init\_\_(self, name, delay, distribution, delay\_dev=None):  
 super().\_\_init\_\_(name, delay, distribution, delay\_dev)  
 self.probability = [1]  
  
  
 def out\_act(self):  
 super().out\_act()  
 self.set\_tnext(self.get\_tcurr() + self.get\_delay())  
 selected\_element = np.random.choice(a=self.next\_element, p=self.probability)  
 selected\_element.in\_act()

Файл “element.py”:

import fun\_rand as fun  
  
  
class Element:  
 nextId = 0  
  
 def \_\_init\_\_(self, name, delay=None, distribution=None, delay\_dev=None):  
 self.distribution = distribution#  
 self.t\_next = 0  
 self.t\_curr = self.t\_next  
 self.delay\_mean = delay#  
 self.delay\_dev = delay\_dev#  
 self.quantity = 0  
 self.state = 0  
 self.next\_element = None#  
 self.id = Element.nextId  
 Element.nextId += 1  
 self.name = name  
  
 def get\_delay(self):  
 if 'exp' == self.distribution:  
 return fun.exp(self.delay\_mean)  
 elif 'norm' == self.distribution:  
 return fun.norm(self.delay\_mean, self.delay\_dev)  
 elif 'uniform' == self.distribution:  
 return fun.uniform(self.delay\_mean, self.delay\_dev)  
 elif 'erlang' == self.distribution:  
 return fun.erlang(self.delay\_mean, self.delay\_dev)  
 else:  
 return self.delay\_mean  
  
 def get\_quantity(self):  
 return self.quantity  
  
 def get\_tcurr(self):  
 return self.t\_curr  
  
 def set\_tcurr(self, t\_curr):  
 self.t\_curr = t\_curr  
  
  
 def in\_act(self):  
 pass  
  
 def out\_act(self):  
 self.quantity += 1  
  
 def get\_tnext(self):  
 return self.t\_next  
  
 def set\_tnext(self, t\_next):  
 self.t\_next = t\_next  
  
 def print\_result(self):  
 print(self.name + " quantity = " + str(self.quantity))  
  
 def print\_info(self):  
 print(self.name + " state= " + str(self.state) + " quantity = " + str(self.quantity) + " tnext= " + str(  
 self.t\_next))  
  
 def do\_statistics(self, delta):  
 pass

Файл “model.py”:

from process import Process  
import numpy as np  
  
class Model:  
 def \_\_init\_\_(self, elements):  
 self.list = elements  
 self.t\_next = 0.0  
 self.event = 0  
 self.t\_curr = self.t\_next  
 self.quantity = 0  
 self.pause\_count = 0  
  
 self.discounts = 0  
 self.time\_profit = 0  
 self.profit = 0  
 self.expenses = 0  
 self.pause\_duration = 0  
 self.onetime\_expenses = 1000000  
 self.money = 0  
 self.in\_queue = 0  
  
 self.profit\_for\_one\_year = 0  
 self.expenses\_for\_one\_year = 0  
 self.payback\_period = 0  
  
 self.curr\_pause\_duration = 0  
 self.tot\_expenses = 0  
  
 def simulate(self, time):  
 while self.t\_curr < time:  
 self.t\_next = float('inf')  
  
 for e in self.list:  
 t\_next\_val = np.min(e.t\_next)  
 if t\_next\_val < self.t\_next:  
 self.t\_next = t\_next\_val  
 self.event = e.id  
  
 for e in self.list:  
 e.do\_statistics(self.t\_next - self.t\_curr)  
  
 self.t\_curr = self.t\_next  
  
 for e in self.list:  
 e.t\_curr = self.t\_curr  
  
 if len(self.list) > self.event:  
 self.list[self.event].out\_act()  
  
 for e in self.list:  
 if np.any(self.t\_curr == e.t\_next):  
 e.out\_act()  
  
 self.print\_info()  
  
 return self.print\_result  
  
 def print\_info(self):  
 for e in self.list:  
 e.print\_info()  
  
