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: «Определение вероятности отказа»

Студент ИУ7-73Б _____ Миронов Г. А.
(Группа) (Подпись, дата) (И. О. Фамилия)

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

*2022 г.*

# 1 Задание

В информационный центр приходят клиенты через интервал времени $10 \pm 2$ минуты. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за $20 \pm 5$, $40 \pm 10$ и $40 \pm 20$, соответственно.

Клиенты стремятся занять свободного оператора с максимальной производительностью. Полученные запросы сдаются в накопитель, откуда выбираются на обработку. На первый компьютер запросы от 1 и 2-ого операторов, на второй – запросы от 3-его. Время обработки запросов первым и 2-м компьютером равны соответственно 15 и 30 мин. Промоделировать процесс обработки 300 запросов.

Для выполнения поставленного задания необходимо создать концептуальную модель в терминах СМО, определить эндогенные и экзогенные переменные и уравнения модели.

За единицу системного времени выбрать 0.01 минуты.

# 2 Теоретическая часть

## 2.1 Общий вид системы

На рисунке 2.1 представлена концептуальная модель моделируемой системы в общем виде.

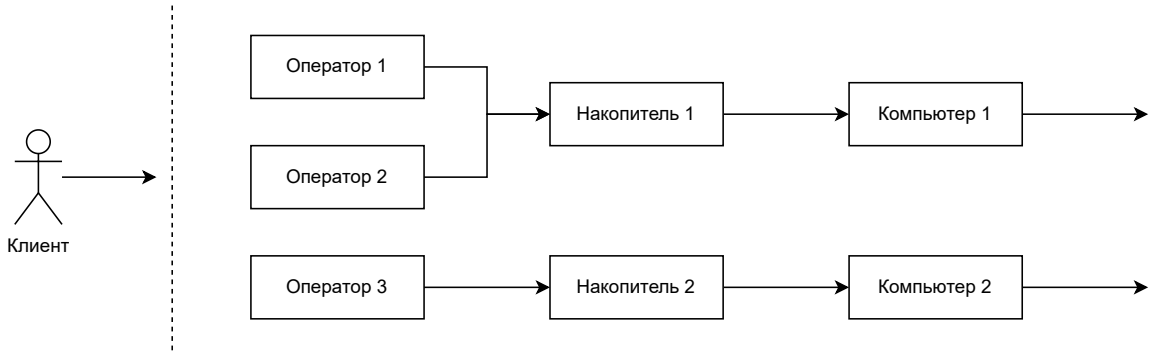


Рисунок 2.1 – Концептуальная модель в общем виде

## 2.2 Концептуальная модель системы в терминах СМО

На рисунке 2.2 представлена концептуальная модель моделируемой системы в терминах СМО.

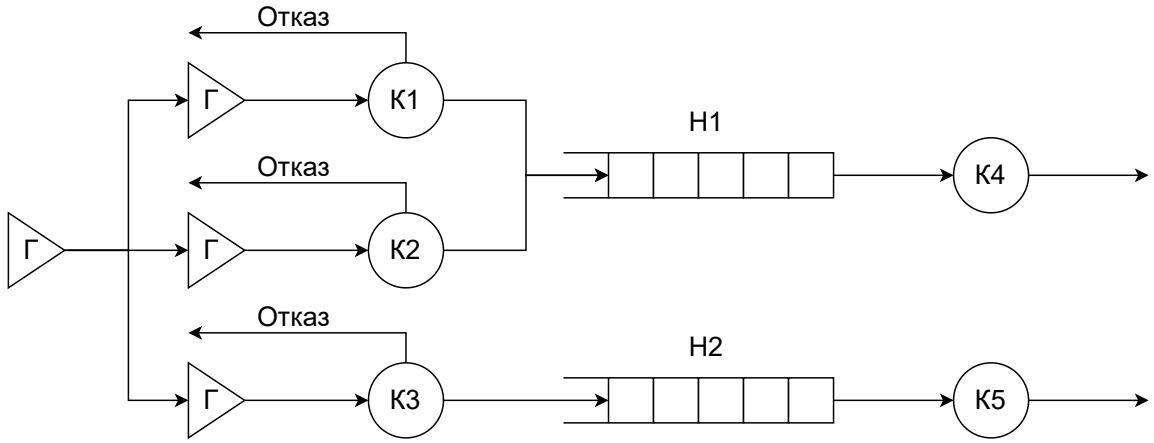


Рисунок 2.2 – Концептуальная модель в терминах СМО

В процессе взаимодействия клиентов с информационным центром возможны:

- 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 имитационной модели

Эндогенные переменные:

- время обработки задания $i$-ым оператором;
- время решения этого задания $j$-ым компьютером.

