ВЕСТНИК ПЕРМСКОГО УНИВЕРСИТЕТА
2016 Математика. Механика. Информатика Вып. 1(32)

УДК 20.01.45

Модель СМО с неоднородными заявками и относительным приоритетом обслуживания

М. В. Шимановская, А. В. Юртаев
Пермская государственная сельскохозяйственная академия
Россия, 614990, Пермь, ул. Петропавловская, 23
mshim@mail.ru; (342) 2-103-413

Разработан имитационный алгоритм функционирования одноканальной СМО с относительным приоритетом обслуживания. При сравнении его с ранее опубликованными работами других авторов выделяется ряд преимуществ: отсутствие операторов безусловного перехода, и, как результат, лучшая наглядность и более легкая сопровождаемость кода, меньшая длина кода при одинаковой сложности алгоритма.

Ключевые слова: *системы массового обслуживания (СМО); относительный приоритет; моделирование; алгоритм.*

Введение

Одна из основных задач специалиста в области информационных систем и прикладной информатики – принимать обоснованные решения с целью повышения эффективности работы экономического объекта.

Одним из наилучших способов обоснования решения является построение и анализ имитационных моделей, описывающих некоторые процессы функционирования предприятия. Функционирование предприятий торговли, производства может быть описано с помощью моделей систем массового обслуживания.

Существуют две разновидности моделей СМО: аналитические и алгоритмические. Аналитические не учитывают полностью действие случайных факторов и поэтому могут использоваться как модели первого приближения. С помощью алгоритмических моделей исследуемый процесс может быть описан с любой степенью точности на уровне его понимания постановщиком задачи.

1. Концептуальная модель

Рассмотрим имитационный алгоритм функционирования СМО с заявками двух приоритетов. На вход системы поступают два потока заявок: высшего и низшего приоритета. Оба потока являются простейшими, т.е. время между соседними заявками имеет показательное распределение. Среднее время между соседними заявками для обоих приоритетов задано (Tzsr), время обслуживания заявок – величина случайная, имеющая равномерное распределение с известным средним временем обслуживания и диапазоном изменения времени обслуживания. Все перечисленные характеристики случайных величин – входные параметры модели. Также входными параметрами являются период функционирования системы и число случайных реализаций.

Правило обслуживания соответствует *относительному приоритету* и состоит в следующем: в канал поступают заявки, отличающиеся приоритетом обслуживания (высший и низший), причем заявки с высшим приоритетом обходят в очереди все заявки с низшим приоритетом. Таким образом, заявка с низшим приоритетом поступает на обслуживание, только если в очереди в данный мо-

мент времени нет заявок с более высоким приоритетом.

Выходными параметрами являются:

- среднее число поступивших заявок для обоих приоритетов;
- среднее число обслуженных заявок обоих приоритетов.

На основании этих показателей легко может быть введен критерий эффективности работы СМО, например средняя ожидаемая прибыль.

2. Моделирование

Применяя принципы объектно-ориентированного программирования, прежде всего выделяем из предметной области объект *заявка* (query), она характеризуется: временем поступления в СМО *Tz*, временем обслуживания *Tk*, целочисленным приоритетом *Priority. A* также введем булевскую переменную *Complete*, которая является показателем, обслужена ли заявка.

Ниже указан код для описания класса заявок на языке C#.

```
public class query
    {
       public double Tz { get; set; }
       public double Tk { get; set; }
       public int Priority { get; set; }
       public bool Complete { get; set; }

        public query(double Tz1, double Tk1, int
Pri, bool Comp)
        {
           Tz = Tz1;
           Tk = Tk1;
           Priority = Pri;
           Comp = Complete;
        }
    }
```

Формирование заявок обоих приоритетов происходит по одному алгоритму (см. рис. 1).

Оператор 1 – объявление и определение нового списка Tz – список объединенных заявок, также он обнуляет модельное время T.

Оператор 2 – начало циклического формирования заявок.

Оператор 3 – определяет случайный диапазон времени между двумя соседними заявками при условии, что среднее время между соседними заявками равно Tzsr. Здесь же изменяется модельное время T.

Оператор 4 – проверяет, закончилось ли модельное время, если закончилось, то циклический процесс (*оператор 2*) останавливается, иначе *оператор 5* добавляет новую заявку в список Tz.

