электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 519.872

Моделирование немарковской системы массового обслуживания с неоднородным входным потоком заявок

Орлова А.А. *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Рудаков И.В..,к.т.н, доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
irudakov@bmstu.ru

Постановка задачи

Моделирование немарковских системы массового обслуживания с неоднородным входным потоком заявок представляет собой интересную задачу. Примером такой системы является СУВ – Система Управления Взаимодействиями. СУВ представляет собой интеграционную платформу, обеспечивающую взаимодействие отдельных информационных систем в составе большой распределенной системы обработки информации.

В основе работы СУВ лежит обработка разнородного потока заявок, связанных между собой. Обрабатываемая заявка может ссылаться на заявку, которая еще только находится в очереди на обслуживание или по каким-то причинам была отказана в системе, что приводит к приостановке обработки текущей заявки. При этом количество заявок, на которые можно ссылаться, не ограничено. Заявка, обработка которой была приостановлена, повторно поступает на обработку тогда, когда все заявки, на которые она ссылалась, успешно были обработаны.

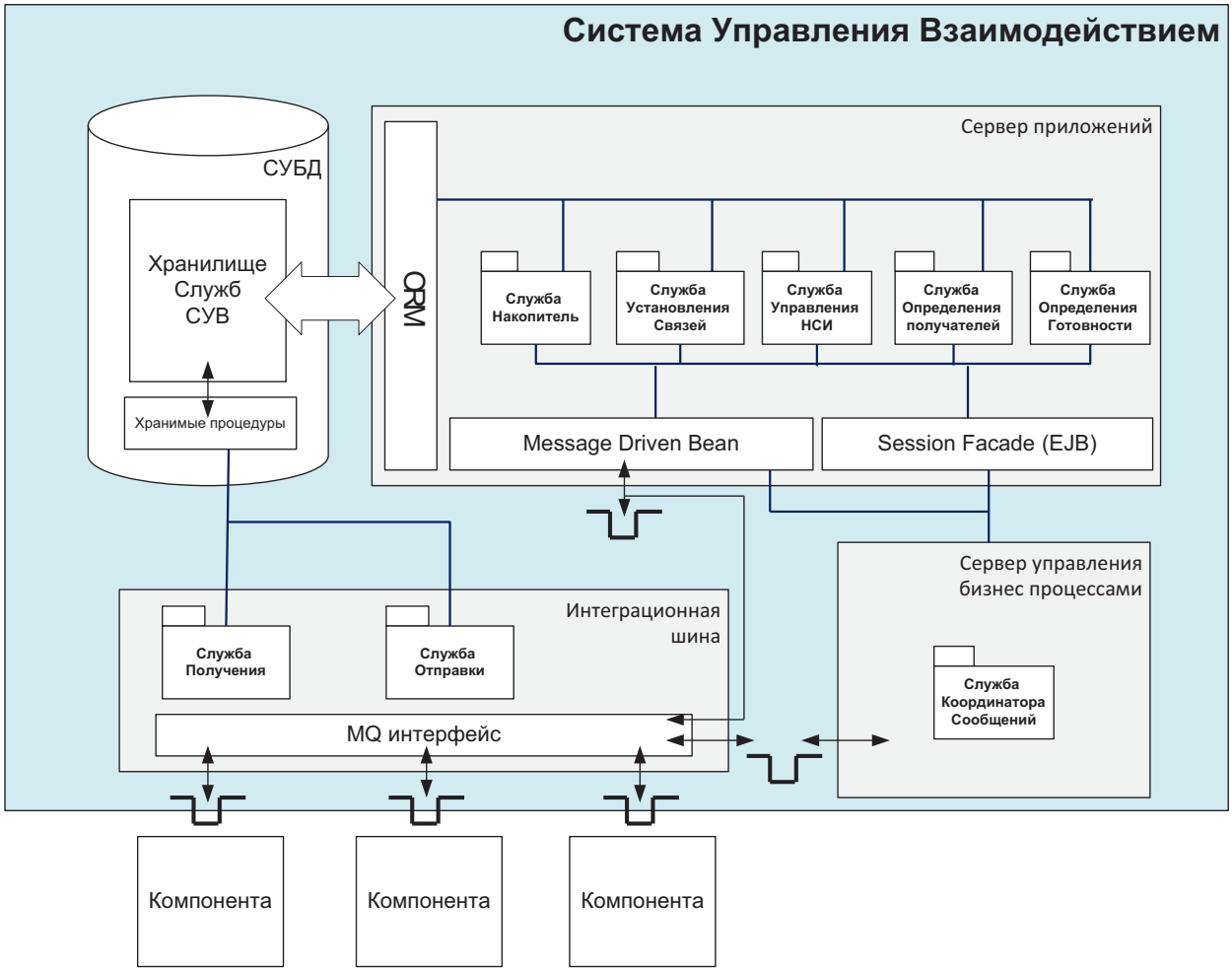

Рис. 1. Логическая модель СУВ

Время обслуживания заявки в СУВ без учета связей с другими заявками в общем случае подчиняется нормальному закону распределения. В случае если заявки становятся взаимосвязанными, время обслуживания в СУВ перестает подчиняться какому-либо закону распределения и является случайной величиной. Кроме того, в СУВ при обработке заявки проверяется наличие связных заявок, т.е. в системе присутствует механизм памяти, что, так же как и отсутствие определенного закона распределения для времени обработки заявки, является одной из характеристик немарковской системы.

