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»

О Т Ч Е Т

по лабораторной работе № 6

Название: «Моделирование системы с очередями»

Дисциплина: Моделирование

Студент ИУ7-76Б (Группа) А. А. Петрова (И.О. Фамилия)

Преподаватель И. В. Рудаков (И.О. Фамилия)

2022 г.

Задание

Для сдачи лабораторных работ по предмету приходят студенты через интервалы времени 2 ± 1 минуты и встают в две очереди к двум соответствующим преподавателям. Первый преподаватель принимает одного студента за 4 ± 2 минуты, а второй – за 20 ± 5 минут. Студенты встают в очередь к первому преподавателю с заданной вероятностью. После сдачи студенты из обеих очередей встают в одну очередь на защиту. Защита одного студента принимается за 5 минут. Смоделировать процесс сдачи лабораторных работ при 100 студентах.

Математическая формализация

Эндогенные переменные: время принятия лабораторной i-ым преподавателем и время принятия защиты.

Экзогенные переменные: число принятых студентов у первого преподавателя и второго.

Ниже представлена схема модели в терминах СМО.

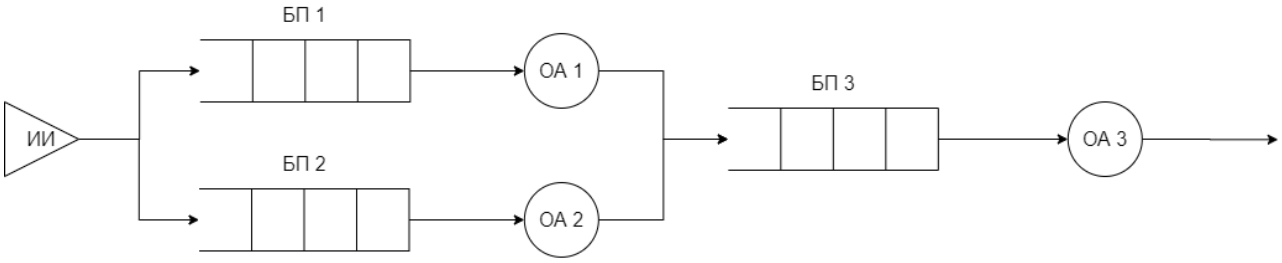

Рисунок 1: модель в терминах СМО

Где ИИ – это источник информации (студент с лабораторной), БП – буферная память (очередь), а ОА – обслуживающий аппарат (преподаватели и защита).

Реализация

В листингах ниже представлена реализация генератора заявок, работы преподавателей и защиты.

Листинг 1: генератор заявок

```
class TimeProcessor:
    def __init__(self, time_distribution):
        self.time_distribution = time_distribution
        self.remaining_time = 0

    def update_time(self):
        if self.remaining_time > 0:
            self.remaining_time -= TIME_DELTA
```

```
        if self.remaining_time <= 1e-5:
            self.remaining_time = self.time_distribution.get_time()
            return Request()

        return None
```

Листинг 2: преподаватель

```
class Teacher:
    def __init__(self, storage, recipient, time_distribution):
        self.time_distribution = time_distribution
        self.requests_storage = storage
        self.recipient = recipient
        self.remaining_time = 0
        self.is_busy = False
        self.processing_request = None
        self.max_queue_len = 0
        self.work_time = 0

    def update_time(self):
        self.remaining_time -= TIME_DELTA
        if len(self.requests_storage) > self.max_queue_len:
            self.max_queue_len = len(self.requests_storage)

        if self.is_busy and self.remaining_time <= 1e-5:
            self.recipient.append(self.processing_request)
            self.is_busy = False
            self.processing_request = None

        if not self.is_busy and len(self.requests_storage) != 0:
            self.processing_request = self.requests_storage.pop(0)
            self.remaining_time = self.time_distribution.get_time()
            self.work_time += self.remaining_time
            self.is_busy = True
```

Листинг 3: защита

```
class Defense:
    def __init__(self, requests_storage, time_distribution):
        self.requests_storage = requests_storage
        self.time_distribution = time_distribution
        self.is_busy = False
        self.processing_request = None
        self.remaining_time = 0
        self.max_queue_len = 0

    def update_time(self):
        if self.remaining_time != 0:
            self.remaining_time -= TIME_DELTA

        if len(self.requests_storage) > self.max_queue_len:
            self.max_queue_len = len(self.requests_storage)

        if self.is_busy and self.remaining_time <= 1e-5:
            self.is_busy = False
            self.processing_request = None
            return FINISH_PROCESS_REQUEST

        if not self.is_busy and len(self.requests_storage) != 0:
            self.processing_request = self.requests_storage.pop(0)
            self.remaining_time = self.time_distribution.get_time()
            self.is_busy = True
```

Результаты работы

Ниже представлены результаты работы программы для 100 студентов с разными вероятностями выбора первого преподавателя.

```
Общее количество запросов:  100
Проверено первым преподавателем:  76
Проверено вторым преподавателем:  24
Макс. очередь у 1-го преподавателя:  23
Макс. очередь у 2-го преподавателя:  14
Макс. очередь на защиту:  31
Общее время проверки студентов 1-м преподавателем (в минутах):  297.7261
Общее время проверки студентов 2-м преподавателем (в минутах):  487.7652
```

Рисунок 2: работа программы при вероятности 80%

```
Общее количество запросов:  100
Проверено первым преподавателем:  55
Проверено вторым преподавателем:  45
Макс. очередь у 1-го преподавателя:  8
Макс. очередь у 2-го преподавателя:  35
Макс. очередь на защиту:  22
Общее время проверки студентов 1-м преподавателем (в минутах):  221.1133
Общее время проверки студентов 2-м преподавателем (в минутах):  884.8889
```

Рисунок 3: работа программы при вероятности 50%

```
Общее количество запросов:  100
Проверено первым преподавателем:  19
Проверено вторым преподавателем:  81
Макс. очередь у 1-го преподавателя:  2
Макс. очередь у 2-го преподавателя:  70
Макс. очередь на защиту:  2
Общее время проверки студентов 1-м преподавателем (в минутах):  74.73
Общее время проверки студентов 2-м преподавателем (в минутах):  1626.1094
```

Рисунок 4: работа программы при вероятности 20%