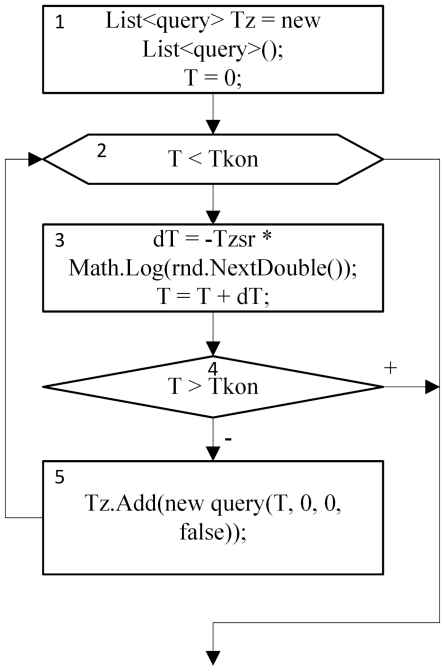


Рис. 1. *Алгоритм формирования потока заявок*

Таким образом, результат работы алгоритма – список заявок Tz (список объектов класса query). Поля заявок из этого списка имеют следующие значения:

время поступления заявки определено случайно, равно T;

время обслуживания и приоритет пока не определены, заполняются нулями;

признак обслуживания установлен в значение false.

С помощью описанного алгоритма будут сформированы два потока заявок (отдельно для каждого приоритета).

В дальнейшем, для моделирования процесса обслуживания, необходимо объединить эти потоки в один, при этом задав соот-

ветствующий приоритет (признак *Priority)* для каждой заявки из списка. Затем необходимо отсортировать общий список по возрастанию времени поступления заявок.

В результате получим очередь, в которой заявки будут расположены по времени поступления независимо от приоритета.

Описанные действия производим по алгоритму (рис. 2):

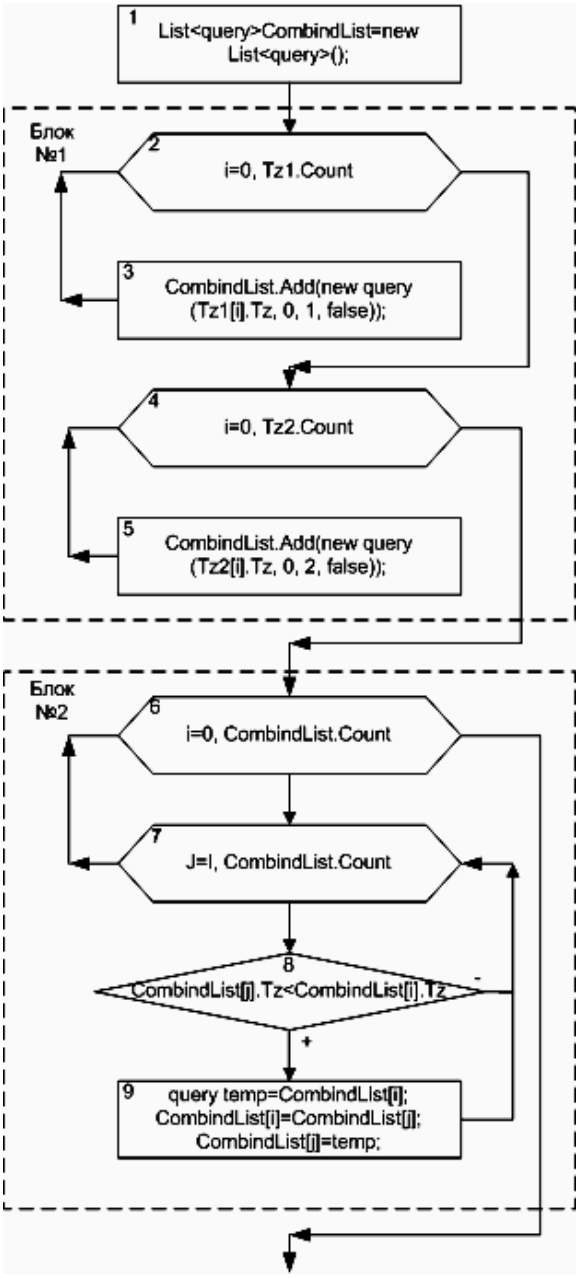


Рис. 2. *Алгоритм объединения потоков заявок*

Оператор 1 – объявление и определение нового списка CombindList – список объединенных заявок.