Для подобных немарковских систем достаточно трудно, а порой и невозможно построить аналитическую модель системы, в связи с чем необходимо построить имитационную модель для того, чтобы изучить процесс обработки коррелирующего потока в зависимости от того, каким образом заявка обрабатывается в системе. Необходимо так же проанализировать возможные варианты обработки заявки для случая отсутствия связных заявок.

Варианты использования

Существующие на данный момент системы имитационного моделирования [4]-[6] поддерживают симуляцию марковских моделей, поскольку моделирование немарковской системы в общем случае требует значительных аппаратных ресурсов и, как правило, отдельных алгоритмов функционирования. Поэтому, для моделирования обработки заявок в СУВ разработана собственная система имитационного моделирования основанная на классическом дискретно-событийном методе протяжки модельного времени. Особенностью системы является работа с несколькими вариантами обработки входного потока заявок.

В системе реализованы следующие модели поведения (здесь и далее под моделью поведения понимается способ обработки заявок в обслуживающем аппарате, проверяющем наличие связных объектов у заявки):

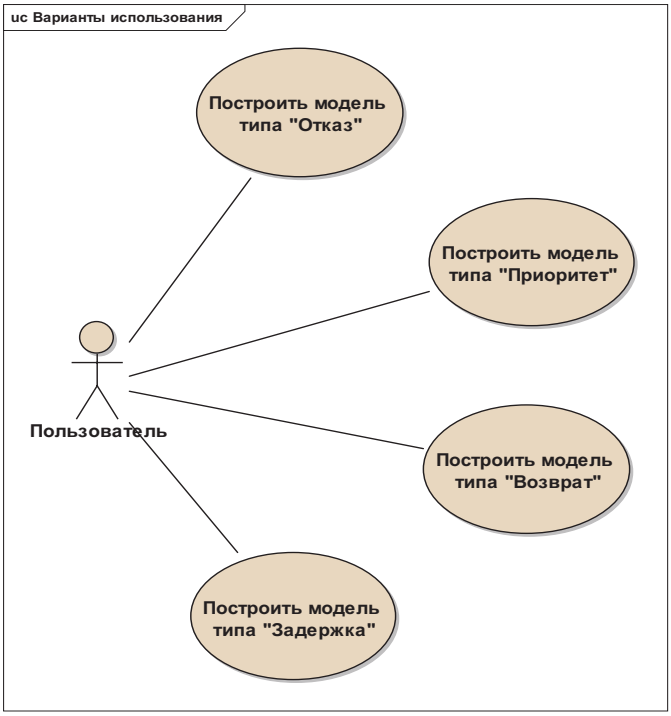

Рис. 2. Варианты использования системы моделирования для построения модели

Для оценки наиболее приемлемой модели поведения их необходимо сравнить по критериям оценки. Критериями оценки для подобной системы выступают следующие параметры системы:

- Среднее время обслуживания заявки (MeanTime);
- Максимальное время обслуживания заявки (MaxTime);

- Процент заявок, обслуживание которых завершилось с ошибкой (Errors).

Рассмотрим подробнее каждую модель поведения.

Модель «Отказ»

Наиболее простым вариантом обработки заявок является модель поведения типа «Отказ». В данном варианте обработка заявки, для которой не найдено связных заявок, завершается с ошибкой. Для данного варианта характерны высокий процент ошибок и достаточно высокая скорость обработки потока заявок. Данный вариант будет принят как базовый, с которым будут в дальнейшем сравнивать остальные модели, так же он используется на практике в реальной системе.

