На правах рукописи

Приступа Андрей Викторович

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ СМО НА ОСНОВЕ ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ МОДЕЛИ ДИСКРЕТНО-СОБЫТИЙНОГО МЕТОДА

Специальность 05.13.11 –
«Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Томск – 2006

Работа выполнена в Томском государственном университете на кафедре прикладной информатики факультета информатики

Научный руководитель: доктор физико-математических наук, доцент Змеев О.А.

Официальные оппоненты: доктор технических наук, профессор Костюк Ю.Л.

кандидат технических наук, Моисеев А.Н.

Ведущая организация – Томский политехнический университет.

Защита состоится 26 октября 2006 г. в 10-30 на заседании Диссертационного совета Д 212.267.08 в Томском государственном университете по адресу: г. Томск, пр. Ленина 36, корп. 2, ауд. 102.

С диссертацией можно ознакомиться в научной библиотеке Томского государственного университета.

Отзывы на автореферат (2 экз.), заверенные печатью, высылать по адресу: 634050, г. Томск, пр. Ленина, 36, ученому секретарю ТГУ.

Автореферат разослан 22 сентября 2006 г.

Ученый секретарь
Диссертационного совета,
доктор технических наук, доцент Скворцов А.В.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы

В разных областях техники, в организации производства, в экономике и медицине, в социальной сфере, в военном деле и во многих других сферах человеческой деятельности постоянно возникает необходимость решения вероятностных задач, связанных с работой систем массового обслуживания. По справедливому замечанию виднейшего советского специалиста в этой области Б.В. Гнеденко, "легче указать ситуации, где не может быть использована теория массового обслуживания (она же – теория очередей), чем перечислить все сферы ее потенциального применения". Многообразие приложений этой теории определяет постоянно растущий интерес к ней, а сложность возникающих задач не позволяет получать решения на базе аналитических методов. По мнению А.М. Лоу и В.Д. Кельтона, имитационное моделирование является одним из наиболее распространенных, а возможно, и самым распространенным методом исследования операций и теории управления, а согласно исследованию Gupta, эта технология исследования сложных систем занимает второе место после математического программирования. Об этом свидетельствуют и Зимние конференции по вопросам имитационного моделирования (Winter Simulation Conference), ежегодно собирающие до 700 участников.

Одним из наиболее распространенных методов имитационного моделирования по праву считается *дискретно-событийный метод*, в основе которого лежит концепция заявок (транзактов). Исторически этот подход восходит к Джеффри Гордону, который в 1960-х гг. разработал язык GPSS. Заявки в этой схеме являются пассивными элементами (детали, сообщения), они двигаются в соответствии со своими маршрутами, при этом время от времени с ними происходят так называемые *события*. Под событиями понимаются мгновенные импульсы, которые приводят к изменению вектора состояний системы (например, поступление заявки в очередь, начало и окончание обслуживания и т.п.).

Целый ряд зарубежных фирм выпускает программные продукты, так или иначе поддерживающие дискретно-событийное моделирование; некоторые общего назначения, большинство нацелено на определенные предметные области: обслуживание, бизнес-процессы, производство, логистика и т.д. В качестве примера российских разработок следует отметить продукт AnyLogic компании XJ Technologies, который получил признание в том числе и за рубежом. Отечественные ученые В.Л. Конюх, Я.Б. Игнатьев, В.В. Зиновьев в 2003 году подготовили и выпустили электронный диск «Методы имитационного модели-

рования дискретных систем», где представили обзор программных средств имитационного моделирования, в том числе и современных.

В последние годы также значительно вырос интерес к распределенным технологиям имитационного моделирования. Вопросы, связанные с применением распределенных вычислений в рамках дискретно-событийного метода, можно проследить в работах U. Chandrasekaran, S. Sheppard, J. Misra, R.M. Fujimoto, D.M. Nicol, R.L. Bagrodia. В качестве альтернативы или возможного пути реализации распределенного подхода рассматривается и объектно-ориентированное моделирование. Заметим, что фактически история ООП берет свое начало именно из задач имитационного моделирования и инструмента, предназначенного для их решения (SIMULA, начало 60-х годов прошлого века). В качестве источников, в которых рассмотрены вопросы объектно-ориентированного моделирования, отметим работы J.A. Levasseur, D.W. Jones, S.D. Roberts.

