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

По лабораторной работе №6

По курсу: «Моделирование»

Тема: «Моделирование защиты НИР»

Студент: Пронин А. С.

Группа: ИУ7-72Б

Преподаватель: Рудаков И. В.

Оценка: _____

Москва

2022

Задание

Заданием данной лабораторной работы является создание модели для придуманного объекта. В качестве моделируемого объекта была выбрана защита студентами НИР.

На защиту приходит весь поток студентов. Для возможности защиты им необходимо сначала пройти нормконтроль. Нормконтроль производится за $3 - 5$ минут и проходится успешно с вероятностью $0,7$. Если результат нормконтроля отрицательный, студет отправляется обратно в очередь, исправлять РПЗ. Если студент проходит нормконтроль успешно, то он готов к защите и ожидает пока освободится одна из комиссий. Если свободны обе комиссии, студент идет к 1-ой. 1-ая комиссия принимает студента за $5 - 10$ минут и засчитывает ему НИР с вероятностью $0,9$. 2-ая комиссия принимает студента за $30 - 40$ минут и засчитывает ему НИР с вероятностью $0,1$. Несдавшие студенты отправляются на защиту в другой день.

Процесс моделируется для 120 студентов.

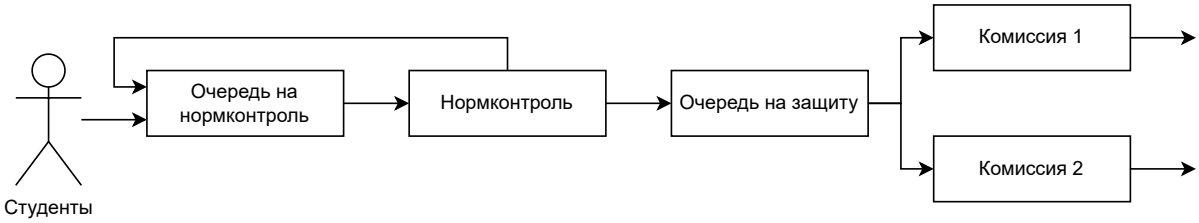

Рис. 1: Структурная схема модели

1 Теория

1.1 Схема модели (каналы и накопители)

Схема модели в виде каналов и накопителей представленна на рисунке 1.1.

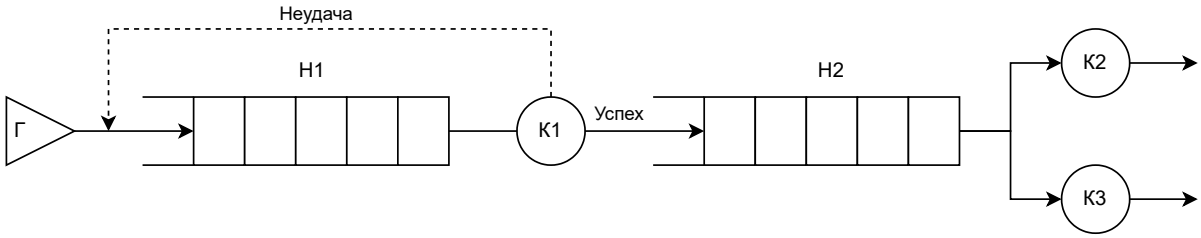

Рис. 1.1: Схема модели (каналы и накопители)

Согласно условию время прохождения нормконтроля и защиты НИР студентом подчиняется закону равномерного распределения. Автоматы обслуживания в модели классифицируются следующим образом:

- К1 симулирует работу нормконтроля;
- К2 и К3 симулируют работу комиссий.

Эндогенные переменные – время прохождения нормконтроля, вероятность успешного прохождения нормконтроля, время защиты НИР и вероятность успешной защиты i-ой комиссии ($i = \overline{0;1}$).

Экзогенные переменные – n_0 равное числу защитившихся студентов и n_1 равное числу несдавших студентов.

Уравнения модели: вероятность сдачи $= n_0/(n_0 + n_1)$

За единицу дискретного времени выбрана 0.01 минуты.

2 Текст программы

В листингах 2.1–2.6 представлен код программы, отвечающий за моделирование.

