ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2017 Управление, вычислительная техника и информатика № 40

УДК 004.42
DOI: 10.17223/19988605/40/8

М.А. Сонькин, А.Н. Моисеев, Д.М. Сонькин, Д.А. Буртовая

ОБЪЕКТНАЯ МОДЕЛЬ ПРИЛОЖЕНИЯ ДЛЯ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ЦИКЛИЧЕСКИХ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

Представлена разработка объектной модели программной системы, предназначенной для имитационного моделирования циклических систем массового обслуживания. Приложение построено на основе дискретно-событийного подхода и включает реализацию различных механизмов группового подключения и переключения обслуживающих приборов между входящими потоками заявок.

Ключевые слова: циклическая система массового обслуживания; имитационное моделирование; объектно-ориентированное проектирование.

Системы массового обслуживания (СМО) [1, 2] являются стохастическими моделями реальных систем. Циклические СМО – это класс систем массового обслуживания, в которых обслуживающие приборы в определенном (циклическом) порядке подключаются к различным накопителям для обслуживания заявок, находящихся в этих накопителях [3, 4]. Заявки попадают в накопители из нескольких входящих в систему потоков заявок. Аналитическим исследованиям циклических систем обслуживания посвящено достаточно много научных трудов (см., например, [5]), но несмотря на это, по вполне понятным причинам во многих случаях не удается получить какие-либо аналитические результаты. Тогда применяют другие методы исследования, среди которых особое место занимает имитационное моделирование [6, 7].

Имитационный подход позволяет воспроизвести поведение системы и таким образом достичь требуемых результатов практически для любой конфигурации исследуемой системы. Конечно, имитационное моделирование имеет и свои недостатки – это в первую очередь числовой, возможно, качественный, но не аналитический характер результатов, что позволяет делать подтвержденные выводы только для модели, определенной с точностью до числовых значений, но не для класса систем. Также выполнение имитационного моделирования требует достаточно долгого времени для проведения комплексного исследования. Однако, несмотря на указанные недостатки, имитационный подход часто используется для решения конкретных задач в тех случаях, когда аналитические исследования невозможны либо возможны при больших упрощениях исходной модели.

Для выполнения имитационного моделирования применяют специализированные программные средства, которые можно разделить на два типа: с полным контролем внутренней структуры со стороны аналитика (типа «белый ящик») и без такового (типа «черный ящик»). Системы первого типа наиболее удобно реализовать с помощью различных готовых программных продуктов, специально предназначенных для имитационного моделирования, например, с помощью систем GPSS [8, 9], MatLab [10], AnyLogic [11] и других программных продуктов. Иногда аналитик использует стандартные языки программирования (например, Fortran или C++) для создания такой программы. В любом из этих случаев от исследователя требуются владение на хорошем уровне навыками программирования и / или достаточно глубокие знания специального языка (GPSS) или устройства соответствующей системы (MatLab).

Системы второго типа представляют собой закрытые приложения, в которых аналитику достаточно только выбрать конфигурацию из предложенного списка и задать параметры, не вдаваясь в подробности устройства программной системы и не написав ни строчки кода на каком-либо языке. В данном случае от пользователя приложения не требуется знания основ программирования и даже глубокого знания внутреннего устройства самого процесса имитационного моделирования.

Каждый тип систем моделирования имеет свои положительные и отрицательные стороны. В настоящей работе представлено решение задачи разработки приложения закрытого типа для имитационного моделирования достаточно широкого класса циклических систем обслуживания.

1. Общее описание модели циклической системы массового обслуживания

Рассматривается класс циклических систем массового обслуживания следующего вида (рис. 1). На вход системы поступает $K \geq 1$ входящих потоков $A_1, \ldots, A_K$, каждый из которых описывается заданной для него моделью случайного потока событий с определенными параметрами. Все заявки k-го входящего потока ($k = \overline{1, K}$) поступают в k-й накопитель (буфер), который может иметь заданный максимальный размер или может быть не ограничен в приеме заявок. Также имеется некоторое число $N \geq 1$ обслуживающих приборов (серверов) $B_1, \ldots, B_N$, которые подключаются к накопителям, забирают из них заявки в порядке, соответствующем некоторой дисциплине (например, FIFO – first in first out), и обслуживают эти заявки в соответствии с некоторым заданным для каждого прибора законом. После этого заявки покидают систему. Функция распределения времени обслуживания может определяться для каждого прибора независимо от накопителя (входящего потока), из которого взята заявка, но также для каждого прибора может быть задан индивидуальный набор из K функций распределения, определяющих продолжительность обслуживания заявок, поступивших из разных входящих потоков. Таким образом, при K входящих потоках (накопителях) и N обслуживающих приборах можно задать $N \times K$ функций распределения $B_{nk}(x)$, $n = \overline{1, N}$, $k = \overline{1, K}$, определяющих продолжительность обслуживания разными приборами заявок из разных входящих потоков.