Заметим, что для работы с существующими инструментами имитационного моделирования пользователю требуется еще изучить их внутренний язык, поскольку редкая модель может быть разработана с помощью изменения параметров какого-то шаблонного варианта – обычно требуется написание кода. Изучение языка в этом случае иногда оказывается более сложной задачей, чем разработка самой модели. Для решения указанной проблемы разрабатываются специальные инструменты, направленные на упрощение процесса создания моделей. Так, например, программная система ISS 2000, разработанная В.Н. Томашевским (являющаяся, по сути, генератором программ на языке GPSS), позволяет с использованием визуальных средств создавать и связывать между собой компоненты модели, не прибегая непосредственно к написанию кода. В результате получается текст программы, которая исполняется в стандартной GPSS-оболочке. Однако подобный подход, к сожалению, сохраняет за собой все те ограничения, которыми обладает сам язык GPSS.

С другой стороны, применение универсальных языков программирования позволяет исследователю добиться большей гибкости при разработке, отладке и испытании модели, чем при использовании специализированных языков и сред. Очевидным преимуществом является и более широкое распространение универсальных языков программирования по сравнению со специализированными языками моделирования. Следует отметить экспериментальную разработку А.Г. Королева, в которой модели систем массового обслуживания представляются в GPSS-подобном стиле, но в виде набора процедур на Object Pascal. Эти тексты компилируются, и в результате создается программа, являю-

щаяся моделью конкретной системы массового обслуживания. С ней можно проводить эксперименты и получать статистику. Однако средства визуального моделирования в данной системе отсутствуют: модель от начала до конца создается в текстовом редакторе.

В этой связи разработка программной системы дискретно-событийного имитационного моделирования, которая с одной стороны была бы по возможности максимально предметно-ориентированной и предоставляла возможности визуального моделирования, а с другой стороны сохранила бы преимущества использования универсальных языков программирования, по мнению автора, является актуальной научно-технической задачей. Настоящая работа посвящена разработке такой системы на основе современных представлений о проектировании программных продуктов. При описании проектных решений используется нотация UML, а также шаблоны проектирования (в качестве источника отметим монографию E. Gamma, R. Helm, R. Johnson, J. Vlissides). В качестве средства выполнения кода, записанного на языке программирования, предлагается использовать библиотеку FastScript, разработанную компанией Fast Reports.

Цель работы

Целью данной работы является разработка универсальной объектно-ориентированной модели дискретно-событийного метода имитационного моделирования и реализация программного комплекса, основанного на этой модели, для визуального моделирования систем массового обслуживания с возможностью доопределения логики работы моделей на одном из языков программирования. В рамках указанной цели поставлены и решены следующие задачи:

1) разработка объектно-ориентированной модели дискретно-событийного метода, построенной на принципах повторного использования;

2) разработка каркаса архитектуры программного комплекса для реализации дискретно-событийного метода имитационного моделирования;

3) реализация программного комплекса имитационного моделирования систем массового обслуживания.

Методика исследований

Для решения задач, обеспечивающих достижение поставленной цели, использовались принципы объектно-ориентированного анализа и проектирования, методы имитационного моделирования, теории массового обслуживания и математической статистики.

Научная новизна работы

1. Предложена оригинальная объектно-ориентированная модель дискретно-событийного метода имитационного моделирования, основанная на принципах повторного использования дизайна.

2. Разработана типовая архитектура программного комплекса для визуального моделирования систем массового обслуживания с возможностью доопределения логики работы моделей. В отличие от других инструментов, в качестве средства написания кода предлагается использовать скриптовые языки, синтаксис и возможности которых приближены к традиционным языкам программирования.

Практическая значимость работы

1. Разработан программный комплекс имитационного моделирования СМО «ObjectSim», позволяющий без изучения синтаксиса конкретного, специализированного языка имитационного моделирования строить имитационные модели, комбинируя работу с визуальными компонентами с написанием собственных обработчиков для различных событий. При этом в качестве средства написания обработчиков может быть использован любой из четырех языков – PascalScript, C++Script, BasicScript и JavaScript. Принципы повторного использования, в соответствии с которыми разрабатывался комплекс, допускают расширение приведенного перечня языков.

2. Проведенное исследование имитационных моделей многозвенных виртуальных каналов с боковым трафиком позволило установить основные операционные характеристики для различных значений буферной емкости в транзитных узлах коммутации и скорости передачи данных в звеньях.

Положения, выносимые на защиту

1. Способ описания графов событий имитационной модели систем массового обслуживания в нотации UML.

2. Объектно-ориентированная модель дискретно-событийного метода имитационного моделирования.

3. Архитектура программного комплекса для моделирования систем массового обслуживания с возможностью доопределения логики работы моделей.

