**Федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**Уфимский государственный авиационный технический университет**

**Кафедра вычислительной математики и кибернетики**

**Расчетно-графическая работа по курсу**

**«Компьютерное моделирование»**

Вариант 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Ф.И.О. | Дата | Оценка | Подпись |
| Выполнил студент группы ПРО-301в | Доронин С.Г. |  |  |  |
| Принял: | Файзрахманов Р.И. |  |  |  |

Уфа – 2014 г.

# Задание на расчетно-графическую работу

Мини-маркет с одним контролером– кассиром обслуживает покупателей, входящий поток которых подчиняется закону Пуассона с параметром 20 покупателей/час.

Провести моделирование описанного процесса и определить вероятность простоя контролера–кассира, среднюю длину очереди, среднее число покупателей в мини-маркете, среднее время ожидания обслуживания, среднее время пребывания покупателей в мини-маркете и дайте оценку его работы.

# Содержание

[Введение 4](#_Toc388020406)

[Описание модели СМО 4](#_Toc388020407)

[Алгоритм моделирования СМО 4](#_Toc388020408)

[Характеристики СМО, определенные по результатам моделирования 5](#_Toc388020409)

[Анализ полученных результатов и рекомендации по повышению эффективности функционирования СМО 5](#_Toc388020410)

[Текст программы 6](#_Toc388020411)

[Кассир 6](#_Toc388020412)

[Покупатель 6](#_Toc388020413)

[Генератор покупателей 7](#_Toc388020414)

[Мини-маркет 7](#_Toc388020415)

[Текст используемых источников 10](#_Toc388020416)

# Введение

За последние десятилетия в самых разных областях народного хозяйства возникла необходимость решения вероятностных задач, связанных с работой систем массового обслуживания. Примерами таких систем служат телефонные станции, ремонтные мастерские, торговые предприятия, билетные кассы и т.д. работа любой системы массового обслуживания состоит в обслуживании поступающего в нее потока требований (вызовы абонентов, приход покупателей в магазин, требования на выполнение работы в мастерской и т. д.).

Математическая дисциплина, изучающая модели реальных систем массового обслуживания, получила название теории массового обслуживания. Задача теории массового обслуживания- установить зависимость результирующих показателей работы системы массового обслуживания (вероятности того, что требование будет обслужено; математического ожидания числа обслуженных требований и т. д.) от входных показателей (количество приборов в системе, параметров входящего потока требований и т. д.) установить такие зависимости в формульном виде можно только для простых систем массового обслуживания. Изучение же реальных систем проводится путем имитации, или моделирования их работы на ЭВМ с привлечением метода статистических испытаний.

# Описание модели СМО

Модель данной задачи эмулирует работу 4 объектов:

* Покупатель – человек совершающий покупки в мини-маркете;
* Генератор покупателей – моделирует работу прихода покупателей;
* Контроллер-кассир – обслуживает покупателей;
* Мини-маркет – моделирует работу всего магазина, используя описанные выше объекты.

# Алгоритм моделирования СМО

Класс моделирующий работу мини-маркета начинает свою работу и запускает генератор покупателей в отдельном потоке, далее запускается поток моделирующий становления покупателей в очередь, также запускается поток, моделирующий обслуживание покупателей в очереди.

В потоке обслуживания покупателей, считается часть статистики (среднее время ожидания обслуживания, среднее время пребывания покупателей).

Запускается таймер подсчета статистики (вероятность простоя кассира, средняя длина очереди, среднее число покупателей), пересчет статистики осуществляется каждую секунду.

# Характеристики СМО, определенные по результатам моделирования

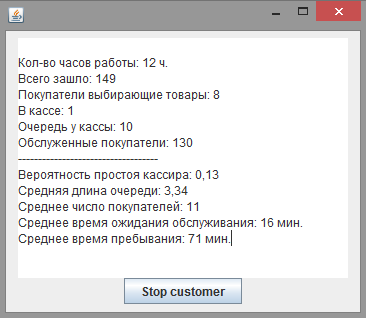


Рисунок . Моделирование 12 часов работы мини-маркета

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Отработано часов** | **Вероятность простоя кассира** | **Средняя длина очереди** | **Среднее число покупателей** | **Средняя время ожидания обслуживания** | **Среднее время пребывания** |
| 12 | 0.13 | 3.34 | 11 | 16 | 71 |
| 86 | 0.04 | 4.89 | 12 | 22 | 79 |
| 154 | 0.02 | 11.56 | 12 | 52 | 108 |
| 246 | 0.01 | 24.06 | 12 | 108 | 164 |

# Анализ полученных результатов и рекомендации по повышению эффективности функционирования СМО

Первые 12 часов магазин работает на приемлемом уровне, кассир почти не простаивает, средняя длина очереди чуть больше 3-х, среднее время ожидания обслуживания 16 минут. Можно уменьшить очередь и время обслуживания добавив еще одного кассира.