Блок №1 – объединение двух списков заявок Tz1 и Tz2.

Оператор 2 – начинает циклический перебор списка заявок первого приоритета Tz1.

Оператор 3 – добавляет в CombindList заявку из списка Tz1, устанавливая ей первый приоритет и статус "не обслужена".

Оператор 4 – начинает циклический перебор списка заявок второго приоритета Tz2.

Оператор 5 – добавляет в CombindList заявку из списка Tz2, устанавливая ей второй приоритет и статус "не обслужена".

Блок № 2 – сортировка объединенного списка заявок по времени их поступления.

Оператор 6 – начало циклического перебора всех заявок.

Оператор 7 – начало циклического перебора неотсортированных заявок.

Оператор 8 – если во время перебора заявок *оператором* 7 найдется заявка с временем поступления меньшим, чем у заявки из перебора *оператором 6,* то *оператор 9* меняет заявки местами и переходит на следующую итерацию.

Итак, входной поток заявок сформирован, и можно приступать к моделированию процесса обслуживания.

Процедура обслуживания заявок описана на рис. 3.

Оператор 1 – обнуляет локальную переменную Tjob (текущее время работы системы).

Блок № 1 – выполняется обслуживание самой первой заявки в списке заявок вне зависимости от ее приоритета.

Оператор 2 – проверяет, приоритет у заявки, если он первый, то *оператор 3* генерирует случайное время обслуживания заявки первого приоритета, в противном случае *оператор 4* генерирует случайное время обслуживания заявки второго приоритета.

Оператор 5 – устанавливает статус первой заявки "обслужена".

Оператор 6 – начало циклического перебора заявок, начиная со второй заявки.

Оператор 7 – фиксирует время освобождения канала обслуживания Tjob, присваивает ему время окончания обслуживания последней заявки.

Оператор 8 – проверяет, не превысило ли текущее время работы Tjob время функционирования системы Tkon, если время превышено, то *оператор 9* устанавливает статус предыдущей заявки как "не обслужена" и прекращает работу циклического перебора *(оператор 6)*.

Блок № 2 – установка времени поступления необработанных заявок на время завершения обслуживания предыдущей заявки (создание очереди из поступивших заявок).

Оператор 10 – начало циклического перебора необслуженных заявок.

Оператор 11 – если заявка поступила раньше текущего времени работы системы Tjob и статус заявки "не обслужена", то *оператор 12* присваивает время поступления заявки на текущее время работы системы Tjob.

Оператор 13 – сброс логических переменных на "ложь" (next1found – означает, найдена ли заявка первого приоритета, next2found – означает, найдена ли заявка второго приоритета).

Блок № 3 – поиск в очереди на обслуживание заявки первого приоритета.

Оператор 14 – начало циклического перебора не обслуженных заявок.

Оператор 15 – если заявка находится в очереди, она является заявкой первого приоритета и ее статус "не обслужена", то *оператор 16* переносит заявку в верх очереди, а *оператор 17* генерирует время обслуживания заявки, устанавливает ее статус как "обслужена", а также устанавливает переменную next1found как "истина" (заявка найдена и обслужена) и прерывает циклический перебор *(оператор 14)*.

Оператор 18 – если в результате работы *блока № 3* не найдено заявки первого приоритета, то включается *блок № 4,* в противном случае циклический перебор *(оператор 6)* переходит на следующую итерацию.

Блок № 4 – поиск в очереди на обслуживание заявки второго приоритета.

Оператор 19 – начало циклического перебора необслуженных заявок.

Оператор 20 – если заявка находится в очереди, она является заявкой второго приоритета и ее статус "не обслужена", то *оператор 21* переносит заявку в верх очереди, а *оператор 22* генерирует время обслуживания заявки, устанавливает ее статус как "обслужена", а также устанавливает переменную next2found как "истина" (заявка найдена и обслужена) и прерывает циклический перебор *(оператор 19)*.

Оператор 23 – если в результате Работы *блока № 4* не найдено заявки второго приоритета, то включается *блок № 5,* иначе циклический перебор *(оператор 6)* переходит на следующую итерацию.

Блок № 5 – обработка заявки, следующей в списке заявок (если нет очереди заявок, то система обслуживает следующую заявку вне зависимости от ее приоритета).

