Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

КУРСОВАЯ РАБОТА

Моделирование ВС

по дисциплине «Архитектура программных систем»

Выплолнил студент
гр. 33534/5 Стойкоски Н.С.

Преподаватель Александрова О.В.

Санкт-Петербург
2018 г.

Оглвление

1. Постановка задачи

Целью курсовой работы является создание модели вычислительной системы (ВС) или ее части на некотором уровне детализации, описывающей и имитирующей ее структуру и функциональность.

Каждый реальный объект (реальная ВС) обладает бесконечной сложностью, множеством характеристик, внутренних и внешних связей. Модель есть приближенное описание объекта с целью получения требуемых результатов с определенной точностью и достоверностью.

При необходимости исследования поведенческих характеристик ВС в процессе исследования выгодно использовать не сам объект, а его модель. Степень приближения модели к описываемому объекту может быть различной и зависит от требований задачи.

Существуют различные типы моделей:

- Аналитические (математические) модели
- Аналоговые модели
- Физические модели
- Имитационные модели

Последний тип моделей является предметом нашего изучения.

Одним из подходов к построению имитационной модели является построение ее в виде системы массового обслуживания (СМО), с характерной для СМО терминологией: источник, буфер, прибор, диспетчер, заявка (требование).

Существуют два подхода к построению моделирующего алгоритма:

1.1. Принцип Δt – универсальный метод построения моделирующего алгоритма, когда состояние объекта проверяется через фиксированный интервал модельного времени. Суть его заключается в следующем: в каждый момент времени $t_i = t_{i-1} + \Delta t_{i-1}$ получают

приближенные значения характеристик исследуемого объекта. Δt можно получить детерминированным способом.

Основной критерий выбора Δt — он должен быть настолько мал, чтобы не пропустить событие в моделируемой системе, которое должно быть учтено при выбранной детальности моделирования. Метод неэффективен, т.к. постоянно проверяет состояние объектов моделирования, не изменяющихся при этом, особенно при малых Δt.

1.2. Принцип особых состояний.

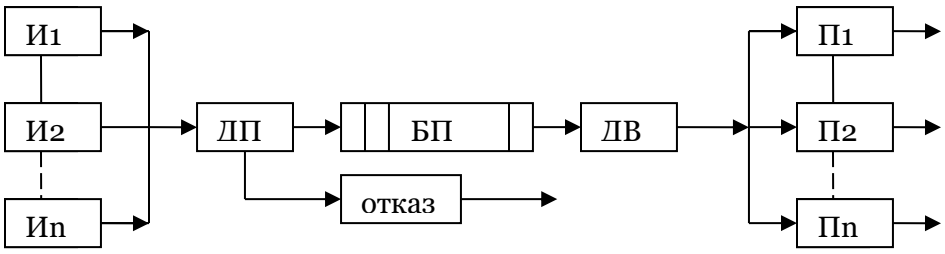
При исследовании реальной системы интервалы, в которых состояние ее не меняется, не представляют интереса. Имеют значение только переходы системы из одного состояния в другое в некоторые моменты времени. Эти переходы определяются особыми состояниями или событиями.

Рассмотрим некоторые типы особых событий, которые изменяют состояние системы:

- Поступление заявки в СМО (момент генерации заявки источником).
- Освобождение прибора (готовность прибора взять заявку на обслуживание).
- Окончание процесса моделирования.

Использование принципа особых событий для построения имитационной модели наиболее эффективно. В настоящей курсовой работе предлагается использовать именно этом принцип.

2. Формализованная схема и описание СМО.



Здесь **Иi (i= 1..n)** – источник заявок, который генерирует заявки, а все вместе n источников создают входной поток заявок в систему.

Каждая заявка приходит в СМО со своими характеристиками. Это $T_{вх}$ — время генерации заявки (время поступления её в СМО) и номер заявки составленный из номера источника, сгенерировавшего заявку, и порядкового номера заявки от этого источника. Например, (2.3) – третья заявка от второго источника.