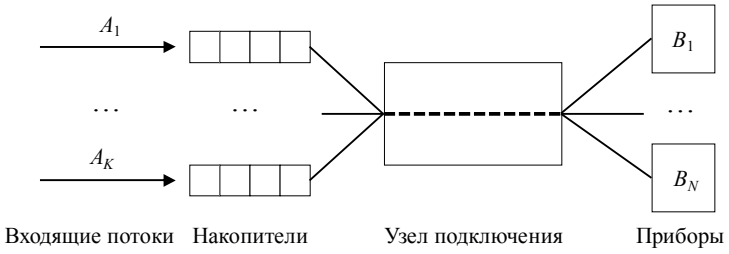

Рис. 1. Модель циклической системы

Порядок, способ и длительность подключения приборов к накопителям определяется заданными параметрами, которые в дальнейшем будем называть параметрами подключения. В случае, когда количество серверов $N > 1$, возможны различные варианты их «группового» поведения. В качестве основных групповых дисциплин выделим три:

1. *Жесткая группа.* Все обслуживающие приборы системы образуют одну группу обслуживания: все приборы в один момент времени работают только с одним накопителем, переключение между накопителями и пребывание в состоянии переключения (см. далее) для всех приборов производится синхронно.

2. *Одиночные приборы*. Обслуживающие приборы подключаются к накопителям независимо друг от друга, причем с одним накопителем в один момент времени может работать только один прибор. В случае, если по окончании сеанса подключения прибора к накопителю следующий накопитель уже обслуживается другим сервером, то прибор ищет следующий по циклическому порядку свободный (не находящийся в сеансе подключения с другим сервером) накопитель и подключается к нему. Эта дисциплина подключения доступна только при условии, что количество серверов не превышает количество накопителей ($N \leq K$).

3. *Гибкие группы*. Обслуживающие приборы подключаются к накопителям независимо друг от друга, в один момент времени к накопителю может быть подключено любое число приборов. По окончании сеанса подключения прибор подключается к следующему накопителю независимо от того, обслуживается он другими серверами или нет.

Кроме дисциплины подключения групп важными параметрами подключения являются параметры, определяющие продолжительность одного сеанса подключения сервера или группы серверов к накопителю. В данной работе будем называть их дисциплиной подключения и рассмотрим два типа:

1. *С разделением времени*. Продолжительность каждого подключения определяется заданной функцией распределения длительности сеанса. По окончании этого периода обслуживание всех заявок, производившееся в рамках данного подключения, прерывается, а сами заявки возвращаются в накопитель (в начало очереди, в порядке времени их поступления в систему). Продолжительность сеанса подключения может определяться одной функцией распределения для всех накопителей либо задаваться индивидуально для каждого накопителя.

2. *До полного исчерпания*. Подключение сохраняется до тех пор, пока не будет закончено обслуживание последней заявки в накопителе. При этом в случае жесткой группы все незадействованные приборы простаивают, пока последний из них не закончит обслуживание, но во время простоя могут принимать и обслуживать заявки, поступающие из текущего входящего потока. В случае гибких групп, если по окончании обслуживания на одном из приборов накопитель оказывается пуст, то сеанс подключения для данного прибора заканчивается и он переключается на другой накопитель.

Когда прибор или группа приборов завершают обслуживание одного накопителя и должны переключиться к следующему, то параметрами задачи может быть задано, что это переключение требует некоторого времени. В случае применения дисциплины подключения «до полного исчерпания» это время должно быть обязательно больше нуля, так как в противном случае, если все накопители оказались временно пусты, модель перейдет в состояние «зацикливания», когда приборы мгновенно переключаются от очереди к очереди, а модельное время не изменяется. Для дисциплины с разделением времени этот параметр опционален. Если параметрами задачи указано, что переключение требует времени, то задается функция распределения, определяющая длительность состояния переключения, в которое прибор или группа приборов переходят по окончании сеанса подключения к одному накопителю. В этом состоянии приборы не выполняют никаких действий. По окончании времени переключения приборы подключаются к следующему накопителю. Функция распределения длительности времени переключения может определяться одной функцией распределения для всех накопителей либо задаваться индивидуально для каждого накопителя.

2. Объектная модель циклической системы обслуживания

2.1. Основные объекты циклической системы

Для имитационного моделирования стохастических систем наиболее подходящим методом моделирования является дискретно-событийный подход [12, 13], основная идея которого заключается в определении событий, которые должны произойти в системе, и сдвиге модельного времени по моментам наступления этих событий. В настоящей работе в качестве платформы для реализации приложения имитационного моделирования циклических систем будут использованы библиотеки программного комплекса ODIS [14–16], предназначенного для имитационного моделирования сетей массового обслуживания. В этих библиотеках уже имеется базовый класс SimulationModel, который реализует основной цикл дискретно-событийного моделирования систем массового обслуживания. Исполнение цикла моделирования базируется на взаимодействии объектов событий, заявок и элементов системы.

Объектная модель разрабатываемого приложения имитационного моделирования циклических СМО представлена на рис. 2, 3. Реализация механизма сеансов подключения и описание основных сценариев имитационного моделирования представлены в следующих подразделах.

Классы Element, Source, PassiveBuffer реализованы каркасом ODIS и предназначены для моделирования элементов СМО общего вида. Классы InputSource, Server и ConnectionManager реализуют объекты, специфичные для циклических СМО.

Класс Element представляет собой абстрактный класс, потомками которого будут любые элементы системы (например, входящий поток, накопитель, прибор обслуживания), которые могут принимать заявки с помощью операции accept(…), а также генерировать и / или обрабатывать события. Обработка событий производится с помощью операции processEvent(…).