Экзогенные переменные:

- число обслуженных клиентов;
- число клиентов, получивших отказ.

Вероятность отказа в обслуживании клиента:

$$\frac{C_{\text{отк}}}{C_{\text{отк}} + C_{\text{обсл}}}$$

где $C_{\text{отк}}$ – количество заявок которым отказали в обслуживании, а $C_{\text{обсл}}$ – количество обслуженных заявок.

# 3 Результат работы

## 3.1 300 заявок

```
Processing time (seconds):  0.8935
Total requests time:  300
Processed requests time:  230
Canceled requests time:  70
Cancelation percent:  0.2333
```

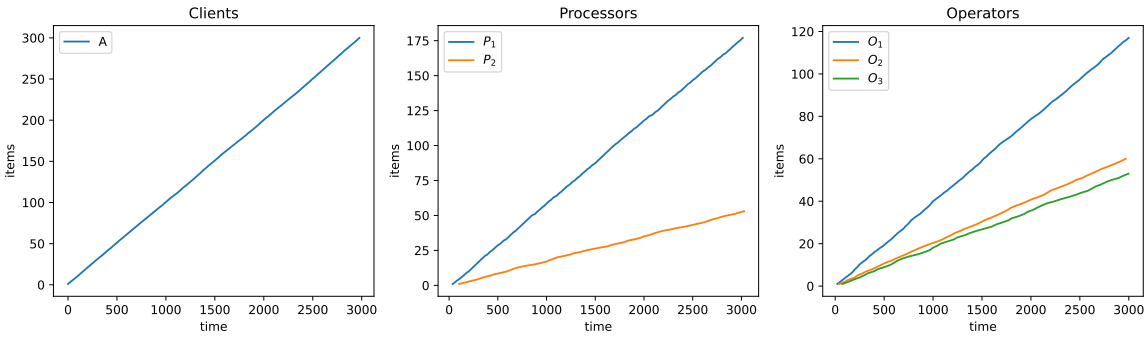


Рисунок 3.1 – Концептуальная модель в терминах СМО

## 3.2 1000 заявок

```
Processing time (seconds):  2.9877
Total requests time:  1000
Processed requests time:  782
Canceled requests time:  218
Cancelation percent:  0.218
```

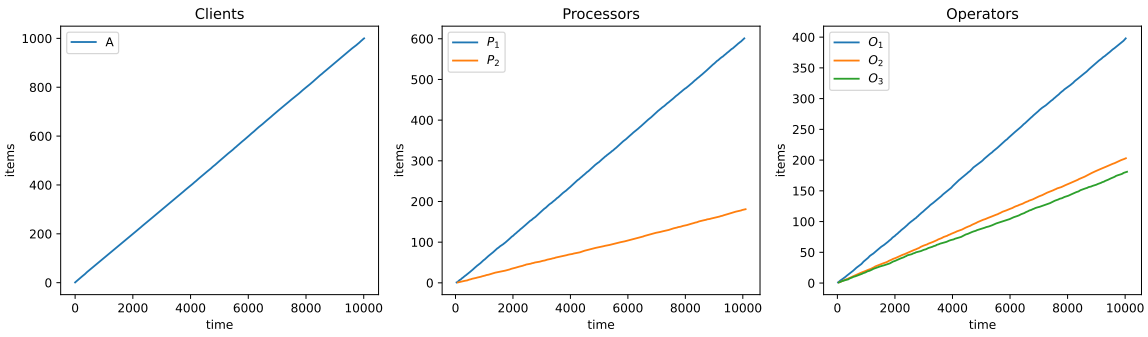


Рисунок 3.2 – Концептуальная модель в терминах СМО

## 3.3 3000 заявок

```
Processing time (seconds):  8.9618
Total requests time:  3000
Processed requests time:  2374
Canceled requests time:  626
Cancelation percent:  0.2087
```