 @property  
 def print\_result(self):  
 print("\n-------------RESULTS-------------")  
 result = list()  
 for e in self.list:  
 e.print\_result()  
 result.append(e.get\_quantity())  
 if isinstance(e, Process):  
 p = e  
 self.quantity += p.get\_quantity()  
 self.discounts += p.discount  
 self.in\_queue += p.count\_in\_queue  
 self.pause\_count += p.pause\_count  
 self.pause\_duration = p.pause\_duration  
 self.curr\_pause\_duration = p.curr\_pause\_duration  
 self.profit += p.profit  
 self.time\_profit += p.time\_profit  
 self.expenses = (self.t\_curr - p.time\_profit)\*100  
 self.tot\_expenses += self.expenses  
 print(f"Mean length of queue = {str(p.get\_mean\_queue() / self.t\_curr)}")  
 print(f"Failure probability = {str(p.get\_failure() / float(p.get\_quantity()))}")  
 print(f"Remove = {p.remove}")  
 print(f"Elements with discount = {p.discount}")  
 print(f"Count elements in queue = {p.count\_in\_queue}")  
 print(f"Pause count = {p.pause\_count}")  
 print(f"Current pause duration = {p.curr\_pause\_duration}")  
 result.append(  
 [p.get\_mean\_queue() / self.t\_curr, p.get\_failure() / float(p.get\_quantity()), p.remove, p.discount])  
 print()  
  
 print(f"Pause duration = {self.pause\_duration}")  
 print(f'Discounts = {self.discounts}')  
 print(f"One-time expenses = {self.onetime\_expenses}")  
  
 self.money = self.profit - self.tot\_expenses  
 self.expenses\_for\_one\_year = self.expenses \* 16 \* 365  
 self.profit\_for\_one\_year = self.profit \* 16 \* 365  
 self.payback\_period = (self.onetime\_expenses + self.expenses\_for\_one\_year) / self.profit\_for\_one\_year  
  
 result.append([self.pause\_duration, self.money, self.expenses\_for\_one\_year, self.profit\_for\_one\_year,  
 self.payback\_period])  
 print(f"Payback period (in years) = {self.payback\_period}")  
  
 print(result)  
 return result

Файл “fun\_rand.py”:

import random  
import numpy as np  
  
  
def exp(time\_mean):  
 a = 0.0  
 while a == 0:  
 a = random.uniform(0, 1)  
 a = -time\_mean \* np.log(a)  
 return a  
  
  
def uniform(time\_min, time\_max):  
 a = 0.0  
 while a == 0:  
 a = random.random()  
 a = time\_min + a \* (time\_max - time\_min)  
 return a  
  
  
def norm(time\_mean, time\_deviation):  
 return time\_mean + time\_deviation \* random.gauss(0.0, 1.0)  
  
  
def erlang(time\_mean, k):  
 return np.random.gamma(k, time\_mean / k)

Файл “process.py”:

from copy import deepcopy  
from datetime import datetime  
  
import numpy as np  
import element as e  
  
  
class Process(e.Element):  
 def \_\_init\_\_(self, name, delay=None, distribution=None, delay\_dev=None, max\_pause\_count=100000, maxi\_queue=100000):  
 super().\_\_init\_\_(name, delay, distribution, delay\_dev)  
 self.queue = 0  
 self.max\_queue = maxi\_queue  
 self.mean\_queue = 0.0  
 self.failure = 0  
 self.state = 0  
 self.t\_next = np.inf  
 self.probability = [1]  
 self.priority = [1]  
  
 self.is\_pause = False # 1-pause, 0-radio  
  
 self.curr\_pause\_duration = 0  
 self.max\_pause\_duration = 0.1 \* 16 \* 60  
  
 self.max\_pause\_count = max\_pause\_count  
 self.pause\_duration = 3.5 # self.max\_pause\_duration / self.max\_pause\_count # тривалість пауз однакова, к-сть пауз задати самостійно  
 self.pause\_count = 0  
 self.end\_t\_curr\_pause = 0  
  
 self.max\_queue\_time = 4 \* 60  
 self.time\_el\_in\_queue = [None] \* self.max\_queue  
 self.remove = 0  
  
 self.discount = 0  
 self.delay = 0  
 self.time\_profit = 0.0  
  
 self.count\_in\_queue = 0  
  
 self.profit = 0.0  
  
 def update\_speaker(self):  
 if self.curr\_pause\_duration < self.max\_pause\_duration and self.pause\_count < self.max\_pause\_count:  
 #print(f'CHECK PAUSE {self.is\_pause}')  
 self.is\_pause = True  
 self.pause\_count += 1  
 self.curr\_pause\_duration += self.pause\_duration  
 self.end\_t\_curr\_pause = self.t\_curr + self.pause\_duration  
  
 def choose\_by\_priority(self):  
 priorities = deepcopy(self.priority)  
 min\_queue = float('inf')  
 with\_min\_q\_index = 0  
 for p in range(len(priorities)):  
 if min(priorities) == float('inf'):  
 break  
 pr\_index = priorities.index(min(priorities))  
 if 0 in self.next\_element[pr\_index].state:  
 return self.next\_element[pr\_index]  
 else:  
 if min\_queue > self.next\_element[pr\_index].queue:  
 min\_queue = self.next\_element[pr\_index].queue  
 with\_min\_q\_index = self.next\_element.index(self.next\_element[pr\_index])  
  
