МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования "Рыбинский государственный авиационный технический университет имени П.А. Соловьева"

КАФЕДРА "ЭМиЭИС"

Контрольня работа №1

по дисциплине "Математическое и имитационное моделирование"

Вариант 2

Группа: ЗИП-14

Студент: Лебедев Е. В.

Преподаватель: Буров Н.Н.

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рыбинск 2017

Содержание

[Введение 3](#_Toc482993027)

[Формулировка проблемы и определение целей имитационного моделирования 3](#_Toc482993028)

[Подготовка исходных данных. Разработка концептуальной модели 4](#_Toc482993029)

[Формализация имитационной модели 5](#_Toc482993030)

[Интерфейс системы. 8](#_Toc482993031)

[Заключение 10](#_Toc482993032)

[Приложение. 12](#_Toc482993033)

# Введение

Целью контрольной работы является овладение технологией и приёмами практического решения задач моделирования процессов функционирования системы на ЭВМ, что необходимо для того, чтобы научиться моделированию и освоить широкий круг его возможностей.

В данной контрольной работе рассматривается моделирование процессов в Q-схемах – одном из важнейших, с точки зрения применения на практике, классов математических схем, разработанных для формализации процессов функционирования систем массового обслуживания (СМО) в теории массового обслуживания. Предметом изучения в теории массового обслуживания являются системы, в которых появление заявок (требований) на обслуживание и завершение обслуживания происходит в случайные моменты времени, т.е. характер их функционирования носит стохастический характер. Следует отметить, что СМО описывают различные по своей физической природе процессы функционирования экономических, производственных, технических и других систем, например потоки поставок продукции некоторому предприятию, потоки деталей и комплектующих изделий на сборочном конвейере цеха, заявки на обработку информации в ЭВМ от удаленных терминалов и т.д.

# Формулировка проблемы и определение целей имитационного моделирования

Основная проблема исследования системы заключается в проверке предположений, выдвинутых относительно ее функционирования, на основе анализа результатов, полученных при выполнении программной реализации модели на ЭВМ.

Объектом моделирования служит система передачи цифровой информации.

Основной целью имитационного исследования является построение достоверной модели, которая позволит изучить особенности моделируемой системы, в частности:

– поведение системы при длительном функционировании;

– определение частот уничтожения транзактов и частот подключения ресурсов.

Также имитация поможет определить возможность оптимизации исследуемой системы.

# Подготовка исходных данных. Разработка концептуальной модели

Из условия задания нам неизвестно как располагаются каналы передачи данных, поэтому предположим два варианта. Первый вариант - это пакеты передаются по двум каналам одновременно, т.е. каналы располагаются параллельно. При этом пакеты подаются поочередно в накопители каналов с интервалом 6±3 мс, таким образом, минимальный интервал поступления пакета в накопитель канала составляет 6 мс (3мс+3мс), что больше времени передачи по каналу (5 мс). Получаем, что частота поступления пакетов в очередь (накопитель) канала ниже, чем частота извлечения их из очереди, следовательно, пакеты не будут задерживаться в буфере и суммарное время передачи не превысит 10 мс.

Вторым вариантом предположим, что пакеты передаются последовательно – сначала по одному каналу, затем по второму. В таком случае, существует вероятность накопления пакетов в буфере первого канала и увеличения времени передачи (более 10 мс). Примем за основу вариант с последовательным расположением каналов.

Анализируя исходные данные, можно выдвинуть следующие гипотезы:

– в накопителе первого канала ожидается один и более пакетов в конкретный момент времени;

– в накопителе второго канала ожидается не более одного пакета в конкретный момент времени;

– предполагается, что часть пакетов будет иметь время передачи более 10 мс, следовательно, ожидается периодическое подключение ресурса;

# Формализация имитационной модели

Исследуемая система массового обслуживания относится к классу систем, в которых требуется ожидание (очередь). В таких системах заявка, поступившая в момент, когда все каналы заняты, становится в очередь и ожидает, пока не освободится один из каналов. Когда канал освобождается, одна из заявок, стоящих в очереди, принимается к обслуживанию. В системе заявки обслуживаются в порядке поступления. Время ожидания неограниченное, т.е. заявка ждет, когда освободится канал, который примет ее к обслуживанию.