П — приборы, которые обслуживают заявки и создают выходной поток заявок после обслуживания.

БП — буферная память (место для хранения очереди заявок).

В общей памяти хранятся заявки от различных источников. Порядок их записи в БП определяется только дисциплиной буферизации.

ДП — диспетчер постановки заявок.

ДВ — диспетчер выбора заявок.

Вариант №7

7.	ИБ	ИЗ2	ПЗ1	Д10З2	Д10О4	Д2П1	Д2Б4	ОР1	ОД1

1. *Тип источников*: бесконечный источник заявок с равномерным законом распределения

2. *Тип приборов*: прибор с экспоненциальныйм законом распределения времени обслуживания

3. *Дисциплина постановки в буфер*: в порядке поступления.

4. *Дисциплина отказа*: самая старая заявка в буфере идёт в отказ.

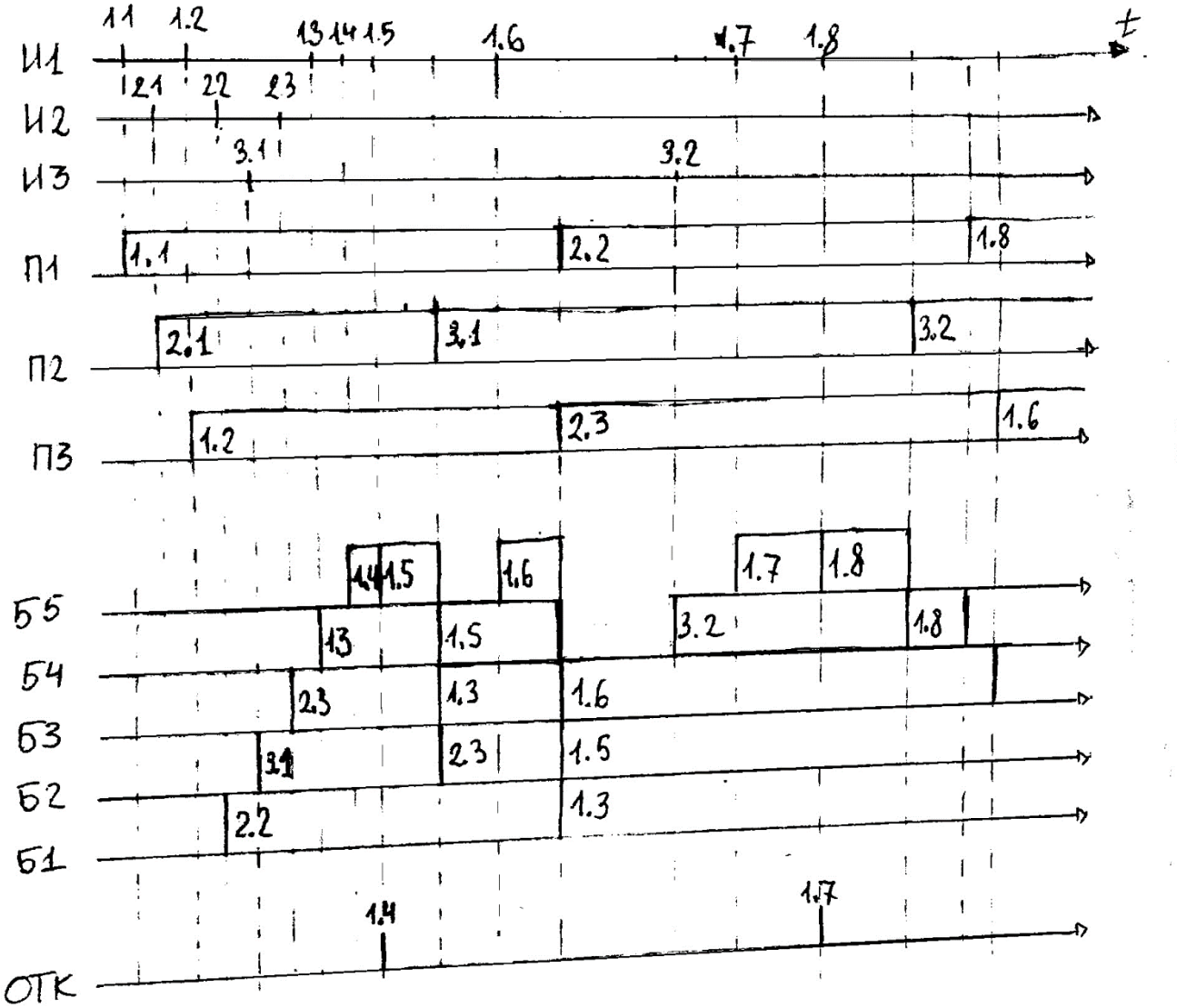
5. *Дисциплина выбора заявки из буфера*: приоритет по номеру источника, по одной заявке. При одинаковых приоритетов выбирается последняя поступившая.