Модель «Приоритет»

Современные системы промежуточного ПО, предоставляющие механизмы передачи сообщений (Message-Oriented Middleware), поддерживают приоритеты сообщений. Кроме того данная возможность реализуется так в POSIX MQ – механизме, поддерживаемом во всех UNIX-системах. Основная идея данной модели состоит в назначении приоритетов сообщениям входного потока. Расстановка приоритетов должна осуществляться на основании типичных зависимостей заявок. Например, известно, что заявки типа 1 всегда зависят от заявок типа 2, благодаря чему приоритет заявок 2 должен быть всегда выше приоритета заявок типа 1. Обработка заявок из входной очереди должна вестись с учетом приоритета заявки: даже если заявка пришла позже всех, но у нее наивысший приоритет, то она будет обработана первой. Если для обрабатываемой заявки отсутствуют связные заявки, то она считается ошибочной.

При достаточно простом варианте зависимостей заявок процент ошибок может свестись к нулю, но на практике это далеко не так. Заявки могут зависеть друг от друга совершенно по-разному. По времени обработки существенных изменений по сравнению с базовым произойти не должно. Дополнительные затраты будут уходить только на выборку сообщений по приоритетам из входной очереди.

Модель «Возврат»

Данная модель предполагает достаточно простой вариант борьбы с ошибками – если заявка не может быть обработана, то она возвращается в конец очереди. Главным недостатком такого подхода является то, что для ряда заявок резко возрастает время обслуживания, что может быть критично при наличии соответствующих требований к системе. Однако, такой подход в теории должен сводить процент ошибок к нулю.

Модель «Задержка»

Данный вид модели представляет собой компромиссный вариант между уже рассмотренными моделями. При таком подходе заявка, в случае невозможности дальнейшей обработки, задерживается или откладывается в буфер, который периодически должен проверяться на предмет наличия заявок, готовых к обработке. Таким образом, среднее время обработки должно вырасти, однако максимальное время обработки заявки будет значительно меньше, чем для модели «Возврат». Процент ошибок так же должен быть нулевым. Данный вариант используется в реальной системе в ряде случаев, и цена такого подхода выражается в значительных накладных расходах и более сложной организации системы.

Функциональные требования к ПО

Система представляет собой набор обслуживающих аппаратов, каждый из которых выполняет соответствующую задачу, представляя собой 1 функциональный модуль реальной системы. На рисунке 3 приведен общий вид процесса обработки заявок в системе. Каждый обслуживающий аппарат является представлением контрольной точки в реальной системе.

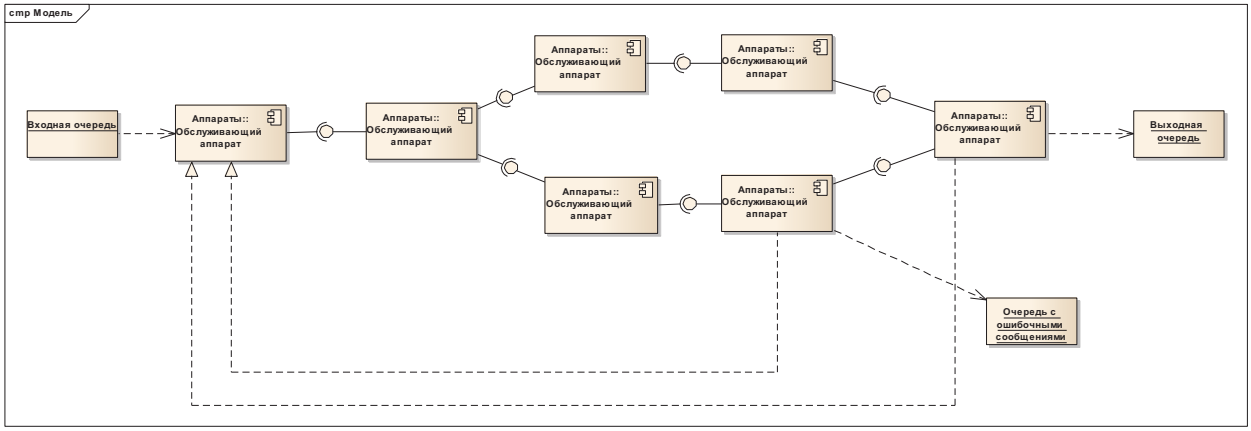

Рис. 3. Общий вид процесса обработки заявок в системе

Под контрольной точкой понимается функциональная часть процесса обработки, связанная с существенными временными затратами или дальнейшей маршрутизацией заявки.