Класс Source представляет входящий поток заявок. В методе accept(…) генерируется исключение, так как этот элемент не может принимать заявки. А в методе processEvent(…) генерируются новые заявки.

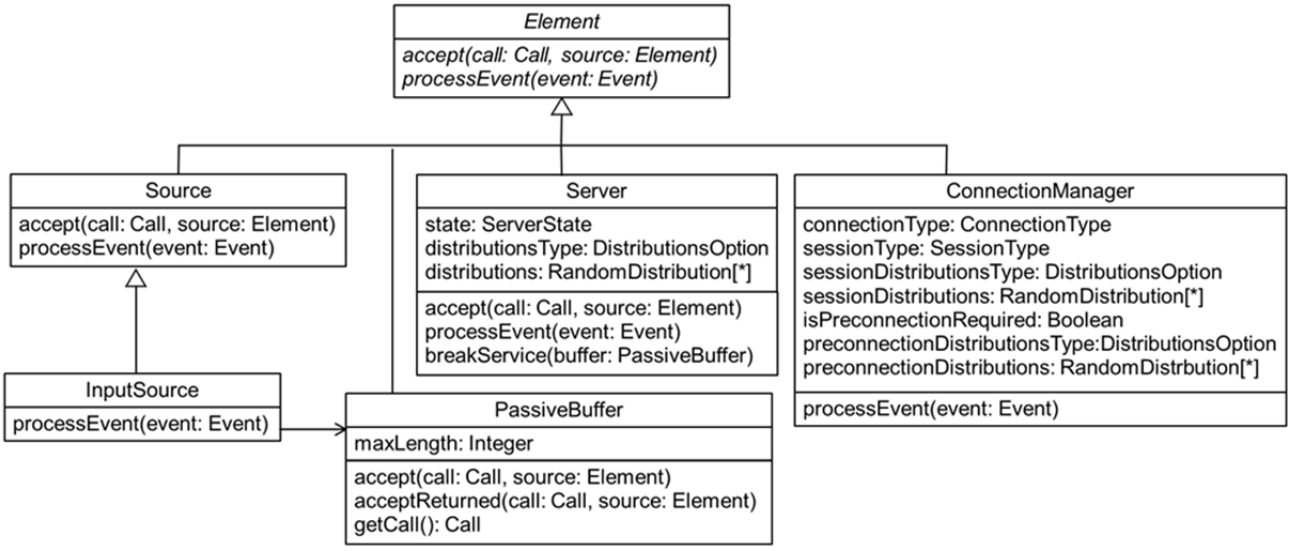

Рис. 2. Объектная модель циклической системы обслуживания

Класс InputSource, в отличие от Source, включает в себя накопитель заявок (объект стандартного класса PassiveBuffer). Метод processEvent(…) вызывается автоматически каркасом платформы, если обрабатываемое системой событие сгенерировано данным объектом InputSource. В дальнейшем объекты класса InputSource будем называть источниками.

Класс PassiveBuffer представляет собой реализацию объекта-накопителя заявок. Он имеет атрибут maxLength, который определяет максимальный объем накопителя (максимальное число заявок, которые можно в нем разместить). Значение maxLength = –1 означает, что накопитель не ограничен в размерах. Методы accept(…) и acceptReturned(…) используются для помещения заявки в накопитель. Если при вызове метода accept(…) оказалось, что накопитель заполнен до предела, то заявка получает отказ и немедленно покидает систему. В случае неограниченного объема накопителя или при вызове метода acceptReturned(…) заявка будет помещена в накопитель в обязательном порядке. В этом случае считается, что накопитель всегда имеет небольшой резерв для размещения заявок, возвращенных в него в экстренных ситуациях (прерывание обслуживания).

Все заявки в накопителе сортируются в порядке времени их первого поступления в накопитель (времени поступления в систему). Операция getCall(…) извлекает самую старую заявку из накопителя и возвращает ее в качестве результата.

Объект Server реализует обслуживающий прибор и имеет два существенных отличия по сравнению с соответствующим объектом платформы, а именно:

1) может пребывать в трех состояниях (поле state): «Свободен», «Занят обслуживанием», «В режиме переключения»;

2) реализует возможность задания отдельных функций распределения для длительности обслуживания заявок из разных накопителей (входящих потоков). Для этого используется опция distributionsType со значениями Single (одна функция распределения для всех накопителей) и ForEachBuffer (функции распределения вероятностей заданы для каждого накопителя в отдельности), а также массив distributions.

Методы accept(…) и processEvent(…) используются соответственно для помещения заявки на обслуживание в данном приборе и для обработки события окончания обслуживания. Метод breakService(…) немедленно прерывает обслуживание заявки и возвращает ее в соответствующий накопитель с помощью метода PassiveBuffer.acceptReturned(…).

Класс ConnectionManager (диспетчер подключений) – искусственно введенный объект, который хранит информацию и реализует логику подключения приборов к очередям. Диспетчер подключений содержит такие атрибуты, как тип соединения connectionType, дисциплина обслуживания очередей sessionType, способ задания функций распределения длительности подключения sessionDistributionsType, функции распределения длительности подключения приборов к накопителям sessionDistributions, флаг

переключения приборов isPreconnectionRequired, способ задания функций распределения длительности времени переключения preconnectionDistributionsType, а также функции распределения длительности переключения приборов preconnectionDistributions.