6. *Дисциплина выбора прибора*: выбор прибора по кольцу

7. *Отображение результатов*: таблица результирующих значений параметров.

8. *Отображение динамики функционирования модели*: календарь событий, буфер и текущее состояние.

3. Временная диаграмма функционирования системы

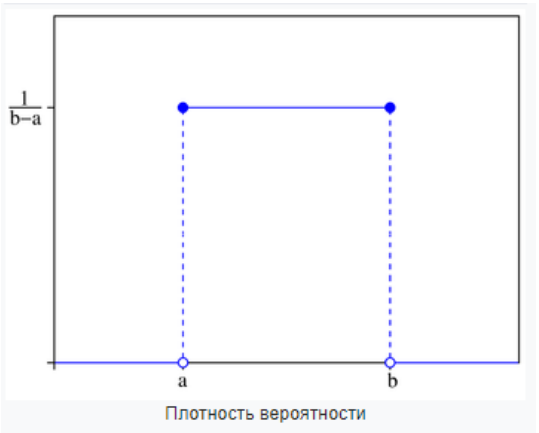


4. Вывод законов распределения

4.1 Равномерный закон распределения

Плотность равномерного распределения:

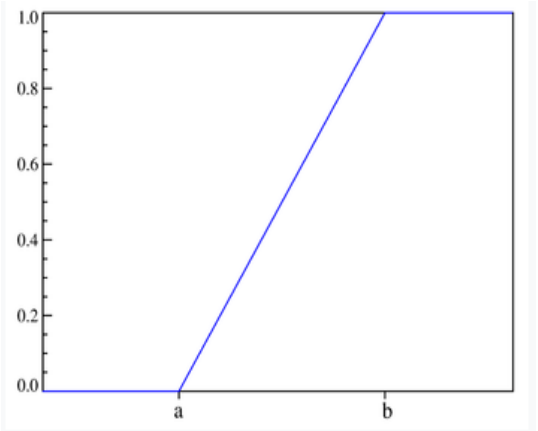
$$f_X(x) = \begin{cases} \frac{1}{b-a}, & x \in [a;b] \\ 0, & x \notin [a;\ b] \end{cases}$$



Плотность вероятности

Функция распределения:

$$F_X(x) = \int f_X(x)dx = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \le x < b \\ 1, & x \ge b \end{cases}$$



Функция распределения

$$F(x) = \frac{x-a}{b-a}$$

$$x - a = (b-a)F(x)$$

$$x = a + (b-a)F(x)$$

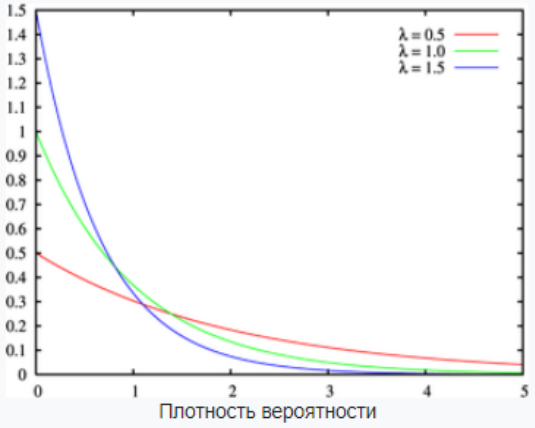
Программная реализация:

```
std::mt19937 rng(std::random_device{}());
std::uniform_real_distribution<double> dist(0, 1);
double TAY = TAY1 + (TAY2-TAY1)*dist(rng);
```