K2

H2

K1

H1

U

W

Рисунок 2 – Структурная схема модели в символике Q-схем

- W – источник, отражающий входящий поток пакетов;

- H1 – очередь перед каналом А;

- H2 – очередь перед каналом B;

- K1 – канал А;

- K2 – канал B;

- Y – выходящий поток пакетов;

- U – управление.

Для реализации системы я выбрал язык программирования java, являющийся современным высокоуровневым языком, имеющий удобные инструменты программирования и позваляющий запускать программу на большенстве платформах скомпилировав код только один раз.

Блок-схема общего алгоритма работы системы показана на рисунке 3. Система поделена на модули (классы): канал, генератор, пакет, ресурс, декодер. Время системы представлено переменной time, где каждую еденицу берем за 1 мс. Каждый тик время увеличивается на 1мс. Генератор генерирует пакеты, используя функцию рандомизации встроенную в стандартную библиотеку языка Java. Получив рандомное число от 3 до 9 (по условию), генератор сохраняет его и каждый тик уменьшает на 1. Как только число станет равно 0, генератор сгенерирует пакет и получит новое рандомное число. В сгенерированном пакете есть счетчик, который увеличивается на 1 в каждый тик. Сгенерированный пакет отправляет в первый канал, где будет находится либо 5 секунд если очередь пуста и ресурс не подключен, либо 4 если ресурс подключен, либо будет ждать своей очереди. На первый пакет в очереди в канале устанавливается счетчик имитируя передачу этого пакета по каналу, каждый тик счетчки уменьшается и как только будет равен 1 пакет будет считать приняты и может передаваться далее во второй канал. Второй канал работает также как и первый. После него пакет передаётся в декодер. Декодер проверяет счетчик пришедшего пакета, это время за которое доставлен пакет. В декодере имеется 3 переменные: общее число пакетов, количество уничтоженых пакетов и процент уничтоженных пакетов. Если время больше 10мс, то декодер считает пакет негодным и увеличивает счетчки уничтоженых пакетов на 1, и увеличивает счетчик общего числа пакетов на 1 в любом случае. Далее проверяется соотношение уничтоженых пакетов и если оно больше 30%, то в каналы добавляется ресурс, тем самым задавая скорость канала 4 мс вместо стандартных 5 мс. Когда соотношение уничтоженых пакетов станет меньше 30%, то ресурс из каналов удаляется и время передачи опять становится 5мс. Программа завершается когда time станет больше определенного нами времени окончании работы равного 10с (10 000мс). И в завершение показывает отчет о работе моделирования.