Метод processEvent(…) класса ConnectionManager обрабатывает событие окончания времени переключения прибора. Другие методы данного класса будут описаны ниже.

2.2. Механизм сеансов подключения

В объектной модели все серверы системы собраны в одну коллекцию – пул серверов (объект класса ServersPool). Все входящие потоки с накопителями также образуют коллекцию. У модели имеется единственный объект класса ConnectionManager, которому предоставлен полный доступ к обеим коллекциям.

Для организации моделирования подключений приборов к накопителям предлагается использовать механизм сеансов подключения. Сеанс подключения (объект Session) – это объект, создаваемый на время подключения пула приборов к одному накопителю (рис. 3). Данный объект отвечает за продолжительность подключения, управляет движением поступающих заявок, а также заявок, находящихся в накопителе и на обслуживании.

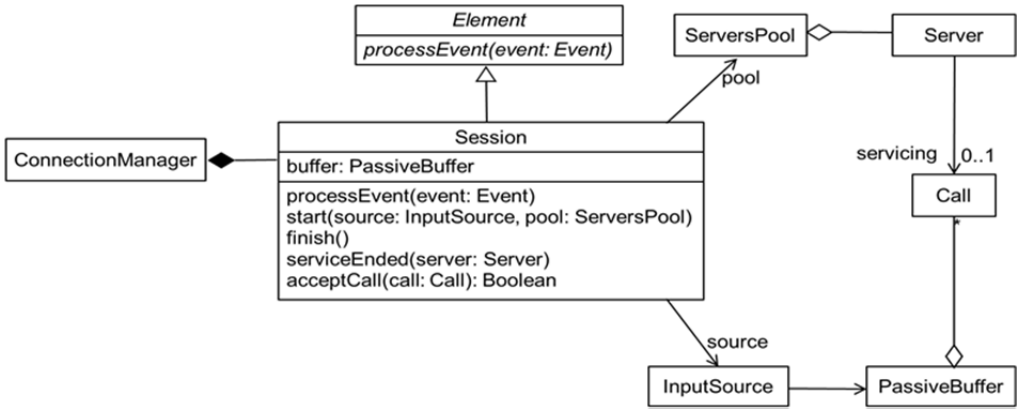

Рис. 3. Структурная схема механизма сеансов подключения

Объект Session реализует операцию processEvent(…) интерфейса Element, в ней обрабатывается событие окончания сеанса. Операции start(…) и finish() используются соответственно для старта и окончания сеанса. Метод serviceEnded(…) вызывается, когда обслуживающий прибор, находящийся в сеансе, закончил обслуживание и готов принять на обслуживание новую заявку. Операция acceptCall(…) используется объектами-источниками InputSource во время поступления заявки в систему для поиска сеанса со свободным прибором и передачи ему этой заявки. Атрибут buffer ссылается на объект-накопитель источника source.

Управление сеансами полностью осуществляет объект ConnectionManager. Для этого он реализует методы (см. рис. 3) startNewSession(…) и sessionEnded(…) соответственно для создания сеанса и обработки события его окончания, а также метод startPreconnection(…) для создания события отложенного старта сеанса (так называемое время переключения).

2.3. Основные сценарии имитационного моделирования циклической системы

Поскольку вся система построена на дискретно-событийной модели управления, то для выполнения полного цикла имитационного моделирования достаточно выделить типы событий системы, реализовать механизмы их генерации, записи в журнал событий, извлечения из него и процедуры обработки этих событий.

Выделим основные типы системных событий:

1. Начало моделирования. Происходит однократно на старте всего процесса моделирования. Не требует записи в журнал.

2. Поступление заявки в систему в определенном входящем потоке.

3. Окончание времени переключения прибора или группы приборов (если задано, что переключение требует времени).

4. Окончание обслуживания заявки на приборе.

5. Завершение сеанса (для дисциплины подключения «с разделением времени»).

Начало моделирования выполняется в методе onInitialization класса PollingModel (рис. 4). Здесь создаются по одному событию на каждый источник заявок. Затем в методе инициализации диспетчера подключений производятся необходимые подготовительные действия в зависимости от заданных параметров моделируемой СМО.