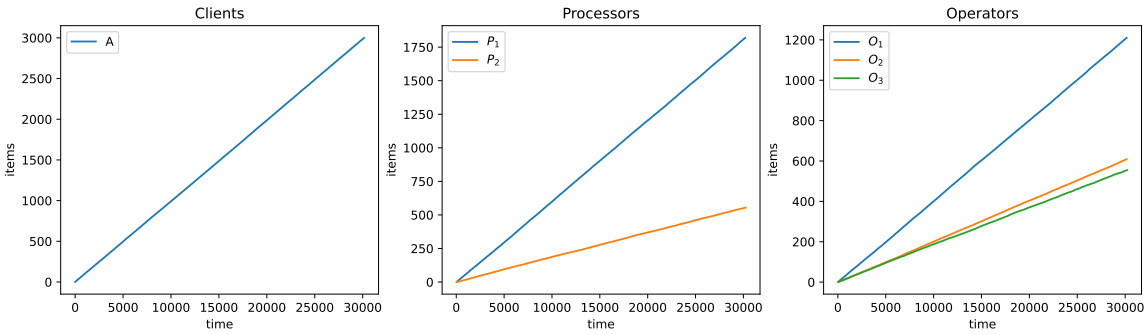


Рисунок 3.3 – Концептуальная модель в терминах СМО

# 4 Исходный код программы

Листинг 4.1 – Исходный код программы. Часть 1

```
from time import time

import numpy as np

import matplotlib.pyplot as plt

TIME_DELTA = 0.01
FINISH_PROCESS_REQUEST = 1
CURRENT_REQUEST = 0
DONT_HAVE_FREE_OPERATOR = -1


class TimeSeries:
    def __init__(self, step: np.float64 = TIME_DELTA):
        self.time = 0
        self.step = step
        self.times = []
        self.counts = []

    def update_time(self):
        self.time += self.step

    def add(self) -> None:
        self.put(self.time, self.counts[-1] + 1 if len(self.
           counts) > 0 else 1)

    def put(self, at: np.float64, val: np.int64) -> None:
        self.times.append(at)
        self.counts.append(val)

    def raw(self):
        return self.times, self.counts


def distribution(a, b):
    return lambda: np.random.uniform(a, b)


class ClientGenerator:
    def __init__(self, time_distribution):
        self.time_distribution = time_distribution
        self.remaining_time = 0
        self.stat = TimeSeries()
```

Листинг 4.2 – Исходный код программы. Часть 2

```
43
44      def update_time(self):
45          self.stat.update_time()
46          if self.remaining_time > 0:
47              self.remaining_time -= TIME_DELTA
48
49          if self.remaining_time <= 1e-5:
50              self.remaining_time = self.time_distribution()
51              self.stat.add()
52              return Request()
53
54          return None
55
56
57  class Operator:
58      def __init__(self, recipient, time_distribution):
59          self.time_distribution = time_distribution
60          self.recipient = recipient
61          self.remaining_time = 0
62          self.is_busy = False
63          self.processing_request = None
64          self.stat = TimeSeries()
65
66      def update_time(self):
67          self.stat.update_time()
68          self.remaining_time -= TIME_DELTA
69          if self.is_busy and self.remaining_time <= 1e-5:
70              self.finish_process_request()
71
72      def start_process_new_request(self, request):
73          self.is_busy = True
74          self.processing_request = request
75          self.remaining_time = self.time_distribution()
76
77      def finish_process_request(self):
78          self.recipient.append(self.processing_request)
79          self.is_busy = False
80          self.processing_request = None
81          self.stat.add()
82
83
84  class Request:
85      request_id = 0
86
87      def __init__(self):
88          global CURRENT_REQUEST
89          self.request_id = CURRENT_REQUEST
90          CURRENT_REQUEST += 1
```

