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 ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

ОТЧЕТ

По лабораторной работе №5

«Определение вероятности отказа»

По курсу «Моделирование»

Студент Т. А. Казаева

Группа ИУ7-76Б

Преподаватель И. В. Рудаков

2022 г.

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 среднего запроса пользователя за 20 ± 5, 40 ± 10, 40 ± 20 минут. Клиенты стремятся занять свободного оператора с максимальной производительностью.

Полученные запросы сдаются в приемный накопитель, откуда они выбираются для обработки. На первый компьютер – запросы от первого и второго оператора, на второй компьютер – от третьего оператора. Время обработки на первом и втором компьютере равны соответственно 15 и 30 минутам.

Смоделировать процесс обработки 300 запросов. В результате определить вероятность отказа. Необходимо построить структурную схему модели.

Нарисовать модель в терминах СМО.

2. Математическая формализация

2.1 Структурная схема модели

На рисунке 2.1 представлена структурная схема модели.

Рис. 2.1: Структурная схема модели

2.2 Концептуальная модель в терминах СМО

На рисунке 2.2 представлена концептуальная модель в терминах СМО.

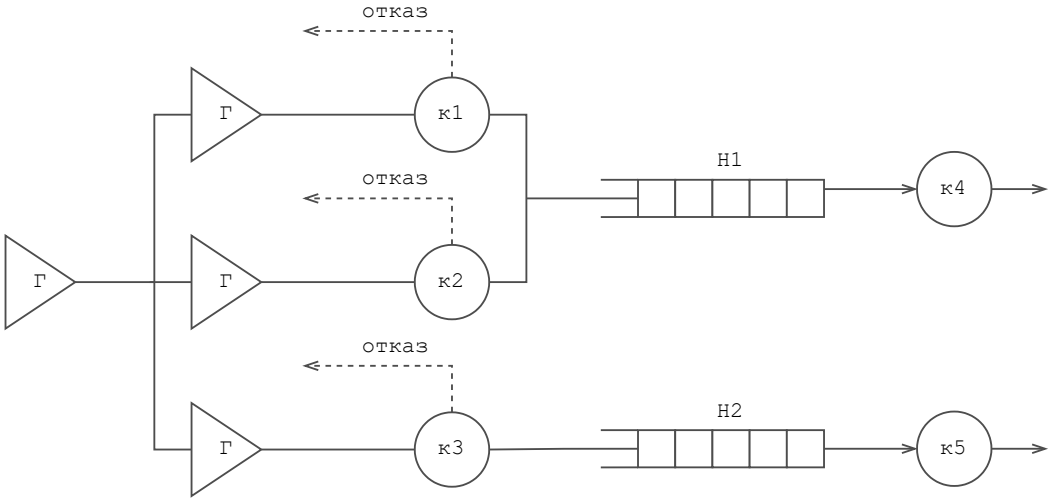

Рис. 2.2: Концептуальная модель в терминах СМО

В процессе взаимодействия клиентов с информационным центром возможен:

- режим нормального обслуживания, т.е. клиент выбирает одного из свободных операторов, отдавая предпочтение тому у которого меньше номер.

- режим отказа в обслуживании клиента, когда все операторы заняты.

2.3 Переменные и уравнения модели

- эндогенные переменные: время обработки задания i-ым оператором, время решения этого задания j-ым компьютером.

- экзогенные переменные: число обслуженных клиентов и число клиентов, получивших отказ.

Вероятность отказа в обслуживании выражается согласно 2.1:

$$P_{\text{отк}} = \frac{C_{\text{отк}}}{C_{\text{отк}} + C_{\text{обсл}}}, \tag{2.1}$$

где $C_{\text{отк}}$ – число клиентов, получивших отказ, а $C_{\text{обсл}}$ – число обслуженных клиентов.