При работе магазина 86 часов увеличивается очередь до почти 5 человек и среднее время ожидания обслуживания становится равным 22 минутам.

При работе магазина 154 часа, он становится неэффективным, потому что очередь возрастает до 12 человек и среднее время ожидания обслуживания становится равным 52 минутам.

Тут также стоит рекомендовать добавить еще одного кассира и запустить моделирование заново.

# Текст программы

## Кассир

/\*\*

\* Кассир

\*/

**case** **class** Cashier() {

/\*\* Свободен или занят \*/

**private** **val** \_free **=** **new** AtomicBoolean(true)

/\*\* Интервал времени обслуживания \*/

**private** **val** serviceTime **=** Const.CashierServiceTime

/\*\* Свободен ли кассир \*/

**def** free: Boolean **=** \_free.get

/\*\* Обслужить покупателя \*/

**def** serviceCustomer(customer: Customer) **=** synchronized {

assert(free)

\_free.set(false)

Thread.sleep(serviceTime.get)

\_free.set(true)

}

/\*\* Количество обслуживаемых покупателей в данный момент \*/

**def** customerServiceNowCount **=** **if** (free) 0 **else** 1

}

## Покупатель

/\*\*

\* Покупатель

\*/

**case** **class** Customer(name: String) {

/\*\* Интервал времени покупок покупателя \*/

**private** **val** ShoppingInterval **=** Const.CustomerShoppingInterval.get

/\*\* Все ли купил покупатель (идет в очередь) \*/

**private** **var** \_allBought **=** **false**

/\*\* Время становления в очередь \*/

**private** **var** \_startWaitTime: Date **=** **null**

/\*\* Время ожидания обслуживания (мс.) \*/

**private** **var** \_waitMillis **=** 0L

/\*\* Время захода в магазин \*/

**private** **var** \_visitTime: Date **=** **null**

// Таймер покупок покупателя

**new** Timer().schedule(new TimerTask {

**override** **def** run() {

\_allBought **=** **true**

}

}, ShoppingInterval)

/\*\* Все ли купил покупатель (идет в очередь) \*/

**def** allBought **=** \_allBought

/\*\* Покупатель встал в очередь ожидания обслуживания \*/

**def** startWait() **=** \_startWaitTime **=** **new** Date

/\*\* Подсчитать время ожидания обслуживания \*/

**def** calcWait() **=** {

assert(\_startWaitTime **!=** null)

\_waitMillis **=** **new** Date().getTime **-** \_startWaitTime.getTime

waitMillis

}

/\*\* Время ожидания обслуживания (мс.) \*/

**def** waitMillis **=** {

assert(\_startWaitTime **!=** null)

\_waitMillis

}

/\*\* Установить время посещения магазина текущим \*/

**def** setVisitTime() **=** \_visitTime **=** **new** Date

/\*\* Время пребывания (мс.) \*/

**def** stayTime **=** **new** Date().getTime **-** \_visitTime.getTime

}

## Генератор покупателей

/\*\*

\* Генератор покупателей

\*/

**class** CustomerGenerator(action: Customer **=>** Unit) {

**private** **val** logger **=** **new** Logger(true, true, true)

**private** **val** enable **=** **new** AtomicBoolean(false)

/\*\* Перерыв между заходами покупателей \*/ //TODO переделать под Пуассона

**private** **val** generationTimeout **=** Const.CustomerGenerationTimeout

/\*\* Поток генерации покупателей \*/

**private** **val** thread **=** **new** Thread(new Runnable {

**override** **def** run() {

**var** customerIndex **=** 0

**while** (enable.get) {

customerIndex **+=** 1

Thread.sleep(generationTimeout.get)

**val** customer **=** Customer("Customer-" **+** customerIndex)

logger.debug("CustomerGenerator generate: " **+** customer)

action(customer)

}

}

})

**def** start() {

logger.debug("CustomerGenerator start")

enable.set(true)

thread.start()

}

**def** stop() {

logger.debug("CustomerGenerator stop")

enable.set(false)

