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»

Отчёт по лабораторной работе №5 по курсу «Моделирование»

Тема Моделирование работы информационного центра

Студент Калашков П. А.

Группа ИУ7-76Б

Оценка (баллы)

Преподаватели Рудаков И. В.

Москва — 2023 г.

Целью данной работы является разработка программы с графическим интерфейсом для моделирования процесса обработки 300 запросов клиентов информационным центром и определения вероятности отказа клиенту в обслуживании. Информационный центр работает следующим образом:

1. Клиенты приходят через интервал времени, равный 10 ± 2 мин.
2. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за 20 ± 5 мин., 40 ± 10 мин. и 40 ± 20 мин. соответственно. Клиенты стремятся занять свободного оператора с максимальной производительностью.
3. Полученные запросы сдаются в приемный накопитель, из которого они выбираются для обработки. На первый компьютер выбираются запросы от первого и второго операторов, на второй компьютер — от третьего оператора. Время обработки на первом и втором компьютерах равны соответственно 15 мин. и 30 мин.

В процессе взаимодействия клиентов и информационного центра предусмотреть: режим нормального обслуживания, когда клиент выбирает одного из свободных операторов с максимальной производительностью, и режим отказа.

Моделирование функционирования системы

При моделировании функционирования системы эндогенными переменными являются:

- время обслуживания клиента i-ым оператором, где $i = \overline{1,3}$;
- время обработки запроса на j-ом компьютере, где $j = \overline{1,2}$.

Экзогенными переменными являются:

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

Уравнение модели имеет следующий вид:

$$P_{\text{отказа}} = \frac{n_1}{n_0 + n_1} \tag{1}$$

Схемы модели

На рисунке 1 показана структурная схема модели.

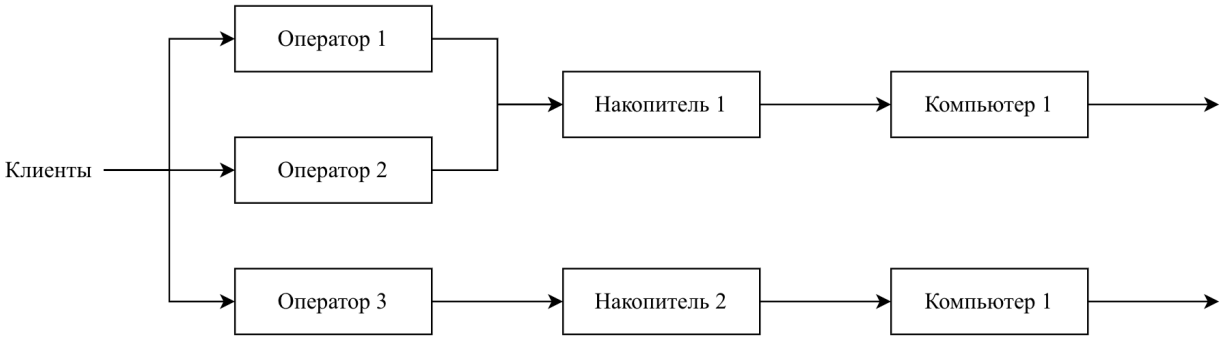

Рисунок 1 – Структурная схема модели информационного центра

На рисунке 2 представлена схема модели в терминах СМО.

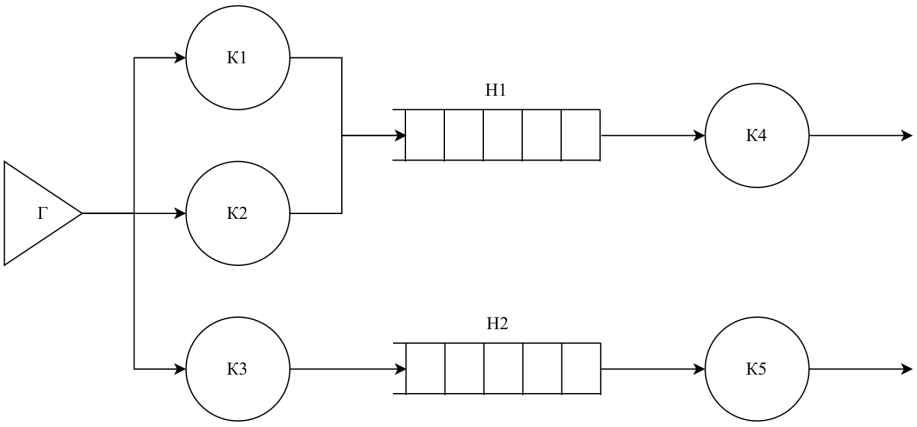

Рисунок 2 – Схема модели в терминах СМО

Результаты работы

Детали реализации

В листинге 1 представлена реализация работы генератора клиентов, а в листингах 2–3 — реализация работы оператора. Реализации работы компьютера и самого информационного центра представлены в листингах 4 и 5–6 соотвественно.