В процессе моделирования реализуются коррелирующий поток и составные заявки, ссылающиеся друг на друга. При поступлении новой заявки система (ее конкретный элемент, обслуживающий аппарат) проверяет, проходили ли связные заявки в системе, если нет, то заявка должна быть перемещена в начало очереди. Очевидно, что в таком случае должна быть реализован некоторый запоминающий аппарат в системе,

позволяющий сохранять связи между заявками, что является характерной чертой немарковских систем.

Так же на систему накладывается требование на количество обрабатываемых заявок в один момент времени – 1. Основанием для такого ограничения в программной реализации является то, что сама система нами рассматривается как один обслуживающий аппарат, декомпозированный в программной реализации на N обслуживающих.

Для реализации данного ограничения необходимо реализовать своеобразные шлюзы на первом и последнем обслуживающих аппаратах, останавливающих и возобновляющих обработку заявок из входной очереди.

Требования к обслуживающему аппарату:

- Для начального аппарата прекращать обслуживание заявок из очереди пока не будет получен сигнал от последнего аппарата или аппарата, завершившего обработку заявки;
- Обслуживающий аппарат должен поддерживать работу с несколькими очередями;
- Обслуживающий аппарат должен поддерживать обработку нескольких типов заявок;
- Аппарат, завершающий обработку заявки, должен отправлять сигнал первому аппарату для возобновления приема заявок.
- Сигнал о возобновлении приема заявок должен представлять собой заявку специального типа.

Заявка в данной системе представляет собой следующую структуру:

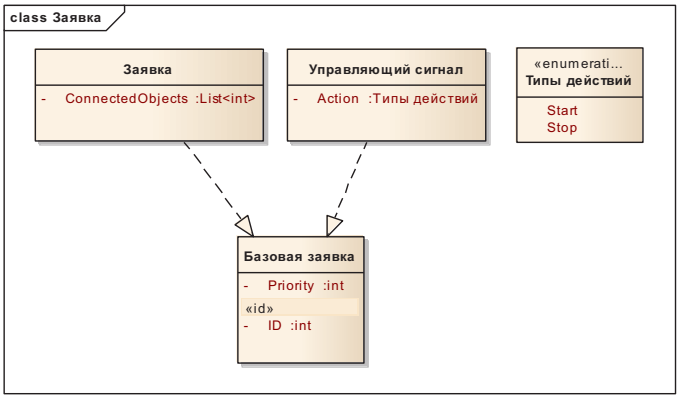

Рис. 4. Структура заявки и ее виды

Объект типа «Заявка» является обычной обслуживаемой заявкой для системы, а «Управляющий сигнал» это заявка служебного вида, регламентирующая начало и остановку обработки заявки.

Генерация входного потока

Одна из первых задач, которые необходимо решить, это генерация входного потока. При генерации необходимо создать связи между заявками, которые возникают из бизнес-модели сущностей (заявок), обрабатываемых системой. На первом этапе будет рассматриваться упрощенный вариант, состоящий из нескольких типов заявок и связей между ними.

На рисунке 4 представлены три типа заявок и связи между ними:

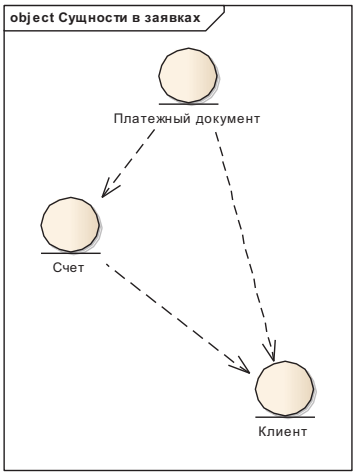

Рис. 5. Модель бизнес-объектов системы

В таком случае, нам сначала надо сгенерировать заявки для объектов «Клиент», «Счет», «Платежный документ» и проставить для них связи согласно схеме.

Алгоритм генерации входного потока выглядит следующими образом:

1. Сгенерировать стартовую заявку.
2. Стартовая заявка выбирается произвольным образом. Стартовая заявка становится текущей.
3. Для текущей заявки для каждого из возможных видом связей сгенерировать связную заявку и назначить связь.
4. Для каждой связной заявки повторить рекурсивно п.2.
5. Поместить во входную очередь сгенерированные заявки.