4.2 Экспоненциальный закон распределения

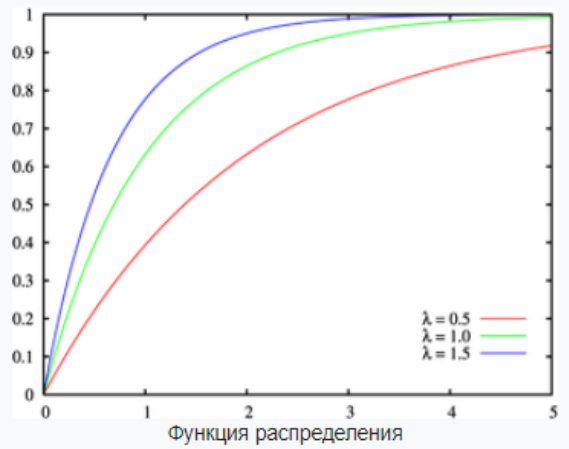
Плотность экспоненциального распределения:

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



Функция распределения:

$$F_X(x) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



$$F(x) = 1 - e^{-\lambda x}$$

$$1 - F(x) = e^{-\lambda x}$$

$$\ln\big(1 - F(x)\big) = -\lambda x$$

$$x = -\frac{1}{\lambda}\ln(1 - F(x))$$

Программная реализация:

```
std::mt19937 rng(std::random_device{}());

std::uniform_real_distribution<double> dist(0, 1);

double TAY = (-1/LAMBDA)*log(dist(rng));
```

5. Техническая система

Техническая система	Система сейсмоактивности
Источники	Источниками являются датчики сейсмической активности, которые отсылают данные на обработку в виде сетевого пакета. Датчики посылают информацию на обработку в виде сетевого пакета размером 64 Кбайт. В целях снижения нагрузки на устройства обработки данных информация посылается лишь при значительном отклонении измеряемых величин. Количество датчиков может быть от 5 до 20. Желательно получать и обрабатывать информацию с максимального числа датчиков.
Приборы	Приборами являются ЭВМ, которые получают и обрабатывают информацию с датчиков. Полученные данные передаются на устройство отображения, а также записываются в базу данных для статистического анализа.
Буфер	Буфером является буфер сетевого коммутатора, который в базовой конфигурации составляет 320 Кбайт (5 заявок), но может быть наращен до 4 Мбайт (64 заявки) .
Дисциплина постановки в буфер	Заявка встанет в очередь на первое от начала свободное место.
Дисциплина выбора из буфера	Приоритет заявки определяется номером источника, её сгенерировавшего. Из буфера выбирается заявка, приоритет которой выше остальных. Такие заявки приходят от сейсмодатчиков, которые установлены в

	наиболее подверженных землетрясениям зонах и их необходимо обрабатывать в первую очередь.
Дисциплина отказа	Самая старая заявка в буфере идёт в отказ.
Дисциплина постановки на обслуживание	Выбор прибора по кольцу.

5.1 Ограничения и требуемые характеристики

Вероятность отказа должна составлять не более 10%.

Загрузка приборов более 90%.

Время пребывания заявки в системе не ограничено, т.к. в зависимости от присланных данных, заявка может обрабатываться длительное время для получения верного результата.

Компоненты системы:

Количество датчиков	5 - 20
Размер заявки	64 Кб
Размер буфера	320Кб - 64 Мб
Количество приборов	1 - 15
Скорость работы сейсмодатчиков	Равномерное распределение с a= 0.4 , b= 0.6
Скорость работы приборов	Экспоненциальное распределение с $\lambda = 2.5$

6. Модульная структура

Разработка проводилась в среде Qt Creator на языке C++ с использованием графической библиотеки Qt.

