



**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 № 6

Дисциплина: Моделирование

Студент

ИУ7-72Б

(Группа)

(Подпись, дата)

В.А. Иванов

(И.О. Фамилия)

Преподаватель

И.В. Рудаков

(Подпись, дата)

(И.О. Фамилия)

Москва, 2021

1. Теоретическая часть

1.1. Задание

Заданием данной лабораторной работы является создание модели для придуманного объекта.

В качестве моделируемого объекта была выбрана сдача лабораторных и получение зачёта.

На зачёт приходят студенты через интервал времени 5 ± 2 минуты. Они делятся на тех, у кого сдано 0, 1 и 2 лабораторных работы. Последние сразу получают билет и становятся в очередь для ответа преподавателю. На приём одного зачёта преподаватель тратит 5 ± 1 минут, при этом есть 5% вероятность того, что студент будет пойман на списывании (реальное количество списывающих студентов не уточняется), в таком случае он получает отказ и уходит без зачёта.

Студенты, не сдавшие все лабораторные, показывают их последовательно магистрам, после чего также становятся в очередь на зачёт. На приём каждой лабораторной также создаётся по очереди. Первая лабораторная принимается одним магистром, обеспечивающим обслуживание работы за 10 ± 1 . С вероятностью в 10% он может найти погрешность и отправить студента доделывать лабораторную в конец своей очереди. Аналогично лабораторная №2 принимается двумя магистрами с интервалами приёма в 15 ± 5 , 10 ± 2 минуты и вероятностью нахождения недочёта в 10% и 20% соответственно.

Промоделировать процесс обработки 100 студентов.

1.2. Описание модели

Структурная схема модели представлена на рисунке 1.2.

Согласно условию время обработки студента магистрами и преподавателем подчиняется закону равномерного распределения. Автоматы обслуживания в модели классифицируются следующим

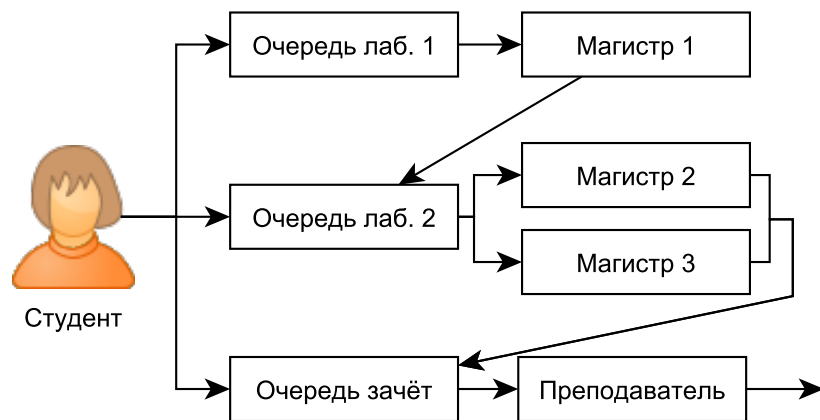


Рис. 1.1 — Схема модели

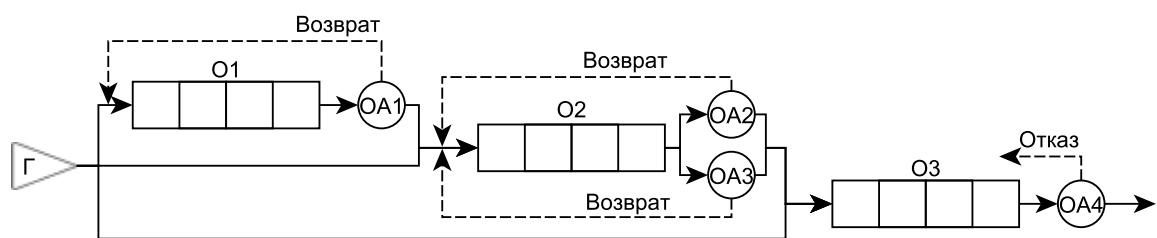


Рис. 1.2 — Структурная схема модели

образом:

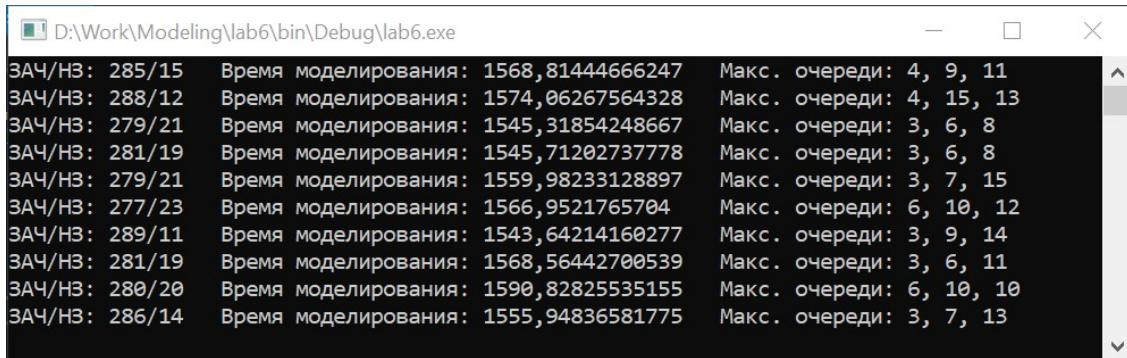
- OA1, OA2, OA3 - АО, симулирующие работу магистров.
- OA4 симулирует работу преподавателя.

Эндогенными переменными системами являются время обработки студентов для магистров и преподавателя.

Экзогенными являются число студентов, которые получили и не получили зачёт, пиковая длина каждой из очередей.

1.3. Работа программы

Результаты моделирования приведены на рисунке 1.3.



ЗАЧ/НЗ: 285/15	Время моделирования: 1568,81444666247	Макс. очереди: 4, 9, 11
ЗАЧ/НЗ: 288/12	Время моделирования: 1574,06267564328	Макс. очереди: 4, 15, 13
ЗАЧ/НЗ: 279/21	Время моделирования: 1545,31854248667	Макс. очереди: 3, 6, 8
ЗАЧ/НЗ: 281/19	Время моделирования: 1545,71202737778	Макс. очереди: 3, 6, 8
ЗАЧ/НЗ: 279/21	Время моделирования: 1559,98233128897	Макс. очереди: 3, 7, 15
ЗАЧ/НЗ: 277/23	Время моделирования: 1566,9521765704	Макс. очереди: 6, 10, 12
ЗАЧ/НЗ: 289/11	Время моделирования: 1543,64214160277	Макс. очереди: 3, 9, 14
ЗАЧ/НЗ: 281/19	Время моделирования: 1568,56442700539	Макс. очереди: 3, 6, 11
ЗАЧ/НЗ: 280/20	Время моделирования: 1590,82825535155	Макс. очереди: 6, 10, 10
ЗАЧ/НЗ: 286/14	Время моделирования: 1555,94836581775	Макс. очереди: 3, 7, 13

Рис. 1.3 — Пример работы программы

2. Текст программы

В листинге 2.1 представлен фрагмент кода программы, отвечающий за моделирование.

Листинг 2.1 — Реализация модели

```
1 public enum EventTypes
2 {
3     NewStudent ,
4     Lab1Done ,
5     Lab2Done ,
6     ExamDone ,
7     End
8 }
9
10 public abstract class Event : IComparable<Event>
11 {
12     public static Random rnd = new Random();
13     public EventTypes type;
14     public double time;
15     public Event(EventTypes type_ , double t_)
16     {
17         this.type = type_ ;
18         this.time = t_ ;
19     }
20
21     public int CompareTo(Event other)
22     {
23         if (this.time > other.time)
24             return 1;
25         else
26             return -1;
27     }
28
29     abstract public void Handle(EventModel model);
30 }
31
```

```

32 public class EGenerated : Event
33 {
34     public EGenerated(double t_) : base(EventTypes.NewStudent ,
        t_) { }
35     public override void Handle(EventModel model)
36     {
37         Request req = model.Gen.New(model.CurT);
38         model.createdN++;
39         if (model.createdN < model.maxCreatedN)
40             model.AddEvent(new EGenerated(model.Gen.WhenReady()));
41
42         var switchOption = rnd.Next() % 3;
43         switch (switchOption)
44         {
45             case 0:
46                 // Student goes to lab 1
47                 model.Queues[0].Push(req, model.CurT);
48                 if (model.Masters[0].IsFree())
49                     model.AddEvent(new ELab1Done(model.CurT));
50                 break;
51
52             case 1:
53                 // Student goes to lab 2
54                 model.Queues[1].Push(req, model.CurT);
55                 if (model.Masters[1].IsFree() && model.Masters[2].
IsFree())
56                     model.AddEvent(new ELab2Done(model.CurT, 1 + rnd.Next()
% 2));
57                 else if (model.Masters[1].IsFree())
58                     model.AddEvent(new ELab2Done(model.CurT, 1));
59                 else if (model.Masters[2].IsFree())
60                     model.AddEvent(new ELab2Done(model.CurT, 2));
61                 break;
62