6. Повторить шаги 1-3 K раз (настраиваемый параметр).
7. Перемешать входную очередь.

Однако у данного алгоритма есть один недостаток – если перемешивать входную очередь равномерно, то разброс связных заявок может оказаться очень большим, что слабо соответствует реальности. Поэтому перемешивать заявки предлагается следующий алгоритм перемешивания входной очереди:

1. Обозначим размер набора сгенерированных за 1 шаг заявок – N.
2. Задать начальные значения интервала $[I_{start};I_{end}]$ - $I_{start} = 1$; $I_{end} = M*N$, где M – параметр обозначающий, сколько наборов заявок будет перемешиваться за одну итерацию; K (общее число наборов заявок) должно нацело делиться на M/2.
3. Выбрать из входной очереди заявки $[I_{start};I_{end}]$ и перемешать их.
4. Увеличить $I_{start} = I_{start} + M*N/2$; $I_{end} = I_{end} + M*N/2$;
5. Повторить шаги 3-4 до тех пор, пока $I_{end} < K*N$;

Отдельно нужно рассмотреть приоритезацию входного потока. Для случая, представленного на рисунке 4, приоритеты назначаются просто: «Клиент» – наивысший, «Счет» – средний, «Платежный документ» - низший.

Назначение приоритетов в общем случае выполняется по алгоритму нахождения кратчайшего пути по графу, получаемому из бизнес-модели. Для этого нужно только выбрать начальное состояние или «стартовую» заявку.

Замечание: Приоритеация используется в модели «Приоритет» и от нее зависит порядок выборки заявок из входной очереди. На практике бизнес-модель может выглядеть следующим образом:

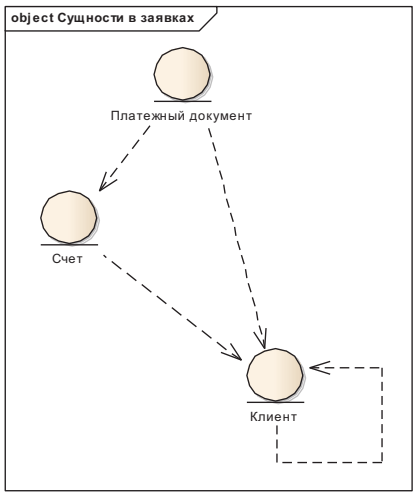

Рис.6. Альтернативная модель бизнес-объектов системы

Приоритеты в данном случае будут расставлены так же, но модель поведения «Приоритет» для такой модели бизнес-объектов может не дать 0% ошибок, т.к. заявки, соответствующие связным клиентам могут быть так же перемешаны. Таким образом, главным условием отсутствия ошибок для модели «Приоритет» является отсутствие циклов в модели бизнес-объектов.

Реализация

Согласно приведенных функциональных требований, а так же общих функциональных требований, не рассматриваемых в данной статье, разработана система имитационного моделирования, позволяющая реализовать перечисленные в п.2 модели поведения системы при обработке заявок.

Структура программного комплекса приведена на рисунке 7.

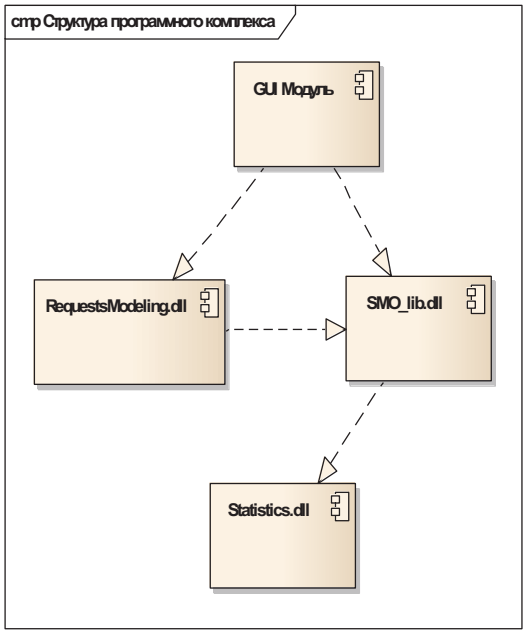

Рис. 7. Структура программного комплекса

Основным модулем программного комплекса, отвечающим за моделирование, является подключаемая библиотека SMO_lib.dll, реализующая основные функциональные требования к системе. Работа с моделью и ее настройка осуществляется посредством интуитивно понятного интерфейса пользователя (GUI модуль). За сбор статистических данных отвечает модуль Statictics.dll. Модуль RequestsModeling.dll отвечает за настройку генерации входного потока, задание возможных связей между типами заявок.