Внедрение полученных результатов

Программный комплекс имитационного моделирования «ObjectSim» внедрен в ООО «Сибирские цифровые приборы», г. Томск. Разработанные модели используются для исследования качества работы коммутаторов, а также зависимости операционных характе-

ристик многозвенных виртуальных каналов от различных параметров, в частности от битовых скоростей и емкости буферных накопителей.

Комплекс «ObjectSim» используется также в учебном процессе на факультете информатики Томского государственного университета при проведении практических занятий по имитационному моделированию СМО по специальности 35.15.00 «Математическое обеспечение и администрирование информационных систем».

Апробация работы

Результаты работы докладывались и обсуждались на следующих конференциях:

1. VII Межрегиональная научно-практическая конференция «Научное творчество молодежи», Анжеро-Судженск, 2003.
2. Всероссийская конференция «Наука и практика: Диалоги нового века», Анжеро-Судженск, 2003.
3. VIII Всероссийская научно-практическая конференция «Научное творчество молодежи», Анжеро-Судженск, 2004.
4. 8-th Korea-Russia International Symposium on Science and Technology. Information Technology, Tomsk Polytechnic University, 2004.
5. II Всероссийская научно-практическая конференция «Имитационное моделирование. Теория и практика», Санкт-Петербург, 2005.

Публикации по теме работы

Основное содержание работы отражено в 7 публикациях, в т.ч. 1 статья опубликована в журнале из списка ВАК.

Личный вклад автора

Основные научные результаты получены автором самостоятельно. Постановка задачи была выполнена автором совместно с научным руководителем. Программная реализация комплекса имитационного моделирования выполнена автором.

Структура диссертации

Диссертация состоит из введения, основного текста, заключения, библиографического списка (100 наименований), и приложения, включающего документы о внедрении. Основной текст состоит из 4 глав и содержит 15 таблиц и 96 рисунков. Общий объем работы 154 страницы, включая 3 страницы приложения.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

В первой главе диссертации проводится сравнение существующих подходов к имитационному моделированию дискретных систем и да-

ется обзор программных инструментов, поддерживающих дискретно-событийное моделирование.

В первом пункте обзора рассмотрены достоинства и недостатки подходов, связанных с применением традиционных языков программирования и с использованием специализированных инструментов имитационного моделирования (в частности, языков и сред). Во втором и третьем пунктах приводится обзор существующих программных средств – от классических языков (GPSS, SIMULA) до современных визуальных сред разработки (ARENA, EXTEND, SIMPLEX3, ANYLOGIC). В четвертом пункте ставятся задачи исследования и разработок диссертации на основе выявленных нерешенных проблем. В частности установлено, что до сих пор нигде не представлено описание дискретно-событийного метода имитационного моделирования с позиций объектно-ориентированного подхода. Кроме того, существующие инструменты имитационного моделирования требуют изучения внутреннего языка, а это сужает круг их использования.

Вторая глава посвящена разработке архитектуры программного комплекса имитационного моделирования, основанной на принципах повторного использования дизайна. Несмотря на то, что моделирование используется довольно широко, а классы моделей очень разнообразны, дискретно-событийные имитационные модели включают в себя, как правило, ряд общих элементов. Логическая организация этих элементов позволяет обеспечить программирование, выполнение, отладку и последующее изменение имитационной модели. В п. 2.1 рассмотрено взаимодействие основных элементов дискретно-событийной модели на примере однократного и многократного запуска.

Динамический характер дискретно-событийных имитационных моделей требует, прежде всего, определенных средств слежения за имитационным временем и его продвижения вперед. Такой механизм обязательно присутствует во всех системах и называется часами модельного времени. Кроме часов модельного времени в дискретно-событийной модели можно выделить следующие элементы: множество компонентов, состояние системы, множество событий, множество статистики, механизм инициализации, механизм синхронизации, механизм обработки событий, механизм генерации случайных величин и генератор отчетов. Всеми этими элементами управляет основная программа, обеспечивающая прогоны имитационной модели. Соответствующая диаграмма активности на языке UML приведена на рис. 1.

