*Государственное образовательное учреждение высшего профессионального образования*

*«Московский Государственный Технический Университет имени Н. Э. Баумана»*

ОТЧЕТ

По лабораторной работе №5

«Система массового обслуживания с несколькими очередями заявок»

По курсу «Моделирование»

|  |  |
| --- | --- |
| Студент: | Марабян К.В. |
| Группа: | ИУ7-75 |
| Преподаватель: | Рудаков И.В. |

Москва, 2019

**Задание**

В информационный центр приходят клиенты через интервал времени 10±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±5; 40±10; 40±20. 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. Необходимо для этого создать концептуальную модель в терминах СМО, определить Эндогенные и Экзогенные переменные и уравнения модели.

В процессе взаимодействия клиентов с информационным центром возможно:

1) Режим нормального обслуживания, т.е. клиент выбирает одного из свободных операторов, отдавая предпочтение тому у которого меньше номер.

1. Режим отказа в обслуживании клиента, когда все операторы заняты

Определить вероятность отказа системы по формуле:



**Реализация**

Ниже приведены фрагменты кода для реализация протяжки модельного времени методом дельта-t и проверки готовности компьютеров и операторов:

|  |
| --- |
| def event\_based\_modelling(client\_m, client\_d,  op0\_m, op0\_d, op1\_m, op1\_d, op2\_m, op2\_d,  comp0\_m, comp1\_m, c\_count):  client\_gen = RequestGenerator(UniformGenerator(client\_m, client\_d))  op0 = RequestProcessor(UniformGenerator(op0\_m, op0\_d), max\_queue\_size=1)  op1 = RequestProcessor(UniformGenerator(op1\_m, op1\_d), max\_queue\_size=1)  op2 = RequestProcessor(UniformGenerator(op2\_m, op2\_d), max\_queue\_size=1)  comp0 = RequestProcessor(ConstGenerator(comp0\_m))  comp1 = RequestProcessor(ConstGenerator(comp1\_m))  client\_gen.add\_receiver(op0)  client\_gen.add\_receiver(op1)  client\_gen.add\_receiver(op2)  op0.add\_receiver(comp0)  op1.add\_receiver(comp0)  op2.add\_receiver(comp1)  devices = [client\_gen, op0, op1, op2, comp0, comp1]  for device in devices:  device.next\_event\_time = 0  dropped\_requests = 0  client\_gen.next\_event\_time = client\_gen.generate\_time()  op0.next\_event\_time = op0.generate\_time()  while client\_gen.generated\_requests < c\_count:  # Find next event  current\_time = client\_gen.next\_event\_time  for device in devices:  if 0 < device.next\_event\_time < current\_time:  current\_time = device.next\_event\_time  for device in devices:  if current\_time == device.next\_event\_time:  if not isinstance(device, RequestProcessor):  assigned\_processor = client\_gen.emit\_request()  if assigned\_processor is not None:  assigned\_processor.next\_event\_time = (current\_time +  assigned\_processor.generate\_time())  else:  dropped\_requests += 1  client\_gen.next\_event\_time = current\_time + client\_gen.generate\_time()  else:  device.process()  if device.queued\_requests == 0:  device.next\_event\_time = 0  else:  device.next\_event\_time = current\_time + device.generate\_time()  return dropped\_requests / c\_count |

Интерфейс и результаты работы программы представлены на рисунке 1.

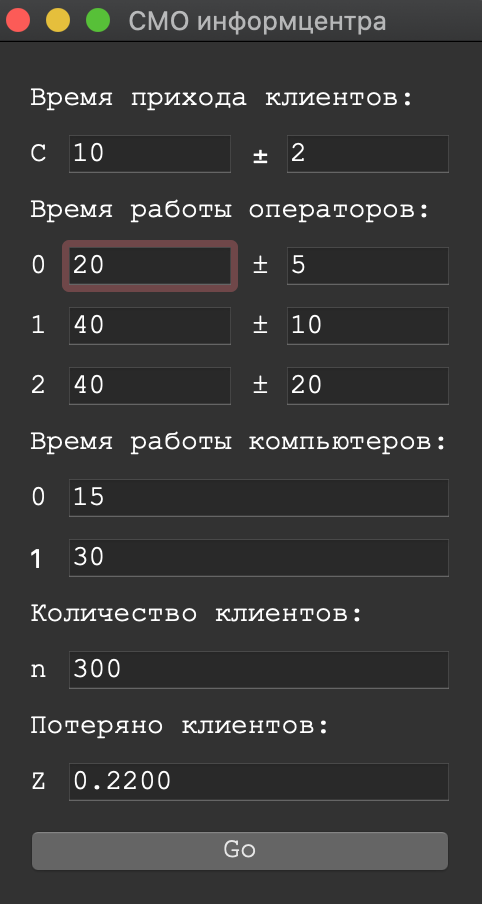


Рисунок 1. Пример работы программы