



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» (ИУ7)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

О Т Ч Е Т

по лабораторной работе № 5

Дисциплина: Моделирование

Студент

ИУ7-72Б
(Группа)

(Подпись, дата)

В.А. Иванов
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

И.В. Рудаков
(И.О. Фамилия)

Москва, 2021

1. 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.

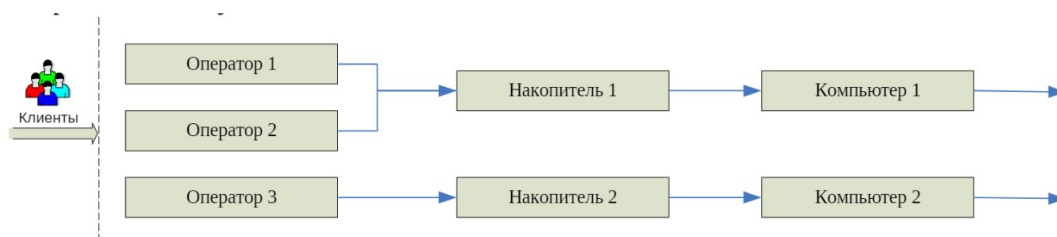


Рис. 1.1 — Визуальное описание задания

2. Результаты

2.1. Описание модели

Визуально данная модель представлена на рисунке 2.1.

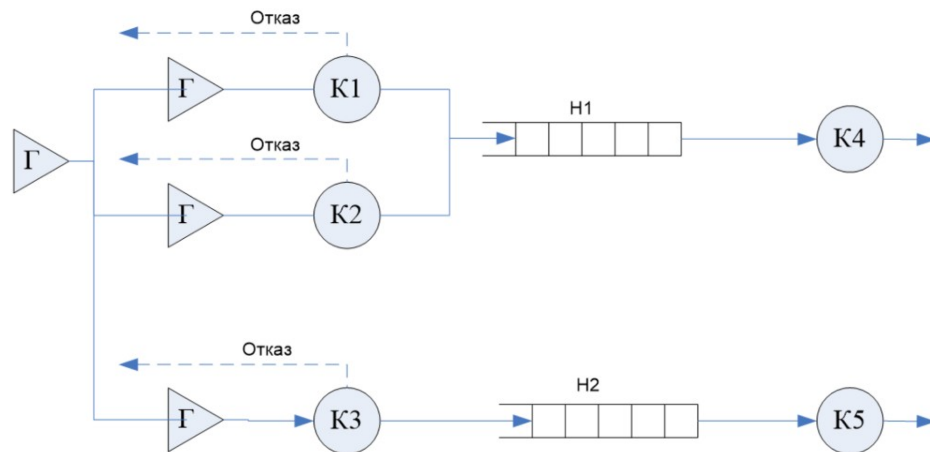


Рис. 2.1 — Структурная схема модели

Согласно условию время обработки заявки оператором подчиняется закону равномерного распределения, а время компьютер выполняет каждую обработку фиксированное время. В терминах СМО данную модель можно описать следующим образом:

- К1, К2, К3 - АО, симулирующие работу операторов. Они являются одноканальными системами с потерями и обозначаются как $G/G/1/1$.
- К4, К5 - АО, симулирующие работу компьютеров. Они являются одноканальными системами с ожиданием и обозначаются как $G/D/1/1$.
- Всю систему можно описать как многоканальную СМО с потерями $G/G/3/1$.

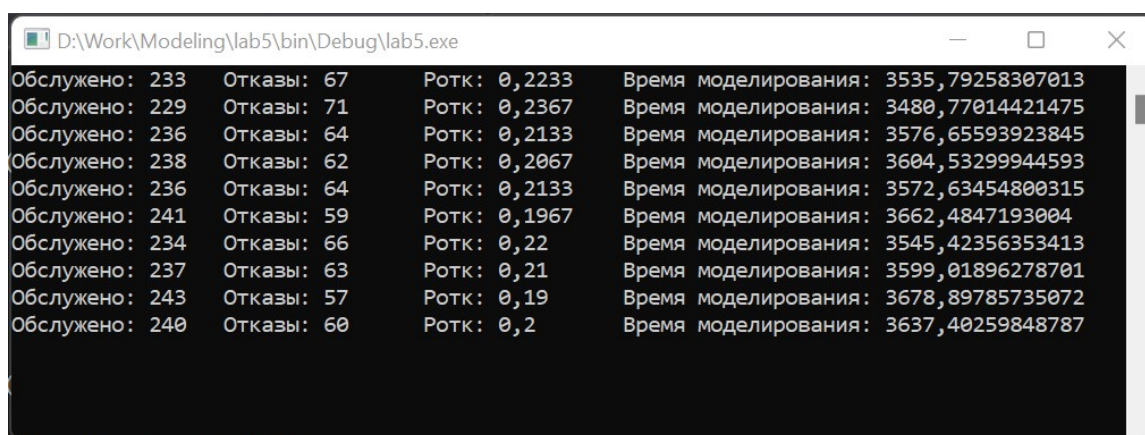
Эндогенными переменными системами являются время обработки заявки для операторов и компьютеров. Экзогенными являются

ся число клиентов, которые были обслужены и число получивших отказ.