Листинг 4.3 – Исходный код программы. Часть 3

```
91
92
93  class Processor:
94      def __init__(self, requests_storage, time_distribution):
95          self.requests_storage = requests_storage
96          self.time_distribution = time_distribution
97          self.is_busy = False
98          self.processing_request = None
99          self.remaining_time = 0
100         self.stat = TimeSeries()
101
102     def update_time(self):
103         self.stat.update_time()
104         if self.remaining_time != 0:
105             self.remaining_time -= TIME_DELTA
106
107         if self.is_busy and self.remaining_time <= 1e-5:
108             self.is_busy = False
109             self.processing_request = None
110             self.stat.add()
111             return FINISH_PROCESS_REQUEST
112
113         if not self.is_busy and len(self.requests_storage) != 0:
114             self.processing_request = self.requests_storage.pop
                    (0)
115             self.remaining_time = self.time_distribution()
116             self.is_busy = True
117
118
119 def find_free_operator(operators):
120     for i in range(len(operators)):
121         if not operators[i].is_busy:
122             return i
123     return DONT_HAVE_FREE_OPERATOR
124
125
126 def iteration(clients, operators, processors, request_info,
        is_new=True):
127     if is_new:
128         request = clients.update_time()
129         if request:
130             request_info["generated_count"] += 1
131             free_operator_number = find_free_operator(operators)
132             if free_operator_number == DONT_HAVE_FREE_OPERATOR:
133                 request_info["lost_count"] += 1
134             else:
135                 operators[free_operator_number].
                        start_process_new_request(request)
```

Листинг 4.4 – Исходный код программы. Часть 4

```
136
137     for operator in operators:
138         operator.update_time()
139
140     for processor in processors:
141         result = processor.update_time()
142         if result == FINISH_PROCESS_REQUEST:
143             request_info["processed_count"] += 1
144
145
146 def modeling(clients, operators, processors, requests_count):
147     statistics_info = {"generated_count": 0, "processed_count":
          0, "lost_count": 0}
148
149     while statistics_info["generated_count"] < requests_count:
150         iteration(clients, operators, processors, statistics_info
              )
151
152     while (
153         statistics_info["lost_count"] + statistics_info["
              processed_count"]
154         < requests_count
155     ):
156         iteration(clients, operators, processors, statistics_info
              , False)
157
158     return statistics_info
159
160
161 if __name__ == "__main__":
162     requests_count = 300
163
164     clients = ClientGenerator(distribution(8, 12))
165
166     storage_1 = []
167     storage_2 = []
168
169     processors = [Processor(storage_1, lambda: 15), Processor(
          storage_2, lambda: 30)]
170
171     operators = [
172         Operator(storage_1, distribution(15, 25)),
173         Operator(storage_1, distribution(30, 50)),
174         Operator(storage_2, distribution(20, 60)),
175     ]
176
177     start_time = time()
```

Листинг 4.5 – Исходный код программы. Часть 5

```
178
179     result = modeling(clients, operators, processors,
            requests_count)
180
181     print("Processing␣time␣(seconds):␣", round((time() -
            start_time), 4))
182     print("Total␣requests␣time:␣", result["generated_count"])
183     print("Processed␣requests␣time:␣", result["processed_count"])
184     print("Canceled␣requests␣time:␣", result["lost_count"])
185     print("Cancelation␣percent:␣", round((result["lost_count"] /
            requests_count), 4))
186
187     fig, axis = plt.subplots(1, 3, figsize=(16, 4))
188
189     axis[0].set_title("Clients")
190     axis[0].plot(*clients.stat.raw(), label="A")
191     axis[0].set_xlabel("time")
192     axis[0].set_ylabel("items")
193     axis[0].legend()
194
195     axis[1].set_title("Processors")
196     axis[1].set_xlabel("time")
197     axis[1].set_ylabel("items")
198     for i, p in enumerate(processors):
199         axis[1].plot(*p.stat.raw(), label=f"$P_{i+1}$")
200     axis[1].legend()
201
202     axis[2].set_title("Operators")
203     axis[2].set_xlabel("time")
204     axis[2].set_ylabel("items")
205     for i, o in enumerate(operators):
206         axis[2].plot(*o.stat.raw(), label=f"$O_{i+1}$")
207     axis[2].legend()
208
209     plt.show()
```