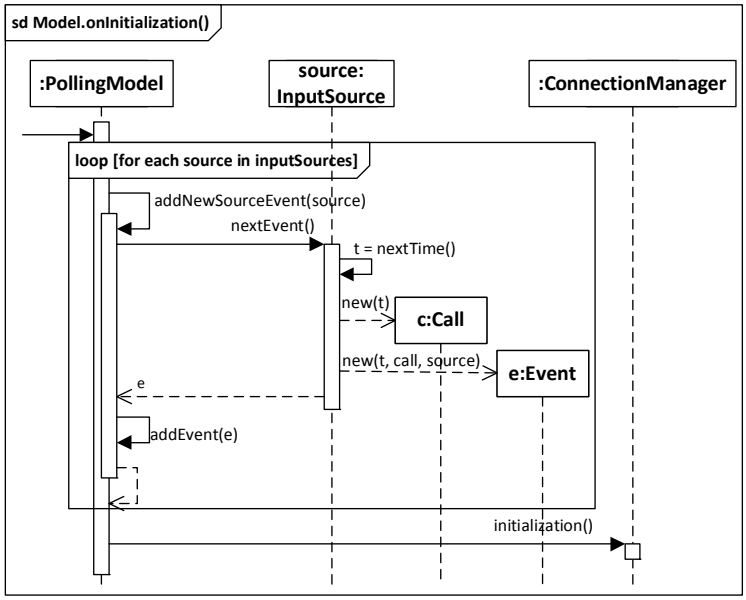

Рис. 4. Сценарий начала моделирования

Событие поступления заявки в систему обрабатывается в методе processEvent(…) класса InputSource (рис. 5). Если источник заявок подключен хотя бы к одному сеансу и накопитель пуст, находим сеанс со свободным прибором и помещаем заявку в этот прибор для обслуживания. Если хотя бы одно из условий нарушено, заявка просто помещается в накопитель. После этого генерируем следующее событие поступления заявки из входящего потока (на диаграмме этот шаг опущен – он выполняется базовым классом Source).

Событие окончания времени переключения группы приборов обрабатывается в методе processEvent(…) класса ConnectionManager. Сначала нужно создать новый сеанс подключения. Алгоритм создания и старта нового сеанса представлен на рис. 6. Во-первых, находим следующий по порядку источник заявок в соответствии с выбранной групповой политикой. Далее создаем новый сеанс и запускаем его. Если выбрана дисциплина подключения «до полного исчерпания» и в накопителе источника отсутствуют заявки, то сеанс подключения тут же прекращается. Если выбрана дисциплина подключения «с разделением времени», то генерируем и регистрируем событие окончания сеанса. Затем все свободные приборы заполняются имеющимися в накопителе заявками.

Событие окончания обслуживания заявки на приборе обрабатывается в методе processEvent(…) класса Server (рис. 7). Обработанная заявка удаляется из прибора, сеансу подключения сообщается об окончании обслуживания. Если в накопителе есть заявка, то она помещается на прибор, иначе если выбрана дисциплина подключения «до полного исчерпания» и ни один прибор пула серверов не занят, то сеанс завершается.