Листинг 2.1: Код генератора

```
class Generator
{
  public Generator(double _a, double _b)
  {
    a = _a;
    b = _b;
  }

  public bool isReady(double t)
  {
    return (t >= gen_time);
  }

  public Request genRequest()
  {
    Request new_r = new Request(gen_time);
    updateGenTime();
    return new_r;
  }

  private void updateGenTime()
  {
    double t_i = a + (b - a) * rnd.NextDouble();
    gen_time += Math.Round(t_i, 2);
  }

  public double a;
  public double b;
  public double gen_time = 0;

  private Random rnd = new Random();
}
```

Листинг 2.2: Код заявки

```
class Request
{
  public Request(double t)
  {
    create_time = t;
  }

  public double create_time = 0;
  public double serve_time = -1;
}
```

Листинг 2.3: Код очереди заявок

```
class ReqQue : Queue<Request>
{
  public bool push(Request r)
  {
    Enqueue(r);
    if (Count > max_size)
    {
      max_size = Count;
      return true;
    }
    return false;
  }
  public Request pop()
  {
    return Dequeue();
  }

  public int max_size = 0;
}
```

Листинг 2.4: Код обслуживающего аппарата

```
class Service
{
  public Service(double _a, double _b, double _p)
  {
    a = _a;
    b = _b;
    pass_prob = _p;
  }

  public bool isFree(double t)
  {
    return (t >= free_time);
  }

  public bool serve(Request r, double t)
  {
    updateFreeTime(t);
    double res = rnd.NextDouble();
    bool pass = res <= pass_prob;
    if (pass)
      r.serve_time = free_time;
    return pass;
  }

  public void updateFreeTime(double t)
  {
    double t_i = a + (b - a) * rnd.NextDouble();
    free_time = t + Math.Round(t_i, 2);
  }

  public double a;
  public double b;
  public double free_time = 0;
  public double pass_prob = 0;

  private Random rnd = new Random();
}
```

Листинг 2.5: Код событий

```
abstract class BaseEvent : IComparable<BaseEvent>
{
  public BaseEvent(double _time)
  {
    time = _time;
  }

  public int CompareTo(BaseEvent other)
  {
    if (this.time > other.time)
      return 1;
    else
      return -1;
  }
  abstract public void Handle(EventModel model);

  public double time;
}

class EStudentCame : BaseEvent
{
  public EStudentCame(Request _student) : base(_student.
   create_time)
  {
    student = _student;
  }

  public override void Handle(EventModel model)
  {
    model.students_n++;

    if (model.students_n < model.max_students_n)
    {
      Request next_student = model.students.genRequest();
      model.addEvent(new EStudentCame(next_student));
    }


```

```
40      model.normcontrol_que.push(student);
41      if (model.normcontrol.isFree(this.time))
42        model.addEvent(new EStudentNormcontrolPass(this.time));
43    }
44
45    private Request student;
46  }
47
48  class EStudentNormcontrolPass : BaseEvent
49  {
50    public EStudentNormcontrolPass(double _time) : base(_time) {}
51
52    public override void Handle(EventModel model)
53    {
54      if (model.normcontrol_que.Count != 0)
55      {
56        Request student = model.normcontrol_que.pop();
57        bool pass = model.normcontrol.serve(student, this.time);
58        if (!pass)
59        {
60          model.normcontrol_que.push(student);
61          model.addEvent(new EStudentNormcontrolPass(model.
    normcontrol.free_time));
62        }
63        else
64        {
65          Request student_rdy = new Request(student.serve_time);
66          model.protection_que.push(student_rdy);
67          if (model.comissions[0].isFree(student_rdy.create_time)
    || model.comissions[1].isFree(student_rdy.create_time))
68            model.addEvent(new EStudentProtection(student_rdy.
    create_time));
69        }
70        model.addEvent(new EStudentNormcontrolPass(model.
    normcontrol.free_time));
71      }
72    }
73  }
74
75
76
```

```
77  class EStudentProtection : BaseEvent
78  {
79    public EStudentProtection(double _time) : base(_time) {}
80
81    public override void Handle(EventModel model)
82    {
83      if (model.protection_que.Count != 0)
84      {
85        for (int i = 0; i < model.comissions.Count; i++)
86        {
87          if (model.comissions[i].isFree(this.time))
88          {
89            Request student_rdy = model.protection_que.pop();
90            bool pass = model.comissions[i].serve(student_rdy,
      this.time);
91            if (!pass)
92              model.refused_n++;
93            else
94              model.passed_n++;
95
96            model.addEvent(new EStudentProtection(model.comissions
      [i].free_time));
97            break;
98          }
99        }
100     }
101   }
102 }
```