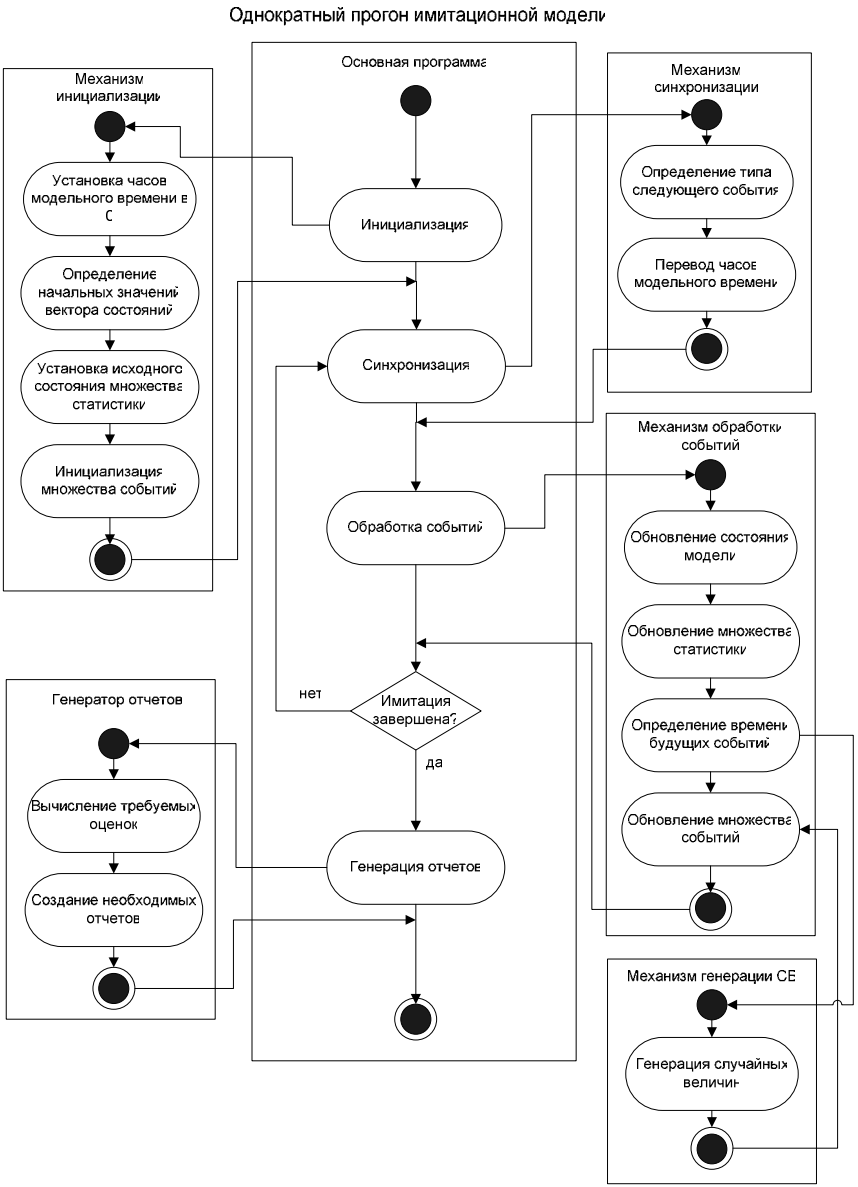

Рис. 1. Логические связи между элементами

В случае многократного запуска отличие состоит в том, что после каждого прогона необходимо обновить множество статистики и, если необходимо, выполнить следующий прогон.

В п. 2.2 рассматривается концептуальная модель системы имитационного моделирования. В п. 2.2.1 – 2.2.6 приведены основные требования к комплексу имитационного моделирования в виде стандартных диаграмм вариантов использования (use case), которые описывают возможные варианты взаимодействия приложения и его составляющих с пользователем.

В п. 2.2.2 рассмотрен вариант использования «Разработка модели», который включает в себя создание и настройку компонентов, определение связей между ними, определение глобальных переменных и создание собственных статистик. Кроме стандартных компонентов, модель может содержать в себе также и компоненты, разработанные пользователем в специальном редакторе. Здесь разработчик модели получает полную свободу, самостоятельно может создавать необходимое ему количество портов, констант и переменных, в том числе и для сбора статистики. Алгоритмы работы целиком определяются пользователем в коде обработчиков.

В п. 2.2.3 описан вариант использования «Запуск модели». Разработчику предоставляется возможность установить длительность процесса моделирования (или задать условия останова), а также настроить сбор статистики. Дело в том, что система переходит в стационарный режим (если он существует) не сразу, а через некоторое время, и данные наблюдений, близких к началу моделирования, могут оказаться нехарактерными для установившегося поведения. Поэтому для получения более точных оценок установившихся параметров используется метод удаления начальных данных.

В п. 2.2.4 рассматривается проведение экспериментов над моделями. В рамках данного варианта использования можно выделить два типа экспериментов – однофакторные и многофакторные. Проведение однофакторного эксперимента подразумевает изменение одного из параметров модели (*фактора*) в определенных пределах с некоторым шагом, при этом для каждого уровня фактора производится некоторое число прогонов. В случае многофакторного эксперимента используется построение *факторных планов*.