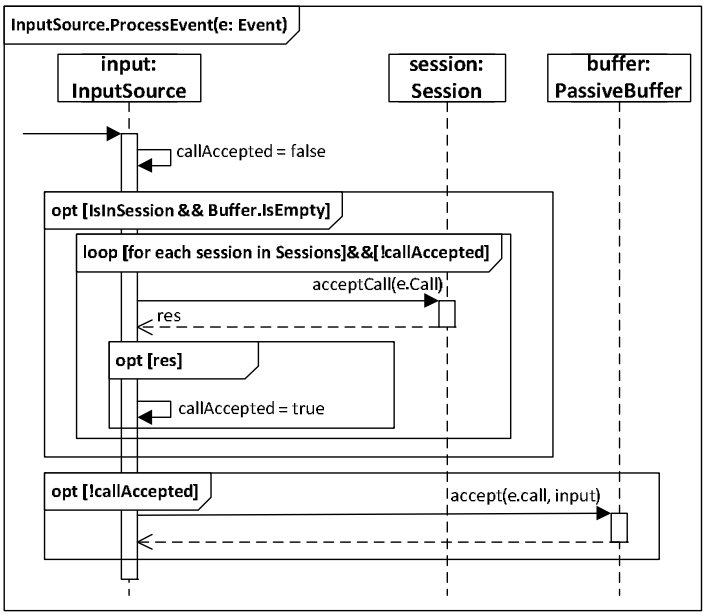

Рис. 5. Обработка события поступления заявки в систему

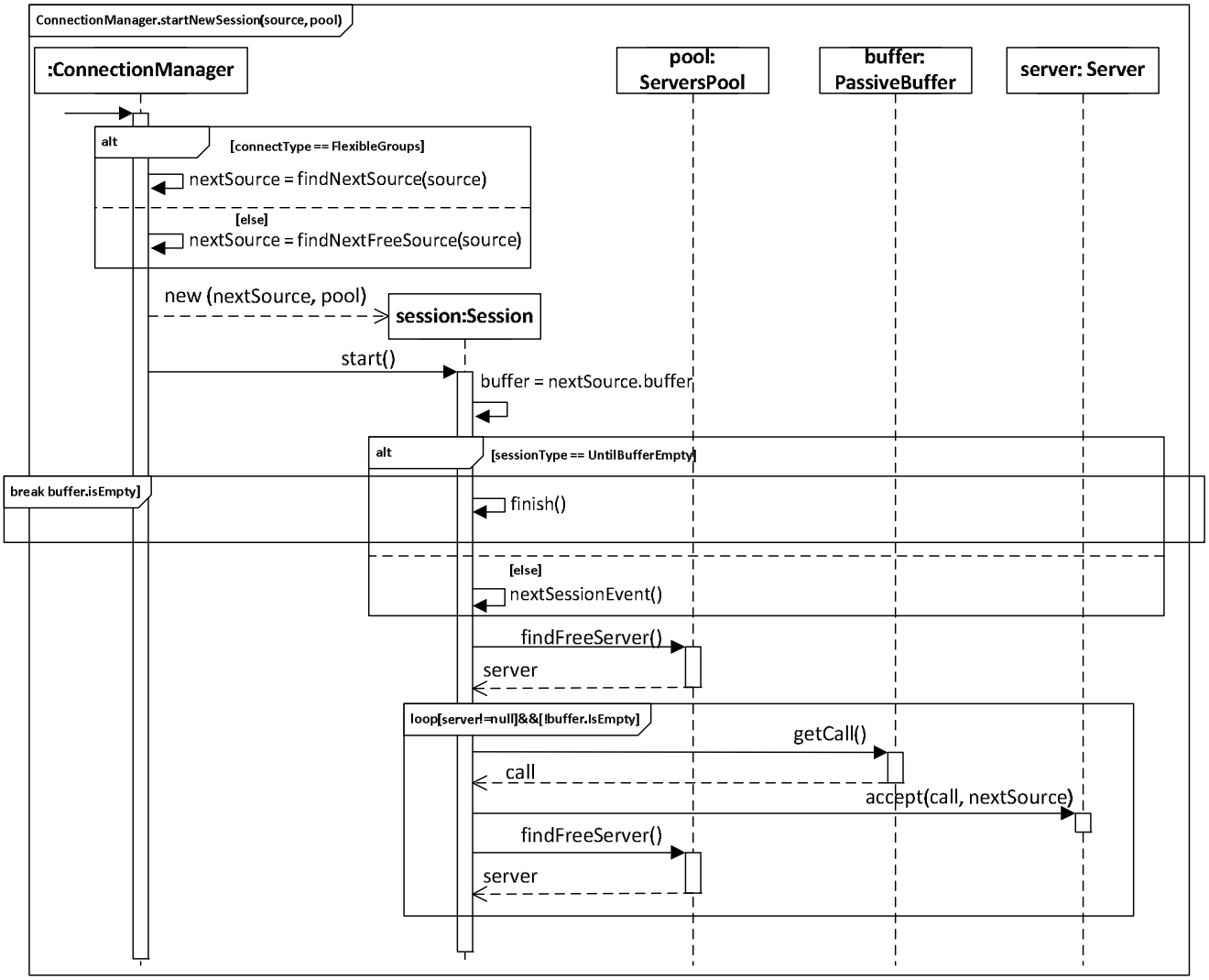

Рис. 6. Алгоритм старта нового сеанса

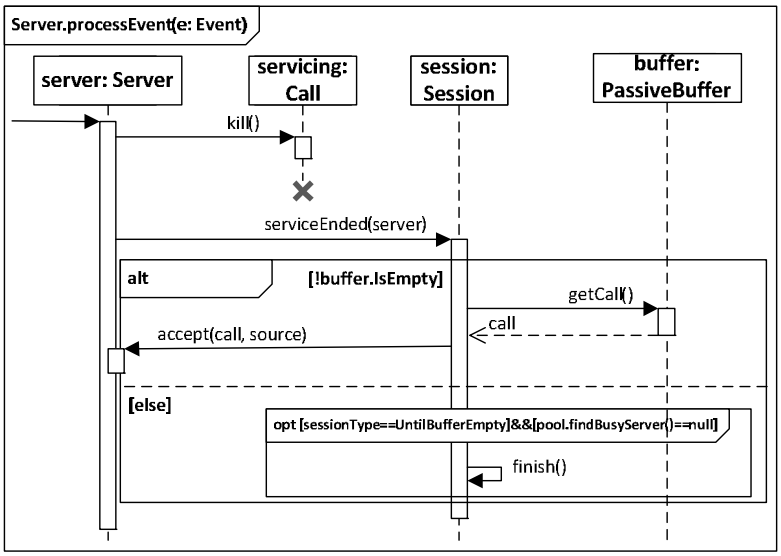

Рис. 7. Обработка события окончания обслуживания заявки

Событие завершения сеанса (для дисциплины подключения «с разделением времени») обрабатывается в методе processEvent(…) класса Session (рис. 8).