3. Результат

Результат выполнения программы (для количества заявок равным 300):

- Число обслуженных клиентов (n_0): 232
- Число клиентов, получивших отказ (n_1): 68
- Процент отказа: 22.67 %

3.1 Вывод

Решение о том, будет ли принята заявка, принимается на этапе обработки заявок оператором. Значит, вероятность отказа не зависит от времени обработки запроса и их количества (эти параметры влияют на быстродействие). Можно сделать вывод, что чем больше операторов или (и) выше их производительность, тем меньше вероятность отказа.

Для заданных данных вероятность отказа в среднем равна 22,6% (данные усреднены после выполнения программы 1000 раз).

4. Программный код

Для реализации программы был выбран язык Python.

На листинге 4.1 представлена генерация числа из заданного интервала.

Листинг 4.1: Генерация числа из заданного интервала

```
class Generator:
    def __init__(self, base, err):
        self.low = base - err
        self.high = base + err

    def generationTime(self):
        return random.randint(self.low, self.high)
```

На листинге 4.2 представлена реализация оператора.

Листинг 4.2: Реализация оператора

```
class Operator(Generator):
    def __init__(self, base, err):
        super().__init__(base, err)
        self.busy = False
```

На листинге 4.3 представлена реализация компьютера.

Листинг 4.3: Реализация компьютера

```
class Computer:
    def __init__(self, time):
        self.generationTime = time
        self.busy = False
```

На листинге 4.4 представлена реализация СМО.

Листинг 4.4: Реализация СМО

```
class QueueNetwork:
    def __init__(self, n):
        self.n = n
        self.jobs = list()
```

```
5           self.clients = Generator(10, 2)
6           self.operators = [Operator(20, 5), Operator(40, 10),
              Operator(40, 20)]
7           self.computers = [Computer(15), Computer(30)]
8           self.lines = [0, 0]
9           self.processed = 0
10          self.lost = 0
11
12      def reset(self, n):
13          self.n = n
14          self.jobs = list()
15          self.clients = Generator(10, 2)
16          self.operators = [Operator(20, 5), Operator(40, 10),
              Operator(40, 20)]
17          self.computers = [Computer(15), Computer(30)]
18          self.lines = [0, 0]
19          self.processed = 0
20          self.lost = 0
21
22      def model(self):
23          self.AddJob([self.clients.generationTime(), 'client'])
24          while self.processed + self.lost < self.n:
25              job = self.jobs.pop(0)
26              if job[1] == 'client':
27                  self.newClient(job[0])
28              elif job[1] == 'operator':
29                  self.OperatorFinished(job)
30              elif job[1] == 'pc':
31                  self.ComputerFinished(job)
32          return self.lost
33
34      def newClient(self, time):
35          i = 0
36          while i < 3 and self.operators[i].busy:
37              i += 1
38          if i == 3:
39              self.lost += 1
40          else:
41              self.operators[i].busy = True
```

```
42              self.AddJob(
43                  [time + self.operators[i].generationTime(),
                        'operator', i])
44          self.AddJob([time + self.clients.generationTime(),
                'client'])
45
46      def OperatorFinished(self, job):
47          self.operators[job[2]].busy = False
48          if job[2] < 2:
49              self.lines[0] += 1
50              job[2] = 0
51          else:
52              self.lines[1] += 1
53              job[2] = 1
54          self.ComputerGetJob(job)
55
56      def ComputerGetJob(self, job):
57          cnum = job[2]
58          if not self.computers[cnum].busy and self.lines[cnum] > 0:
59              self.computers[cnum].busy = True
60              self.AddJob(
61                  [job[0] + self.computers[cnum].generationTime,
                        'pc', cnum])
62              self.lines[cnum] -= 1
63
64      def ComputerFinished(self, job):
65          self.computers[job[2]].busy = False
66          self.processed += 1
67          self.ComputerGetJob(job)
68
69      def AddJob(self, job: list):
70          i = 0
71          while i < len(self.jobs) and self.jobs[i][0] < job[0]:
72              i += 1
73          self.jobs.insert(i, job)
```