В п. 2.2.5 рассматривается вариант использования «Работа с результатами», при этом выделяются 2 типа результатов: покомпонентная и дополнительная статистика. Первая представляет собой набор объектов-статистик, индивидуальных для каждого компонента (например, для очереди это ее длина, время пребывания в ней заявок и

т.п.). Под дополнительными понимаются статистики, созданные и изменяемые разработчиком модели.

Загрузка и сохранение модели рассмотрены в п. 2.2.6.

В п. 2.3 приведено описание логической модели системы. Вначале рассматривается иерархия классов (п. 2.3.1), представляющих модельные компоненты. В основу положен базовый класс TModelComponent. Операции, реализованные на уровне данного класса, выполняют действия, общие для всех компонентов, независимо от их типа. Здесь происходит компиляция и выполнение обработчиков событий, возникающих при входе и выходе заявок (если таковые заданы), добавление заявок в список обработки при входе и удаление их при выходе. Здесь же содержится информация о структуре компонента.

Ниже по иерархии находятся классы TStandardComponent и TUserComponent (представляющие соответственно стандартные и пользовательские компоненты). Классы стандартных компонентов (источник, очередь, устройства и т.п.) наследуются от TStandardComponent. Коротко рассмотрим их основные характеристики.

TSource (источник заявок). Характеризуется законом распределения времени между поступлением заявок, задержкой первой заявки, приоритетом заявок, ограничением числа заявок, размером группы заявок, циклом работы.

TEntitiesQueue (очередь). Характеризуется дисциплиной, максимальной длиной, пределом времени ожидания, задержкой первой заявки, приоритетом заявок, ограничением числа заявок, размером группы заявок, циклом работы.

Поскольку дисциплины выбора заявок из очереди определяют, вообще говоря, семейство алгоритмов, то в данном контексте целесообразно использовать шаблон проектирования *Стратегия*, согласно которому в базовом классе следует объявить лишь интерфейс для выполнения операции, конкретные же варианты ее выполнения реализовать в подклассах (рис. 2). Данный подход позволяет легко добавлять новые дисциплины: для этого достаточно лишь определить новый класс.

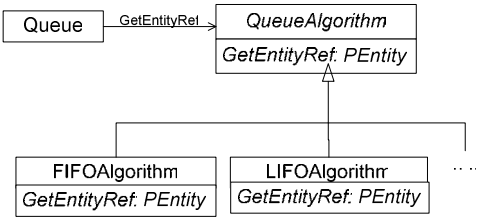

Рис. 2. Выбор очередной заявки

TSingleServer и TMultiServer (устройства обслуживания). Характеризуются законом распределения времени обслуживания, дисциплиной обслуживания, дисциплиной выбора очередной заявки (в случае, если к устройству подходит несколько маршрутов), циклом работы. Для режима с конфликтами можно задать закон распределения времени оповещения о конфликте. Для данных объектов предусмотрена также возможность записывать обработчики событий «прерывание обслуживания» и «возобновление обслуживания» (для приоритетных СМО). Многоканальное устройство характеризуется также числом каналов.

Дисциплины обслуживания и дисциплины выбора, как и в предыдущем случае, определяют семейства алгоритмов. Поэтому для их реализации используется аналогичный подход (шаблон *Стратегия*).

TMerge и TFurcate (объединитель и разветвитель потоков). Объединитель содержит несколько входных портов и один выходной, при этом может работать в режиме «склейки» заявок. Разветвитель, напротив, имеет один входной и несколько выходных портов. Для него реализованы различные алгоритмы разделения входящих потоков. Таким образом, оба объекта характеризуются числом портов и дисциплиной.

TSynchronizer (синхронизатор). Имеет равное количество входных и выходных портов. Принцип работы заключается в следующем: заявки, поступающие во входные порты компонента, ждут, пока в каждом порту не окажется хотя бы по одной заявке, после чего единовременно покидают компонент – каждая через свой выходной порт.

TBusyKey (ключ занятости). Может направить заявку в стандартный выходящий порт, если компонент, следующий непосредственно за ним, не занят (число заявок в компоненте меньше заданного предела), и в порт аварийного выхода в противном случае (при этом объект-приемник извещается о предпринятой попытке его занять).

TStopper и TRecallSrc (стоппер и источник повторных вызовов). Назначение объектов данных классов состоит в задержке заявок на некоторое время. Отличие от устройств обслуживания заключается в том, что эти компоненты не бывают заняты, перед ними не возникает очереди и они могут одновременно задерживать любое количество заявок.