# Интерфейс системы.

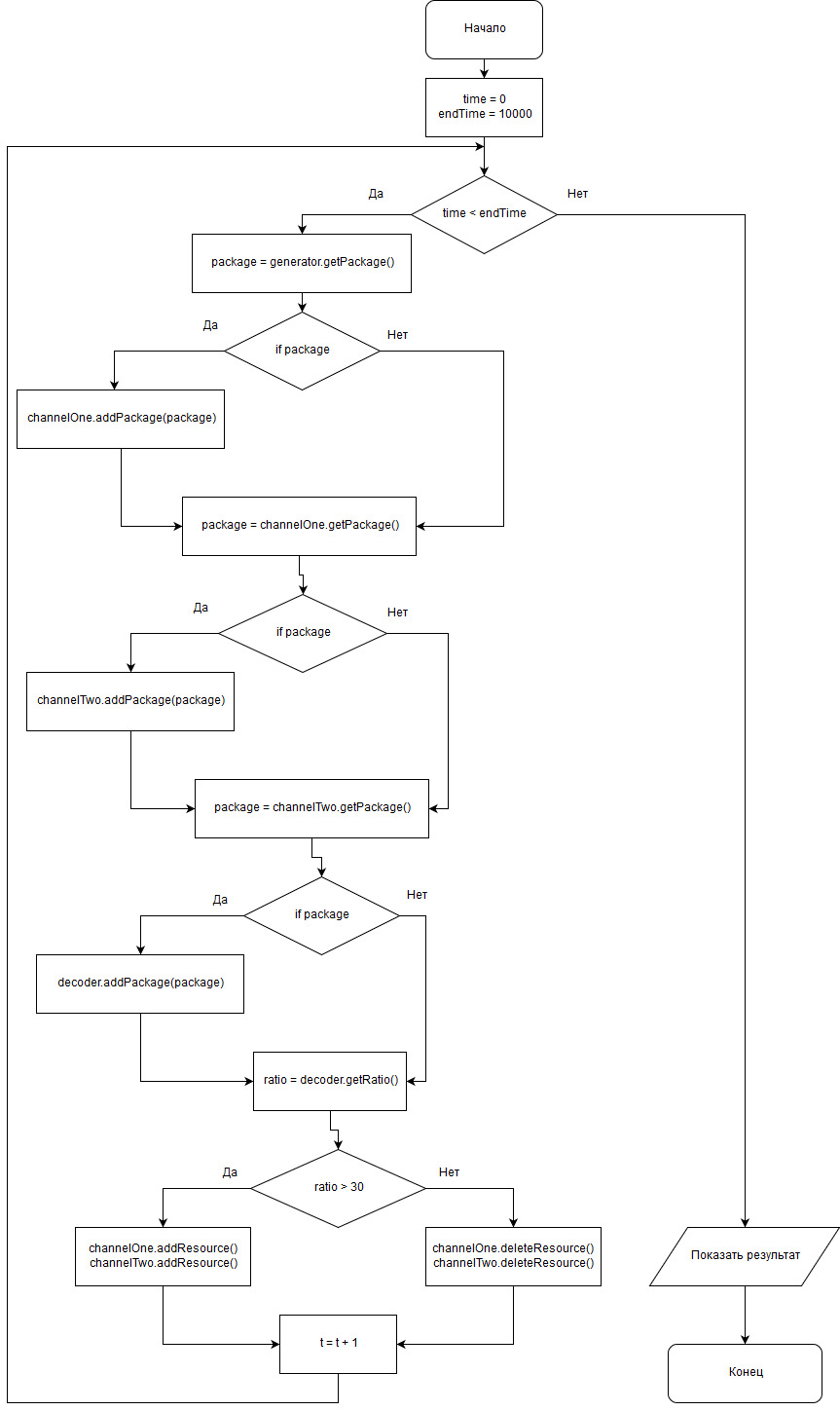
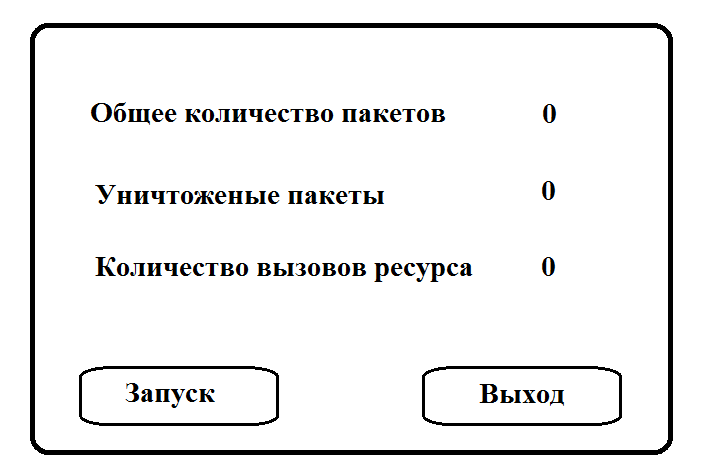


