Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления (ИУ)»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии (ИУ7)»

ОТЧЕТ

по Лабораторной работе №5

по курсу «Моделирование»

на тему: «Моделирование работы информационного центра»

Студент ИУ7-73Б _____ К.Э. Ковалец
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ И.В. Рудаков
(Подпись, дата) (И. О. Фамилия)

2022 г.

Содержание

1 Задание

В информационный центр приходят клиенты через интервалы времени 10±2 минуты. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднее запросы за 20±5, 40±10, 40±20 минут. Клиенты стремятся занять свободного оператора с максимальной производительностью. Полученные запросы сдаются в приемные накопители, откуда они выбираются для обработки. На первый компьютер – запросы от первого и второго операторов, на второй компьютер – от третьего оператора. Время обработки на первом и втором компьютере равны соответственно 15 и 30 минутам. Смоделировать процесс обработки 300 запросов. Определить вероятность отказа.

2 Теоретическая часть

2.1 Схемы модели

На рисунке 2.1 представлена структурная схема модели.

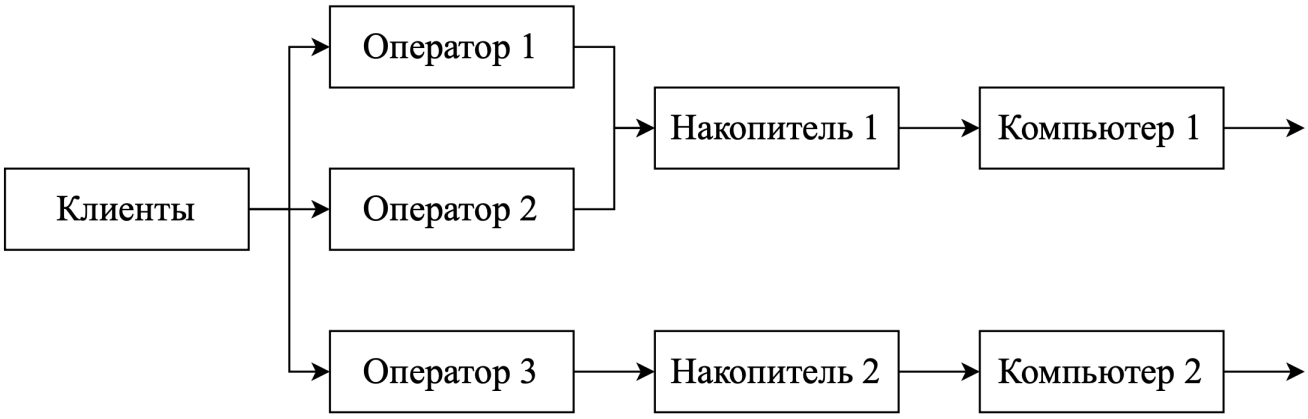

Рисунок 2.1 – Структурная схема модели

В процессе взаимодействия клиентов с информационным центром возможно два режима работы:

- режим нормального обслуживания, когда клиент выбирает одного из свободных операторов, отдавая предпочтение тому, у кого максимальная производительность;
- режим отказа клиенту в обслуживании, когда все операторы заняты.

На рисунке 2.2 представлена схема модели в терминах систем массового обслуживания (СМО).

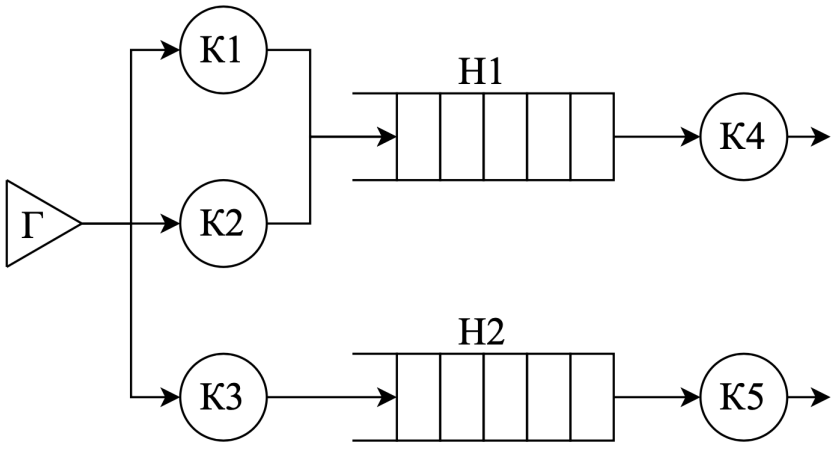

Рисунок 2.2 – Схема модели в терминах СМО

2.2 Переменные и уравнение имитационной модели

Эндогенные переменные:

- время обработки задания i-ым оператором;
- время решения задания на j-ом компьютере.