TSink (уничтожитель заявок). Компонент содержит единственный входящий порт; заявки, войдя в него, удаляются из модели.

TUserComponent (компонент пользователя). Разработчик сам определяет, какие порты, переменные и константы будут входить в состав компонента, а также полностью определяет его поведение.

В п. 2.3.2 рассматривается представление модели в виде класса. На уровне модели происходит автоматический подсчет общего количества сгенерированных заявок и событий, уничтоженных заявок, обработанных событий и т.д. Модель также является владельцем всех основных объектов и структур: компонентов, связей, списков событий и статистики, пользовательских переменных и т.д.

В п. 2.3.3 рассматривается работа с событиями. Работа со списком событий должна вестись независимо от типов событий, которые там хранятся. Для достижения указанной цели в основу положен абстрактный класс TEvent, в конкретных подклассах которого реализованы операции для каждого типа событий. В обработчиках событий независимо от их типа происходит сбор той или иной статистической информации и, возможно, создание другого события. Однако каждый тип события уникален, поэтому при их наступлении нужно выполнять и специфичные действия (шаблон проектирования *Шаблонный метод*).

В качестве шаблонного метода в данном случае выступает операция Handle, определенная в родительском классе (рис. 3). Ее назначение состоит в том, чтобы жестко зафиксировать последовательность шагов обработки, при этом в подклассах происходит перекрытие операций DoStart, DoAbstract и DoFinish для реализации уникального поведения. Первая реализует начальный этап обработки, вторая запускает на выполнение соответствующий скрипт, и, наконец, последняя описывает завершающий этап разработки.

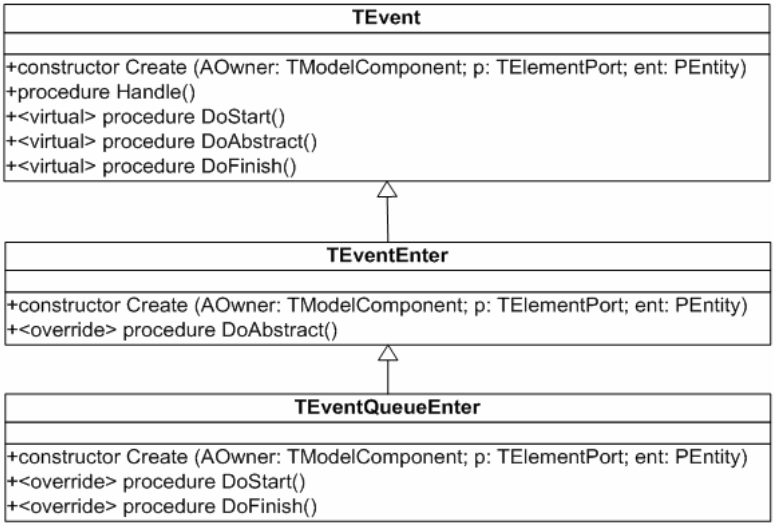

Рис. 3. Механизм обработки событий

В п. 2.3.4 рассмотрены механизмы инициализации (локальной и глобальной), а также построена диаграмма последовательности UML, демонстрирующая взаимодействие базовых классов в рамках однократного прогона имитационной модели. На уровне глобальной инициализации происходит кроме всего прочего компиляция обработчиков событий для каждого компонента. Применение шаблона проектирования *Фабричный метод* позволило использовать в качестве языка не только любой из четырех скриптовых языков, но и реализовывать собственные, не изменяя при этом существующие классы.

В п. 2.3.5 дается описание механизмов генерации случайных величин. Для получения независимых случайных величин, равномерно распределенных на интервале (0,1) используется алгоритм Мерсенна-Твистера. Остальные распределения получаются путем различных преобразований равномерного. Для воспроизведения экспериментов с одними и теми же параметрами (на уровне какого-либо из источников случайности) можно задавать номер потока. В этом случае генерация случайных значений идет по одной и той же схеме для каждого эксперимента. Начавшись с какого-то детерминированного значения, случайный поток полностью воспроизводится. Важной особенностью является также возможность создавать собственные распределения.

В п. 2.3.6 рассмотрен класс «Статистика» и приводятся формулы для вычисления основных оценок требуемых характеристик.