Оператор 24 – проверяет, какой приоритет у поступившей заявки, если первый, то *оператор 25* генерирует случайное время обслуживания заявки первого приоритета, в противном случае *оператор 26* генерирует случайное время обслуживание заявки второго приоритета.

Оператор 27 – устанавливает статус поступившей заявки "обслужена", циклический перебор *(оператор 6)* переходит на следующую итерацию.

На рис. 4 представлен обобщенный алгоритм функционирования СМО с отказами. Рассмотрим его подробнее.

Оператор 1 – осуществляет перевод исходных данных из символьной формы в числовую.

Оператор 2 – обнуление переменных SNobs1 и SNobs2 (количество обслуженных заявок первого и второго приоритета) и SNz1 и SNz2 (количество поступивших заявок первого и второго приоритета). Все эти переменные вводятся для подсчета соответствующих параметров для всех случайных реализаций.

Оператор 3 – начинает циклический перебор случайных реализаций.

Оператор 4 – обнуление локальных переменных Nobs1 и Nobs2 (число обслуженных заявок первого и второго приоритета в данной реализации).

Оператор 5 – обращается к автономной функции формирования потока заявок Tz1. В результате работы этой функции формируется список потока заявок с значениями времени поступления заявок.

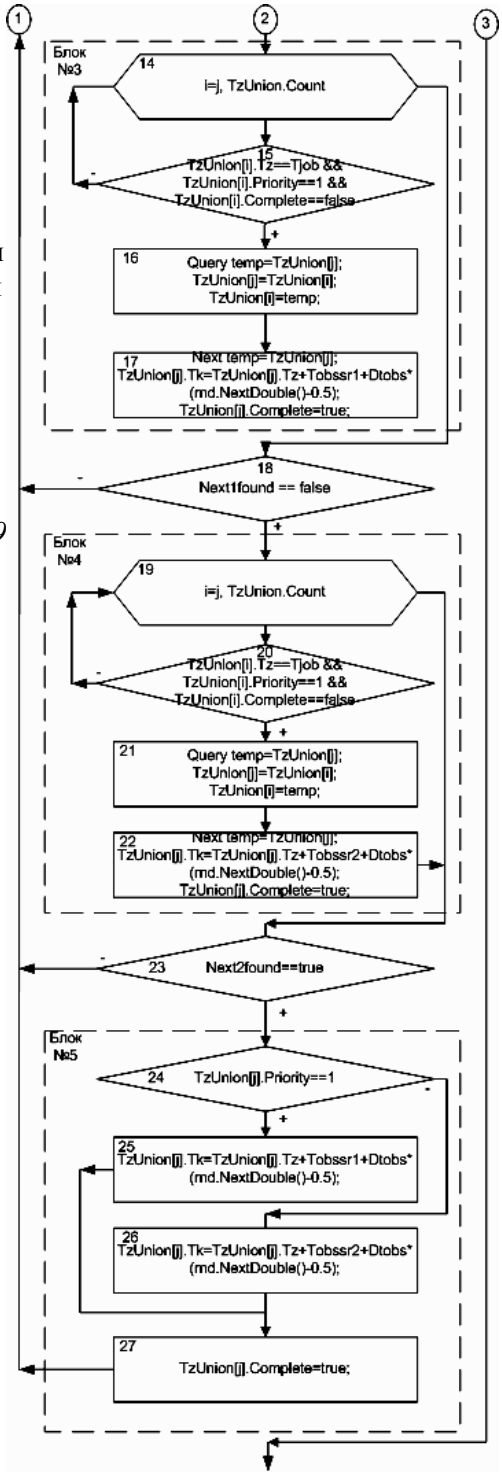
Оператор 6 – обращается к автономной функции формирования потока заявок Tz2. В

результате работы этой функции формируется список потока заявок с значениями времени поступления заявок.

Оператор 7 – обращается к автономной функции объединения и сортировки заявок. В результате работы этой функции два потока заявок Tz1 и Tz2 объединяются в поток заявок TzUnion и сортируются по времени поступления.

Оператор 8 – обращается к автономной функции обслуживания заявок. В результате работы этой функции заявки из списка дополняются информацией: время завершения обслуживания, статус (обслужена или не обслужена).

Блок № 1 – подсчет обслуженных заявок первого и второго приоритета.



Оператор 9 – начало циклического перебора заявок.

Оператор 10 – если заявка первого приоритета и она обслужена, то *оператор 11* увеличивает переменную Nobs1 на единицу, иначе осуществляется переход на *оператор 12*.