Рисунок 3

Испытание и исслеждование имитационной модели

Теблица 1. результаты испытаний имитационной модели

|  |  |  |  |
| --- | --- | --- | --- |
| Сгенерировано пакетов | Доставлено пакетов во время | Уничтожено пакетов | Количество использования ресурса |
| 1654 | 1141 | 513 | 144 |
| 1669 | 1156 | 513 | 158 |
| 1667 | 1153 | 514 | 166 |
| 1660 | 1154 | 506 | 164 |
| 1676 | 1161 | 515 | 186 |
| 1651 | 1140 | 511 | 132 |
| 1664 | 1149 | 515 | 142 |
| 1675 | 1156 | 519 | 176 |
| 1668 | 1155 | 513 | 152 |
| 1687 | 1167 | 520 | 168 |

Мат ожидание:

|  |  |  |  |
| --- | --- | --- | --- |
| 1667.1 | 1153.2 | 513.9 | 158.8 |

Средне квадратичное отклонение

|  |  |  |  |
| --- | --- | --- | --- |
| 1667 +- 7 | 1153 +- 5 | 514 +- 3 | 159 +- 11 |

Оценим чувствительность результатов имитационного моделирования на изменение времени прихода пакеторв с 5+-3 мс до 7+-3 мс в среднем.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2004 | 1384 | 620 | 180 | х=5 |
| 1424 | 1195 | 231 | 2 | х=7 |
| -34% | -15% | -91% | -196% | dx |

Планирование и проведение направленного вычислительного эксперимента модели

## Заключение

Результатом данной работы стало построение программы, моделирующего процесс функционирования заданной системы. Также были построена схема моделирующего алгоритма.

Таким образом, показано, что машинное моделирование – это эффективное средство решения задач в системе управления технологическим процессом, где появление сообщений (сигналов) на обработку происходит в случайные моменты времени, т.е. характер их функционирования носит стохастический характер.

Список литературы

1. Бусленко Н.П. Моделирование сложных систем. -М.: Наука, 1978. – 399 с.
2. Василенков В.П. Математическое моделирование социально-экономических процессов: практический курс для студентов специальностей «Менеджмент организации» и «Государственное и муниципальное управление» / В.П.Василенков, И.Б.Болотин; Смол.гос.ун-т. – Смоленск: Изд-во СмолГУ, 2009.
3. Варфоломеев В.И. Алгоритмическое моделирование элементов экономических систем. Практикум.-М.: Финансы и статистика, 2004. – 265 с.
4. Введение в математическое моделирование: Учеб.пособие / Под ред. П.В. Трусова. – М.: Университетская книга, Логос, 2007. – 440 с.
5. Емельянов А.А. Структурный анализ и динамические имитационные модели в экономике. - М.: Финансы и статистика, 1998. – 358 с.
6. Емельянов А.А. Имитационное моделирование экономических процессов. – М., 2004.
7. Интрилигатор М. Математические методы оптимизации и экономическая теория / Пер. с англ. Г. И. Жуковой, Ф. Я. Кельмана. – М.: Айрис-прсс, 2002. – 576 с.: ил. – (Высшее образование).
8. Калянов Г.Н. CASE структурный системный анализ (автоматизация и применение).-М.:Лори, 1996. – 243 с.

# Приложение.

Исходный код программы

Class Main

**public class** Main {  
  
 **public int all** = 0;  
 **public int fault** = 0;  
 **public int resource** = 0;  
  
 **public void** start() {  
 **long** time = 0;  
 **long** endTime = 10 \* 1000;  
 Generator generator = **new** Generator();  
 Channel channelOne = **new** Channel();  
 Channel channelTwo = **new** Channel();  
 Decoder decoder = **new** Decoder();  
 Resource resource = **new** Resource();  
 **while** (time < endTime) {  
 Package newPackage = generator.getPackage();  
 **if** (newPackage != **null**) {  
 channelOne.addPackage(newPackage);  
 }  
 Package package1 = channelOne.getPackage();  
 **if** (package1 != **null**) {  
 channelTwo.addPackage(package1);  
 }  
 Package package2 = channelTwo.getPackage();  
 **if** (package2 != **null**) {  
 decoder.addPackage(package2);  
 }  
 **int** ratio = decoder.getGetDeletedRatio();  
 **if** (ratio > 30) {  
 channelOne.addResource(resource);  
 channelTwo.addResource(resource);  
 } **else** {  
 channelOne.deleteResource();  
 channelTwo.deleteResource();  
 }  
 time += 1;  
 channelOne.updatePackagesTime();  
 channelTwo.updatePackagesTime();  
 }  
   