В п. 2.4 диссертантом предлагается перенести логику известного способа описания имитационных моделей с помощью графов событий на унифицированный язык моделирования. Обычно каждое событие является отдельной вершиной в графе, а отношения планирования между событиями представляются направленными дугами. При этом дуга может начинаться и заканчиваться в одной и той же вершине, а также между двумя событиями может быть более одной дуги. Каждой дуге ставится в соответствие некоторая функция (или число), задающая разницу во времени наступления двух событий, связанных этой дугой. Также каждой дуге может ставиться в соответствие некоторое условие, которое является булевой функцией состояния системы. Планируемое событие наступает тогда и только тогда, когда условие истинно. Проиллюстрируем данное представление логики имитационной модели, используя стандартные элементы диаграммы состояний UML.

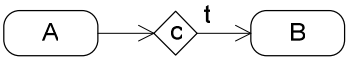

Рис. 4. Стандартные элементы графа событий в нотации UML

Данную запись можно прокомментировать так. «Когда наступает событие *A*, выполняются все изменения состояния модели, связанные с этим событием. Если условие *C* истинно, планируется наступление события *B* через *t* единиц времени». В случаях, когда некоторое событие имеет собственные параметры, все дуги графа, ведущие в соответствующую этому событию вершину, имеют список аргументов, которые должны быть при вычислении подставлены вместо параметров.

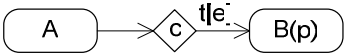

Рис. 5. Элементы диаграммы состояний с параметрами

Интерпретация выглядит следующим образом. «Когда наступает событие A, выполняются все изменения состояния модели, связанные с A. Если условие C истинно, планируется наступление события B через t единиц времени. При этом параметр p события B принимает значение выражения e».

В третьей главе диссертации приводится описание разработанного программного комплекса имитационного моделирования систем массового обслуживания «ObjectSim».

В п. 3.1 рассматриваются базовые принципы работы с комплексом. Компоненты, входящие в состав модели, должны размещаться на рабочем пространстве и соединяться между собой связями, при этом из выходных портов компонента не может выходить более одной связи, однако несколько связей могут вести в один и тот же порт.

В п. 3.2 подробно рассматривается работа с компонентами: п. 3.2.1 посвящен стандартным компонентам, настройка которых осуществляется с помощью формы свойств (уникальной для каждого компонента), а п. 3.2.2 – компонентам пользователя, создание и редактирование которых производится в специальном редакторе.

В п. 3.3 приводятся варианты реализации случайных процессов поступления заявок. Показано, как можно моделировать стационарные пуассоновские процессы (п. 3.3.1), процессы с последействием (п. 3.3.2), нестационарные пуассоновские процессы (п. 3.3.3) и групповые поступления заявок (п. 3.3.4). Комплекс также позволяет моделировать несколько эффектов вместе (например, прибытие заявок группами и переменная интенсивность процесса поступления и т.д.).

В п. 3.4 рассмотрены варианты реализации процессов обслуживания. Процесс обслуживания заявок характеризуется дисциплиной выбора требований из очереди (если таковая имеется перед устройством) и собственно дисциплиной обслуживания. В комплексе «ObjectSim» реализованы несколько стратегий выбора из очереди, в том числе с

учетом приоритетов (`PFIFO`, `PLIFO`), без учета приоритетов (`FIFO`, `LIFO`, `RANDOM`), циклический обход и пассивный режим. Что касается дисциплин обслуживания, то комплекс допускает использование стратегий с относительными и абсолютными приоритетами (в т.ч. с дообслуживанием и с обслуживанием заново), с отказами и конфликтами.

В п. 3.5 рассматриваются дополнительные средства по управлению параметрами, относящимися к модели в целом. В частности, пользователь может объявлять собственные переменные, устанавливать им начальные значения, а также создавать собственные статистики. Здесь же рассмотрен порядок запуска модели на выполнение, а также приведено краткое описание текстовых результатов моделирования (список произошедших событий и точечные оценки характеристик). Каждое событие характеризуется собственным уникальным номером, временем наступления, именем компонента, в котором оно произошло, типом и номером заявки, с которой данное событие связано. Что касается точечных оценок соответствующих характеристик, то некоторые из них могут усредняться по времени (например, занятость устройства) или по числу заявок (время обслуживания), некоторые не усредняются, а вместо этого используются достигнутые значения (входы, выходы заявок).

В п. 3.6 описаны графические результаты моделирования, которые может предоставлять комплекс «ObjectSim». Каждая характеристика оценивается с нескольких сторон: строится график изменения на основе одной реализации, а также графики математического ожидания, дисперсии и коэффициентов автокорреляции на основе нескольких прогонов. Примеры графиков приводятся в п. 3.6.1. Однако разработчика модели часто интересует поведение системы в установившемся режиме. Для определения точки перехода в стационарный режим используется графическая процедура Велча, рассмотренная в п. 3.6.2

В п. 3.6.3 описывается проведение над моделью однофакторных и многофакторных экспериментов.