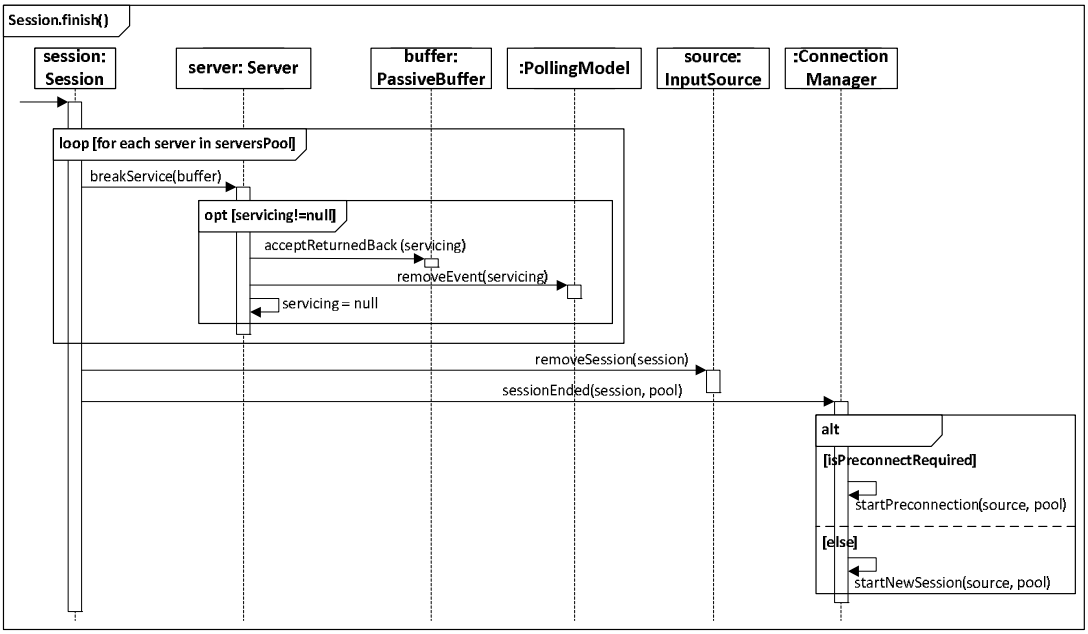

Рис. 8. Обработка события завершения сеанса

Сначала удаляются все признаки сеанса у источника и приборов в пуле. Если какие-то заявки находятся на обслуживании, то их облуживание прерывается и заявки принудительно возвращаются в накопитель. Затем диспетчеру подключений сообщается об окончании сеанса. Он, в свою очередь, если требуется, переводит пул приборов в режим переключения, регистрируя событие окончания времени переключения. Если времени переключения не требуется, то для текущего пула приборов начинается новый сеанс.

Заключение

Разработанная и представленная выше объектная модель позволяет выполнять имитационное моделирование циклических система массового обслуживания с применением дискретно-событийного

подхода. На основе представленной модели было реализовано приложение, позволяющее выполнять имитационное моделирование для широкого класса циклических систем обслуживания за счет вариативности задаваемых параметров конфигурации: различные типы входящих потоков и их параметры для каждого источника, индивидуальные параметры соответствующих им накопителей, индивидуальные параметры для каждого обслуживающего прибора, в том числе в зависимости от обслуживаемого источника, учет возможных затрат времени на переключение приборов от одного источника к другому, три дисциплины подключения групп приборов, две стратегии, определяющие продолжительность сеансов, в том числе для дисциплины «с разделением времени»; реализована возможность задавать продолжительность сеанса подключения в зависимости от обслуживаемого источника.

Разработанное приложение может использоваться для исследования широкого класса циклических систем обслуживания, в том числе и тогда, когда аналитические исследования не представляются возможными.

ЛИТЕРАТУРА

1. Sztrik J. Basic queueing theory. University of Debrecen, 2011. 193 p.
2. Cooper R.B. Introduction to queueing theory, 2nd ed. New York : Elsevier North Holland, 1981. 347 p.
3. Takagi H. Analysis and applications of polling models // Lecture notes in computer science. 2000. V. 1769. P. 423–442.
4. Borst S.C. Polling systems. Amsterdam : Centrum voor wiskunde en informatica, 1996. 232 p.
5. Вишневский В.М., Семенова О.В. Математические методы исследования систем поллинга // Автоматика и телемеханика. 2006. № 2. С. 3–56.
6. Лоу А., Кельтон В. Имитационное моделирование. 3-е изд. СПб. : Питер, 2004. 848 с.
7. Advances in intelligent modelling and simulation: simulation tools and applications / A. Byrski, Z. Oplatková, M. Carvalho, M. Kisiel-Dorohinicki. Springer, 2012. 368 p.
8. GPSS world. URL: http://www.minutemansoftware.com/simulation.htm (дата обращения: 27.01.2017).
9. Schriber T.J. Simulation using GPSS. New York : Wiley, 1974. 592 p.
10. Моделирование и симуляция динамических систем для Simulink. URL: http://matlab.ru/products/simulink (дата обращения: 27.01.2017).
11. Инструмент многоподходного имитационного моделирования AnyLogic. URL: http://www.anylogic.ru (дата обращения: 27.01.2017).
12. Robinson S. Simulation: the practice of model development and use. Hoboken : Wiley, 2004. 336 p.
13. Use cases of discrete event simulation: appliance and research / ed. by S. Bangsow. Springer, 2012. 373 p.
14. Моисеев А.Н. Программная система имитационного моделирования сетей массового обслуживания // Хроники объединенного фонда электронных ресурсов «Наука и образование». 2015. № 11 (78). С. 34.
15. Мещеряков Р.В., Моисеев А.Н., Демин А.Ю., Дорофеев В.А., Матвеев С.А. Применение параллельных вычислений в имитационном моделировании сетей массового обслуживания // Известия Томского политехнического университета. 2014. Т. 325, № 5. С. 99–109.
16. Моисеев А.Н., Синяков М.В. Разработка объектно-ориентированной модели системы имитационного моделирования процессов массового обслуживания // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2010. № 1 (10). С. 89–93.

Сонькин Михаил Аркадьевич, д-р техн. наук, доцент. E-mail: sonkin@tpu.ru
Национальный исследовательский Томский политехнический университет
***Моисеев Александр Николаевич*,** д-р физ.-мат. наук, доцент. E-mail: moiseev.tsu@gmail.com
Национальный исследовательский Томский государственный университет
Сонькин Дмитрий Михайлович, канд. техн. наук, доцент. E-mail: sonkind@tpu.ru
Национальный исследовательский Томский политехнический университет
Буртовая Дарья Алексеевна. E-mail: gottok.inbox@gmail.com
Национальный исследовательский Томский государственный университет

Поступила в редакцию 1 февраля 2017 г.

Sonkin Mikhail A., Moiseev Alexander N., Sonkin Dmitriy M., Burtovaya Daria A. (Tomsk State University, Tomsk Polytechnic University, Russian Federation).

