



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»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по Лабораторной работе №5

по курсу «Моделирование»

на тему: «Моделирование работы информационного центра»

Студент группы ИУ7И-74Б

(Подпись, дата)

Динь Вьет Ань
(Фамилия И.О.)

Преподаватель

(Подпись, дата)

Рудаков И. В.
(Фамилия И.О.)

2023 г.

Содержание

1	Задание	3
2	Теоретическая часть	4
2.1	Схемы модели	4
2.2	Переменные и уравнение имитационной модели	5
3	Результаты работы	6
3.1	Листинги программы	6
3.2	Демонстрация работы программы	9

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.

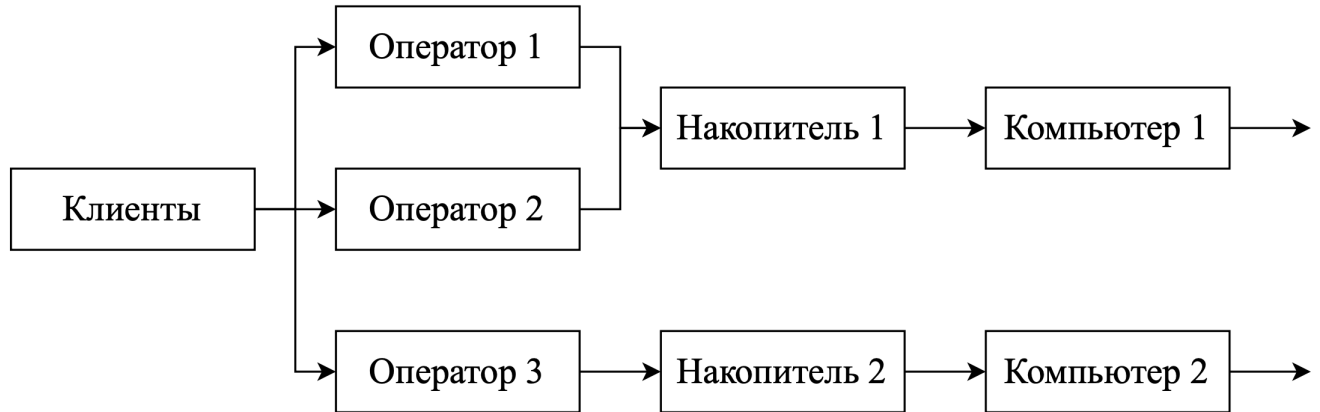


Рисунок 2.1 – Структурная схема модели

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).

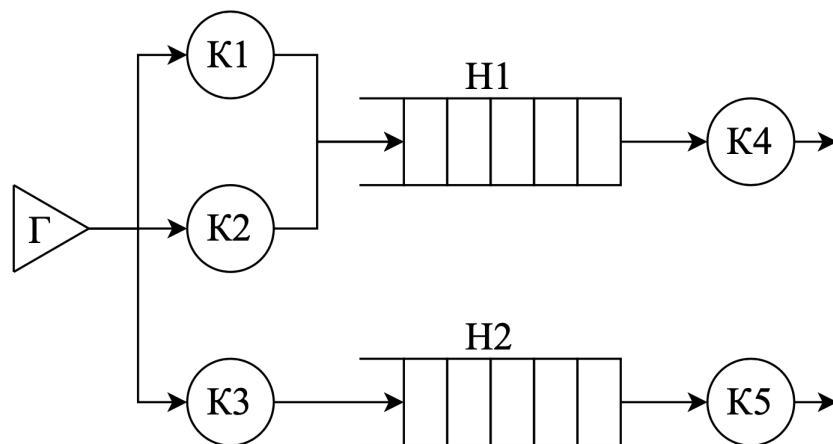


Рисунок 2.2 – Схема модели в терминах СМО

2.2 Переменные и уравнение имитационной модели

Эндогенные переменные:

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

Экзогенные переменные:

- n_0 — число обслуженных клиентов;
- n_1 — число клиентов, получивших отказ.