В четвертой главе приводятся примеры построения реальных моделей с использованием комплекса имитационного моделирования «ObjectSim».

В п. 4.1 производится постановка общей задачи моделирования многозвенных виртуальных каналов. В п. 4.2 приводятся результаты моделирования трехзвенных виртуальных каналов с боковым трафиком, в т.ч. с одним мешающим потоком (в первом узле) и с двумя мешающими потоками. В рассмотренных моделях звенья передачи данных промоделированы одноканальными устройствами, а транзитные узлы коммутации очередями ограниченной длины.

В п. 4.3 рассмотрено моделирование 4-портового коммутатора, где выходной порт представлен в виде одноканального устройства обслуживания, а входные порты в виде очередей ограниченной длины.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1. Впервые продемонстрированы в нотации UML (а именно в виде диаграмм активности) основные механизмы работы дискретно-событийного метода и их взаимодействие для случаев однократного и многократного запуска.

2. Предложен оригинальный способ работы с событиями, возникающих в ходе выполнения имитационной модели. Он заключается в использовании скриптовых языков, синтаксис и возможности которых приближены к традиционным языкам программирования.

3. Разработана архитектура программного комплекса, реализующего дискретно-событийное моделирование. В рамках данной архитектуры построена концептуальная модель системы в виде стандартных диаграмм вариантов ее использования, а также логическая модель, представленная в виде диаграмм классов и взаимодействия. Проведенная адаптация шаблонов проектирования под конкретные задачи предметной области позволила создать такую архитектуру, при которой можно добавлять новые классы и алгоритмы поведения, не вдаваясь существенно в реализацию уже разработанных классов.

4. Предложен способ представления графа событий имитационной модели в виде стандартных диаграмм состояния UML. В результате получено решение, соответствующее одновременно и методам, принятым при описании дискретно-событийных моделей, и стандарту UML. Предложенный подход представления логики работы имитационной модели позволил унифицированным способом документировать и обсуждать имитационные модели независимо от того, в какой среде или на каком языке эти модели разрабатываются.

5. Создан программный комплекс «ObjectSim», позволяющий без изучения синтаксиса специализированного языка имитационного моделирования строить модели систем массового обслуживания. Имея при этом навыки программирования на одном из традиционных языков, пользователь может доопределять логику работы моделей.

6. Проведено исследование имитационных моделей виртуальных каналов с боковым (мешающим) трафиком, позволившее оценить основные операционные характеристики каналов для широкого класса нагрузочных показателей.

СПИСОК ПУБЛИКАЦИЙ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Змеев О.А., Приступа А.В. Диаграммы состояния UML как способ представления графа событий имитационной модели дискретной системы массового обслуживания // Обработка данных и управление в сложных системах. Томск: Изд-во Том. ун-та, 2004. – Вып.6. – С.76-81.

2. Змеев О.А., Приступа А.В. Классификация коммерческих систем имитационного моделирования // Материалы Всероссийской конференции "Наука и практика: Диалоги нового века". – Анжеро-Судженск, 14 ноября 2003г. – Часть 3. – С.93-95.

3. Змеев О.А., Приступа А.В. Применение паттернов проектирования при построении систем имитационного моделирования // Материалы VIII Всероссийской научно-практической конференции "Научное творчество молодежи". – Анжеро-Судженск, 16-17 апреля 2004г. – С.36-38

4. Змеев О.А., Приступа А.В. Разработка объектно-ориентированного программного комплекса имитационного моделирования систем массового обслуживания // Вестник Томского государственного университета. – 2004. – №284. – С.181-184.

5. Приступа А.В. Имитационное моделирование систем массового обслуживания. Шаг моделирования // Материалы VII Межрегиональной научно-практической конференции "Научное творчество молодежи". – Анжеро-Судженск, 18 апреля 2003г. – Часть 2. – С.39-40.

6. Приступа А.В. Инструмент для объектного моделирования систем с дискретными событиями ObjectSim // Материалы Второй всероссийской научно-практической конференции ИММОД-2005. – Санкт-Петербург, 19-21 октября 2005г. – СПб.: ФГУП "ЦНИИТС". – 2005. – Т.1. – С.264-268.

7. Pristupa A.V., Zmeyev O.A. Design Patterns in Discrete Event Simulation // 8th Korea-Russia International Symposium on Science and Technology. PROCEEDINGS. KORUS. – New York City, NY: IEEE, 2004. – Vol. 1. – PP. 141-144.