|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

*к лабораторной работе №5*

*По курсу: «Моделирование»*

*Тема:* ***«Моделирование работы информационного центра»***

Студент ИУ7-74Б

Жабин Д.В.

Преподаватель

Рудаков И.В.

*Москва, 2022 г.*

# 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 минут. Клиенты стремятся занять свободного оператора с максимальной производительностью. Полученные запросы поступают в накопитель, откуда затем выбираются на обработку. На первый компьютер поступают запросы от

1-ого и 2-ого операторов, на второй – от 3-его оператора. Время обработки запросов первым и вторым компьютерами равны соответственно 15 и 30 мин. Промоделировать процесс обработки 300 запросов. Определить вероятность отказа.

Создать концептуальную модель в терминах СМО, определить эндогенные и экзогенные переменные и уравнения модели. За единицу системного времени принять 0.01 минуты.



Рисунок 1 – Схема моделируемой системы

# Теоретическая часть

В процессе взаимодействия клиентов с информационным центром возможны следующие ситуации:

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

2) Режим отказа в обслуживании клиента, когда все операторы заняты.

**Эндогенные переменные**: время обработки задания i-ым оператором, время решения задания j-ым компьютером.

**Экзогенные переменные**: число обслуженных клиентов и число клиентов, получивших отказ.

Структурная схема представлена на рисунке 2, К1-К3 моделируют работу операторов, а К4-К5 – компьютеров.



Рисунок 2 – Структурная схема модели

Вероятность отказа вычисляется следующим образом:



# Текст программы

|  |
| --- |
| from random import random  EPS = 1e-5  dt = 0.01  class EvenDistribution:  def \_\_init\_\_(self, a, b):  self.a = a  self.b = b  def generate(self):  return self.a + (self.b - self.a) \* random()  class Request:  cur\_id = 0  def \_\_init\_\_(self):  self.id = Request.cur\_id  Request.cur\_id += 1  class Generator:  def \_\_init\_\_(self, distribution):  self.distrib = distribution  self.time\_left = 0  def update(self):  self.time\_left -= dt  if self.time\_left <= EPS:  self.time\_left = self.distrib.generate()  return Request()  return None  class Operator:  def \_\_init\_\_(self, storage, distribution):  self.distrib = distribution  self.busy = False  self.storage = storage  self.current\_req = None  self.time\_left = 0  def acccept\_req(self, request):  self.busy = True  self.current\_req = request  self.time\_left = self.distrib.generate()  def complete\_req(self):  self.storage.append(self.current\_req)  self.busy = False  self.current\_req = None  def update(self):  self.time\_left -= dt  if self.busy and self.time\_left <= EPS:  self.complete\_req()  class Processor:  def \_\_init\_\_(self, req\_queue, distribution):  self.distrib = distribution  self.busy = False  self.req\_queue = req\_queue  self.current\_req = None  self.time\_left = 0  def update(self):  self.time\_left -= dt  if self.busy and self.time\_left <= EPS:  self.busy = False  self.current\_req = None  return 'completed'  if not self.busy and len(self.req\_queue) > 0:  self.current\_req = self.req\_queue.pop(0)  self.time\_left = self.distrib.generate()  self.busy = True  def choose\_operator(ops):  for i in range(len(ops)):  if not ops[i].busy:  return i  return -1  def time\_step(generator, operators, processors, req\_info, gen\_flag = True):  if gen\_flag:  request = generator.update()  if request:  req\_info['generated'] += 1  op\_ind = choose\_operator(operators)  if op\_ind == -1:  req\_info['denied'] += 1  else:  operators[op\_ind].acccept\_req(request)  for op in operators:  op.update()  for proc in processors:  res = proc.update()  if res == 'completed':  req\_info['processed'] += 1  def model(generator, operators, processors, total\_reqs):  req\_info = {'generated': 0, 'denied': 0, 'processed': 0}  time = 0  while req\_info['generated'] < total\_reqs:  time\_step(generator, operators, processors, req\_info)  time += dt  while req\_info['denied'] + req\_info['processed'] < total\_reqs:  time\_step(generator, operators, processors, req\_info, False)  time += dt  return req\_info, time  def main():  queue1 = []  queue2 = []    clients = Generator(EvenDistribution(8, 12))  operators = [  Operator(queue1, EvenDistribution(15, 25)),  Operator(queue1, EvenDistribution(30, 50)),  Operator(queue2, EvenDistribution(20, 60))  ]  processors = [  Processor(queue1, EvenDistribution(15, 15)),  Processor(queue2, EvenDistribution(30, 30))  ]  total\_reqs = 300  res, time = model(clients, operators, processors, total\_reqs)  print('Заявок сгенерировано:', res['generated'])  print('Заявок обработано:', res['processed'])  print('Заявок отклонено:', res['denied'])  print('Модельное время:', round(time, 2))  print('Вероятность отказа:', round(res['denied'] / total\_reqs, 3))  if \_\_name\_\_ == '\_\_main\_\_':  main() |

# Результаты работы

Результаты моделирования процесса обработки 300 заявок с заданными начальными параметрами представлены на рисунках 3-5.

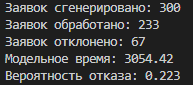


Рисунок 3 – Пример работы №1

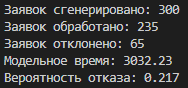


Рисунок 4 – Пример работы №2

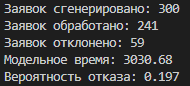


Рисунок 5 – Пример работы №3

Вероятность отказа в данной системе оказалась приблизительно равна 0.20-0.22.