Листинг 1 – Реализация работы генератора клиентов

```
class ClientGenerator:
    def __init__(self, time_value, time_limit, operators, number):
        self.generator = TimeGenerator(time_value - time_limit,
           time_value + time_limit)
        self.operators = sorted(operators, key= lambda operator:
           operator.max_time)
        self.time_next = 0
        self.number = number

    def generate_client(self, time_prev):
        self.time_next = time_prev + self.generator.get_interval()

    def choose_operator(self):
        for operator in self.operators:
            if operator.is_free():
                return operator
        return None
```

Листинг 2 – Реализация работы оператора (часть 1)

```
class Operator:
    def __init__(self, time_value, time_limit, computer):
        self.time_generator = TimeGenerator(time_value -
           time_limit, time_value + time_limit)
        self.computer = computer
        self.max_time = time_value + time_limit
        self.time_next = 0
        self.free = True

    def generate_time(self, prev_time):
        self.time_next = prev_time +
           self.time_generator.get_interval()

    def is_free(self):
        return self.free

    def set_free(self):
        self.free = True
```

Листинг 3 – Реализация работы оператора (часть 2)

```
    def set_busy(self):
        self.free = False

    def get_computer(self):
        return self.computer
```

Листинг 4 – Реализация работы компьютера

```
class Computer:
    def __init__(self, time_value, time_limit):
        self.time_generator = TimeGenerator(time_value -
            time_limit, time_value + time_limit)
        self.queue = []
        self.time_next = 0
        self.free = True

    def generate_time(self, prev_time):
        self.time_next = prev_time +
            self.time_generator.get_interval()

    def is_free(self):
        return self.free

    def set_free(self):
        self.free = True

    def set_busy(self):
        self.free = False

    def is_empty(self):
        if self.queue:
            return False
        return True

    def add_request(self):
        self.queue.append(request)

    def pop_request(self):
        self.queue.pop(0)
```

Листинг 5 – Реализация работы информационного центра (часть 1)

```
class Center:
    def __init__(self, client_generator):
        self.client_generator = client_generator

    def service_clients(self):
        failures = 0
        self.client_generator.generate_client(0)
        generated_clients = 1
        events = [Event(self.client_generator,
           self.client_generator.time_next)]

        while generated_clients < self.client_generator.number:
            events = sort_events(events)
            event = events.pop(0)

            if isinstance(event.creator, ClientGenerator):
                operator = self.client_generator.choose_operator()
                if operator is None:
                    failures += 1
                else:
                    operator.set_busy()
                    operator.generate_time(event.time)
                    events.append(Event(operator,
                       operator.time_next))
                self.client_generator.generate_client(event.time)
                generated_clients += 1
                events.append(Event(self.client_generator,
                   self.client_generator.time_next))

            elif isinstance(event.creator, Operator):
                operator = event.creator
                operator.set_free()
                computer = operator.get_computer()
                computer.add_request()
                if computer.is_free() and not computer.is_empty():
                    computer.pop_request()
                    computer.set_busy()
                    computer.generate_time(event.time)
                    events.append(Event(computer,
                       computer.time_next))
```

Листинг 6 – Реализация работы информационного центра (часть 2)

```
            elif isinstance(event.creator, Computer):
                computer = event.creator
                computer.set_free()
                if not computer.is_empty():
                    computer.pop_request()
                    computer.set_busy()
                    computer.generate_time(event.time)
                    events.append(Event(computer,
                        computer.time_next))

        return failures
```

Примеры работы

На рисунке 3 представлен пример работы разработанной программы для описанного информационного центра с обработкой 300 заявок.

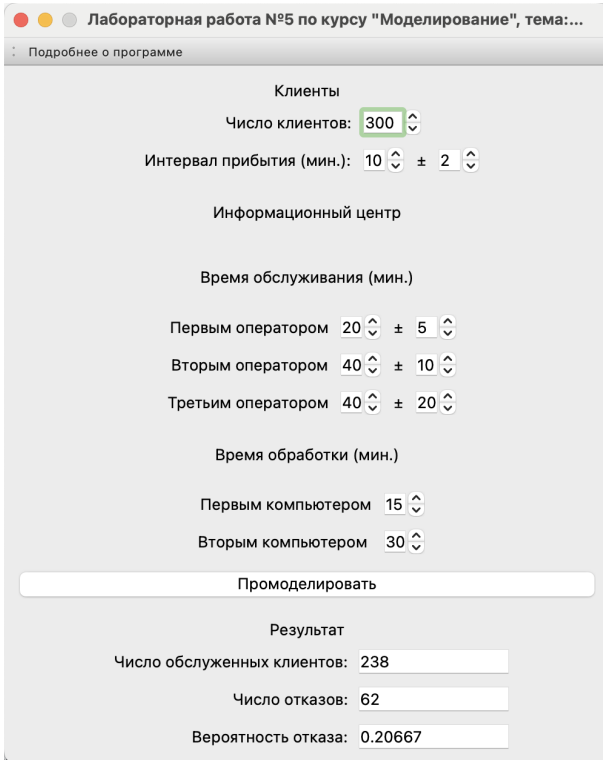

Рисунок 3 – Пример работы программы, обработка 300 заявок

Вывод

В ходе выполнения лабораторной работы была реализована программа с графическим интерфейсом для моделирования процесса обработки 300 запросов клиентов информационным центром и определения вероятности отказа клиенту в обслуживании. Вероятность отказа равна примерно 0.2.