Приложение является объектно-ориентированным и содержит следующий набор классов:

1. Request – содержит описание заявки, реализует методы для получения этих описаний.
2. Istocnik – реализует методы генерации заявок.
3. Buffer – реализует методы проверки свободного места в буфере, добавления заявки в буфер, выбора заявки из буфера.
4. Pribor – реализует метод обслуживания заявки.
5. Modulator – устанавливает связь между параметрами и систему. Модулирует данную систему с использованием метода особых состояний. Собирает статистику.
6. MainWindow – реализует графический интерфейс.

6.1 Алгоритм определяющий количества реализаций

Существует количественная связь между количеством реализаций (количество заявок, проходящих через ВС), относительной точностью, достоверной вероятностью и случайной величиной p(A) — вероятностью некоторого события A. Эта связь выражается формулой

$$N = \frac{t_\alpha^2 (1 - p)}{p\delta^2}, \qquad (2)$$

Где p — вероятность отказа заявкам в обслуживании; $t_\alpha = 1{,}643$ для $\alpha = 0{,}9$; $\delta = 0{,}1$ — относительная точность.

Формула представляет собой уравнение с двумя неизвестными (N и p). Поскольку искомой величиной является N, то необходимо иметь представление о значении p. Для этого обычно производят приблизительную оценку («пристрелку»), назначая какое-либо значение N (например, N_0=100), с которым проводят процесс моделирования, т.е. через систему пропускают 100 заявок, получают на выходе программной модели рассчитанные выходные характеристики, в этом числе и p_0, которое подставляют в формулу (2) и получают N_1, с которым снова проводят процесс моделирования.

Полученное на этом этапе значение p_1 сравнивают со значением p_0. Если разница $|p_0 - p_1|$ меньше 10 % от значения p_0, то N=100 удовлетворяет заданной точности результатов. Если же $|p_0 - p_1| \geq 10\%$ от p_0, то процесс моделирования продолжается с новым N_2 и т. д. до достижения необходимой точности.

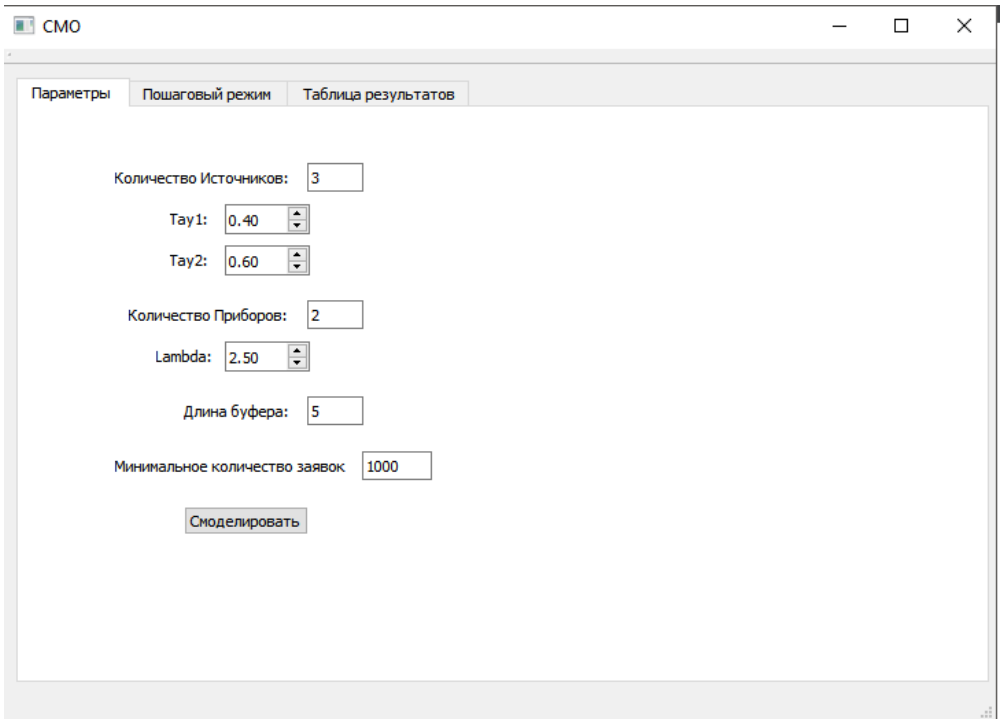
Весь этот итерационный процесс нахождения необходимого количества заявок нужно реализовать в автоматическом режиме. После окончания моделирования на экран выдается таблица результатов, полученных с заданной точностью.