 priorities[pr\_index] = 10000000  
 return self.next\_element[with\_min\_q\_index]  
  
 def choose\_next\_el(self):  
 if self.probability == [1] and self.priority == [1]:  
 return self.next\_element[0]  
 elif self.probability != [1] and self.priority != [1]:  
 raise Exception('Error: Probability and priority are both defined.')  
 elif self.probability != [1]:  
 next\_element = np.random.choice(a=self.next\_element, p=self.probability)  
 return next\_element  
 elif self.priority != [1]:  
 next\_element = self.choose\_by\_priority()  
 return next\_element  
  
 def in\_act(self):  
 self.update\_speaker()  
 if self.state == 0 and self.is\_pause:  
 self.state = 1  
 self.delay = self.get\_delay()  
 if self.delay > self.pause\_duration:  
 self.discount += 1  
 self.profit += 0.9 \* self.pause\_duration \* 300  
 self.time\_profit += self.pause\_duration  
 self.t\_next = self.t\_curr + self.delay  
 self.is\_pause = False  
 else:  
 self.profit += self.delay \* 300  
 self.time\_profit += self.delay  
 self.t\_next = self.t\_curr + self.delay  
 self.is\_pause = False  
 else:  
 if self.queue < self.max\_queue:  
 self.queue += 1  
 self.count\_in\_queue += 1  
 self.time\_el\_in\_queue[self.queue - 1] = self.t\_curr  
 else:  
 self.failure += 1  
  
 def out\_act(self):  
 super().out\_act()  
 self.t\_next = np.inf  
 self.state = 0  
 for i, time\_in\_queue in enumerate(self.time\_el\_in\_queue):  
 if time\_in\_queue is not None and self.t\_curr - time\_in\_queue >= self.max\_queue\_time:  
 print('!!!=========================!!!')  
 print(f'Element removed from the queue: {self.name}')  
 print(f'Time spent in queue: {self.t\_curr - time\_in\_queue}')  
 self.queue -= 1  
 self.remove += 1  
 self.time\_el\_in\_queue[i] = None  
 self.failure += 1  
 if self.queue > 0:  
 self.update\_speaker()  
 if self.is\_pause:  
 self.queue -= 1  
 self.state = 1  
 self.delay = self.get\_delay()  
 if self.delay > self.pause\_duration:  
 self.profit += 0.7 \* self.pause\_duration \* 300 \* 0.1  
 self.time\_profit += self.pause\_duration  
 else:  
 self.profit += 0.7 \* self.delay \* 300  
 self.time\_profit += self.delay  
 self.t\_next = self.t\_curr + self.delay  
 if self.next\_element is not None:  
 next\_el = self.choose\_next\_el()  
 next\_el.in\_act()  
  
 def get\_failure(self):  
 return self.failure  
  
  
 def print\_info(self):  
 super().print\_info()  
 print(f'failure= {str(self.failure)}, queue= {str(self.queue)}')  
  
 def get\_mean\_queue(self):  
 return self.mean\_queue  
  
 def do\_statistics(self, delta):  
  
 self.mean\_queue = self.get\_mean\_queue() + self.queue \* delta

Файл “main.py”:

import numpy as np  
import openpyxl  
from openpyxl.styles import PatternFill, Font  
  
from create import Create  
from tabulate import tabulate  
from model import Model  
from process import Process  
import pandas as pd  
  
#  
# c = Create('CREATOR', 20, 'erlang', 2)  
#  
# p1 = Process('SPEAKER-1', 2.5, 'uniform', 3.5, 9, 5)  
# p2 = Process('SPEAKER-2', 2.5, 'uniform', 3.5, 9, 5)  
# p3 = Process('SPEAKER-3', 2.5, 'uniform', 3.5, 9, 5)  
#  
# c.next\_element = [p1, p2, p3]  
# c.probability = [1 / 3, 1 / 3, 1 / 3]  
#  
# # p1.next\_element = [p2, p3]  
# # p1.probability = [0.5, 0.5]  
# # p2.next\_element = [p3, p1]  
# # p2.probability = [0.5, 0.5]  
# # p3.next\_element = [p2, p1]  
# # p3.probability = [0.5, 0.5]  
#  
# # p1.max\_queue = 5  
# # p2.max\_queue = 5  
# # p3.max\_queue = 5  
#  
# # elements = [c, p1]  
# elements = [c, p1, p2, p3]  
# model = Model(elements)  
# res = model.simulate(16 \* 60)  
  