```

```

63     case 2:
64         // Student goes to exam
65         model.Queues[2].Push(req, model.CurT);
66         if (model.Lecturer.IsFree())
67             model.AddEvent(new EExamDone(model.CurT));
68         break;
69
70     default:
71         break;
72 }
73 }
74 }
75
76 public class ELab1Done : Event
77 {
78     public ELab1Done(double t_) : base(EventTypes.Lab1Done, t_)
79     {}
80
81     public override void Handle(EventModel model)
82     {
83         Request req = model.Masters[0].Get();
84         if (req != null)
85         {
86             if (rnd.NextDouble() < model.Masters[0].ReturnP)
87                 model.Queues[0].Push(req, model.CurT);
88             else
89             {
90                 model.Queues[1].Push(req, model.CurT);
91                 if (model.Masters[1].IsFree() && model.Masters[2].
IsFree())
92                     model.AddEvent(new ELab2Done(model.CurT, 1 + rnd.Next
93                     () % 2));
94                 else if (model.Masters[1].IsFree())
95                     model.AddEvent(new ELab2Done(model.CurT, 1));

```

```

94         else if (model.Masters[2].IsFree())
95             model.AddEvent(new ELab2Done(model.CurT, 2));
96     }
97 }
98
99     if (model.Queues[0].Count == 0)
100         return;
101     req = model.Queues[0].Pop();
102     if (req != null)
103     {
104         model.Masters[0].Put(req, model.CurT);
105         model.AddEvent(new ELab1Done(model.Masters[0].WhenReady
106             ()));
107     }
108 }
109
110 public class ELab2Done : Event
111 {
112     private int Num;
113     public ELab2Done(double t_, int num) : base(EventTypes.
114         Lab2Done, t_) {Num = num;}
115
116     public override void Handle(EventModel model)
117     {
118         Request req = model.Masters[Num].Get();
119         if (req != null)
120         {
121             if (rnd.NextDouble() < model.Masters[Num].ReturnP)
122                 model.Queues[1].Push(req, model.CurT);
123             else
124             {
125                 model.Queues[2].Push(req, model.CurT);
126                 if (model.Lecturer.IsFree())

```



```

126         model.AddEvent(new EExamDone(model.CurT));
127     }
128 }
129
130     if (model.Queues[1].Count == 0)
131         return;
132     req = model.Queues[1].Pop();
133     if (req != null)
134     {
135         model.Masters[Num].Put(req, model.CurT);
136         model.AddEvent(new ELab2Done(model.Masters[Num].
WhenReady(), Num));
137     }
138 }
139 }
140
141 public class EExamDone : Event
142 {
143     public EExamDone(double t_) : base(EventTypes.ExamDone, t_)
144     { }
145     public override void Handle(EventModel model)
146     {
147         Request req = model.Lecturer.Get();
148         if (req != null)
149         {
150             if (rnd.NextDouble() < model.Lecturer.ReturnP)
151                 model.deniedN++;
152             else
153                 model.servedN++;
154         }
155
156         if (model.Queues[2].Count == 0)
157             return;
158         req = model.Queues[2].Pop();

```

```

158     if (req != null)
159     {
160         model.Lecturer.Put(req, model.CurT);
161         model.AddEvent(new EExamDone(model.Lecturer.WhenReady()
162             ));
163     }
164 }
165
166 public class EventModel
167 {
168     public double CurT;
169     public Generator Gen;
170     public List<Service> Masters;
171     public Service Lecturer;
172     public List<ReqQueue> Queues;
173
174     public int createdN = 0;
175     public int servedN = 0;
176     public int deniedN = 0;
177     public int maxCreatedN = 300;
178     private List<Event> Events;
179
180     public EventModel(Generator generator, List<Service>
181         masters, Service lecturer)
182     {
183         Gen = generator;
184         Masters = masters;
185         Lecturer = lecturer;
186     }
187
188     private void Reset()
189     {
190         createdN = 0;

```

```

190     servedN = 0;
191     deniedN = 0;
192     CurT = 0;
193     Gen.New(CurT);
194     Events = new List<Event> {
195         new EGenerated(Gen.WhenReady())
196     };
197     Queues = new List<ReqQueue> { new ReqQueue(), new
198 ReqQueue(), new ReqQueue() };
199
200     public ModelingResult Run()
201     {
202         Reset();
203         while (Events.Count > 0)
204         {
205             Event e = this.Events[0];
206             this.Events.RemoveAt(0);
207             CurT = e.time;
208             e.Handle(this);
209         }
210         return new ModelingResult(this.servedN, this.deniedN,
211 this.CurT, this.Queues);
212
213     public void AddEvent(Event e)
214     {
215         Events.Add(e);
216         Events.Sort();
217     }
218 }

```