6.2 Реализация алгоритма

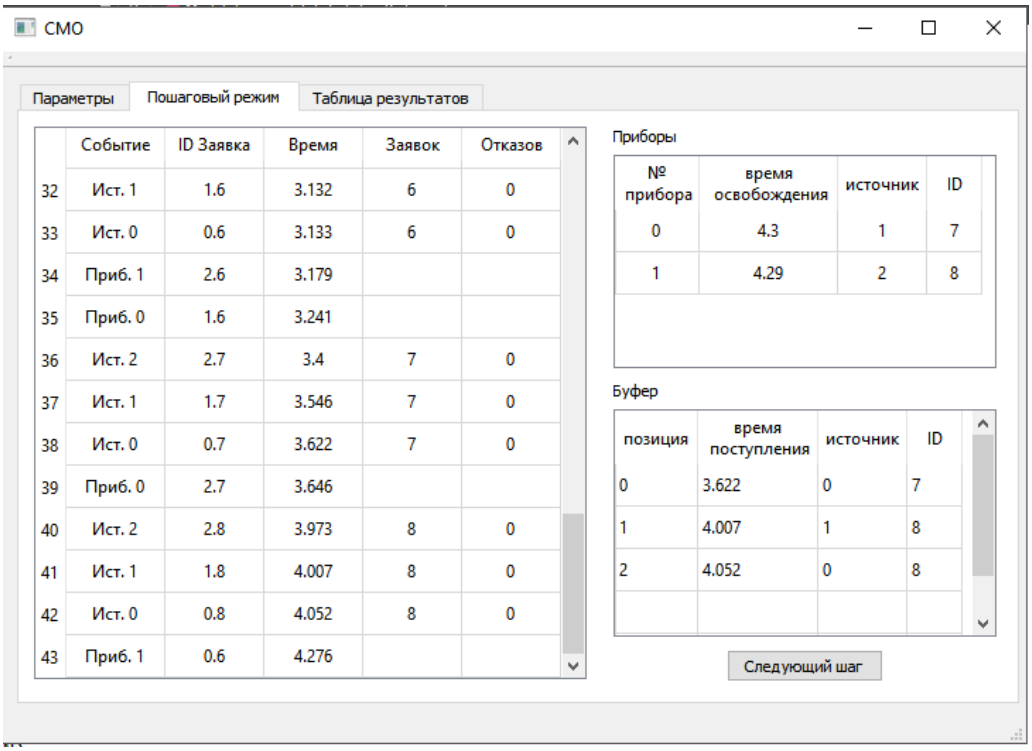
```
  std::vector<double> prevBOTK(n_ist); //массив где записываем вероятностей
отказа каждого источника в предыдущем особом состоянием.

  int N = KMIN;
  bool first_iteration = true;

  int TKOL = 0;    //общее число сгенерированных заявок,
                   //при поступ. заявка ++TKOL;
  while (true)
  {
    if(TKOL >= N)
    {
      const double ta2 = std::pow(1.643, 2);
      const double d2 = std::pow(0.1, 2);

      for (int i = 0; i < n_ist; i++)
        N = std::max(N, (int)((ta2 * (1.0 - ist[i].getBOTK())) /
(ist[i].getBOTK() * d2)));

      if (first_iteration)
      {
        first_iteration = false;

        if (N <= KMIN)
          return;

        for (int i = 0; i < n_ist; i++)
          prevBOTK[i] = ist[i].getBOTK();
      }
      else
      {
        bool diffLessThan_10p = true;
        for (int i = 0; i < n_ist; i++)
        {
          if (std::abs(prevBOTK[i] - ist[i].getBOTK()) >= 0.1 * prevBOTK[i])
          {
            diffLessThan_10p = false;
            break;
          }
        }

        if (diffLessThan_10p)
          return;

        for (int i = 0; i < n_ist; i++)
          prevBOTK[i] = ist[i].getBOTK();
      }
     }

    process_osob_sob();  //блок обработки особого события.

  }
```