Целью проведения экспериментов с различными моделями поведения системы является анализ выходных параметров моделирования для каждого типа модели поведения.

Основными параметрами, на основе которых выполняется сравнение и анализ, являются:

- Среднее время обслуживания заявки (*MeanTime*);
- Максимальное время обслуживания заявки (*MaxTime*);
- Процент заявок, обслуживание которых завершилось с ошибкой (*Errors*).

Таблица 1

План эксперимента

Этап	**Описание этапа**	**Комментарий**
Построение модели.	На данном этапе в системе имитационного моделирования собирается модель, задаются начальные и конечные обслуживающие аппараты, выбираются аппараты, проверяющие наличие связей.	Данный этап является общим для всех моделей поведения.
Настройка типов заявок	Задаются возможные типы заявок и возможные связи между ними	Общий этап для всех моделей поведения
Задание приоритетов	Задаются приоритеты типов заявок автоматически или вручную	Данный этап необходим для проведения эксперимента с моделью поведения «Приоритет»
Настройка проверяющих аппаратов	Задается необходимая модель поведения	Вариативный пункт плана
Запуск симуляции	Запускается симуляция модели с указанием объема входного потока	Параллельно с процессом симуляции собирается необхо-димая статистика
Анализ результатов	Визуализация собранной статистики и подведение итогов	

Результаты

Были проведены эксперименты со всеми предложенными моделями поведения. В качестве бизнес-модели заявок использована модель с рисунка 5.

Результаты экспериментов в зависимости от рассматриваемого параметра приведены далее в таблице 2:

Таблица 2

Результаты экспериментов

Параметр / Модель	**MeanTime, ед. времени**	**MaxTime, ед. времени**	**Errors, %**
«Отказ»	47,264	51,591	37
«Возврат»	89,346	128,334	0
«Приоритет»	54,284	59,057	9
«Задержка»	60,108	75,046	0

Как видно по сводной таблице наиболее оптимальными вариантами с точки зрения отсутствия ошибок при обработке являются модели поведения «Возврат» и «Задержка». Оба этих варианта не являются самыми быстрыми, однако, они самые надежные. В случае, если приоритетом является скорость обработки, то оптимальным вариантом будет модель «Приоритет». Среднее время обработки (*MeanTime*) незначительно больше модели «Отказ», максимальное время обработки также сравнимо, однако процент ошибок значительно меньше.

Список литературы

1. Назаров А. А., Моисеева Е. А. Исследование RQ-системы MMPP|M|1 методом Асимптотического анализа в условии большой нагрузки // Вестник Томского государственного университета. Сер. Управление, вычислительная техника и информатика. 2013. № 4. С. 84-94.
2. Назаров А. А., Семенова И. А. Исследование RQ-систем методом асимптотических семиинвариантов // Известия Томского Политехнического Университета. Управление. Вычислительная техника и информатика. 2010. № 3. С. 85-96.
3. Назаров А. А., Моисеева С. П., Морозова А. С. Исследования СМО с повторным обращением и неограниченным числом обслуживающих приборов методом предельной декомпозиции // Вычислительные технологии. 2008. Т. 13. Спец. вып. С. 272-277.
4. Обзор — Инструмент имитационного моделирования AnyLogic. Режим доступа: http://www.anylogic.ru/overview (дата обращения 15.05.2014).

5. Arena Simulation Software by Rockwell Automation Home. Режим доступа: http://www.arenasimulation.com/Arena_Home.aspx (дата обращения 15.05.2014).
6. GPSS World - General Information. Режим доступа: http://www.minutemansoftware.com/general_information.htm (дата обращения 15.05.2014).
7. Рудаков И. В., Шляева А. В. Моделирование входных потоков данных для Стохастических моделей дискретных систем // Вестник МГТУ им. Н. Э. Баумана. Сер. Приборостроение. 2008. № 2. С. 65-72.
8. Кельтон В. Дэвид, Лоу М. Аверилл. Имитационное моделирование. СПб.: Питер, 2004. 846 с.
9. Лоскутов А. Ю., Михайлов А. С. Основы теории сложных систем. Ижевск: нститут компьютерных исследований, 2007. 620 с.