 **this**.**all** = decoder.**all**;  
 **this**.**fault** = decoder.**fault**;  
 **this**.**resource** = resource.getCalled();  
 }  
  
 **public static void** main(String[] args) {  
 Gui dialog = **new** Gui();  
 dialog.pack();  
 dialog.setVisible(**true**);  
 System.*exit*(0);  
 }  
}

Class Channel

**public class** Channel {  
  
 **private** Queue<Package> **packages**;  
 **private int speed**;  
 **private int counter**;  
 **private boolean resourceAdded**;  
  
 **public** Channel() {  
 **this**.**speed** = 5;  
 **this**.**counter** = **this**.**speed**;  
 **this**.**packages** = **new** LinkedList<Package>();  
 }  
  
 **public void** addPackage(Package newPackage) {  
 **if** (**this**.**packages**.isEmpty()) {  
 **this**.**counter** = **this**.**speed**;  
 }  
 **this**.**packages**.add(newPackage);  
 }  
  
 **public** Package getPackage() {  
  
 **if** (**this**.**counter** == 0) {  
 **this**.**counter** = **this**.**speed**;  
 **this**.**counter** -= 1;  
 **return this**.**packages**.poll();  
 }  
 **this**.**counter** -= 1;  
 **return null**;  
 }  
  
 **public void** updatePackagesTime()  
 {  
 **for** (Package pack : **this**.**packages**) {  
 pack.tick();  
 }  
 }  
  
 **public void** addResource(Resource resource) {  
 **if** (**this**.**resourceAdded**) {  
 **return**;  
 }  
 resource.call();  
 **this**.**resourceAdded** = **true**;  
 **this**.**speed** = 4;  
 **if** (**this**.**counter** > 0) {  
 **this**.**counter** -= 1;  
 }  
 }  
  
 **public void** deleteResource() {  
 **this**.**speed** = 5;  
 **this**.**resourceAdded** = **false**;  
 }  
}

Class Decoder

**public class** Decoder {  
 **public int all**;  
 **public int fault**;  
  
 **public** Decoder() {  
 **this**.**all** = 0;  
 **this**.**fault** = 0;  
 }  
  
 **public void** addPackage(Package pack) {  
 **if** (pack.getTime() > 10) {  
 **this**.**fault** += 1;  
 }  
 **this**.**all** += 1;  
 }  
  
 **public int** getGetDeletedRatio() {  
 **if** (**this**.**fault** > 0) {  
 **int** result = (**this**.**fault** \* 100) / **this**.**all**;  
 **return** result;  
 }  
 **return** 0;  
 }  
}

Class Generator

**public class** Generator {  
  
 **private** Random **random**;  
 **private int timer**;  
  
 **public** Generator() {  
 **this**.**random** = **new** Random();  
 **this**.**timer** = 1;  
 }  
  
 **public** Package getPackage() {  
 **this**.**timer** -= 1;  
 **if** (**this**.**timer** == 0) {  
 **this**.**timer** = **this**.**random**.nextInt(7) + 3;  
 System.***out***.println(**this**.**timer**);  
 **return new** Package();  
 } **else** {  
 **return null**;  
 }  
 }  
}

Class Package

**public class** Package {  
  
 **private int time**;  
  
 **public** Package() {  
 **this**.**time** = 0;  
 }  
  
 **public void** tick() {  
 **this**.**time** += 1;  
 }  
  
 **public int** getTime() {  
 **return this**.**time**;  
 }  
}

Class Resource

**public class** Resource {  
 **private int called**;  
  
 **public** Resource() {  
 **this**.**called** = 0;  
 }  
 **public void** call()  
 {  
 **this**.**called** += 1;  
 }  
 **public int** getCalled()  
 {  
 **return this**.**called**;  
 }  
}