7. Описание работы программы

Ввод параметров системы:



Отображение результатов в пошаговом режиме:

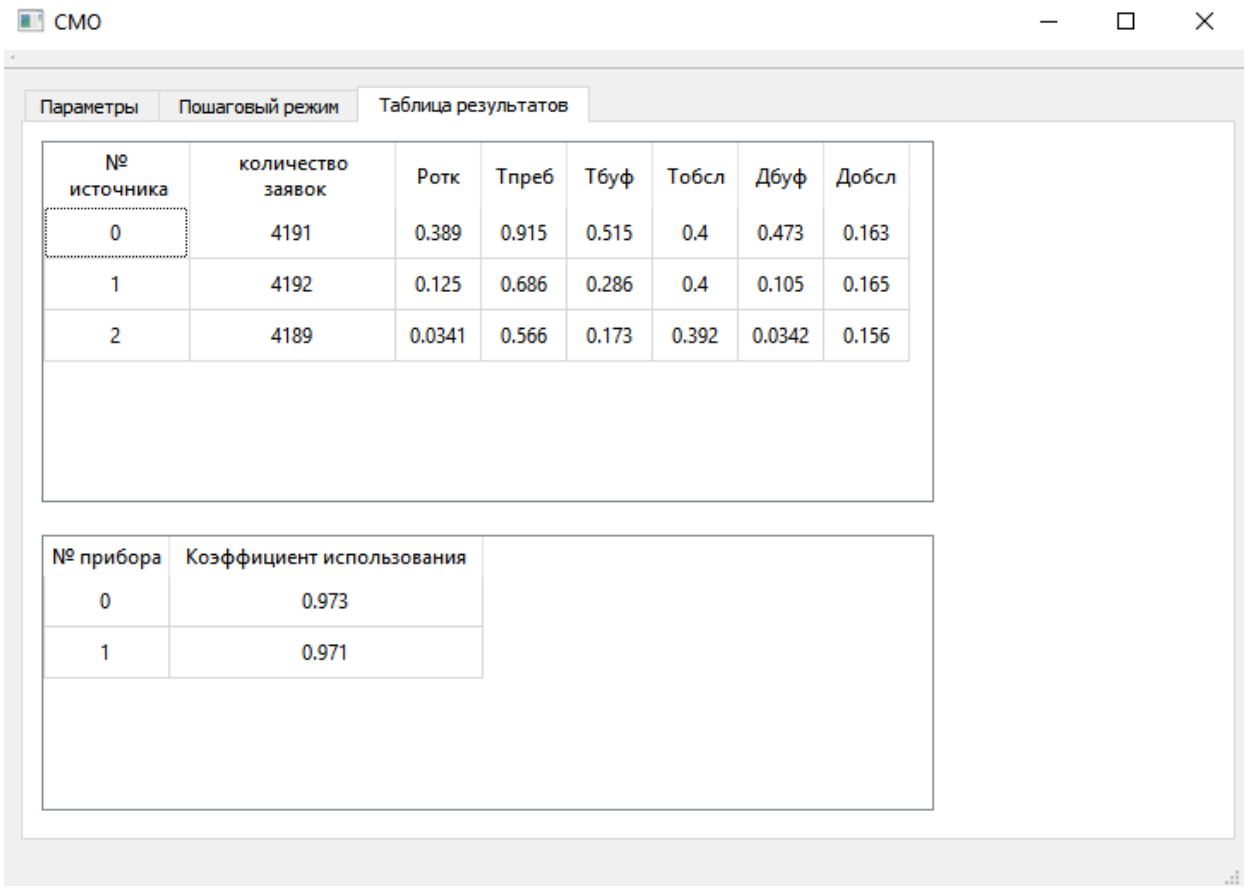


	Событие	ID Заявка	Время	Заявок	Отказов
32	Ист. 1	1.6	3.132	6	0
33	Ист. 0	0.6	3.133	6	0
34	Приб. 1	2.6	3.179		
35	Приб. 0	1.6	3.241		
36	Ист. 2	2.7	3.4	7	0
37	Ист. 1	1.7	3.546	7	0
38	Ист. 0	0.7	3.622	7	0
39	Приб. 0	2.7	3.646		
40	Ист. 2	2.8	3.973	8	0
41	Ист. 1	1.8	4.007	8	0
42	Ист. 0	0.8	4.052	8	0
43	Приб. 1	0.6	4.276		

№ прибора	время освобождения	источник	ID
0	4.3	1	7
1	4.29	2	8

позиция	время поступления	источник	ID
0	3.622	0	7
1	4.007	1	8
2	4.052	0	8

Отображение результатов в автоматическом режиме:



8. Анализ результатов, выводы и рекомендации по выбору конфигурации системы.

Т. к. целью моделирования является выбор конфигурации системы, требующей наименьшее количество ресурсов и обрабатывающей максимальный поток информации, то начнем с проверки конфигурации с макс. числом источников и минимальным числом приборов и мин. размером буфера.

Число ист.	Число приб.	тау1 ист.	тау2 ист..	лямбда Приб.	размер буф.	$p_{отк}$	$K_{исп}1$	$K_{исп}2$	$K_{исп}3$	$K_{исп}4$
10	1	0.4	0.6	3.0	10	0.8458	0.9972			
10	2	0.4	0.6	3.0	10	0.7033	0.9992	0.9992		
10	3	0.4	0.6	3.0	10	0.5421	0.9996	0.9996	0.9995	
10	4	0.4	0.6	3.0	10	0.3992	0.9998	0.9998	0.9998	0.9998