Экзогенные переменные:

- $n0$ — число обслуженных клиентов;
- $n1$ — число клиентов, получивших отказ.

Вероятность отказа в обслуживании клиента будет вычисляться как:

$$P = \frac{n_0}{n_0 + n_1} \tag{2.1}$$

3 Результаты работы

3.1 Листинги программы

В листинге 3.1 представлена реализация генератора.

Листинг 3.1 — Реализация генератора

```
class Generator:
    def __init__(self, distribution, countClients):
        self.distribution = distribution
        self.receivers = []
        self.numbRequests = countClients
        self.next = 0

    def nextTime(self):
        return self.distribution.generate()

    def generateRequest(self):
        self.numbRequests -= 1

        for receiver in self.receivers:
            if receiver.receiveRequest():
                return receiver
```

В листинге 3.2 представлена реализация канала обслуживания.

Листинг 3.2 — Реализация канала обслуживания

```
from generator import Generator

class Processor(Generator):
    def __init__(self, distribution, maxQueue):
        self.distribution = distribution
        self.maxQueueSize = maxQueue
        self.currentQueueSize  = 0
        self.processedRequests = 0
        self.receivedRequests  = 0
        self.next = 0

    # Обработка запроса при его наличии
    def processRequest(self):
        if self.currentQueueSize > 0:
            self.processedRequests += 1
            self.currentQueueSize  -= 1

    # Добавление реквеста в очередь
```

```
    def receiveRequest(self):
        if self.maxQueueSize == -1 or self.maxQueueSize > self.currentQueueSize:
            self.currentQueueSize += 1
            self.receivedRequests += 1
            return True
        else:
            return False

    def nextTime(self):
        return self.distribution.generate()
```

В листинге 3.3 представлена реализация моделирования работы информационного центра.

Листинг 3.3 — Реализация моделирования работы информационного центра

```
from processor import Processor

class EventModel:
    def __init__(self, generator, operators, computers):
        self.generator = generator
        self.operators = operators
        self.computers = computers

    def run(self):
        refusals = 0
        processed = 0
        generatedRequests = self.generator.numbRequests
        generator = self.generator

        generator.receivers = self.operators.copy()
        self.operators[0].receivers = [self.computers[0]]
        self.operators[1].receivers = [self.computers[0]]
        self.operators[2].receivers = [self.computers[1]]

        generator.next = generator.nextTime()
        self.operators[0].next = self.operators[0].nextTime()

        blocks = [
            generator,
            self.operators[0], self.operators[1], self.operators[2],
            self.computers[0], self.computers[1]
        ]

        while generator.numbRequests >= 0:
```

```
30              # Находим наименьшее время
31              currentTime = generator.next
32              for block in blocks:
33                  if 0 < block.next < currentTime:
34                      currentTime = block.next
35
36              for block in blocks:
37                  # Событие наступило для этого блока
38                  if currentTime == block.next:
39                      if not isinstance(block, Processor):
40                          # Проверяем, может ли оператор обработать
41                          nextGenerator = generator.generateRequest()
42                          if nextGenerator is not None:
43                              nextGenerator.next = currentTime +
                                ↪  nextGenerator.nextTime()
44                              processed += 1
45                          else:
46                              refusals += 1
47
48                          generator.next = currentTime + generator.nextTime()
49                      else:
50                          block.processRequest()
51                          if block.currentQueueSize == 0:
52                              block.next = 0
53                          else:
54                              block.next = currentTime + block.nextTime()
55
56          return [processed, refusals, refusals / generatedRequests * 100]
```

3.2 Демонстрация работы программы

На рисунке 3.1 представлен пример работы программы.

Рисунок 3.1 – Результат работы программы