}

}

## Мини-маркет

/\*\*

\* Мини-маркет

\*/

**class** MiniMarket {

**private** **val** logger **=** **new** Logger(true, true, true)

/\*\* Очередь покупателей \*/

**private** **val** queue **=** **new** MultiThreadQueue[Customer]

/\*\* Работает-ли магазин \*/

**private** **val** enable **=** **new** AtomicBoolean(false)

/\*\* Кассир \*/

**private** **val** cashier **=** Cashier()

/\*\* Не обслуженные покупатели \*/

**private** **val** notServiceCustomerList **=** ListBuffer[Customer]()

/\*\* Обслуженные покупатели \*/

**private** **val** serviceCustomerList **=** ListBuffer[Customer]()

/\*\* Счетчик зашедших покупателей \*/

**private** **val** customerCount **=** **new** AtomicInteger(0)

/\*\* Генератор покупателей \*/

**private** **val** customerGenerator **=** **new** CustomerGenerator({ customer **=>**

customerCount.getAndAdd(1)

// Установка времени посещения

customer.setVisitTime()

synchronized(notServiceCustomerList **+=** customer)

})

/\*\* Статистика \*/

**private** **val** stat **=** MiniMarketStat(new Stat, **new** Stat, **new** Stat, **new** Stat, **new** Stat)

/\*\* Время начала работы \*/

**private** **var** startTime: Date **=** **null**

/\*\* Поток становления покупателей в очередь \*/

**private** **val** queueThread **=** **new** Thread(new Runnable {

**override** **def** run() {

while(enable.get) {

synchronized {

notServiceCustomerList.filter(\_.allBought).foreach { customer **=>**

// Старт подсчета времени ожидания

customer.startWait()

queue **+=** customer

notServiceCustomerList **-=** customer

}

}

Thread.sleep(Const.ThreadSleepMilis)

}

}

})

/\*\* Поток обслуживания очереди покупателей \*/

**private** **val** serviceThread **=** **new** Thread(new Runnable {

**override** **def** run() {

while(enable.get) {

**if** (cashier.free **&&** queue.nonEmpty) {

**val** customer **=** queue.dequeue()

// Подсчет среденго времени ожидания обслуживания

stat.waitServiceTime.newElementAndAdd(customer.calcWait())

cashier.serviceCustomer(customer)

// Подсчет среденго времени пребывания покупателей

stat.stayTime.newElementAndAdd(customer.stayTime)

serviceCustomerList **+=** customer

}

}

}

})

/\*\* Таймер подсчета статистики \*/

**private** **val** statTimer **=** **new** Timer(1000, **new** ActionListener {

**override** **def** actionPerformed(e: ActionEvent) **=** synchronized {

stat.queueLength.newElementAndAdd(queue.size)

stat.cashierUpTime.newElementAndAdd(cashier.customerServiceNowCount)

stat.customerCount.newElementAndAdd(cashier.customerServiceNowCount **+** notServiceCustomerList.size)

}

})

/\*\* Время работы (мс.) \*/

**def** workTime **=** **new** Date().getTime **-** startTime.getTime

/\*\* Запуск работы магазина \*/

**def** start() {

logger.debug("MiniMarket start")

startTime **=** **new** Date()

enable.set(true)

// Запуск подсчета статистики

statTimer.start()

// Запуск обслуживания

serviceThread.start()

// Запуск становления покупателей в очередь

queueThread.start()

// Запуск генератора покупателей

customerGenerator.start()

}

/\*\* Остановка работы магазина \*/

**def** stop() {

logger.debug("MiniMarket stop")

enable.set(false)

customerGenerator.stop()

statTimer.stop()

}

/\*\* Хранилище информации о магазине \*/

**case** **class** MiniMarketInfo(customerCount: Int, serviceCustomerCount: Int, notServiceCustomerCount: Int, customerServiceNowCount: Int, queueLength: Int,

pCashierDownTime: Float, avgQueueLength: Float, avgCustomerCount: Float, avgWaitServiceTime: Float, avgStayTime: Float)

/\*\* Информация о работе магазина \*/

**def** getInfo **=** synchronized {

MiniMarketInfo(customerCount.get, serviceCustomerList.size, notServiceCustomerList.size, cashier.customerServiceNowCount, queue.size,

1**-**stat.cashierUpTime.avg, stat.queueLength.avg, stat.customerCount.avg, stat.waitServiceTime.avg, stat.stayTime.avg)

}

/\*\* Остановка генерирования покупателей \*/

**def** stopCustomerGenerator() **=** customerGenerator.stop()

}

# Текст используемых источников

1 Емельянов А.А. Имитационное моделирование экономических процессов

[Текст]: Учеб. пособие для вузов/ А.А. Емельянов, Е.А. Власова, Р.В. Дума. -М. : Финансы и статистика, 2002. - 368с.

2 Бусленко, Н.П. Моделирование сложных систем[Текст]/ Н.П. Бусленко.-М. : Наука, 1978. - 399с.

3 Советов Б.Я. Моделирование систем[Текст]: Учеб. для вузов/ Б.Я. Сове-тов, С.А. Яковлев. -М. : Высш. школа, 1985. - 271 с.