Object model of application for simulation of cyclic queueing systems.

Key words: cyclic queueing system; simulation; object-oriented design.

DOI: 10.17223/19988605/40/8

In this paper, we consider the problem of developing the application for simulation of cyclic queueing systems. Cyclic (polling) systems is a class of queuing systems in which servers cyclically connect to various buffers (or inputs) to serve customers from these buff-

ers. In the study of cyclic systems, in most cases, analytical results cannot be obtained, therefore, the development and using software programs for the simulation of such systems is an important problem.

In the paper, the following model of cyclic queuing systems is considered. The system has $K \geq 1$ inputs where incoming customers arrive. All customers from the input number k go to a single buffer connected to this input. Also, the system has $N \geq 1$ servers that periodically connect to the buffers, take customers from them and serve the customers according to the specified service laws. When a service is complete, customers leave the system. In the case of the number of servers $N > 1$, different versions of their group behavior (disciplines) is possible. In the paper, we consider such group disciplines as a hard group, single devices and a flexible group. Also, there are disciplines of two types for connecting of separate or grouped servers to buffers: time-sharing or exhaustive. Moreover, when the server or group of servers have completed a service of one buffer and they need to switch to the next buffer, it maybe requires a certain switching time. In addition, in the case of the exhaustive discipline, this time should be greater than zero.

We use libraries of software package ODIS as a framework for the application implementation. This framework designed for simulation of queuing networks. The main classes of the framework that used in the work are the following: Element which is an abstract parent of all elements of the simulation model. Source which simulates arrivals of customers, Server which implements the service unit, PassiveBuffer which is a buffer with queue organized for customers in it, Call which implements a customer. In addition, the following special classes for simulation of cyclic systems is designed in the work: InputSource combines two objects – a Source instance and a PassiveBuffer one; ConnectionMannager is an object which implements the logic of connecting servers to buffers; Session is an object which is created at the instant when servers pool is connecting to a buffer, it responds for the duration of the connection and controls the movements of incoming customers, customers in the buffer and in servers.

Special technique of system events is designed to realize the simulation based on the discrete event model. This technique implements mechanisms of generation, logging and processing of system events. For the simulation of the cyclic systems, it is identified five types of the system events: start of simulation; customer arrival in a particular input; the end of switching time period (if it set that switching requires some time); customer service completion; the end of the session (for time-sharing connection discipline). Processing procedures for scenarios of all these events was designed and implemented. An application was implemented on the designed basis. The application is used for investigations in the field of queuing theory.

REFERENCES

1. Sztrik, J. (2011) *Basic Queueing Theory*. University of Debrecen.
2. Cooper, R.B. (1981) *Introduction to Queueing Theory*. 2nd ed. New York: Elsevier North Holland.
3. Takagi, H. (2000) Analysis and applications of polling models. *Lecture Notes in Computer Science*. 1769. pp. 423–442. DOI: 10.1007/3-540-46506-5_18
4. Borst, S.C. (1996) *Polling systems*. Amsterdam: Centrum voor Wiskunde en Informatica.
5. Vishnevskiy, V.M. & Semenova, O.V. (2006) Mathematical methods to study the polling systems. *Automation and Remote Control.* 67(2). pp. 173–220. DOi: 10.1134/S0005117906020019
6. Law, A.M. & Kelton, W.D. (2000) *Simulation Modeling and Analysis*. McGraw-Hill.
7. Byrski, A., Oplatková, Z., Carvalho, M. & Kisiel-Dorohinicki, M. (eds) (2012) *Advances in intelligent modelling and simulation: simulation tools and applications*. Springer.
8. *GPSS world*. (n.d.) [Online] Available from: http://www.minutemansoftware.com/simulation.htm. (Accessed: 27th January 2017).
9. Schriber, T.J. (1974) *Simulation using GPSS*. New York: Wiley.
10. Matlab.ru (n.d.) *Modeling and Simulation of dynamic systems in Simulink*. [Online] Available from: http://matlab.ru/products/simulink. (Accessed: 27th January 2017).
11. Anylogic.ru (n.d.) *The tool of multi-approach simulation AnyLogic*. [Online] Available from: http://www.anylogic.ru. (Accessed: 27th January 2017).
12. Robinson, S. (2004) *Simulation: the practice of model development and use*. Hoboken: Wiley.
13. Bangsow, S. (ed.) (2012) *Use cases of discrete event simulation: appliance and research*. Springer.
14. Moiseev, A.N. (2015) Programmnaya sistema imitatsionnogo modelirovaniya setey massovogo obsluzhivaniya [Software for simulation of queueing networks]. *Khroniki ob"edinen-nogo fonda elektronnykh resursov "Nauka i obrazovanie"*. 11(78). p. 34. (In Russian).
15. Meshcheryakov, R.V., Moiseev, A.N., Demin, A.Yu., Dorofeev, V.A. & Matveev, S.A. (2014) Using parallel computing in queueing network simulation. *Bulletin of Tomsk Polytechnic University*. 325(5). pp. 99–109. (In Russian).
16. Moiseev, A.N. & Sinyakov, M.V. (2010) Development of the object-oriented model of the software for simulation of queueing processes. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitel'naya tekhnika i informatika – Tomsk State University Journal of Control and Computer Science*. 1(10). pp. 89–93. (In Russian).