Из таблицы видно, что при данном числе источников, вероятность отказов слишком высока при любом числе приборов данного типа.

Попробуем достигнуть необходимой вероятности отказов заменой приборов на более производительные.

Число ист.	Число приб.	тау1 ист.	тау2 ист..	ламбда Приб.	размер буф.	$p_{отк}$	$K_{исп}1$	$K_{исп}2$	$K_{исп}3$
10	1	0.4	0.6	6.0	10	0.6963	0.9992		
10	2	0.4	0.6	6.0	10	0.4018	0.9999	0.9999	
10	3	0.4	0.6	6.0	10	0.1121	0.9867	0.9868	0.9868

Конфигурация из макс. числа самых производительных приборов также не справилась с нагрузкой. Будем уменьшать входной поток.

Число ист.	Число приб.	тау1 ист.	тау2 ист..	ламбда Приб.	размер буф.	$p_{отк}$	$K_{исп}1$	$K_{исп}2$	$K_{исп}3$
10	3	0.6	0.8	3.0	10	0.3726	0.9998	0.9998	0.9998
10	3	0.8	1.0	3.0	10	0.1920	0.9972	0.9971	0.9972
10	3	1.0	1.2	3.0	10	0.05053	0.9589	0.9591	0.959

Следовательно, оптимальной является следующая конфигурация:

Число ист.	Число приб.	тау1 ист.	тау2 ист..	ламбда Приб.	размер буф.	$p_{отк}$	$K_{исп}1$	$K_{исп}2$	$K_{исп}3$
10	3	1.0	1.2	3.0	10	0.05053	0.9589	0.9591	0.959

9. Вывод

В ходу курсовой работы была написана система массового обслуживания на языке C++ с использованием графической библиотеки Qt. С помощью данной программы была проанализирована реальная система и подобрана максимально выгодная комплектация данной системы