Оператор 12 – если заявка второго приоритета и она обслужена, то *оператор 13* увеличивает переменную Nobs2 на единицу, иначе – переход на следующую итерацию.

Оператор 14 – служит для расчета суммарного числа поступивших и обслуженных заявок для всех случайных реализациях.

Оператор 15 – выводит на экран средние числа поступивших и обслуженных заявок во всех случайных реализациях.

Рис. 3. *Алгоритм объединения потоков заявок*

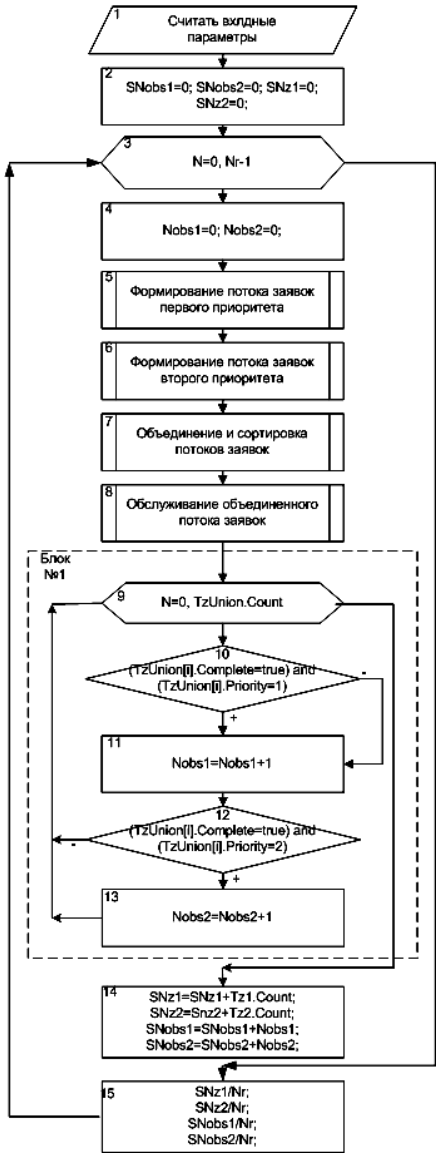


Рис. 4. *Обобщенный алгоритм функционирования СМО с относительным приоритетом обслуживания*

Результаты. Разработанный алгоритм реализован на языке C#. Проведены расчеты с комбинацией переменных и параметров, предложенные в работе [1].

Разработанный алгоритм используется при работе со студентами факультета Прикладной информатики Пермской ГСХА. При изучении темы "Имитационное моделирование" на лабораторных работах студентам предлагается реализовать алгоритм в выбранной ими среде программирования.

Выводы. Алгоритмическое моделирование СМО может быть успешно использовано для отработки у студентов различных специальностей навыков написания прикладных программ. А также выработки навыков решения задач моделирования типовых экономических процессов, которые необходимы для успешной экономической и управленческой деятельности.

Список литературы

1. *Варфоломеев В.И., Назаров С.В.* Алгоритмическое моделирование элементов экономических систем // Практикум: учеб. пособие. 2-е изд., доп. и перераб. / под ред. С.В. Назарова. М.: Финансы и статистика, 2004.
2. *Снетков Н.Н.* Имитационное моделирование экономических процессов: учеб.-практ. пособие. М.: Изд. центр ЕАОИ, 2008.
3. *Шимановская М.В.* Модель СМО с неоднородными заявками и абсолютным приоритетом обслуживания // Вестник Пермского университета. Математика. Механика. Информатика. 2013. Вып. 4(23). С. 103–106.

Model of a queuing system with heterogeneous customers and the absolute priority of service

M. V. Shimanovskaja, **A. V. Yurtaev**
Perm State Agricultural Academy, Russia, 614990, Perm, Petropavlovckaja st., 23
mshim@mail.ru; (342) 2-103-413

The simulation algorithm of functioning of single channel queuing system has been worked out. It has the relative priority of service. When comparing it with the previous published works of other authors, it is distinguished a number of advantages: the absense of *go to* operators and as a result, better visibility and more easier maintainbility of the code? Smaller codelength with the same complexity of the algorithm.

Key words: *queuing system; relative priority; modeling; algorithm.*

Рис. 2. *Алгоритм объединения потоков заявок.*