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 №6

Моделирование работы многофункционального центра

Студент Соколов Ефим
Группа ИУ7-73Б
Дисциплина Моделирование

Преподаватель: _____ Рудаков И.В.
подпись, дата Фамилия, И.О.

Оценка _____

Москва — 2022 г.

Задание

В многофункциональный центр приходят посетители через интервал времени 5 +- 2 минуты. При входе производится проверка наличия маски в течение 1 минуты. Если у посетителя нет маски, ему отказывается в обслуживании. Вероятность отсутствия маски у посетителя составляет 5%. После проверки наличия маски производится проверка температуры в течение 3 +- 1 минуты. С вероятностью 2% у посетителя будет температура и ему будет отказано в обслуживании. Далее посетители проходят к терминалам для получения талона очереди в течение 4 +- 1 минуты. Вероятность того, что в многофункциональный центр не предоставляют необходимую услугу, равняется 5% Если все три терминала заняты, ему будет отказано в обслуживании. Если в очереди на окно набралось 10 человек, посетителю отказывают. Всего есть 3 окна, которые работают 15 +- 5 , 10 +- 2 , 20 +- 5 минут соответственно. Окно на терминале выбирается по равномерному распределению.

Промоделировать процесс обработки 800 запросов. Определить вероятность отказа.

Схема системы

На рисунке 1 приведена схема моделируемой системы.

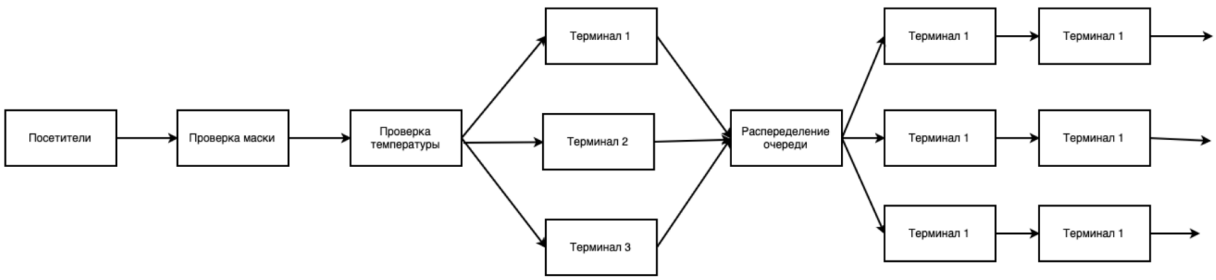

Рисунок 1: Схема системы

Листинги

На листинге 1 приведена реализация класса Model; с помощью метода event_mode выполняется моделирование системы.

Листинг 1: Класс Model

```
class Model:
  def __init__(self, clients, mask_check, temp_check, terminals, windows):
    self.clients = clients
    self.mask_check = mask_check
    self.temp_check = temp_check
    self.terminals = terminals
    self.windows = windows

  def event_mode(self):
    clients = self.clients
    mask = self.mask_check

    clients.receivers = self.mask_check.copy()
    self.mask_check[-1].receivers = self.temp_check.copy()
    self.temp_check[-1].receivers = self.terminals.copy()

    self.terminals[0].receivers = self.windows.copy()
    self.terminals[1].receivers = self.windows.copy()
    self.terminals[2].receivers = self.windows.copy()

    clients.next = clients.delay()

    blocks = []
    blocks += [clients]
    blocks += self.mask_check
    blocks += self.temp_check
    blocks += self.terminals
    blocks += self.windows

    refusals = {}

    for block in blocks:
      refusals[block.name] = 0
    refusals["query_window"] = 0

    while clients.num_requests >= 0:
      current_time = clients.next
      for block in blocks:
        if (0 < block.next) and (block.next < current_time):
          current_time = block.next

    for block in blocks:
      if current_time == block.next:
        if not isinstance(block, ProcessRequest):
          next_clients = clients.generate_request()
          if next_clients is not None:
            next_clients.next = current_time + next_clients.delay()
          else:
            refusals[block.name] += 1
          clients.next = current_time + clients.delay()
        else:
```

```
52            next_process = block.process_request()
53            if block.queue == 0:
54              block.next = 0
55            else:
56              block.next = current_time + block.delay()
57
58            if next_process == "kick":
59              if block.name.find("terminal") != -1:
60                refusals["query_window"] += 1
61              else:
62                refusals[block.name] += 1
63              continue
64
65            if block.end:
66              continue
67
68            if next_process is not None:
69              next_process.next = current_time + next_process.delay()
70            else:
71              refusals[block.name] += 1
72
73      return refusals
```

Результаты выполнения работы

На рисунке 2 приведен результат работы системы с 800 заявок.

```
❯ python3 main.py
Время прихода посетителей: 5
Дельта прихода посетителей: 2
Время проверки наличия маски: 1
Дельта проверки наличия маски: 0
Время проверки температуры: 3
Дельта проверки температуры: 1
Время получения очереди в терминалах: 4
Дельта получения очереди в терминалах: 1
Время работы в окне 1: 15
Дельта работы в окне 1: 5
Время работы в окне 2: 10
Дельта работы в окне 2: 2
Время работы в окне 3: 20
Дельта работы в окне 3: 5
Количество посетителей: 800
```

Наименование этапа	min	max
Отказы при проверке маски	29	61
Отказы при проверке температуры	8	25
Отказы на терминале 1	33	55
Отказы на терминале 2	33	59
Отказы на терминале 3	30	60
Отказы при распределении очереди	28	56
Отказы на окне 1	0	0
Отказы на окне 2	0	0
Отказы на окне 3	0	0

```
Всего отказов:
   минимум - 202 (25.25%)
   максимум - 263 (32.88%)
```

Рисунок 2: Результат моделирования системы с 800 заявками