2.2. Работа программы

Моделирование происходит при помощи событийного метода. По достижению 300 созданных заявок события генерации новых клиентов более не помещается в очередь событий.

Результаты моделирования приведены на рисунке 2.2.



Обслужено: 233	Отказы: 67	Ротк: 0,2233	Время моделирования: 3535,79258307013
Обслужено: 229	Отказы: 71	Ротк: 0,2367	Время моделирования: 3480,77014421475
Обслужено: 236	Отказы: 64	Ротк: 0,2133	Время моделирования: 3576,65593923845
Обслужено: 238	Отказы: 62	Ротк: 0,2067	Время моделирования: 3604,53299944593
Обслужено: 236	Отказы: 64	Ротк: 0,2133	Время моделирования: 3572,63454800315
Обслужено: 241	Отказы: 59	Ротк: 0,1967	Время моделирования: 3662,4847193004
Обслужено: 234	Отказы: 66	Ротк: 0,22	Время моделирования: 3545,42356353413
Обслужено: 237	Отказы: 63	Ротк: 0,21	Время моделирования: 3599,01896278701
Обслужено: 243	Отказы: 57	Ротк: 0,19	Время моделирования: 3678,89785735072
Обслужено: 240	Отказы: 60	Ротк: 0,2	Время моделирования: 3637,40259848787

Рис. 2.2 — Пример работы программы

3. Текст программы

В листинге 3.1 представлен фрагмент кода программы, отвечающий за моделирование.

Листинг 3.1 — Реализация модели

```
1 public class EGenerated : Event
2 {
3     public EGenerated(double t_) : base(EventTypes.NewClient ,
4         t_) {}
5     public override void Handle(EventModel model)
6     {
7         Request req = model.Gen.New(model.CurT);
8         model.createdN++;
9         if (model.createdN < model.maxCreatedN)
10             model.AddEvent(new EGenerated(model.Gen.WhenReady()));
11
12         for (int i = 0; i < model.Operators.Count; i++)
13         {
14             var oper = model.Operators[i];
15             if (oper.IsFree())
16             {
17                 oper.Put(req, model.CurT);
18                 model.AddEvent(new EOperatorServed(oper.WhenReady(),
19                     i));
20                 return;
21             }
22         }
23         model.deniedN++;
24     }
25
26 public class EOperatorServed : Event
27 {
28     private int num;
29     public EOperatorServed(double t_, int num_) : base(
```

```

    EventTypes.OperatorServed, t_)
30 {
31     this.num = num_;
32 }
33
34 public override void Handle(EventModel model)
35 {
36     Request req = model.Operators[num].Get();
37     int targetQueue = (num == 3) ? 1 : 0;
38     model.CompQueue[targetQueue].Push(req, model.CurT);
39     if (model.Computers[targetQueue].IsFree())
40         model.AddEvent(new EComputerServed(model.CurT,
        targetQueue));
41 }
42 }
43
44 public class EComputerServed : Event
45 {
46     private int num;
47     public EComputerServed(double t_, int num_) : base(
        EventTypes.ComputerServed, t_)
48     {
49         this.num = num_;
50     }
51     public override void Handle(EventModel model)
52     {
53         Request req = model.Computers[num].Get();
54         if (req != null)
55             model.servedN++;
56
57         if (model.CompQueue[num].Count == 0)
58             return;
59         req = model.CompQueue[num].Pop();
60         if (req != null)

```

```

61     {
62         model.Computers[num].Put(req, model.CurT);
63         model.AddEvent(new EComputerServed(model.Computers[num]
        ].WhenReady(), num));
64     }
65 }
66 }
67
68 public class EventModel
69 {
70     public double CurT;
71     public Generator Gen;
72     public List<Service> Operators;
73     public List<Service> Computers;
74     public List<ReqQueue> CompQueue;
75
76     public int createdN = 0;
77     public int servedN = 0;
78     public int deniedN = 0;
79     public int maxCreatedN = 300;
80
81     private List<Event> Events;
82
83     public EventModel(Generator generator, List<Service>
        operators, List<Service> computers)
84     {
85         Gen = generator;
86         Operators = operators;
87         Computers = computers;
88         CompQueue = new List<ReqQueue> { new ReqQueue(), new
        ReqQueue() };
89     }
90
91     private void Reset()

```

```

92  {
93      createdN = 0;
94      servedN = 0;
95      deniedN = 0;
96      CurT = 0;
97
98      Gen.New(CurT);
99      Events = new List<Event> { new EGenerated(Gen.WhenReady()
100 ) };
101 }
102
103 public ModelingResult Run()
104 {
105     Reset();
106     while (Events.Count > 0)
107     {
108         Event e = this.Events[0];
109         this.Events.RemoveAt(0);
110
111         CurT = e.time;
112         e.Handle(this);
113     }
114
115     return new ModelingResult(this.servedN, this.deniedN,
116 this.CurT);
117 }
118
119 public void AddEvent(Event e)
120 {
121     Events.Add(e);
122     Events.Sort();
123 }
124 }

```