Листинг 2.6: Код событийной модели

```
class EventModel
{
  public EventModel(int _max_clients_n = max_default)
  {
    students = new Generator(0, 0);
    normcontrol_que = new ReqQue();
    normcontrol = new Service(3, 5, 0.7);
    protection_que = new ReqQue();
    comissions = new List<Service> { new Service(5, 10, 0.9),
   new Service(30, 40, 0.1) };
    max_students_n = _max_clients_n;
  }

  public double getResults()
  {
    reset();

    while (events.Count > 0)
    {
      BaseEvent e = events[0];
      events.RemoveAt(0);
      end_time = e.time;
      e.Handle(this);
    }

    return (double)refused_n / (refused_n + passed_n);
  }

  public void addEvent(BaseEvent e)
  {
    events.Add(e);
    events.Sort();
  }

  public string getResultsStr()
  {
    string res = String.Format("Прошло студентов: {0:D} | ", this.
   passed_n);
    res += String.Format("Несдавших студентов: {0:D} | ", this.
   refused_n);
```

```
38      res += String.Format("Вероятность сдачи: {0:F4} | ", Math.Round
    ((double)passed_n / (refused_n + passed_n), 4));
39      res += String.Format("Время моделирования: {0:F2} мин[] ",
    this.end_time);
40
41      return res;
42    }
43
44    private void reset()
45    {
46      students.gen_time = 0;
47      normcontrol_que.max_size = 0;
48      normcontrol.free_time = 0;
49      protection_que.max_size = 0;
50      for (int i = 0; i < comissions.Count; i++)
51        comissions[i].free_time = 0;
52
53      end_time = 0;
54      refused_n = 0;
55      passed_n = 0;
56
57      students_n = 0;
58
59      Request first_student = students.genRequest();
60      events = new List<BaseEvent> { new EStudentCame(
    first_student) };
61    }
62
63    public Generator students;
64    public ReqQue normcontrol_que;
65    public Service normcontrol;
66    public ReqQue protection_que;
67    public List<Service> comissions;
68    const int max_default = 120;
69    public int max_students_n = max_default;
70    public double end_time = 0;
71    public int refused_n = 0;
72    public int passed_n = 0;
73    public int students_n = 0;
74    private List<BaseEvent> events = new List<BaseEvent>();
75  }
```

3 Результаты

Пример работы программы приведен на рисунке 3.1.

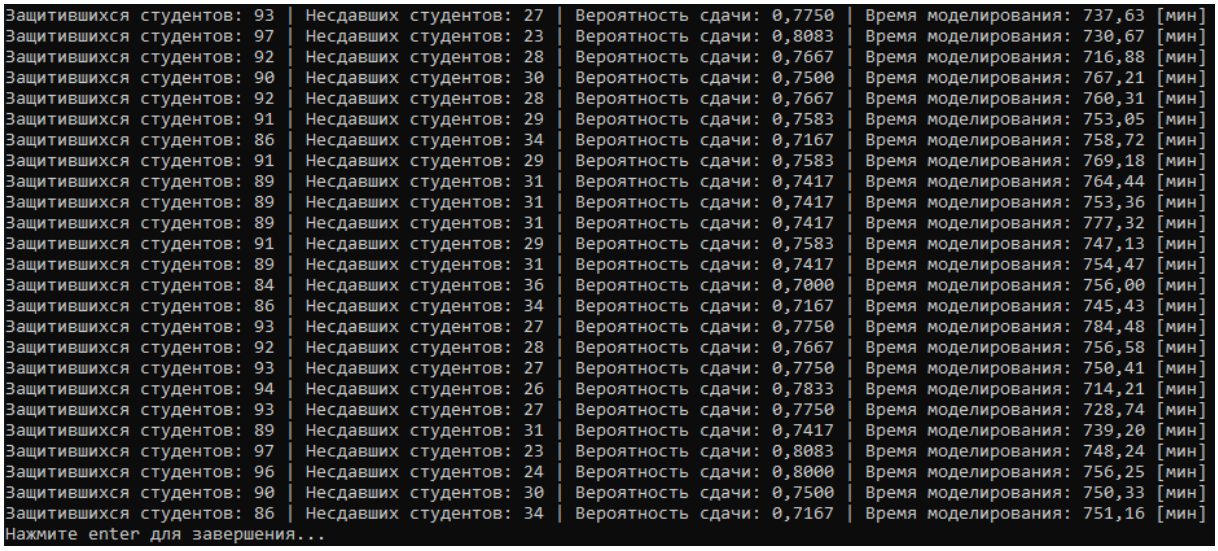

Рис. 3.1: Пример работы программы