workbook = openpyxl.Workbook()  
sheet = workbook.active  
  
min\_payback\_period = float('inf')  
sum\_payback\_period = 0  
min\_payback\_period\_row = 0  
tests = 48  
max\_pause\_count = [6, 6, 6, 6, 6, 6,  
 10, 10, 10, 10, 10, 10,  
 13, 13, 13, 13, 13, 13,  
 15, 15, 15, 15, 15, 15,  
 17, 17, 17, 17, 17, 17,  
 19, 19, 19, 19, 19, 19,  
 21, 21, 21, 21, 21, 21,  
 25, 25, 25, 25, 25, 25]

pause\_duration = [3, 3, 3.5, 3.5, 4, 4, 3, 3, 3.5, 3.5, 4, 4, 3, 3, 3.5, 3.5, 4, 4, 3, 3, 3.5, 3.5, 4, 4, 3, 3, 3.5, 3.5, 4, 4, 3, 3, 3.5, 3.5, 4, 4, 3, 3, 3.5, 3.5, 4, 4, 3, 3, 3.5, 3.5, 4, 4]  
column\_names = [  
 'quantity\_create',  
 'max\_pause\_count',  
 'pause\_duration',  
 'quantity\_p1',  
 'fail1',  
 'queue1',  
 'remove1',  
 'count\_discount1',  
 'quantity\_p2',  
 'fail2',  
 'queue2',  
 'remove2',  
 'count\_discount2',  
 'quantity\_p3',  
 'fail3',  
 'queue3',  
 'remove3',  
 'count\_discount3',  
 'Payback period in years',  
 'Avg Payback Period in years'  
]  
  
for i in range(tests):  
 c = Create('CREATOR', 20, 'erlang', 2)  
  
 p1 = Process('SPEAKER-1', 2.5, 'uniform', 3.5, max\_pause\_count[i], 5)  
 p2 = Process('SPEAKER-2', 2.5, 'uniform', 3.5, max\_pause\_count[i], 5)  
 p3 = Process('SPEAKER-3', 2.5, 'uniform', 3.5, max\_pause\_count[i], 5)  
  
 c.next\_element = [p1, p2, p3]  
 c.probability = [1/3, 1/3, 1/3]  
 p1.pause\_duration = 3.5 #pause\_duration[i]  
 p2.pause\_duration = 3.5 #pause\_duration[i]  
 p3.pause\_duration = 3.5 #pause\_duration[i]  
  
 elements = [c, p1, p2, p3]  
 model = Model(elements)  
 res = model.simulate(16 \* 60)  
 if res[-1][4] < min\_payback\_period:  
 min\_payback\_period = res[-1][4]  
 min\_payback\_period\_row = i + 2  
  
 result1 = {  
 'quantity\_create': res[0],  
 'max\_pause\_count': max\_pause\_count[i],  
 'pause\_duration': res[-1][0],  
 'quantity\_p1': res[1],  
 'fail1': res[2][1],  
 'queue1': res[2][0],  
 'remove1': res[2][2],  
 'count\_discount1': res[2][3],  
 'quantity\_p2': res[3],  
 'fail2': res[4][1],  
 'queue2': res[4][0],  
 'remove2': res[4][2],  
 'count\_discount2': res[4][3],  
 'quantity\_p3': res[5],  
 'fail3': res[6][1],  
 'queue3': res[6][0],  
 'remove3': res[6][2],  
 'count\_discount3': res[6][3],  
 'Payback period in years': res[-1][4],  
 'Avg Payback Period in years': None  
 }  
 sum\_payback\_period += res[-1][4]  
 if (i+1) % 20 == 0:  
 avg\_payback\_period = sum\_payback\_period / 20  
 result1['Avg Payback Period in years'] = avg\_payback\_period  
 sum\_payback\_period = 0  
 payback\_period\_count = 0  
  
 if i == 0:  
 for col\_num, column\_name in enumerate(column\_names, start=1):  
 sheet.cell(row=1, column=col\_num, value=column\_name)  
  
 for col\_num, value in enumerate(result1.values(), start=1):  
 sheet.cell(row=i + 2, column=col\_num, value=value)  
  
  
for cell in sheet[min\_payback\_period\_row]:  
 cell.fill = PatternFill(start\_color="FFFF00", end\_color="FFFF00", fill\_type="solid")  
  
  
for cell in sheet[1]:  
 cell.font = Font(bold=True)  
  
workbook.save("RESULTS.xlsx")

# **Додаток В. Критерій Фішера при α = 0.05**