Вероятность отказа в обслуживании клиента будет вычисляться как:

$$P = \frac{n_0}{n_0 + n_1} \quad (2.1)$$

3 Результаты работы

3.1 Листинги программы

В листинге 3.1 представлена реализация генератора.

Листинг 3.1 — Реализация генератора

```
1  class Generator:
2      def __init__(self, distribution, countClients):
3          self.distribution = distribution
4          self.receivers = []
5          self.numbRequests = countClients
6          self.next = 0
7
8      def nextTime(self):
9          return self.distribution.generate()
10
11     def generateRequest(self):
12         self.numbRequests -= 1
13
14         for receiver in self.receivers:
15             if receiver.receiveRequest():
16                 return receiver
```

В листинге 3.2 представлена реализация канала обслуживания.

Листинг 3.2 — Реализация канала обслуживания

```
1  from generator import Generator
2
3  class Processor(Generator):
4      def __init__(self, distribution, maxQueue):
5          self.distribution = distribution
6          self.maxQueueSize = maxQueue
7          self.currentQueueSize = 0
8          self.processedRequests = 0
9          self.receivedRequests = 0
10         self.next = 0
11
12         # Обработка запроса при его надичии
13     def processRequest(self):
14         if self.currentQueueSize > 0:
15             self.processedRequests += 1
16             self.currentQueueSize -= 1
17
18         # Добавление реквеста в очередь
```

```

19     def receiveRequest(self):
20         if self.maxQueueSize == -1 or self.maxQueueSize > self.currentQueueSize:
21             self.currentQueueSize += 1
22             self.receivedRequests += 1
23             return True
24         else:
25             return False
26
27     def nextTime(self):
28         return self.distribution.generate()

```

В листинге 3.3 представлена реализация моделирования работы информационного центра.

Листинг 3.3 — Реализация моделирования работы информационного центра

```

1  from processor import Processor
2
3  class EventModel:
4      def __init__(self, generator, operators, computers):
5          self.generator = generator
6          self.operators = operators
7          self.computers = computers
8
9      def run(self):
10         refusals = 0
11         processed = 0
12         generatedRequests = self.generator.numbRequests
13         generator = self.generator
14
15         generator.receivers = self.operators.copy()
16         self.operators[0].receivers = [self.computers[0]]
17         self.operators[1].receivers = [self.computers[0]]
18         self.operators[2].receivers = [self.computers[1]]
19
20         generator.next = generator.nextTime()
21         self.operators[0].next = self.operators[0].nextTime()
22
23         blocks = [
24             generator,
25             self.operators[0], self.operators[1], self.operators[2],
26             self.computers[0], self.computers[1]
27         ]
28
29         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 +
44                     ↪ nextGenerator.nextTime()
45                     processed += 1
46                 else:
47                     refusals += 1
48
49                 generator.next = currentTime + generator.nextTime()
50             else:
51                 block.processRequest()
52                 if block.currentQueueSize == 0:
53                     block.next = 0
54                 else:
55                     block.next = currentTime + block.nextTime()
56
57     return [processed, refusals, refusals / generatedRequests * 100]

```


3.2 Демонстрация работы программы

На рисунке 3.1 представлен пример работы программы.

Лабораторная работа №5 (Ковалец Кирилл ИУ7-73Б)

ПАРАМЕТРЫ		
Количество заявок	300	
Интервал прихода клиента	10	+/- 2 минут(ы)

ОПЕРАТОРЫ		
Оператор 1	20	+/- 5 минут(ы)
Оператор 2	40	+/- 10 минут(ы)
Оператор 3	40	+/- 20 минут(ы)

КОМПЬЮТЕРЫ		
Компьютер 1	15	минут(ы)
Компьютер 2	30	минут(ы)

РЕЗУЛЬТАТ		
Обработано	Отказано	Процент отказа
236	64	21.67
Решить		

О ПРОГРАММЕ
Информация о программе

Рисунок 3.1 – Результат работы программы